Training Tied-State Models

Rita Singh and Bhiksha Raj

Recap and Lookahead

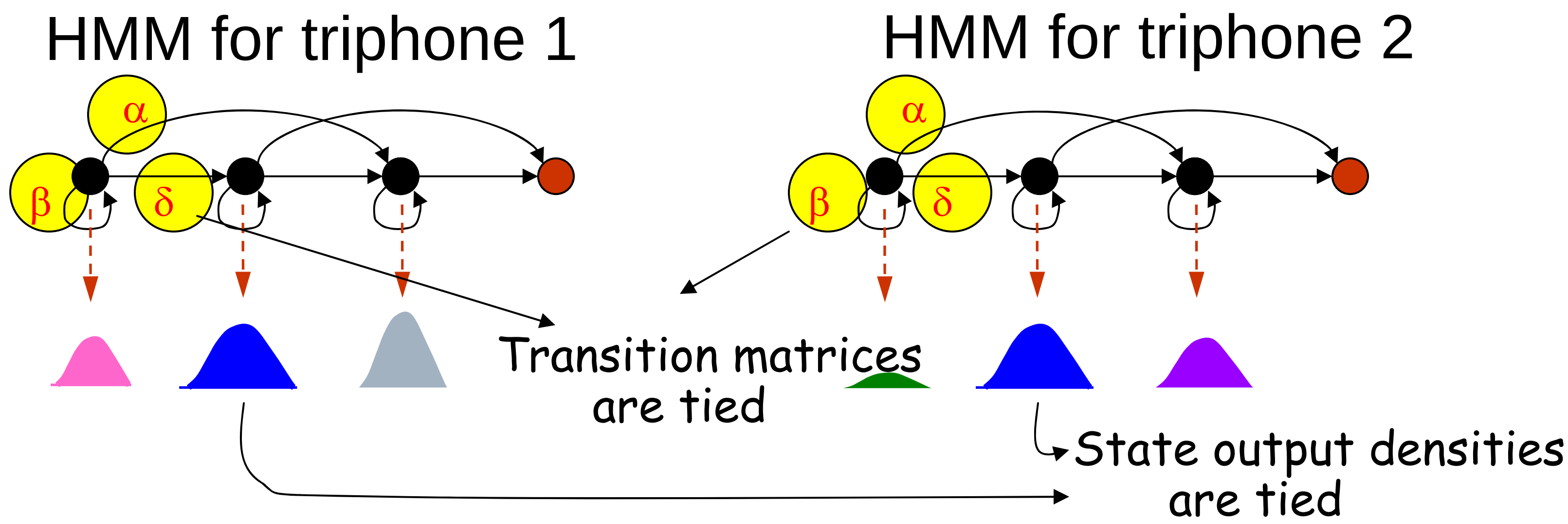
- Covered so far:
  - String Matching based Recognition
  - Introduction to HMMs
  - Recognizing Isolated Words
  - Learning word models from continuous recordings
  - Building word models from phoneme models
  - Context-independent and context-dependent models
  - Building decision trees

  - Exercise: Training phoneme models
  - Exercise: Training context-dependent models
  - Exercise: Building decision trees
- Training tied-state acoustic models

Training Acoustic Models

- The goal of training is to train HMMs for all sound units
  - Models for triphones to represent spoken sounds
  - Models for other types of sounds
- What we really train is an *acoustic model*
- An acoustic model is a collection of *component parts* from which we can compose models that we require
- What follows:
  - Modelling spoken sounds: How triphone models are built
    - Including a quick recap of parameter sharing and state tying
  - Issues relating to triphone models
  - Modelling non-speech sounds
  - Forced alignment
  - And an exercise

