SPRING 2023

CS 378: INTRO TO SPEECH AND AUDIO PROCESSING

Hidden Markov Models 3

DAVID HARWATH
Assistant Professor, UTCS

The University of Texas at Austin
Department of Computer Science
College of Natural Sciences

Training HMMs on a dataset

- In practice, we don't want to train an HMM on a single observation sequence (too little data)
- We often have a *collection* D of K observation sequences we can use for training:

$$D = \{O^1, \dots, O^K\}$$

- Assuming independence between the sequences, maximum likelihood training becomes:

$$\text{Maximize}_\lambda \prod_{k=1}^{K} P(O^k|\lambda)$$

Training HMMs on a dataset

- It's straightforward to modify Baum-Welch so that we can train on multiple observation sequences.
- E-Step: compute a separate $\gamma_t^k(i)$ and $\tau_t^k(i,j)$ for each observation sequence O^k
- M-Step: Accumulate statistics over *all* $\{O^1, \ldots, O^K\}$

$$\pi_i^* = \frac{\sum_{k=1}^{K} \gamma_1^k(i)}{K} \quad a_{ij}^* = \frac{\sum_{k=1}^{K} \sum_{t=1}^{T^k-1} \tau_t^k(i,j)}{\sum_{k=1}^{K} \sum_{t=1}^{T^k-1} \gamma_t^k(i)} \quad b_i^*(o) = \frac{\sum_{k=1}^{K} \sum_{t=1}^{T^k} \gamma_t^k(i) \mathbf{1}(o = o_t^k)}{\sum_{k=1}^{K} \sum_{t=1}^{T^k} \gamma_t^k(i)}$$

Modification: Viterbi Training

- Instead of computing the full marginal distribution over states $\gamma_t(i)$, use Viterbi to find the single best state sequence for each observation sequence

- Modify the update equations on the previous slide by removing the expectations and just using hard counts instead of soft counts

Today’s agenda

- HMM motivation and intuitive introduction
- HMM mathematical formulation
- HMM algorithms
  - Scoring: Forward-Backward Algorithm
  - Decoding: Viterbi and Forward-Backward Algorithms
  - Training: Baum-Welch Algorithm
- HMMs for phone and word modeling in ASR

Continuous density HMMs

- In ASR, our observations are not symbols, instead they are real-valued vectors
- Typically we use a Gaussian Mixture Model (GMM) instead of a multinomial distribution for $b_i(o)$:

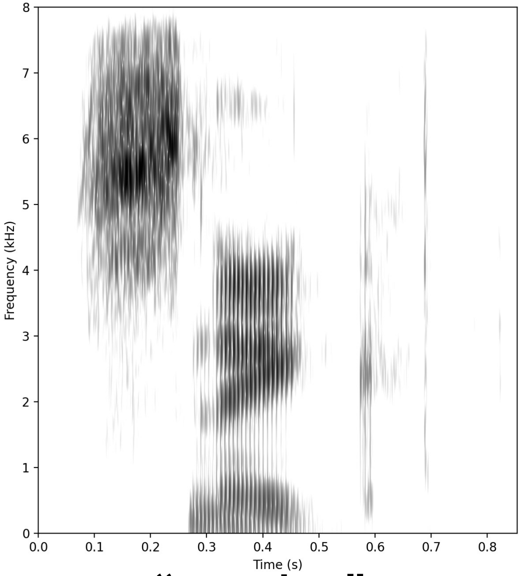
$$b_i(o) = \sum_{m=1}^{M} w_{jm} \mathcal{N}(o; \mu_{jm}, \Sigma_{\mathrm{jm}})$$

- The Baum-Welch update equation for $b_i(o)$ is modified to reflect the ML estimate for the GMM (it gets messy).

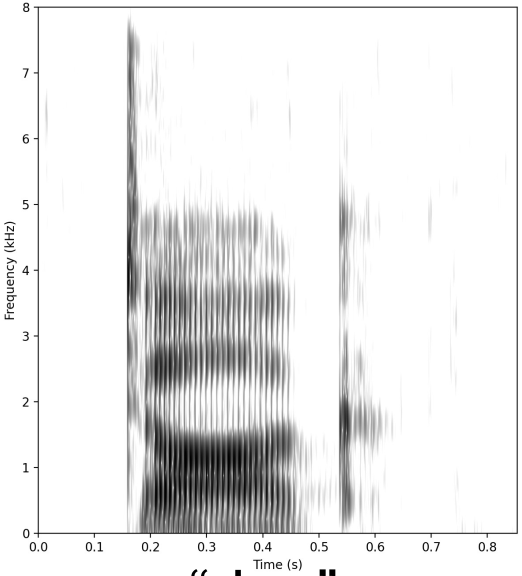
HMMs in an ASR system

How should we design HMMs for ASR systems?

