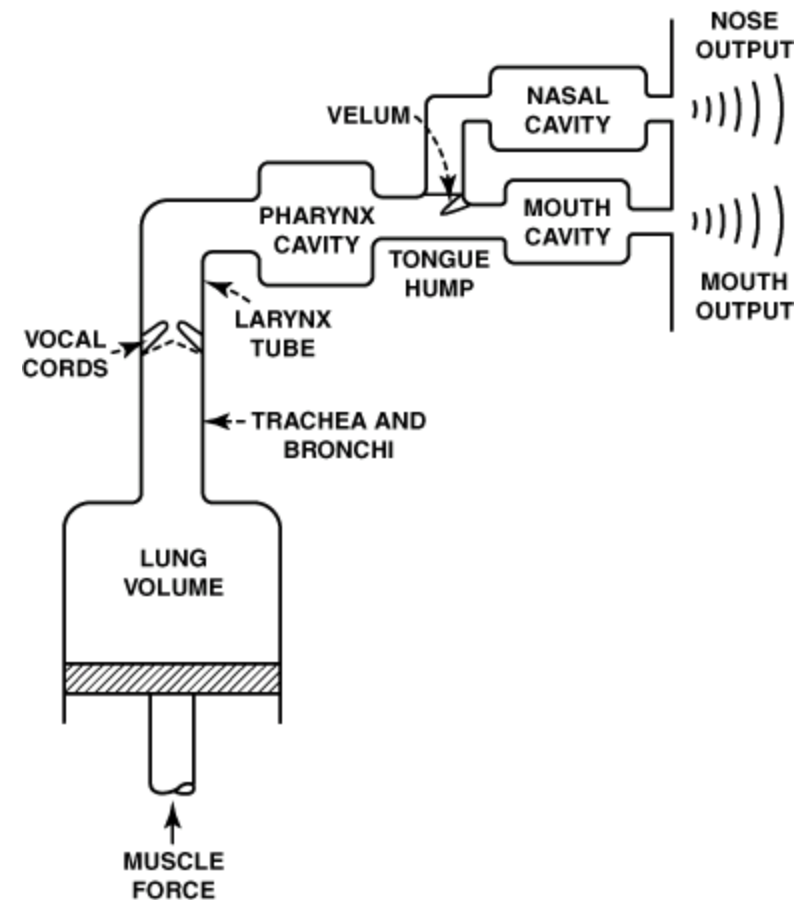
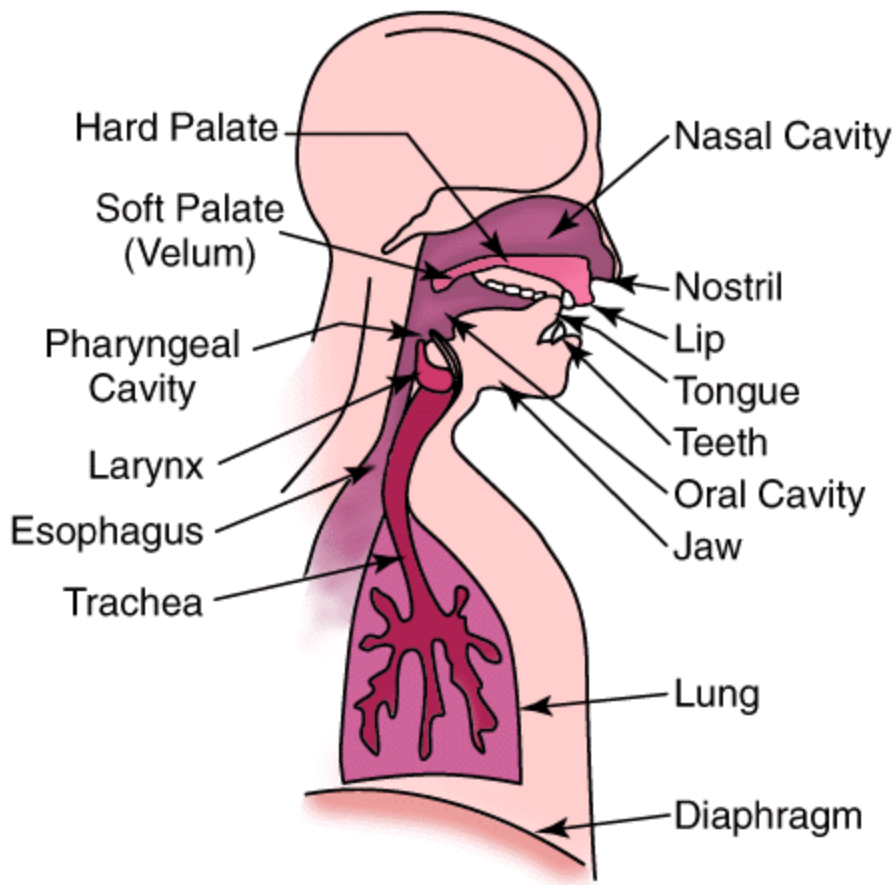


Speech Recognition

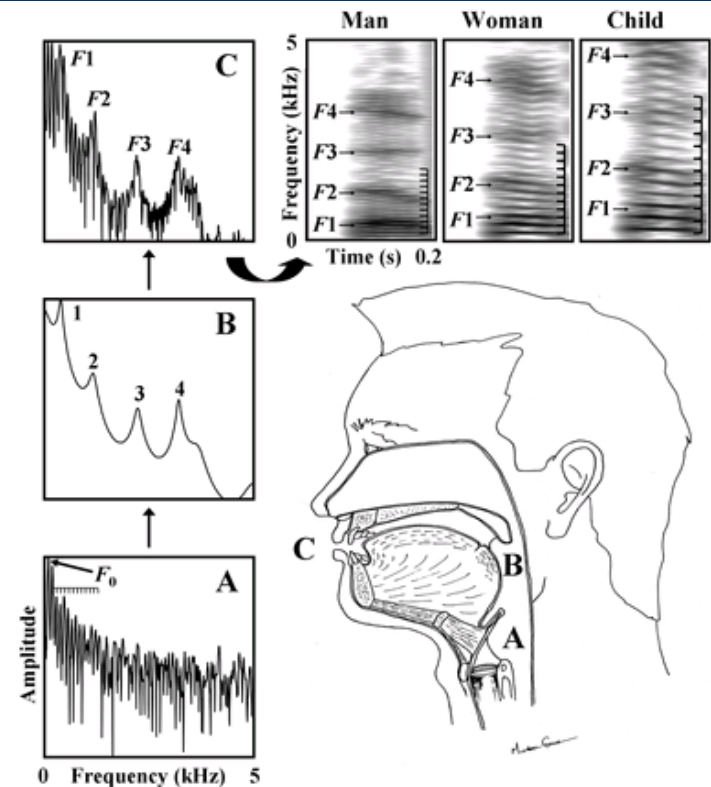
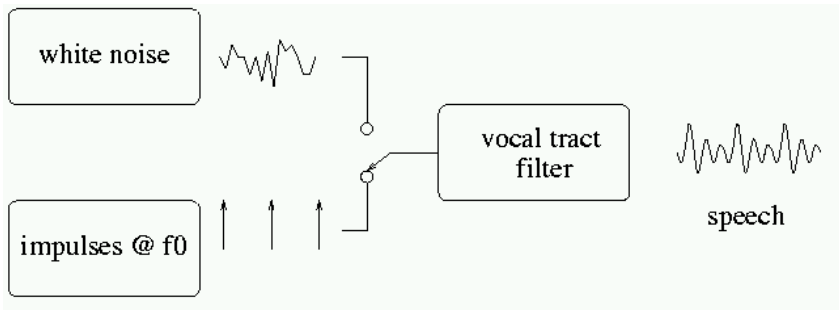
Chapter 15.1 – 15.3, 15.6