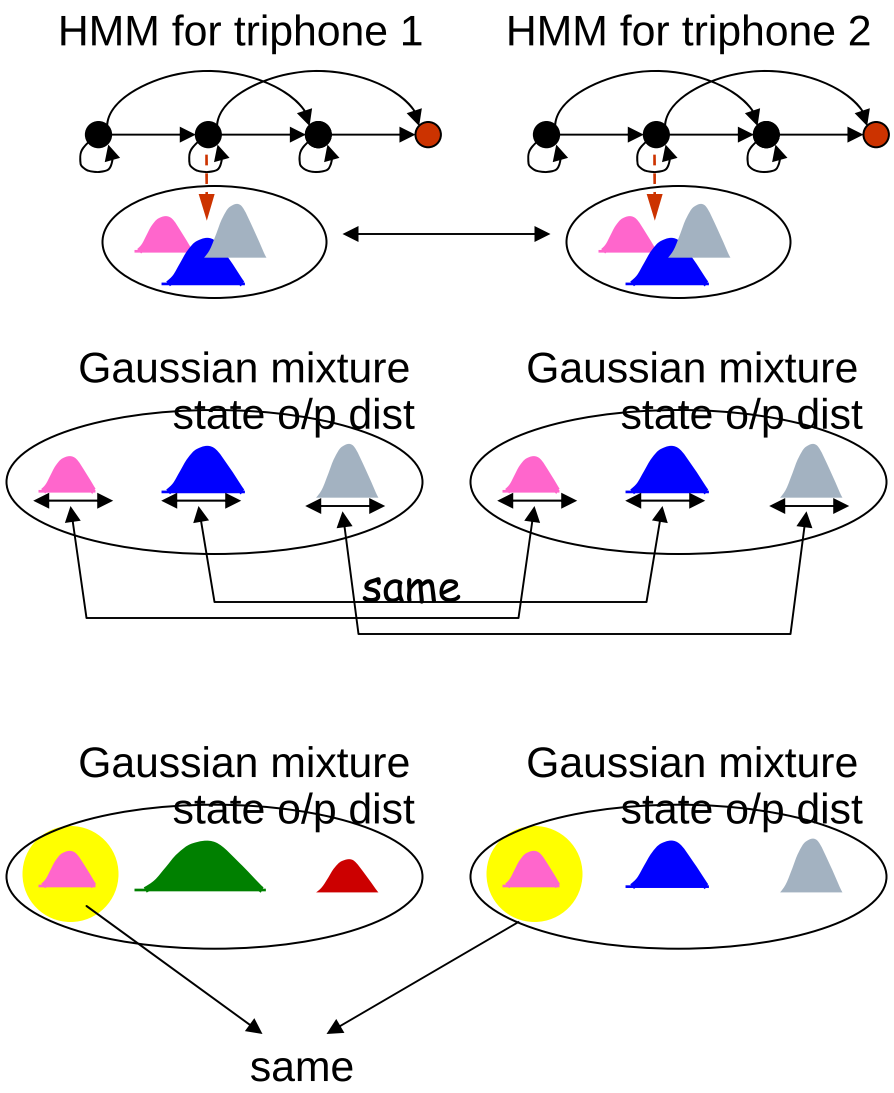
Recap: What is Parameter Tying



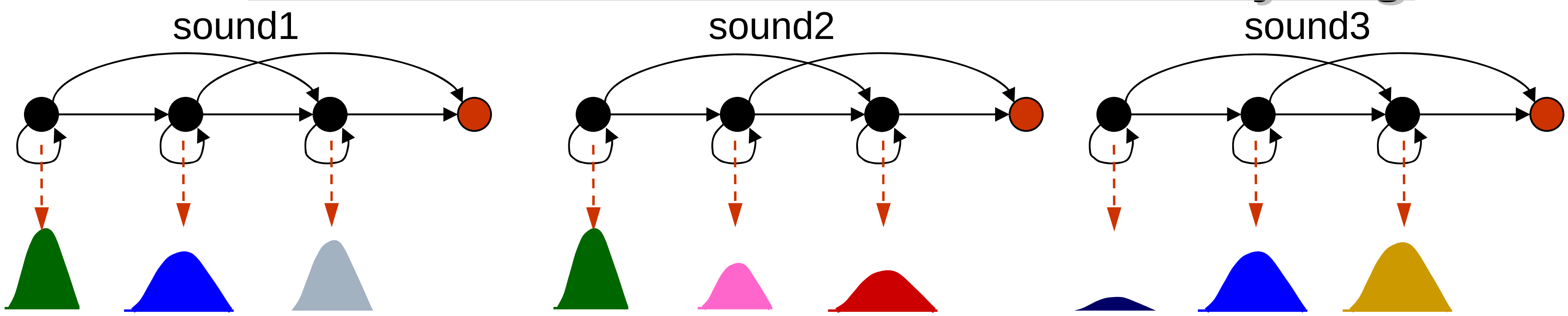
- HMMs have many parameters
  - Transition matrices
  - HMM state-output distribution parameters
- A parameter is said to be tied in the HMMs of two sound units if it is identical for both of them
  - E.g. if transition probabilities are assumed to be identical for both, the transition probabilities for both are "tied"
  - **Tying affects training**
  - **The data from both sounds are pooled for computing the tied parameters**

More on Parameter Tying

- Parameter tying can occur at any level
- Entire state output distributions for two units may be tied
- Only the *variances* of the Gaussians for the two may be tied
  - Means stay different
- Individual Gaussians in state output distributions may be tied
- Etc.