General rule: HMM states should capture stationary observations



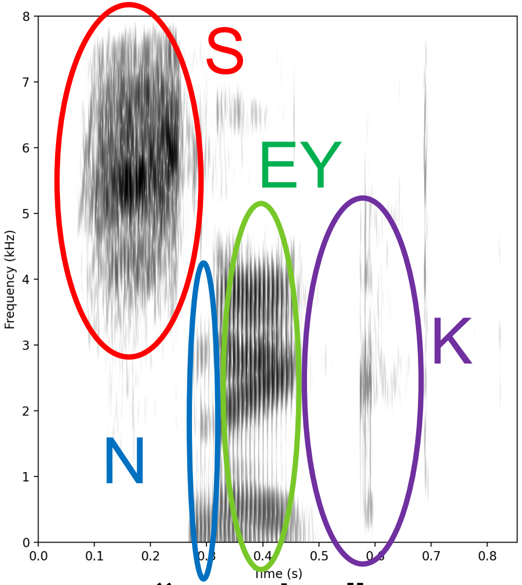
“snake”



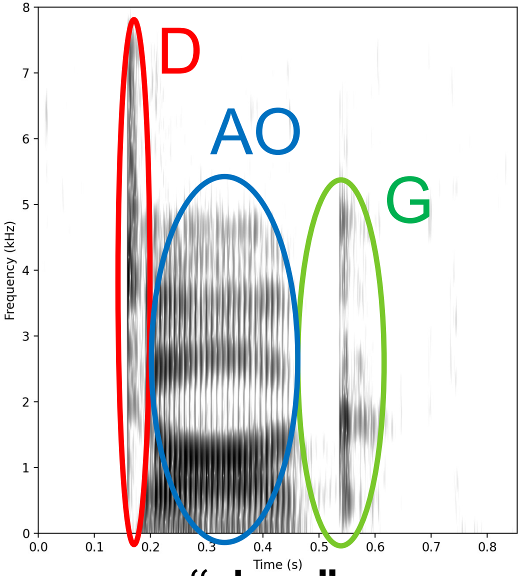
“dog”

Whole-word HMMs for ASR

One idea: use a separate HMM for each word, with one state for each phoneme in the word