Speech Production



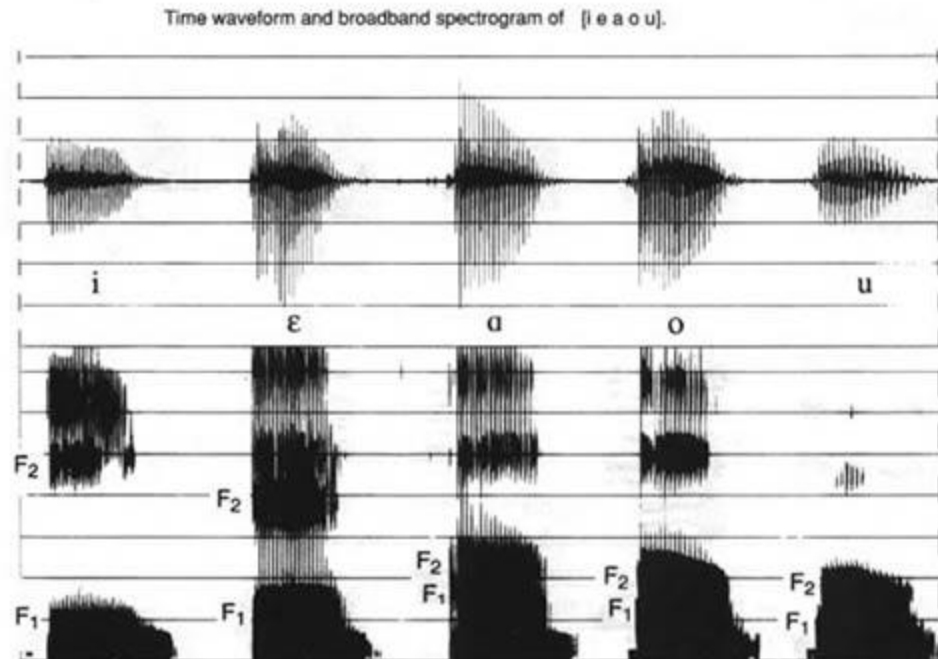
Speech Production



- Demo: jaw harp
- X-ray speech

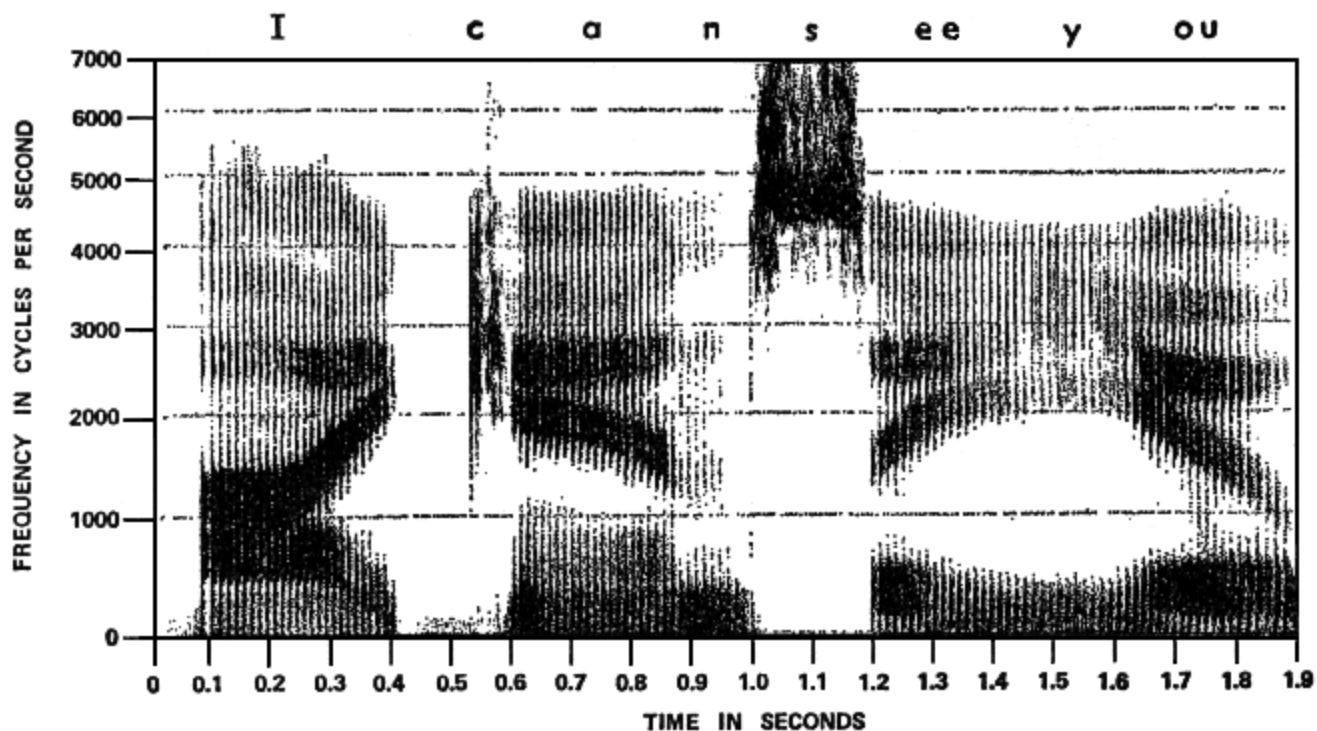
- <http://hundertmarkblog.de/christine-ericsson-x-ray-analyses-of-speech/>
- http://www.ling.su.se/STAFF/ericsson/projects/xray_info.html

Spectrograph



More spectrograph

- “I can see you”



Phones

- Human languages are limited to a set of about 40 to 50 distinct sounds called **phones**, e.g.,
 - [ey] bet
 - [ah] but
 - [oy] boy
 - [em] bottom
 - [en] button
- These phones are characterized in terms of acoustic features, e.g., frequency and amplitude, that can be extracted from the sound waves

International Phonetic Alphabet (IPA)

- **IPA:** <http://www.phonetics.ucla.edu/course/chapter1/consonants2.html>
- **Consonants**

CONSONANTS (PULMONIC)

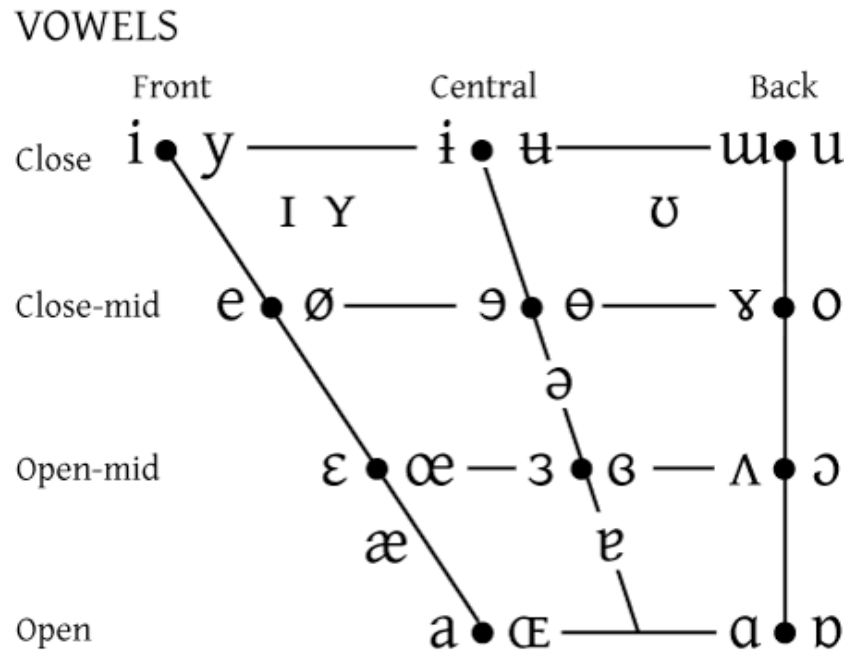
	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b			t d		ʈ ɖ	c ɟ	k ɡ	q ɢ		ʔ
Nasal	m	ɱ		n		ɳ	ɲ	ŋ	ɴ		
Trill	ʙ			r					ʀ		
Tap or Flap				ɾ		ɽ					
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Lateral fricative				ɬ ɮ							
Approximant		ʋ		ɹ		ɻ	j	ɰ			
Lateral approximant				l		ɭ	ʎ	ʟ			

Where symbols appear in pairs, the one to the right represents a voiced consonant. Shaded areas denote articulations judged impossible.