Still more on Parameter Tying

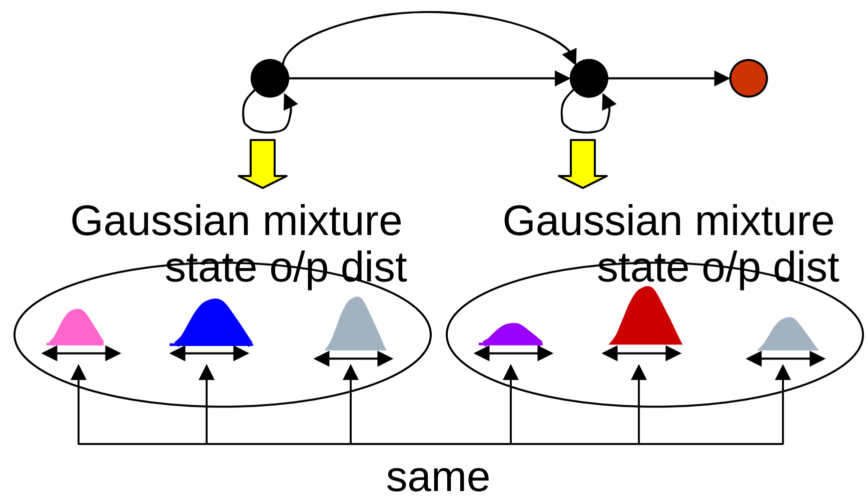


- Parameter tying may be different for different components
  - E.g. the state output distributions for the first state of HMMs for sound1 and sound2 are tied
  - But the state output distribution of the *second* state of the HMMs for sound1 and *sound3* are tied
- This too affects the training accordingly
  - Data from the first states of sound1 and sound2 are pooled to compute state output distributions
  - Data from the second states of sound1 and sound3 are pooled

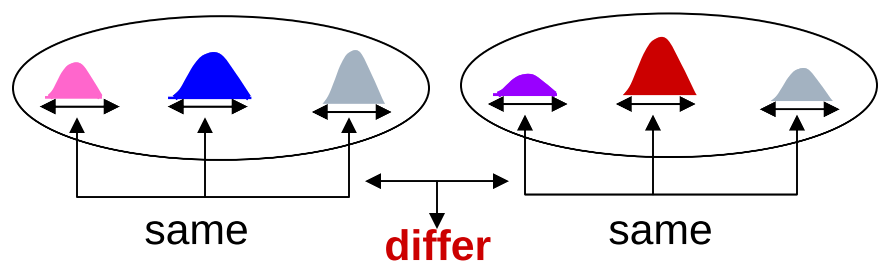
And yet more on parameter tying

- Parameters may even be tied *within* a single HMM

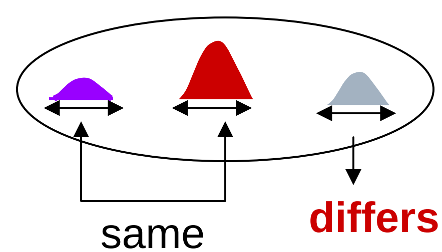
- E.g. the variances of all Gaussians in the state output distributions of all states may be tied



- The variances of all Gaussians within a state may be tied
  - But different states have different variances