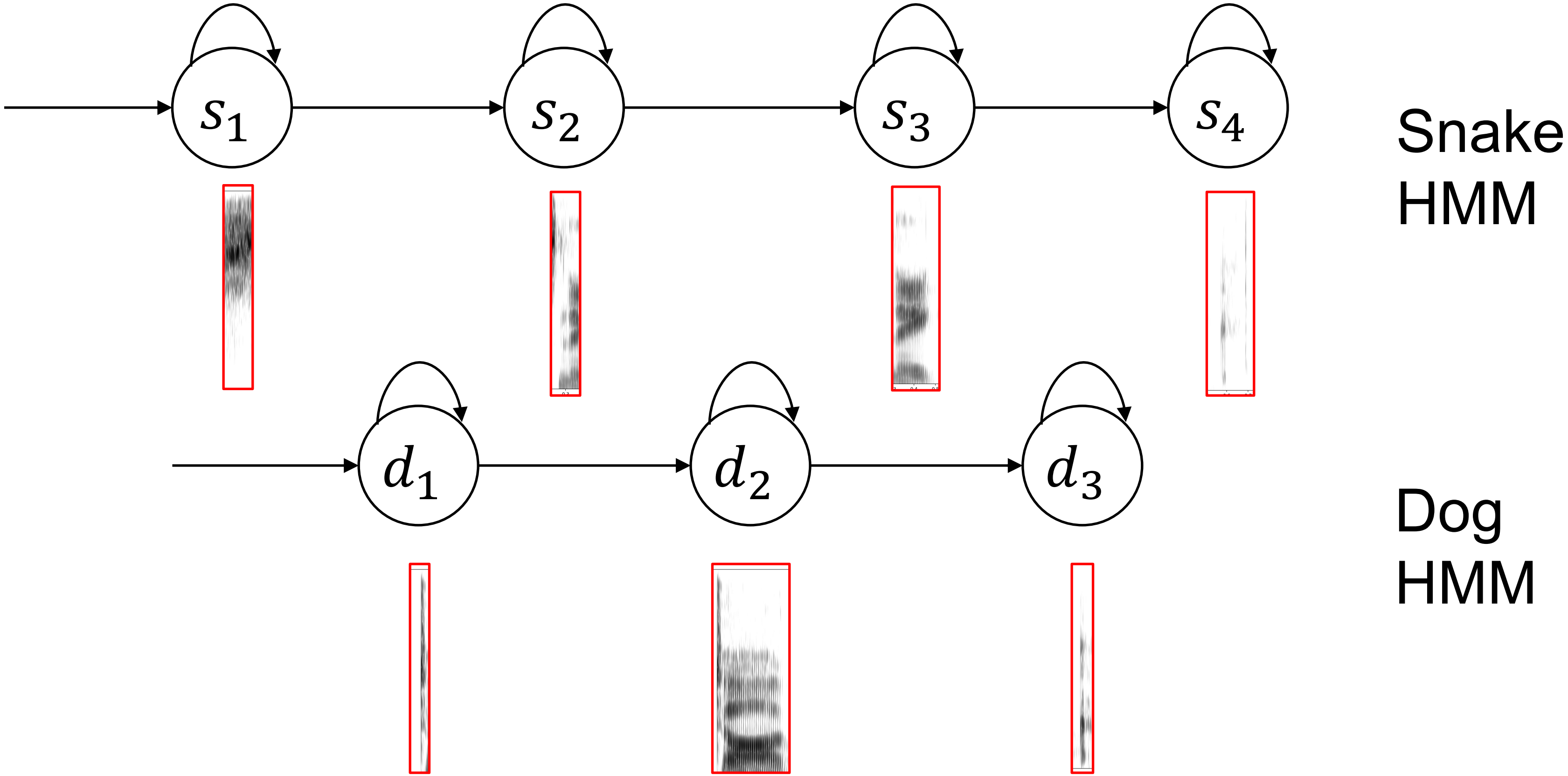


“snake”



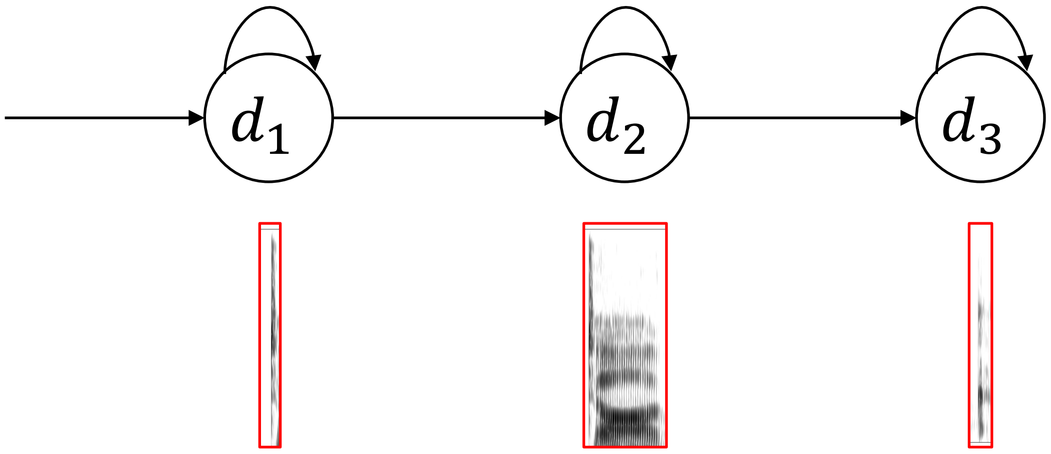
“dog”

Whole-word HMMs for ASR



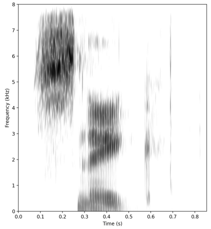
Whole-word HMMs for ASR

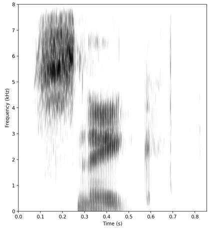
In ASR, we almost always use left-to-right HMMs with self loops, which is how we can model variable-duration speech segments.



Whole-word HMMs for ASR

We can easily build an ASR system to recognize a small number of isolated words this way.

p( | Snake HMM) = likelihood “snake” was spoken

p( | Dog HMM) = likelihood “dog” was spoken

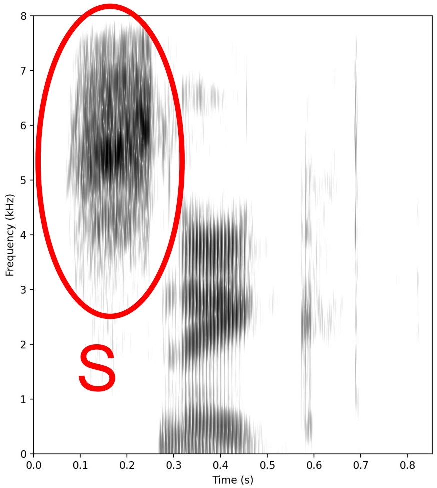
All you need for training is some recordings of “snake” and “dog” being spoken.

How to get to LVCSR?