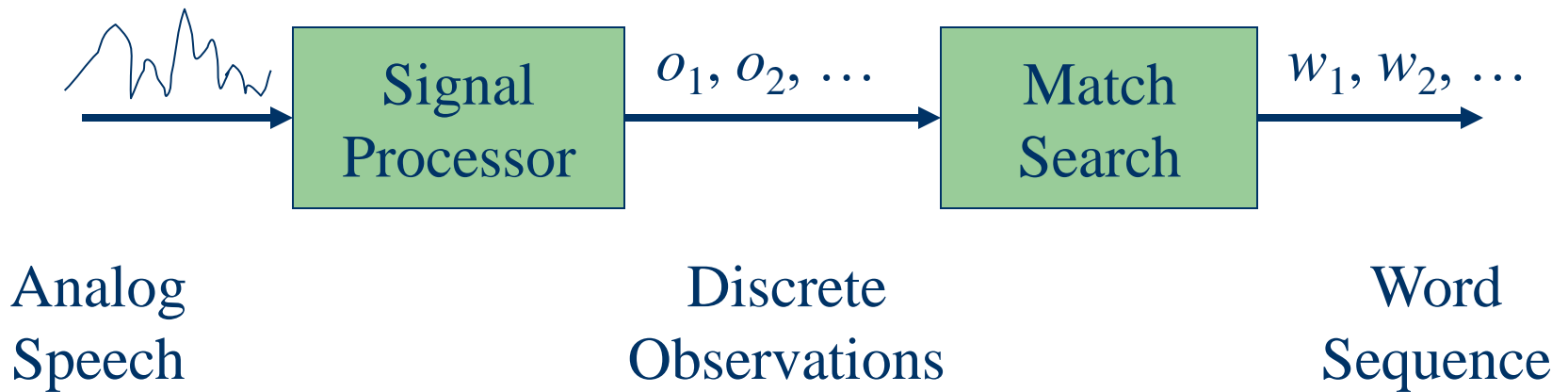
- **Demo: click language**
http://www.youtube.com/watch?v=D_I7ty_MH_Y

International Phonetic Alphabet (IPA)

- Vowels



Speech Recognition Task



Hearing speech with your eyes

- **Demo: the McGurk effect**
<http://www.youtube.com/watch?v=aFPtc8BVdJk>
- **Computer vision for lip reading can improve speech recognition accuracy, especially in high noise conditions (e.g., fighter cockpit)**

Introduction

- **Why isn't this easy?**

- just develop a dictionary of pronunciation
e.g., coat = [k] + [ow] + [t] = [kownt]
- **but** “recognize speech” ≈ “wreck a nice beach”

- **Problems:**

- **Homophones**: different fragments sound the same
 - e.g., “rec” and “wreck”
- **Segmentation**: determining breaks between words
 - e.g., “nize speech” and “nice beach”
- Speaker variation, condition (cellphone, meeting...), signal processing problems

Signal Processing

- **Sound is an analog energy source resulting from pressure waves striking an eardrum or microphone**
- **An analog-to-digital converter can be used to record the speech sounds**
 - **Sampling**: the number of times per second that the sound level is measured
 - **Quantization**: the number of bits of precision for the sound level measurements
 - Telephone: 3 KHz (3000 times per second)
 - Speech recognizer: 8 KHz with 8 bits/sample so that 1 minute takes about 500K bytes

Signal Processing

- **Wave encoding**
 - group into ~10 msec **frames** (larger blocks) that are analyzed individually
 - frames overlap to ensure important acoustical events at frame boundaries aren't lost
 - frames are analyzed in terms of features
 - amount of energy at various frequencies
 - total energy in a frame
 - differences from prior frame
 - **vector quantization** further encodes by mapping frame into regions in n -dimensional feature space

Input Sequence

- **Vector Quantization encodes each successive frame as one of, say, 256 possible observation values: C_1, C_2, \dots, C_{256}**
- **Input is a sequence such as**
 $C_{82}, C_{44}, C_{63}, C_{44}, C_{25}, \dots$
 $= O_1, O_2, O_3, O_4, O_5, \dots$

Signal Processing

- Goal is **speaker independence** so that *representation of sound is independent of a speaker's specific pitch, volume, speed, and other aspects such as dialect*
- **Speaker identification** does the opposite, i.e., the specific details are needed to decide who is speaking
- A significant problem is dealing with background noises that are often other speakers

Speech Recognition Model

- **Bayes's Rule** is used break up the problem into manageable parts:

$$P(\text{words} \mid \text{signal}) = \frac{P(\text{words}) P(\text{signal} \mid \text{words})}{P(\text{signal})}$$

- $P(\text{signal})$: is ignored (normalizing constant)
- $P(\text{words})$: **Language model**
 - likelihood of words being heard
 - e.g., “recognize speech” more likely than “wreck a nice beach”
- $P(\text{signal} \mid \text{words})$: **Acoustic model**
 - likelihood of a signal given words
 - accounts for differences in pronunciation of words
 - e.g., given “nice,” likelihood that it is pronounced [nuys]

• *Signal* = **observation sequence** $O = o_1, o_2, o_3, \dots, o_t$

• *Words* = **string of words** $W = w_1, w_2, w_3, \dots, w_n$

• Best match metric: **probability** $\hat{W} = \arg \max_{W \in L} P(W | O)$

• **Bayes's rule:**

$$\hat{W} = \arg \max_{W \in L} \frac{P(O | W)P(W)}{P(O)}$$

$$= \arg \max_{W \in L} P(O | W)P(W)$$

observation likelihood
(acoustic model)

prior probability
(language model)

Language Model (LM)

- * $P(\text{words})$ is the joint probability that a sequence of words = $w_1 w_2 \dots w_n$ is likely for a specified natural language
- This joint probability can be expressed using the chain rule (order reversed):
$$P(w_1 w_2 \dots w_n) = P(w_1) P(w_2 / w_1) P(w_3 / w_1 w_2) \dots P(w_n / w_1 \dots w_{n-1})$$
- Collecting the probabilities is too complex; it requires statistics for m^{n-1} starting sequences for a sequence of n words in a language of m words
 - Simplification is necessary

Language Model (LM)

- **First-order Markov Assumption**

- Probability of a word depends only on the previous word:

$$P(w_i / w_1 \dots w_{i-1}) \approx P(w_i / w_{i-1})$$

- **The LM simplifies to**

$$P(w_1 w_2 \dots w_n) = P(w_1) P(w_2 / w_1) P(w_3 / w_2) \dots P(w_n / w_{n-1})$$

- called the **bigram model**
- relates consecutive pairs of words

Language Model (LM)

- More context could be used, such as the two words before, called the **trigram model**, but it's difficult to collect sufficient data to get accurate probabilities
- A weighted sum of **unigram, bigram, trigram models** could be used in combination:

$$P(w_1 w_2 \dots w_n) = \prod (c_1 P(w_i) + c_2 P(w_i / w_{i-1}) + c_3 P(w_i / w_{i-1} w_{i-2}))$$

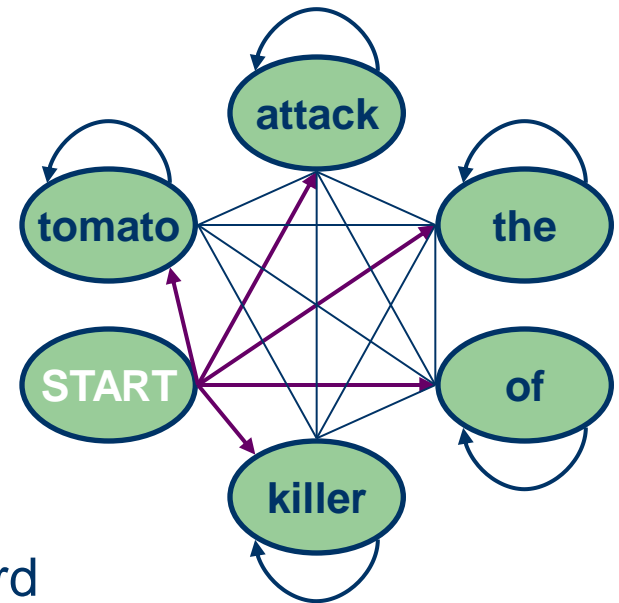
- **Bigram and trigram models account for**
 - *local* context-sensitive effects
 - e.g., "bag of tricks" vs. "bottle of tricks"
 - some *local* grammar
 - e.g., "we was" vs. "we were"

Language Model (LM)

- **Probabilities are obtained by computing statistics of the frequency of all possible pairs of words in a large training set of word strings:**
 - if "the" appears in training data 10,000 times and it's followed by "clock" 11 times then
$$P(\text{clock} \mid \text{the}) = 11/10000 = .0011$$
- **These probabilities are stored in**
 - a probability table
 - a probabilistic finite state machine

Language Model (LM)

- **Probabilistic finite state machine:** a (almost) fully connected directed graph:
 - **nodes (states):** all possible words and a START state
 - **arcs:** labeled with a probability
 - from START to a word is the prior probability of the destination word
 - from one word to another is the probability of the destination word given the source word