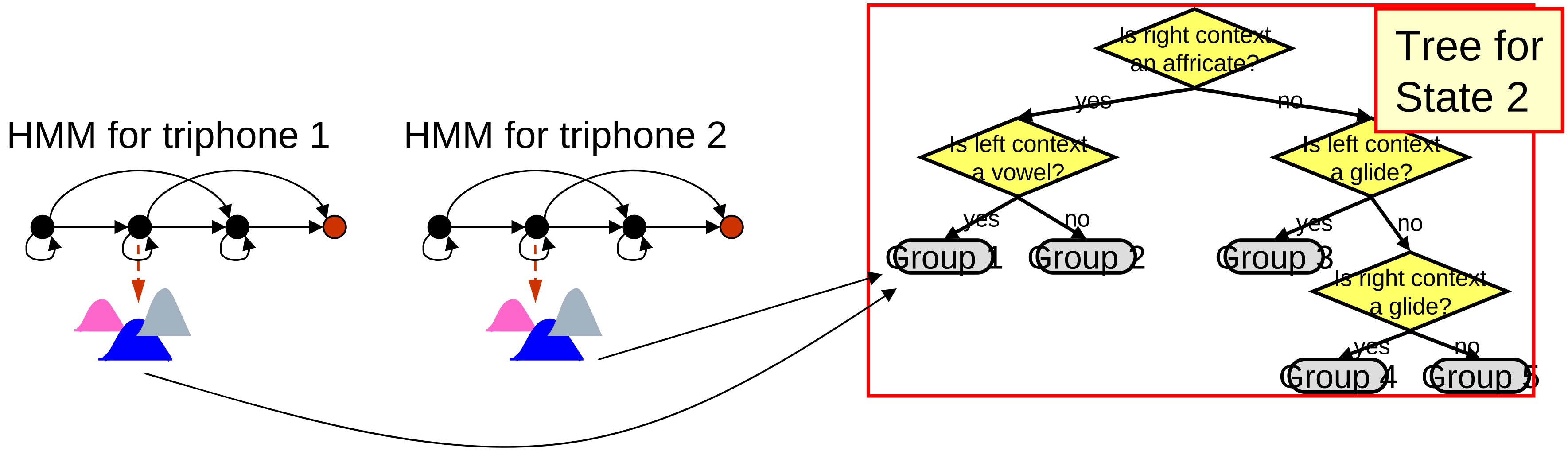


- The variances of *some* Gaussians within a state may be tied



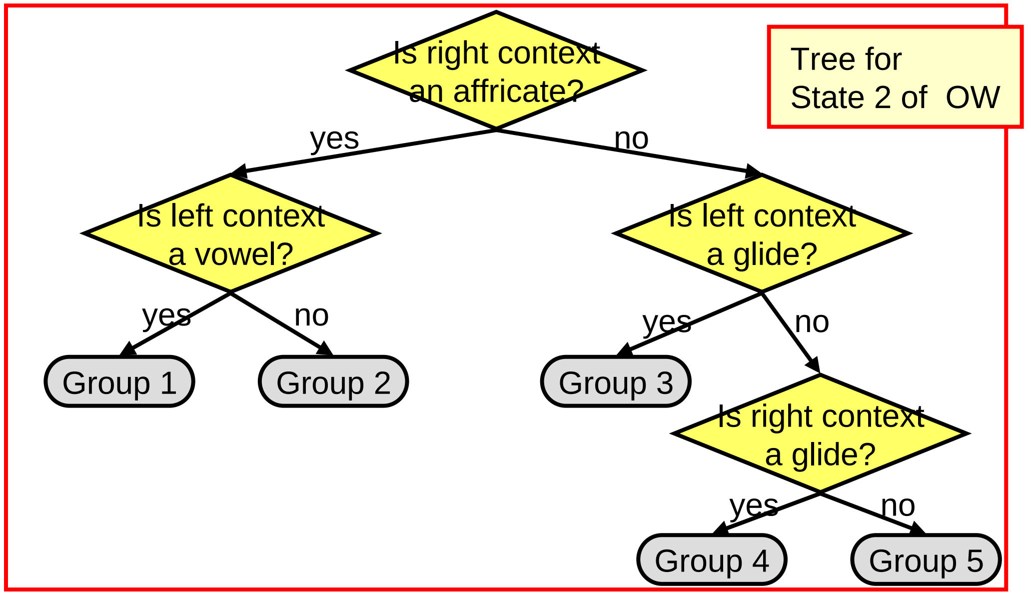
- All of these are not unusual.