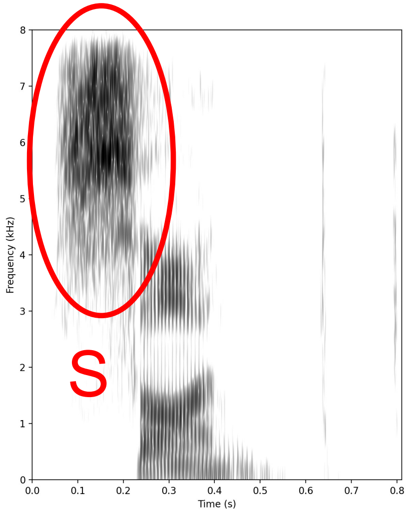
- What we really want in general is Large Vocabulary, Continuous Speech Recognition (LVCSR)
- “Large Vocabulary” is a problem for whole-word HMM modeling because we run into data sparsity problems
- “Continuous” (aka recognizing multi-word utterances) is also a problem if we only have HMMs for individual words

Modeling a large vocabulary

- For modeling a large vocabulary, it is better to use *sub-word* HMMs to take advantage of the fact that sub-word units (e.g. phones, phonemes, syllables…) are shared across many different words

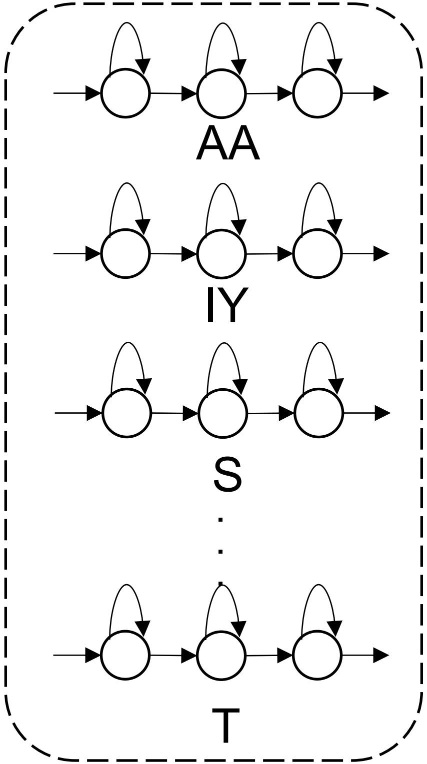


"snake"

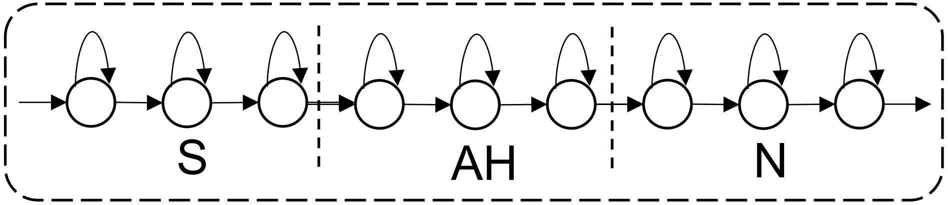


"sun"

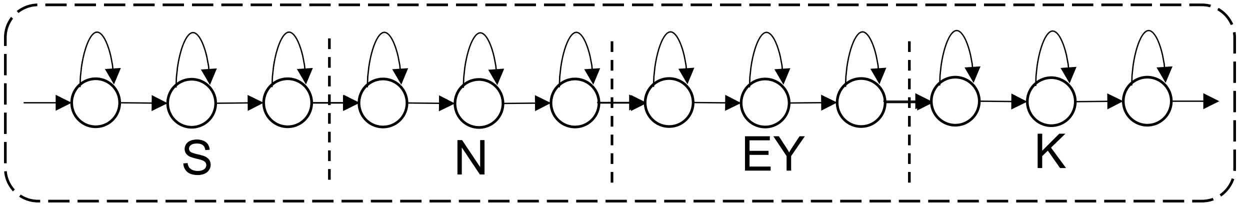
Sub-word unit modeling



Sub-word HMM Inventory



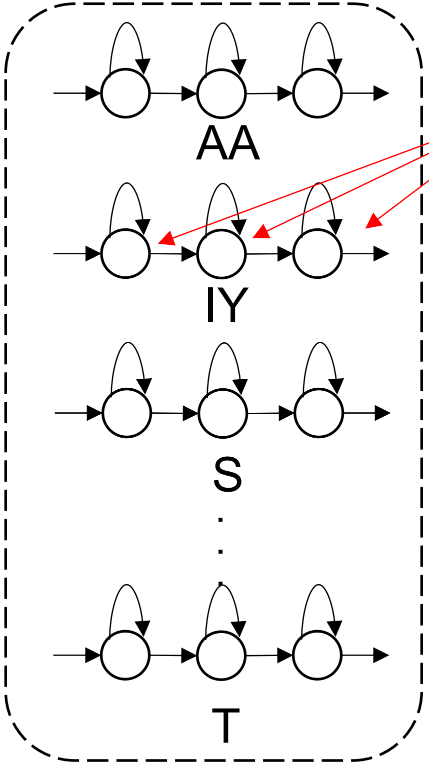
"sun"



"snake"