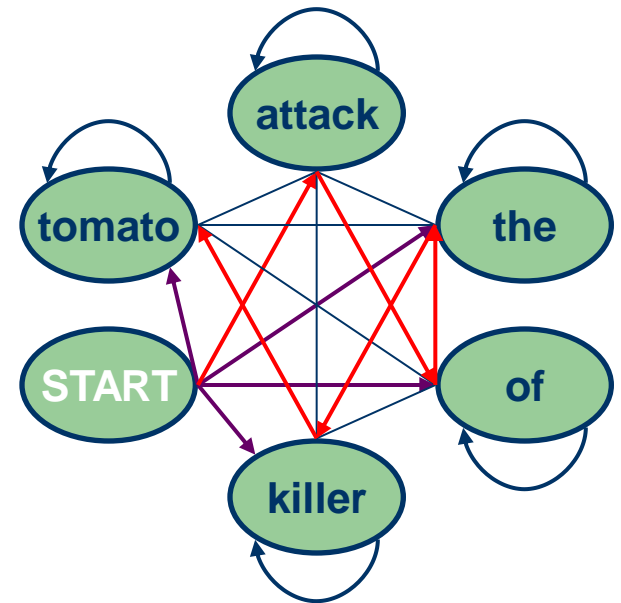


Language Model (LM)

- **Probabilistic finite state machine:** a (almost) fully connected directed graph:

- joint probability is estimated for the *bigram* model by starting at START and multiplying the probabilities of the arcs that are traversed for a given sentence

- $P(\text{"attack of the killer tomato"}) = P(\text{attack}) P(\text{of} \mid \text{attack}) P(\text{the} \mid \text{of}) P(\text{killer} \mid \text{the}) P(\text{tomato} \mid \text{killer})$



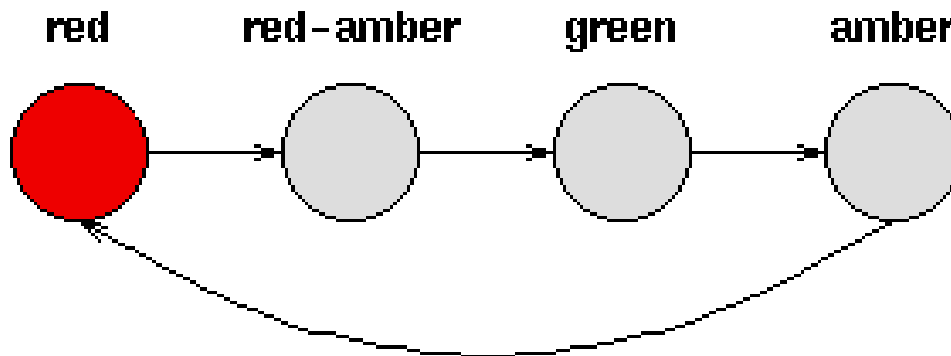
Acoustic Model (AM)

- $P(\text{signal} \mid \text{words})$ is the conditional probability that a signal is likely given a sequence of words for a particular natural language
- This is divided into two probabilities:
 - $P(\text{phones} \mid \text{word})$: probability of a sequence of phones given word
 - $P(\text{signal} \mid \text{phone})$: probability of a sequence of vector quantization values from the acoustic signal given phone

Finding Patterns

- Speech is an example of a more general problem of finding patterns over time (or any sequence)
- Deterministic patterns have a fixed sequence of states, where the next state is dependent solely on the previous state

European
Stop light



Non-Deterministic Patterns

- Discrete set of states, but can't model deterministically the sequence
- Example: Predicting the weather
 - **States**: Sunny, Cloudy, Rainy
 - **Arcs**: Probability, called the **state transition probability**, of moving from one state to another
- **N th-order Markov assumption**: Today's weather can be predicted solely given knowledge of the last N days' weather
- **1st-order Markov assumption**: Today's weather can be predicted solely given knowledge of yesterday's weather

1st-Order Markov Model

- Markov process is a process that moves from state to state probabilistically based on the **state transition matrix** associated with the graph
- Sum of values in each row and each column is 1
- 1st-order Markov model for weather prediction:



1st-Order Markov Model

- To initialize the process, also need to state the probability of the **initial state** at time $t=0$, called the **π vector**. For example, if we know the first day was sunny, then $\pi =$

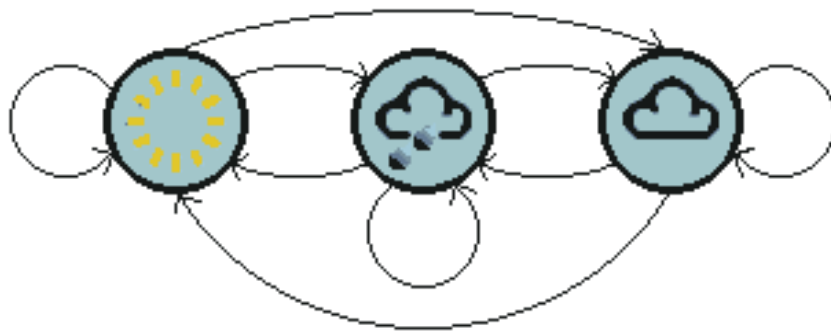
$$\begin{array}{ccc} & \text{Sun} & \text{Cloud} & \text{Rain} \\ \left(\right. & 1.0 & 0.0 & 0.0 \end{array}$$

- For simplicity, we will assume a single, given state is the start state

1st-Order Markov Model

- **Markov Model $M = (A, \pi)$ consists of**
 - Discrete set of states, s_1, s_2, \dots, s_N
 - $\boldsymbol{\pi}$ vector, where $\pi_j = P(q_1=s_j)$
 - State transition matrix, $\mathbf{A} = \{a_{ij}\}$ where $a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i)$
- **The state transition matrix is fixed for all times and describes probabilities associated with a (completely-connected) graph of the states**

Example: Using a Markov Model for Weather Prediction



		weather today		
		Sun	Cloud	Rain
weather yesterday	Sun	0.5	0.25	0.25
	Cloud	0.375	0.125	0.375
	Rain	0.125	0.625	0.375

$$\pi = \begin{pmatrix} \text{Sun} & \text{Cloud} & \text{Rain} \\ 1.0 & 0.0 & 0.0 \end{pmatrix}$$

Given that today is sunny, what is the probability of the next two days being sunny and rainy, respectively?

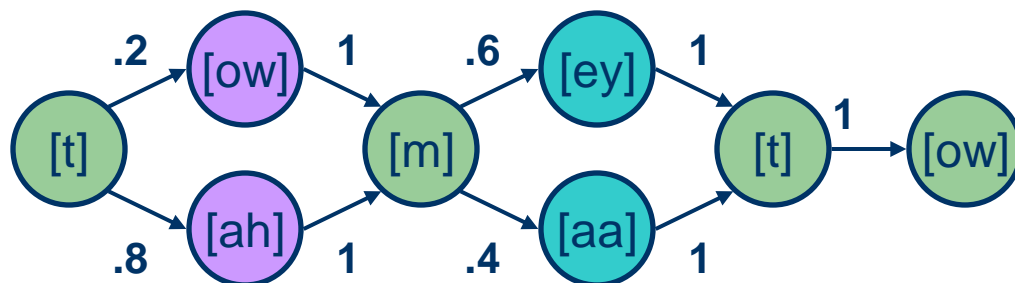
Weather Prediction Example (cont.)

- $P(w_2 = \text{Sun}, w_3 = \text{Rain} \mid w_1 = \text{Sun}) = ?$

- $P(w_3 = \text{Rain} \mid w_1 = \text{Cloudy}) = ?$

Acoustic Model (AM)

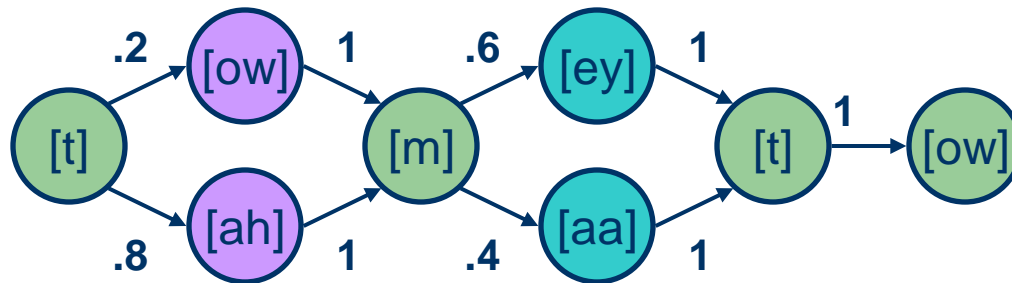
- $P(\text{phones} / \text{word})$ can be specified as a **Markov model**, which is a way of describing a process that goes through a series of states, e.g., “tomato”



- **Nodes:** correspond to the production of a phone
 - **sound slurring** (coarticulation) typically from quickly pronouncing a word
 - **variation in pronunciation** of words typically due to dialects
- **Arcs:** probability of transitioning from current state to another