State Tying



- State-tying is a form of parameter sharing where the state output distributions of different HMMs are the same
- All state-of-art speech recognition systems employ state-tying at some level or the other
- The most common technique uses decision trees

Decision Trees



- Decision trees categorize triphone states into a tree based on linguistic questions
  - Optimal questions at any level of the tree are determined from data
  - All triphones that end up at the same leaf of the tree for a particular state have their states tied
  - Decision trees are phoneme and state specific

Building Decision Trees

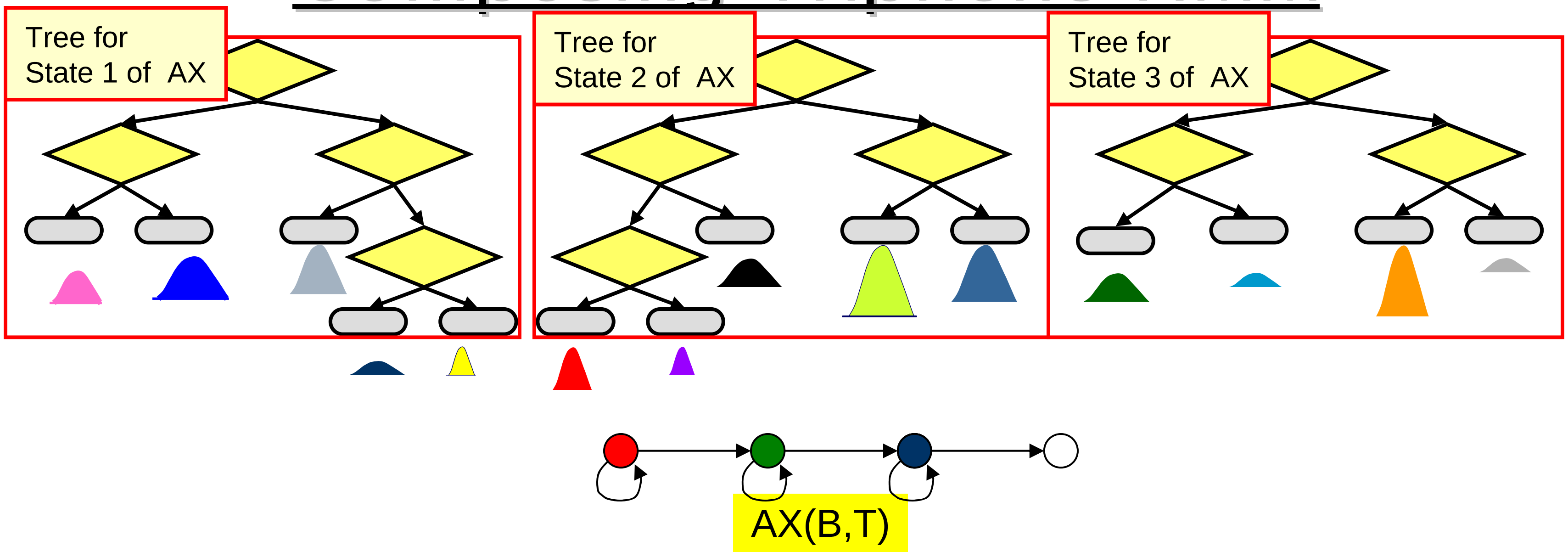
- For each phoneme (AX, AH, AY, ... IY,.., S, .. ZH)
- For each state (0,1,2..)
  - Gather all the data for that state for all triphones of that phoneme together
  - Build a decision tree
  - The distributions of that state for each of the triphones will be tied according to the composed decision tree

- Assumption: All triphones of a phoneme have the same number of states and topology
- If the HMM for all triphones of a phoneme has K states, we have K decision trees for the phoneme
- For N phonemes, we will learn N*K decision trees