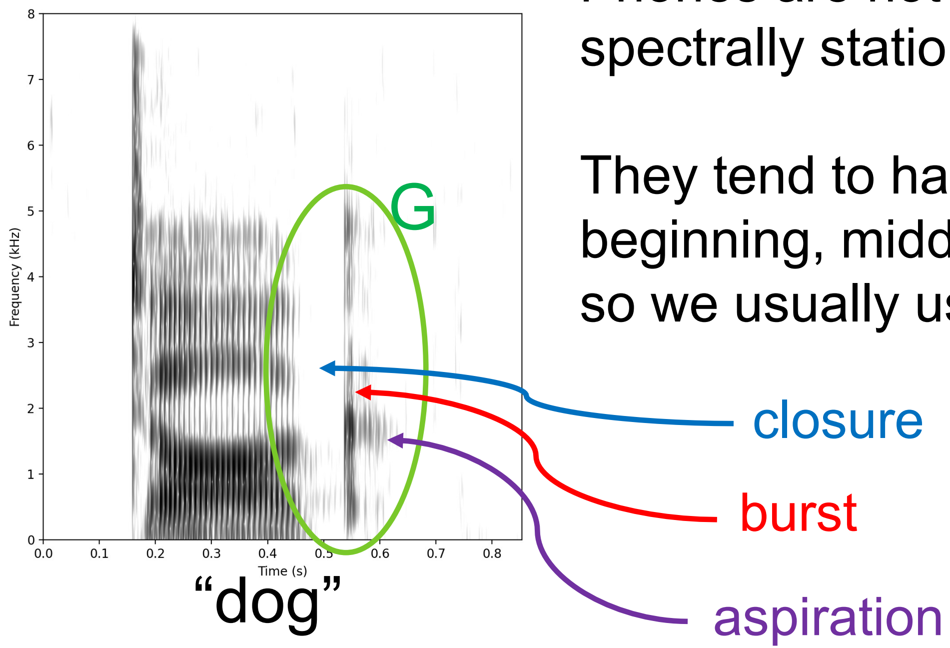
Word HMM Inventory

Sub-word unit modeling



Sub-word HMM Inventory

Why multiple states for each sub-word unit?

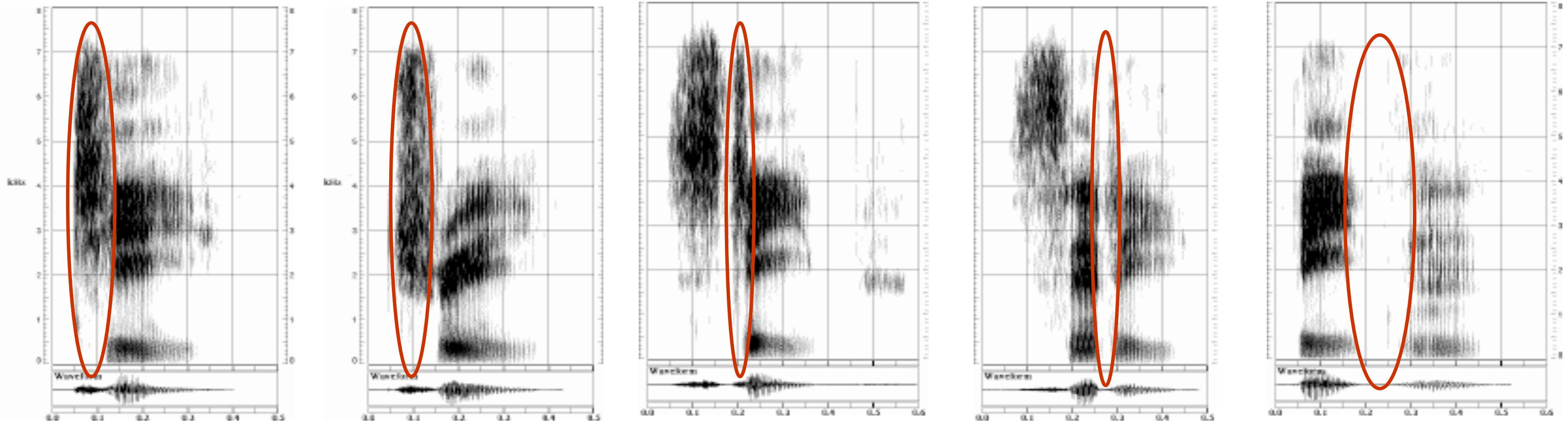


"dog"

Phones are not always spectrally stationary.

They tend to have a beginning, middle, and end, so we usually use 3 states.