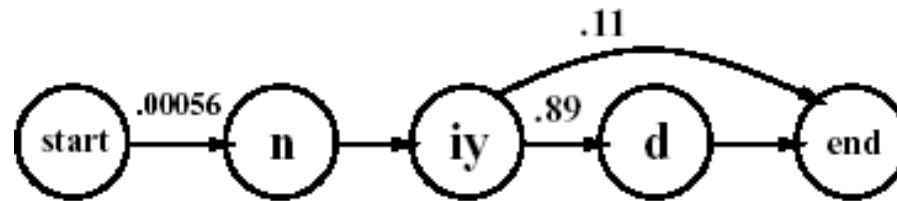
Acoustic Model (AM)

- $P(\text{phones} / \text{word})$ can be specified as a **Markov model**, which is a way of describing a process that goes through a series of states, e.g., “tomato:”



- $P(\text{phones} / \text{word})$ is a **path** through the diagram, i.e.,
 - $P[\text{towmeytow} / \text{tomato}] = 0.2 * 1 * 0.6 * 1 * 1 = 0.12$
 - $P[\text{towmaatow} / \text{tomato}] = 0.2 * 1 * 0.4 * 1 * 1 = 0.08$
 - $P[\text{tahmeytow} / \text{tomato}] = 0.8 * 1 * 0.6 * 1 * 1 = 0.48$
 - $P[\text{tahmaatow} / \text{tomato}] = 0.8 * 1 * 0.4 * 1 * 1 = 0.32$

Acoustic Model for “Need”



Word model for "need"

- **Look at probabilities of various phonemes as we listen:**
 - In corpus “need” always starts with "n" sound
 - What are the possibilities for the next sound? With probability 1, we know that next sound will be "iy"
 - What are possibilities for next sound? 11% of the time, “d” sound will be omitted
 - Probability of transitioning from "iy" to the "d" sound is 89
- **Circles represent two things—states and observations**
- **In real world, state is hidden: For sound [iy], we don't know whether we are at second phoneme of the word “knee” or the second phoneme of the word “need”**
- **Simplify: we know the word as well as the phoneme**

Problem

- We don't know the sequence of phones, we only have the observation sequence o_1, o_2, o_3, \dots
- How do we relate the given input sequence to phone sequences?

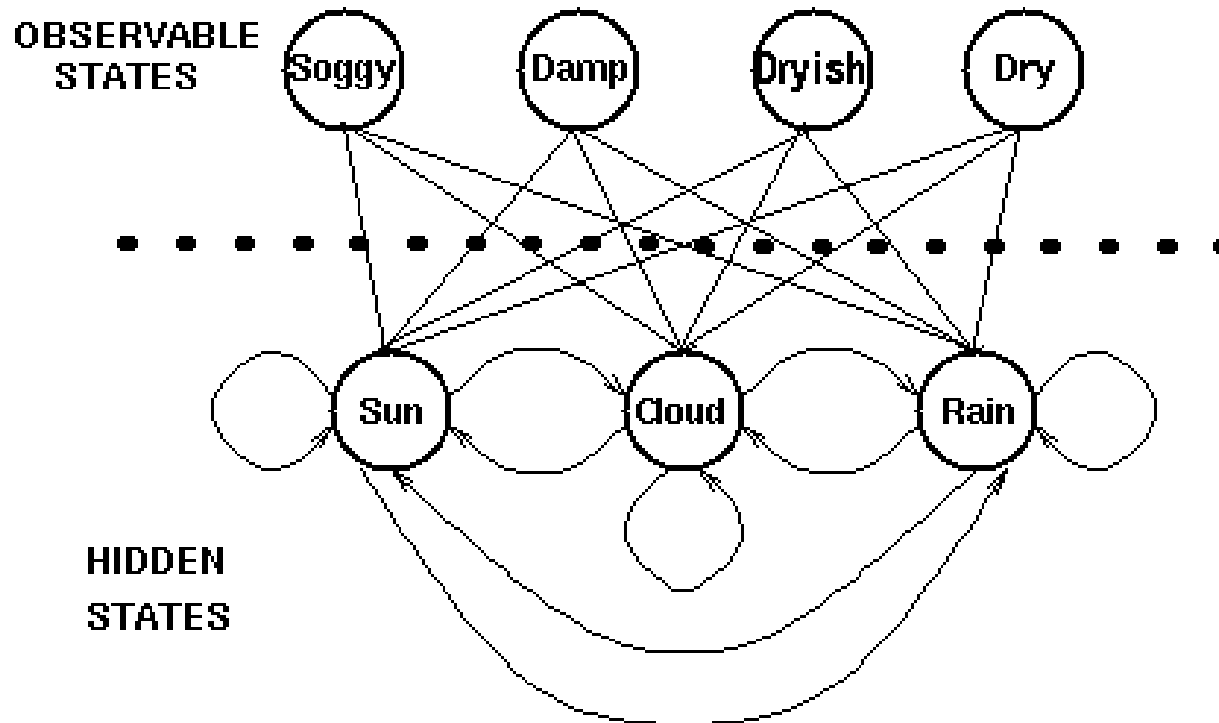
Hidden Markov Models (HMMs)

- Sometimes the states we want to predict are *not directly observable*; only observations available are indirect evidence
- Example: A CS major does not have direct access to the weather, but can only observe the state of a piece of **corn** (dry, dryish, damp, soggy)
- Example: In speech recognition we can observe features of the changing sound, i.e., O_1, O_2, \dots , but there is no direct evidence of the words being spoken

HMMs

- **Hidden States:** The states of real interest, e.g., the true weather or the sequence of words spoken; represented as a 1st-order Markov model
- **Observable Values:** A discrete set of observable values; the number of observable values is *not*, in general, equal to the number of hidden states. The observable values are related *somehow* to the hidden states (i.e., *not* 1-to-1 correspondence)

Hidden Markov Model



Arcs and Probabilities in HMMs

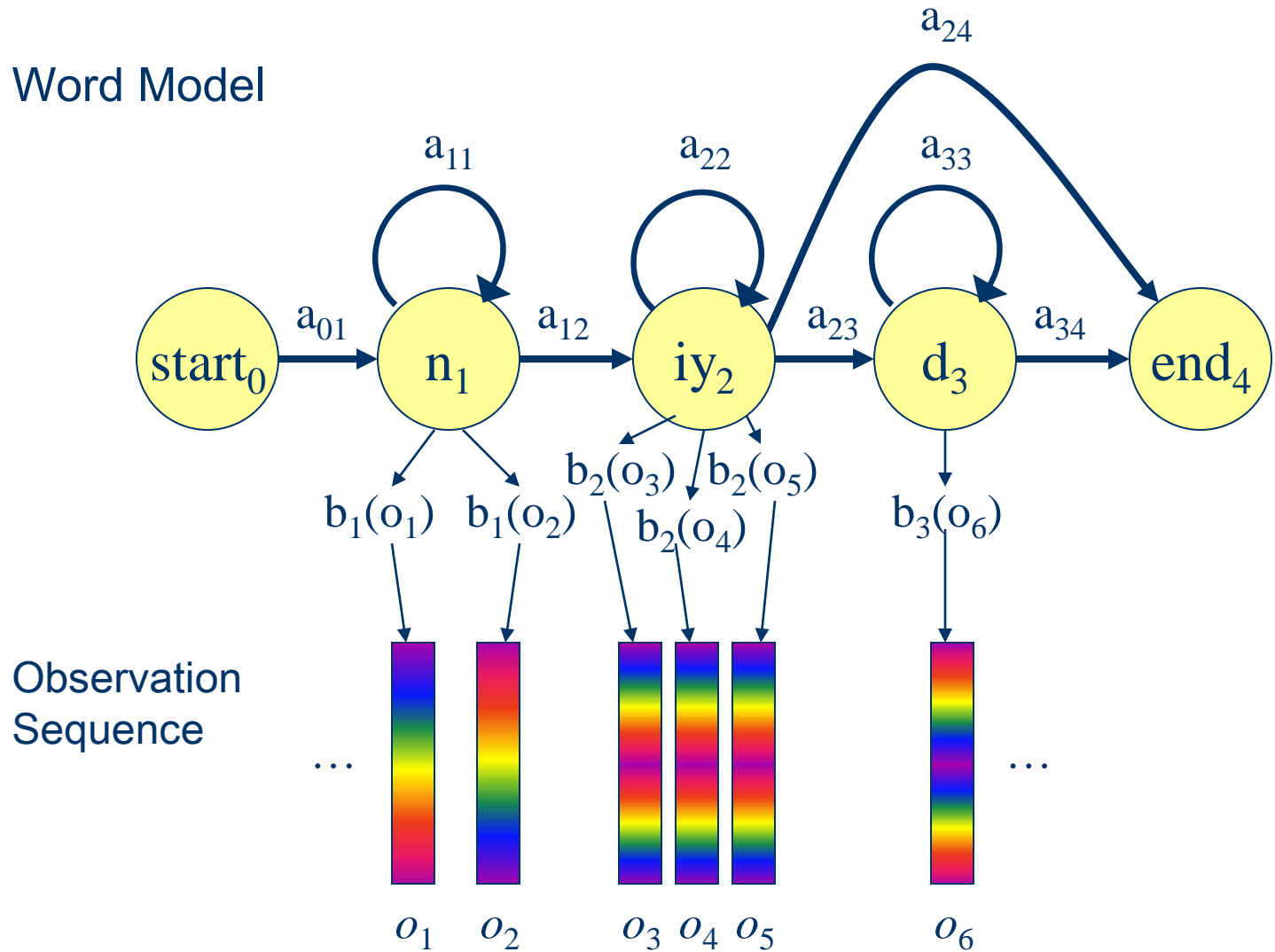
- Arcs connecting hidden states and observable values represent the probability of generating an observed value given that the Markov process is in a hidden state
- **Observation Likelihood matrix, B, (aka Confusion matrix or output probability distribution)** stores probabilities associated with arcs from hidden states to observable values, i.e., $P(\text{Obs} \mid \text{Hidden})$