The Triphone Models

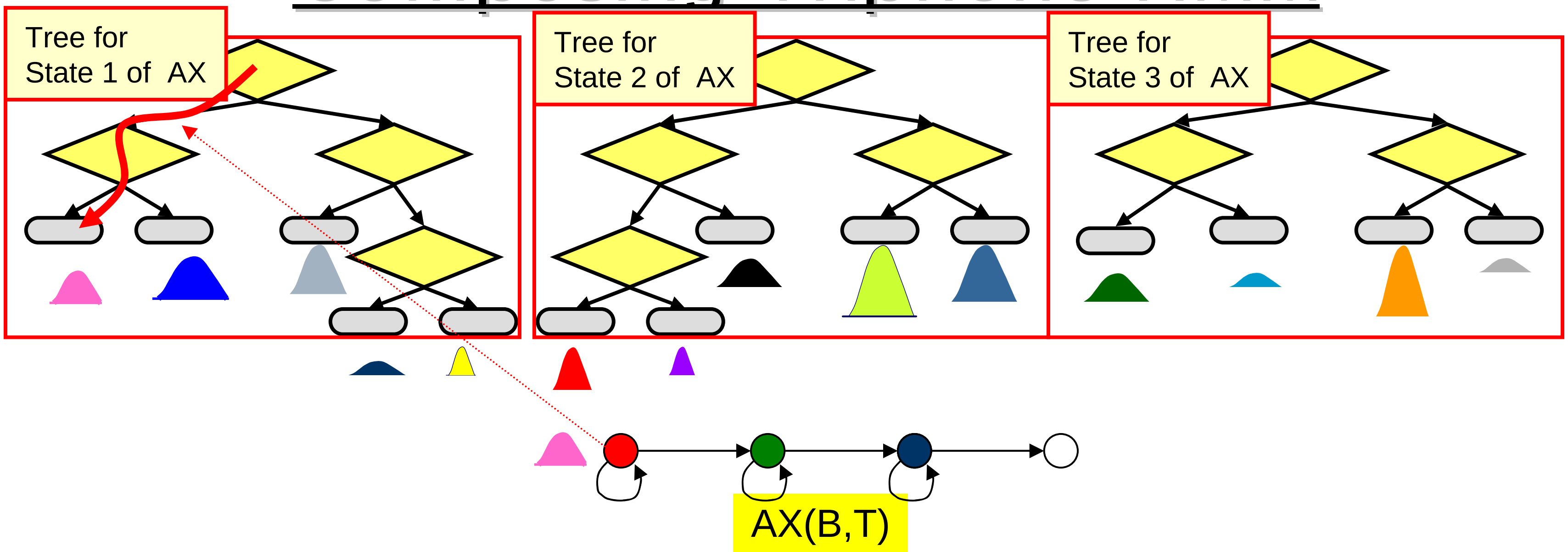
- We never actually train triphone HMMs!
- We only learn all constituent parts
  - The transition matrix, which is common to all triphones of a phoneme
  - The distributions for all tied states
- Triphones models are *composed* as necessary
  - If a specific triphone is required for a word or word sequence, we identify the necessary tied states
    - Either directly from the decision trees or a pre-computed lookup table
  - We identify the necessary transition matrix
  - We combine the two to compose the triphone HMM
- Triphone HMMs by themselves are not explicitly stored

Composing Triphone HMM



- Select all decision trees associated with the primary phoneme for the triphone
  - E.g. for AX (B, T), select decision trees for AX
  - There will be one decision tree for each state of the triphone
  - Each leaf represents a tied state and is associated with the corresponding state output distribution