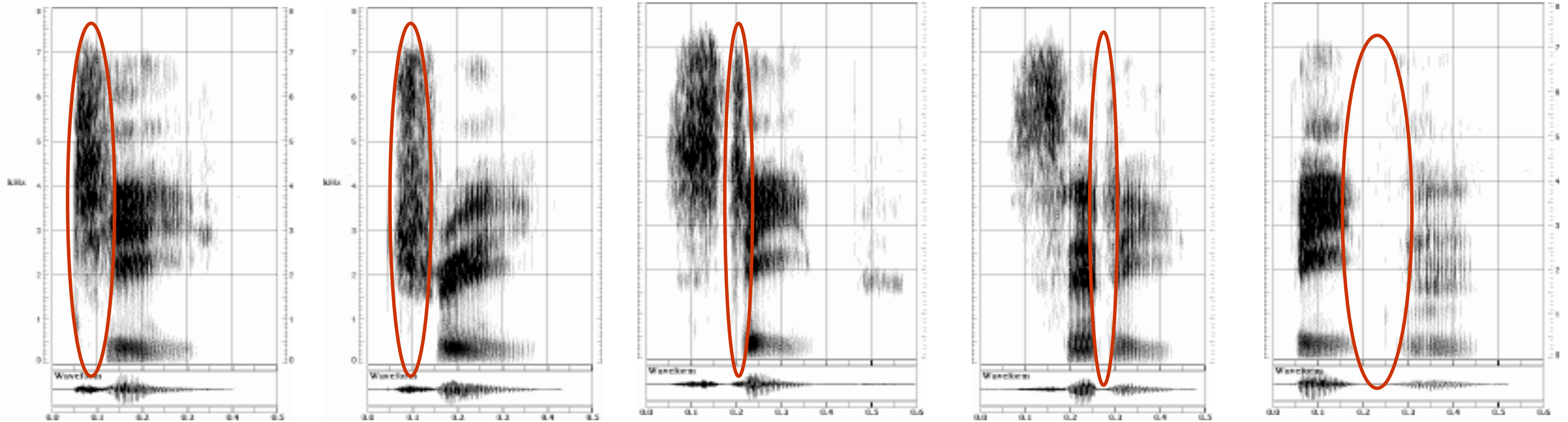
Problem: phonological variation



TEA TREE STEEP CITY BEATEN

Adapted from *James Glass, and Victor Zue. 6.345 Automatic Speech Recognition. Spring 2003. Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu. License: [Creative Commons BY-NC-SA](https://creativecommons.org/licenses/by-nc-sa/4.0/).*

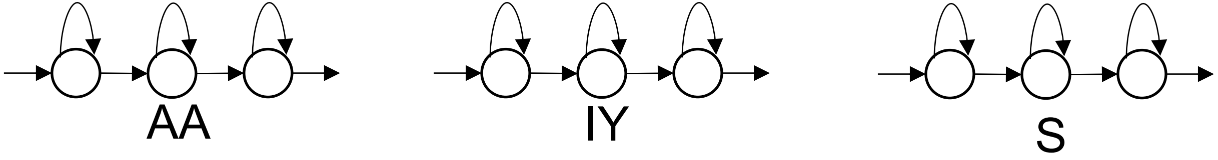
Solution: context dependent models



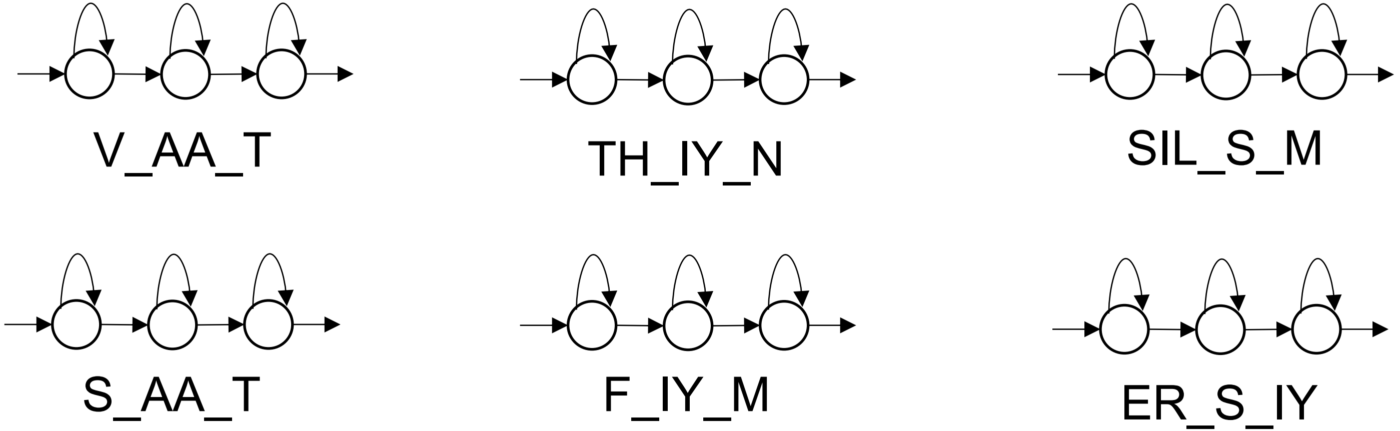
TEA	TREE	STEEP	CITY	BEATEN
SIL_T_IY	SIL_T_R	S_T_IY	IH_T_IY	IY_T_N

Context Dependent Phonetic HMMs

Context Independent (CI) Models



Context Dependent (CD) Models (triphone)