		corn			
		Dry	Dryish	Damp	Soggy
weather	Sun	0.60	0.20	0.15	0.05
	Cloud	0.25	0.25	0.25	0.25
	Rain	0.05	0.10	0.35	0.50

HMM Summary

- **An HMM contains 2 types of information**
 - Hidden states: s_1, s_2, s_3, \dots
 - Observable values
 - In speech recognition, the vector quantization values in the input sequence $\mathbf{O} = o_1, o_2, o_3, \dots$
- **An HMM, $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$, contains 3 sets of probabilities**
 - $\boldsymbol{\pi}$ vector, $\boldsymbol{\pi} = (\pi_i)$
 - State transition matrix, $\mathbf{A} = (a_{ij})$ where $a_{ij} = P(q_t = s_j \mid q_{t-1} = s_i)$
 - Observation likelihood, $\mathbf{B} = b_j(o_k) = P(y_t = o_k \mid q_t = s_j)$

Example: An HMM Word Model



Acoustic Model (AM)

- $P(\text{signal} / \text{phone})$ can be specified as an **HMM**, e.g., phone model for [m]:



Possible
outputs:

C1: 0.5
C2: 0.2
C3: 0.3

C3: 0.2
C4: 0.7
C5: 0.1

C4: 0.1
C6: 0.5
C7: 0.4

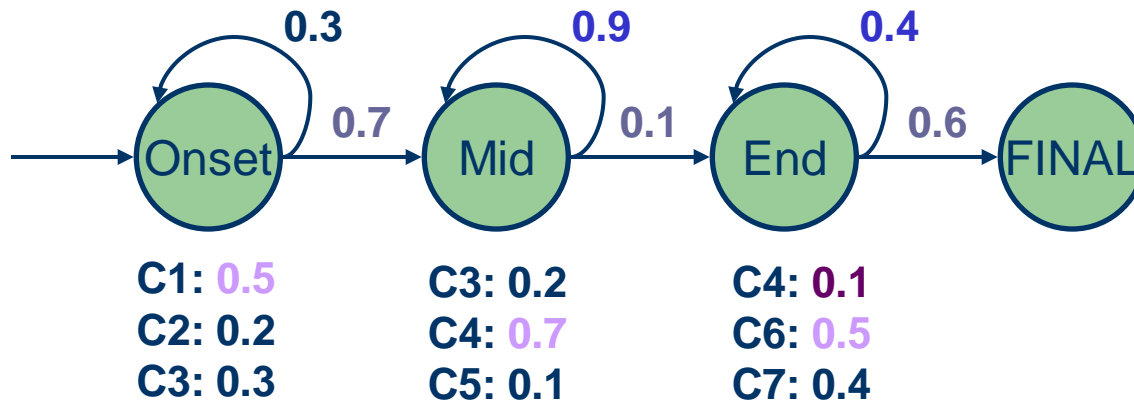
- **nodes**: probability distribution over a set of vector quantization values
- **arcs**: probability of transitioning from current state to another

Generating HMM Observations

- Choose an initial state, $q_1 = s_i$, based on π
- For $t = 1$ to T do
 - Choose output value $z_t = o_k$ according to the symbol probability distribution in state s_i , $b_i(k)$
 - Transition to a new state $q_{t+1} = s_j$ according to the state transition probability distribution for state s_i , a_{ij}
- So, transition to new state and *then* output value at the new state

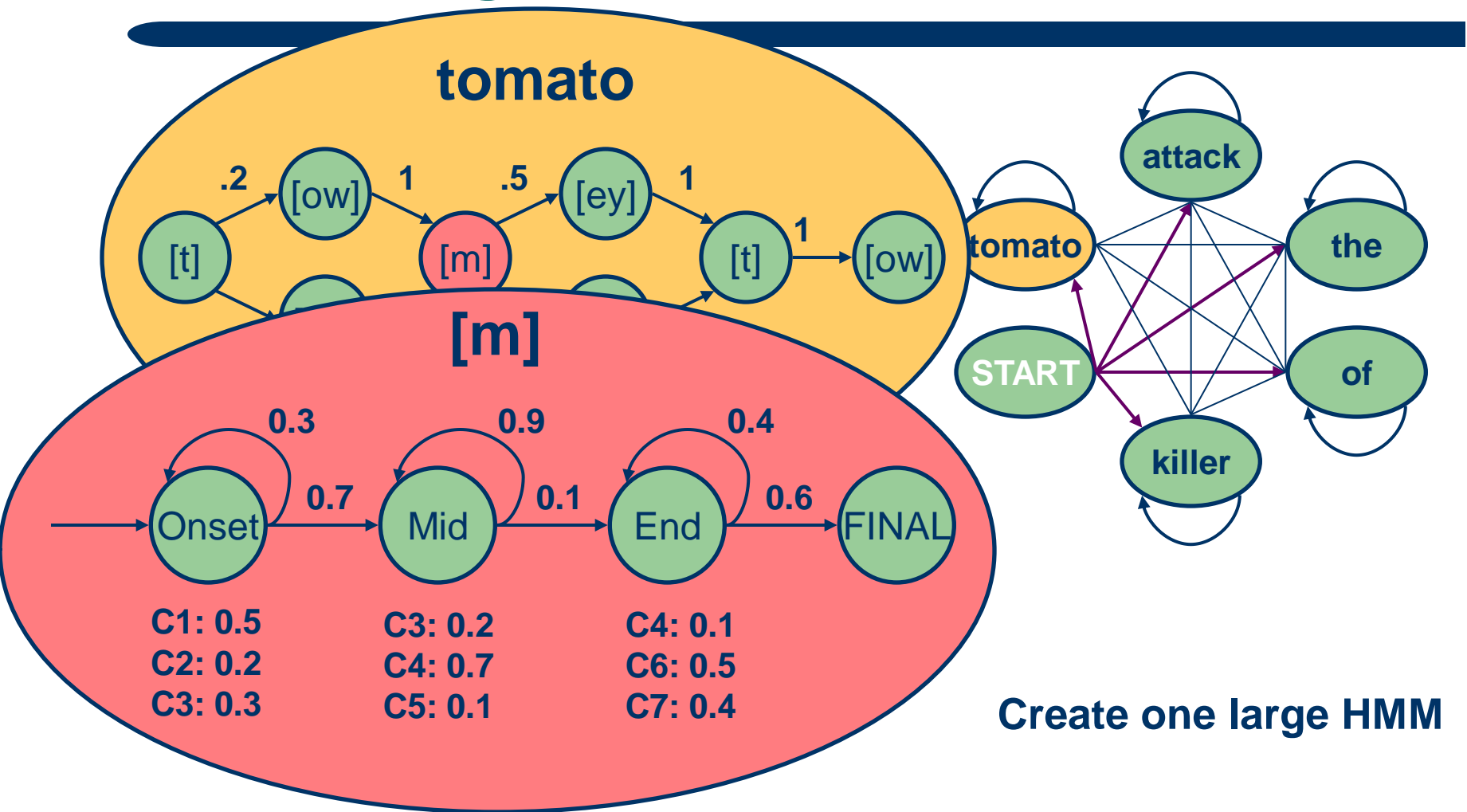
Acoustic Model (AM)

- $P(\text{signal} / \text{phone})$ can be specified as an **HMM**, e.g., phone model for [m]:



- $P(\text{signal} / \text{phone})$ is a path through the diagram, i.e.,
 - $P[\text{C1}, \text{C4}, \text{C6}] / [m]) = (0.7 * 0.1 * 0.6) * (0.5 * 0.7 * 0.5) = 0.00735$
 - $P[\text{C1}, \text{C4}, \text{C4}, \text{C6}] / [m]) = (0.7 * 0.9 * 0.1 * 0.6) * (0.5 * 0.7 * 0.7 * 0.5)$
 - $(0.7 * 0.1 * 0.4 * 0.6) * (0.5 * 0.7 * 0.1 * 0.5) = 0.0049245$

Combining Models



Evaluation/Scoring Problem: $P(O | \lambda)$

- Find the probability of an observed sequence given an HMM. For example, given several HMMs such as a “summer HMM” and a “winter HMM” and a sequence of corn observations, which HMM most likely generated the given sequence?
- In speech recognition, use one HMM for each word, and an observation sequence from a spoken word. Compute $P(O|\lambda)$. Recognize the word by identifying the most probable HMM.
- Use with Bayes’s Rule to solve State Estimation problems, i.e., given an observation sequence, what’s the most likely state that I’m in?
- Use **Forward algorithm**

Decoding/Matching Problem

- Find the most probable sequence (i.e., path) of hidden states given an observation sequence
- Compute $Q^* = \operatorname{argmax}_Q P(Q | O)$
- Use **Viterbi algorithm**

Learning/Training Problem

- **Generate an HMM given a sequence of observations and a set of known hidden states**
- **Learn the most probable HMM**
- **Compute $\lambda^* = \operatorname{argmax}_{\lambda} P(O | \lambda)$**
- **Use Forward-Backward algorithm and Expectation-Maximization (EM) algorithm**

Evaluation Problem: Forward Algorithm

- $P(\mathbf{O} | \lambda) = ?$ where $\mathbf{O} = o_1, o_2, \dots, o_T$ is the observation sequence and λ is an HMM model
- Let $\mathbf{Q} = q_1, q_2, \dots, q_T$ be a specific hidden state sequence

$$P(\mathbf{O} | \lambda) = \sum_{\mathbf{Q}} P(\mathbf{O} | \mathbf{Q}, \lambda) P(\mathbf{Q} | \lambda) \quad \text{by Conditioning rule}$$

$$P(\mathbf{O} | \mathbf{Q}, \lambda) = \prod_{t=1}^T P(o_t | q_t, \lambda) = b_{q_1}(o_1) b_{q_2}(o_2) \dots b_{q_T}(o_T)$$

$$P(\mathbf{Q} | \lambda) = \pi_{q_1} a_{q_1 q_2} \dots a_{q_{T-1} q_T}$$

$$\text{So, } P(\mathbf{O} | \lambda) = \sum_{\mathbf{Q}} \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) a_{q_2 q_3} b_{q_3}(o_3) \dots a_{q_{T-1} q_T} b_{q_T}(o_T)$$

- $(\pi, \mathbf{A}, \mathbf{B})$ known for given HMM λ

Computation of $P(O|\lambda)$

- Since there are N^T sequences (N hidden states and T observations), $O(T N^T)$ calculations required!
- For $N=5$, $T=100$, 10^{72} computations

Forward Algorithm

- Compute Bayesian likelihood

$$P(O | \lambda)$$

- incrementally by following the state transitions for the HMM model λ

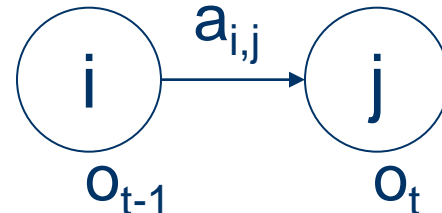
- Let $O_t = o_1 o_2 \dots o_{t-1} o_t$

- $P(O_t, j | \lambda) = P(O_{t-1}, i | \lambda) a_{i,j} b_{jt}$

- in state j at o_t , previously state i at o_{t-1}

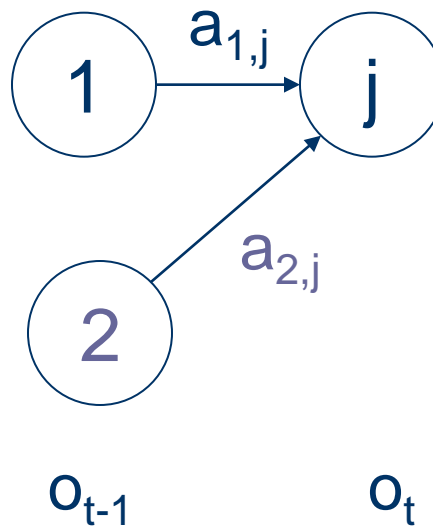
- $a_{i,j}$ is the **transition probability** between state i and state j

- $b_{i,t}$ is the **observation likelihood** that the state j label matches o_t



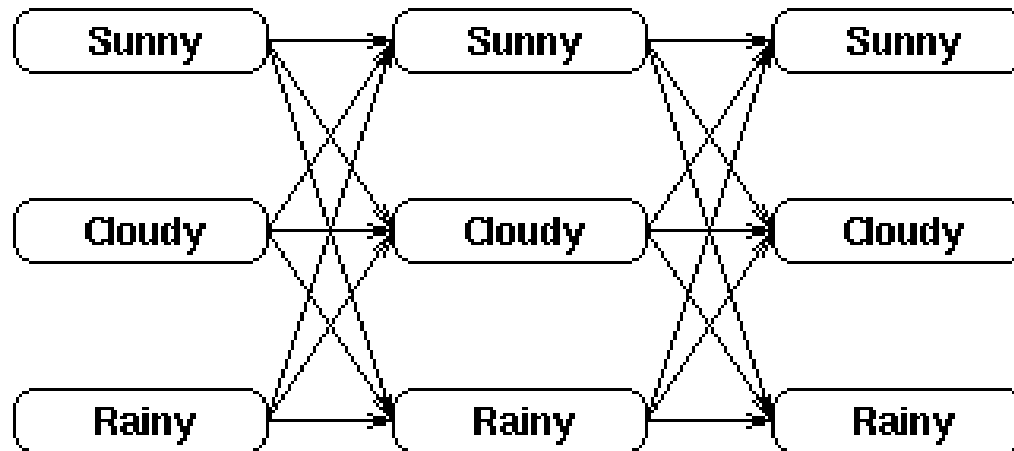
Forward Algorithm

- If there are multiple states leading to state j
 - $P(O_t, j | \lambda) = \sum_{\text{all } i} P(O_{t-1}, i | \lambda) a_{i,j} b_{jt}$
 - i.e., *sum the probabilities*



Evaluation using Exhaustive Search

- Given observation sequence (dry, damp, soggy), “unroll” the hidden state sequence as a trellis (matrix):



Observations : **dry** **damp** **soggy**

Evaluation using Exhaustive Search

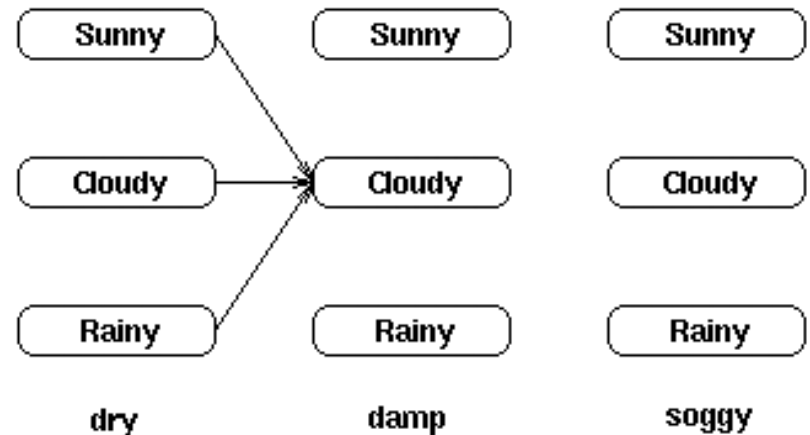
- Each column in the trellis (matrix) shows a possible state of the weather
- Each state in a column is connected to each state in the adjacent columns
- Sum the probabilities of each possible sequence of the hidden states; here, $3^3 = 27$ possible weather sequences
- $P(\text{dry,damp,soggy} \mid \text{HMM } \lambda) =$
 $P(\text{dry,damp,soggy} \mid \text{sunny,sunny,sunny}) +$
 $P(\text{dry,damp,soggy} \mid \text{sunny,sunny,cloudy}) + \dots +$
 $P(\text{dry,damp,soggy} \mid \text{rainy,rainy,rainy})$
- Not practical since the number of paths is $O(N^T)$ where N is the number of hidden states and T is number of observations