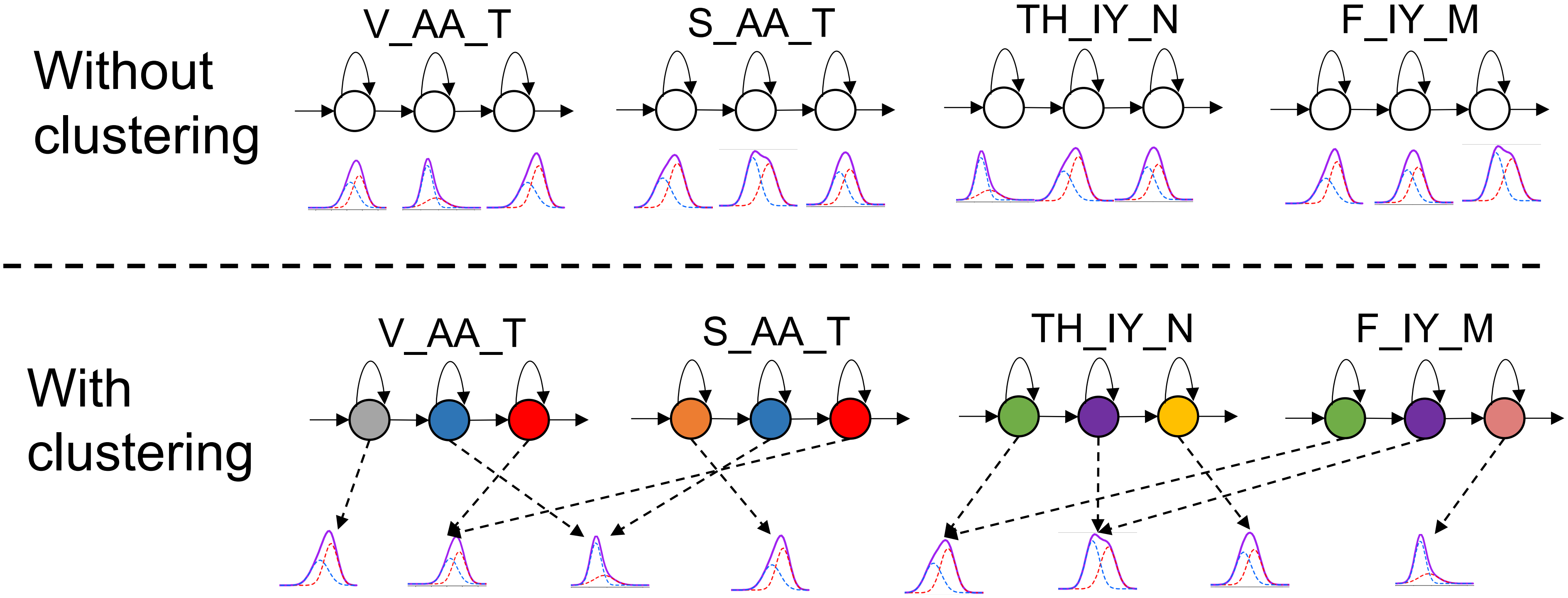
Problem: too many models!

- How many sub-word HMMs do we need for various degrees of contextual modeling?
- American English has approximately 40 phones
  - # Monophone (context independent) models = 40
  - # Triphone models = $40^3 = 64{,}000$
  - # Quinphone models = $40^5 = 102{,}400{,}000$
- Don’t forget that each HMM has 3 states, each with its own GMM that has multiple mixture components!

Triphone Model Parameters

- Triphone system: $40^3 = 64{,}000$ HMMs
- 3 states per HMM → $3 * 64{,}000 = 192{,}000$ GMMs
- 10 Gaussians per GMM → $10 * 192{,}000 = 1{,}920{,}000$ Gaussians
- 39-dim MFCCs and diagonal covariance Gaussians → $2 * 39 + 1 = 79$ parameters per Gaussian
- $79 * 1{,}920{,}000 = 151{,}680{,}000$ total GMM parameters! (Need lots of data to train a model this big)
- Worse yet, the data won't be spread ”evenly” across the individual GMMs due to the “long tail” of infrequent triphones.

Solution: state clustering



Phonetic State Clustering

- Basic idea: Multiple states across different triphones are allowed to share the same GMM

- Allows for shared training data among similar acoustic states as well as far fewer overall model parameters

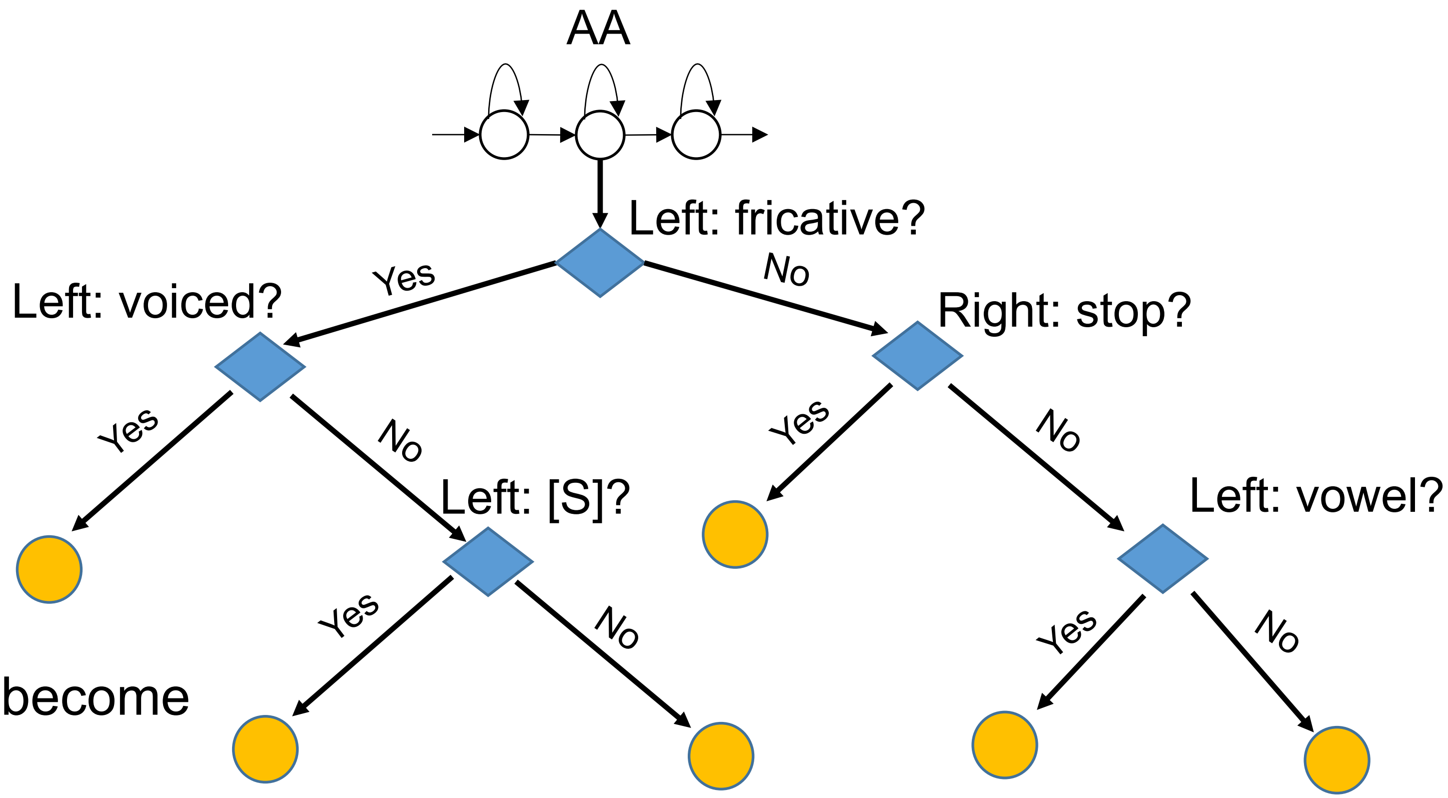
- Main problem we need to solve: how to cluster states?

Phonetic State Clustering

- One obvious strategy: train all 40k triphone models, then cluster their states bottom-up to get better models
  - Doesn’t work well; we may not even observe many triphones during training

- Better approach: Use top-down strategy to gradually break the states of context independent models into more specific states

Phonetic Decision Trees



Leaf nodes become GMMs

Phonetic Decision Trees

- Decision tree splits each CI state into multiple CD states
- Leaf nodes determine the final set of GMMs
- Each decision node is a Y/N question of the form “Is [left,right] context phone [X]?
- X can be manner class (vowel vs. fricative), distinctive feature (voiced vs. unvoiced), place of articulation (alveolar vs. dental), or an individual phone

Choosing Decision Tree Questions

To build the tree: start with large list of candidate questions, then choose the next question to add based upon whichever one maximizes log-likelihood gain

$$L(S_1) + L(S_2) - L(S)$$

Where

$$L(S) = \sum_{s \in S} \sum_{x \in X} \log p(x | \mu_s, \Sigma_s) \gamma_s(x)$$

States in cluster S

Cluster Gaussian

State occupancy

Building the Tree

- When building the decision tree, initially use *only 1 Gaussian per state*
  - Makes it easy to compute $L(S)$ at each split step
- Keep splitting states until likelihood gain becomes small, or cluster occupancy counts hit a lower threshold
- Once we've built the tree, turn each leaf node into a full GMM by adding more Gaussians and doing E-M training

State GMM Training Details

- To train a K-component GMM, we don't simply use K Gaussians right off the bat
  - Start with 1 Gaussian, do several E-M iterations, then split the Gaussian into two and perturb the means, continue training
  - Continue "Mixing up" e.g. $1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \ldots$ components until we hit K
  - Decide which Gaussians to split based on highest occupancy
- Usually have to use a variance floor to prevent Gaussians from having 0 variance (when occupancy=1)

Phone recognition on TIMIT

- TIMIT: very small (4 hours) phonetically balanced dataset, manually transcribed at phone level
- Kaldi HMM-GMM system
- Monophone PER: 31.7%
- Triphone WER: 25.1%