Forward Algorithm Intuition

- Idea: compute and cache values $\alpha_t(i)$ representing probability of generating the first t observations, o_1, o_2, \dots, o_t and being in state i at time t
- Each cell expresses the probability
$$\alpha_t(i) = P(o_1, o_2, \dots, o_t, q_t=i)$$
- $q_t = i$ means "the probability that the t^{th} state in the sequence of hidden states is state i "
- Compute α by **summing** over extensions of all paths leading to current cell
- An extension of a path from a state i at time $t-1$ to state j at t is computed by multiplying together:
 - i. previous path probability from the previous cell $\alpha_{t-1}(i)$
 - ii. transition probability a_{ij} from previous state i to current state j
 - iii. observation likelihood b_{jt} that current state j matches observation symbol t

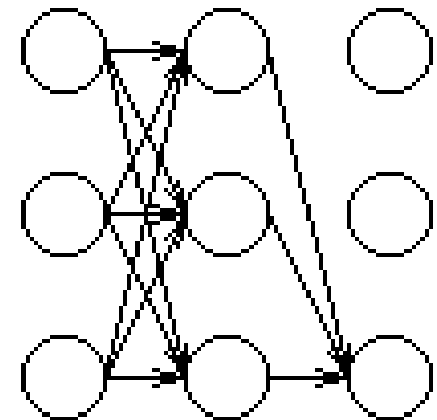
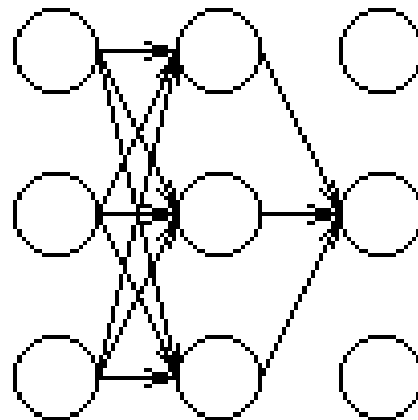
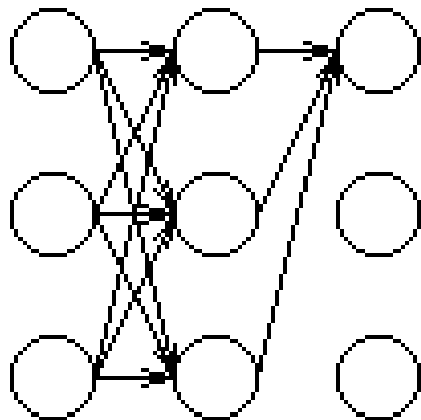
Evaluation using Forward Algorithm

- Example: Given $O = (\text{dry}, \text{damp}, \text{soggy})$, compute $P(O, q_2 = \text{cloudy} \mid \text{HMM } \lambda)$
- $\alpha_2(\text{cloudy}) =$
 $P(O \mid q_2 = \text{cloudy}) * P(\text{all paths to } q_2 = \text{cloudy})$



Forward Algorithm (cont.)

- $P(\mathbf{O}, q_T=s_j | \lambda) = \text{sum of all possible paths through the trellis}$

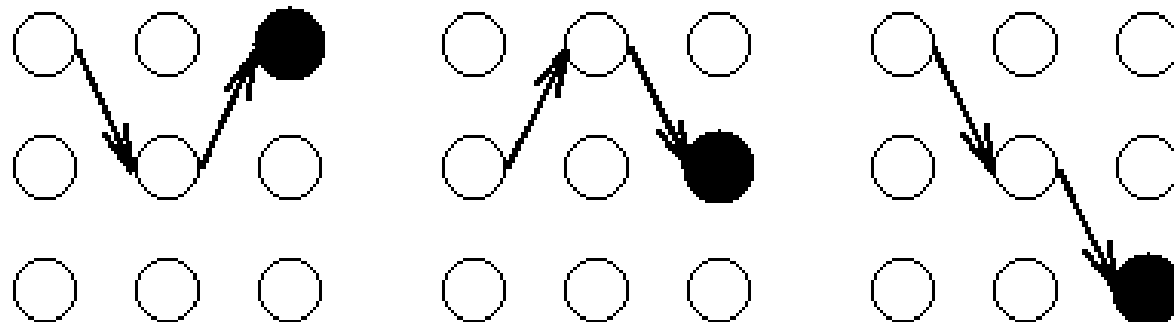


Forward Algorithm (cont.)

- Compute α recursively:
 - $\alpha_1(j) = \pi_j b_j(o_1)$ for all states j
 - $\alpha_t(j) = P(\mathbf{O}, q_t = s_j \mid \lambda)$
 - $= \left[\sum_{i=0}^N \alpha_{t-1}(i) a_{ij} \right] b_j(o_t)$ for $t > 0$
- $P(\mathbf{O} \mid \lambda) = \sum_{j=1}^N \alpha_T(s_j)$
- $O(N^2 T)$ computation time (i.e., linear in T , the length of the sequence)

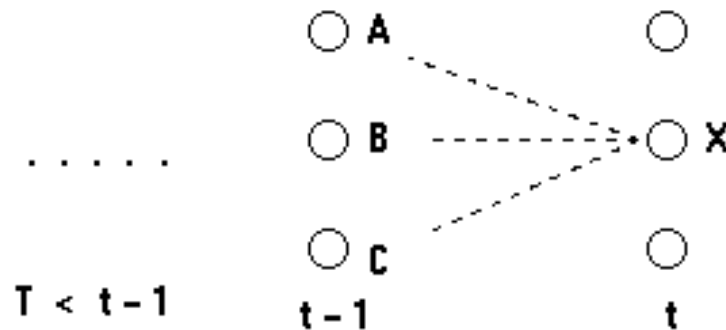
Decoding Problem

- Most probable sequence of hidden states is the state sequence **Q** that **maximizes** $P(\mathbf{O}, \mathbf{Q} | \lambda)$
- Similar to the Forward Algorithm except uses **MAX** instead of SUM, thereby computing the probability of the **most probable path to each state** in the trellis



Decoding using Viterbi Algorithm

- For each state q_i and time t , compute recursively $\delta_t(i) =$ **maximum** probability of all sequences ending at state q_i and time t , and the partial best path to that state
- Assumption: **Dynamic Programming invariant**
 - If ultimate best path for \mathbf{O} includes state q_i , then it includes the best path up to and including q_i



Viterbi Algorithm

- Variant of forward algorithm that considers all words simultaneously and computes most likely *path*
- A type of dynamic programming algorithm
- Input is sequence of observations, and an HMM
- Output is most probable state sequence $q = (q_1, q_2, q_3, q_4, \dots, q_T)$, together with its probability
- Works by computing **max** of previous paths instead of sum
- Looks at the whole sequence before deciding on the best final state and then follows back pointers to recover the best path
- Linear time and linear space algorithm in T

Viterbi Algorithm Intuition

- **Idea: Compute probability by taking the max over extensions of all paths leading to current cell in the trellis**
- **An extension of a path from a state i at time $t-1$ to state j at t is computed by multiplying together:**
 - **previous path probability** from the previous cell, $\delta_{t-1}(i)$
 - **transition probability** a_{ij} from previous state i to current state j
 - **observation likelihood** b_{jt} that current state j matches observation symbol t

Summary of Viterbi Algorithm

- Given an HMM, the Viterbi algorithm finds the most probable sequence of hidden states given a sequence of observed values
- Exploits time invariance of the probabilities to avoid examining every possible path through the trellis
- Looks at the whole sequence before deciding on the best final state and then follows back pointers to recover the best path
- Uses entire context to make its decision and is therefore robust with respect to noise (e.g., a bad observation value in the middle of the sequence)

Summary

- **Speech recognition systems work best if**
 - high SNR (low noise and background sounds)
 - small vocabulary
 - good language model
 - pauses between words
 - trained to a specific speaker
- **Current systems**
 - vocabulary of ~200,000 words for single speaker
 - vocabulary of ~5,000 words for multiple speakers
 - accuracy depends on the task

Error Rates: Machine vs. Human*

Task	Machine	Human
Digits (10)	0.72%	0.009%
Letters (26)	5.0%	1.6%
Transactional speech (1,000)	3.6%	0.1%
Sentences read from <i>WSJ</i> (5,000)	12.8 - 7.2%	1.1 - 0.9%
Telephone speech (14,000)	43.0% (19.3% in 2000)	4.0%

* R. Lippmann, *Speech Comm.* **22**(1), 1997

Are HMMs Useful?

You bet !!

- **Robot planning + sensing when there's uncertainty**
- **Robot learning control**
- **Spam deobfuscation (mis-spelling words)**
- **Speech Recognition / Understanding**
 Phones → Words, Signal → phones
- **Human Genome Project**
 - Complicated stuff your lecturer knows nothing about
- **Consumer decision modeling**
- **Economics & Finance**
- **Plus at least 7 other things**