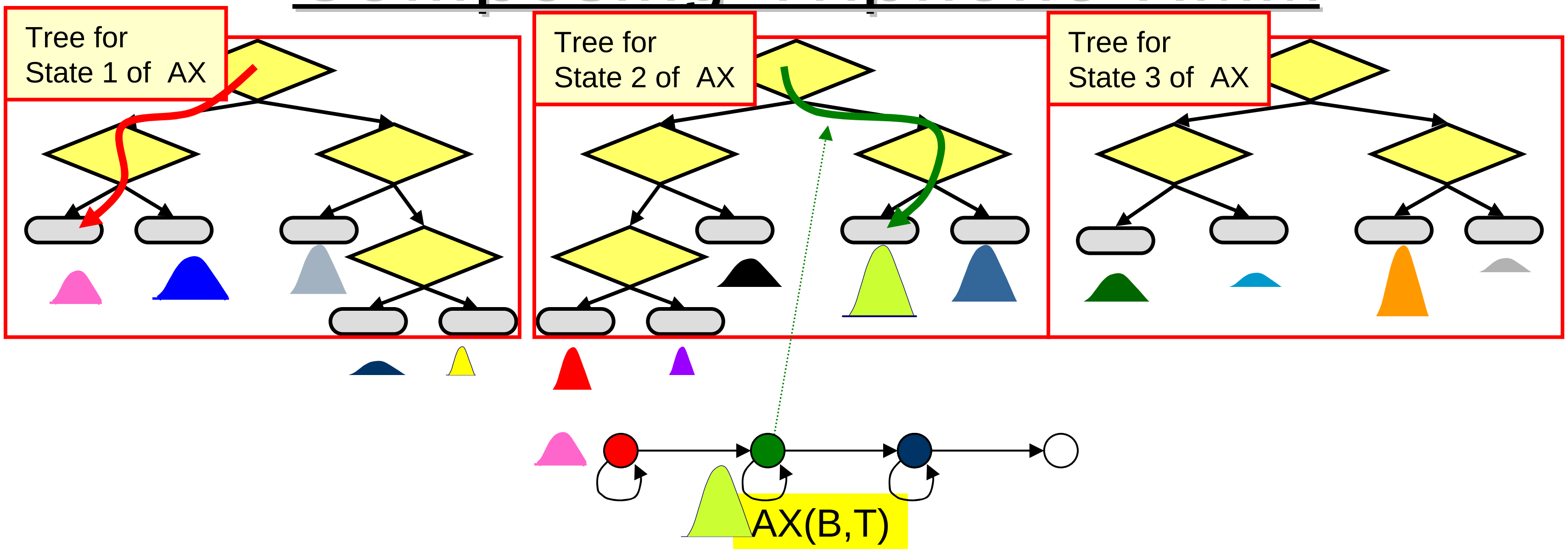
Composing Triphone HMM



- Pass each state of the triphone down its corresponding tree
- Select the state output distribution associated with the leaf it ends up at

- Finally select the transition matrix of the underlying base (context independent) phoneme
  - E.g. AX(B,T) uses the transition matrix of AX

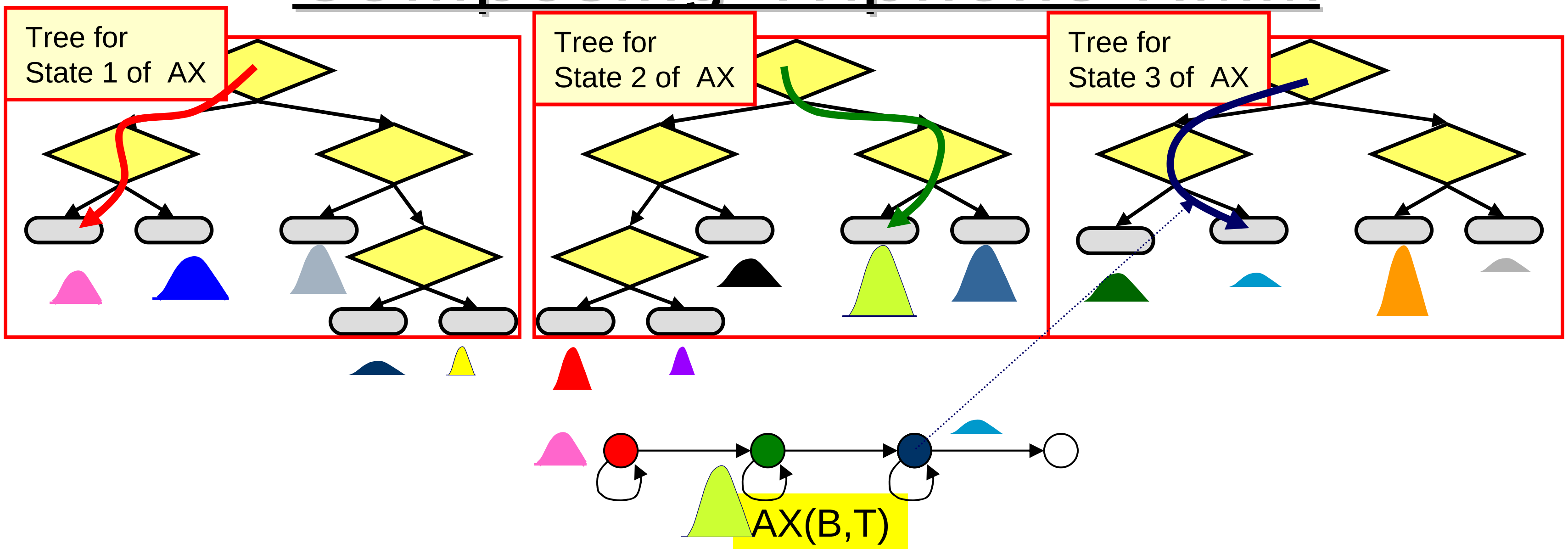
Composing Triphone HMM



- Pass each state of the triphone down its corresponding tree
- Select the state output distribution associated with the leaf it ends up at

- Finally select the transition matrix of the underlying base (context independent) phoneme
  - E.g. AX(B,T) uses the transition matrix of AX

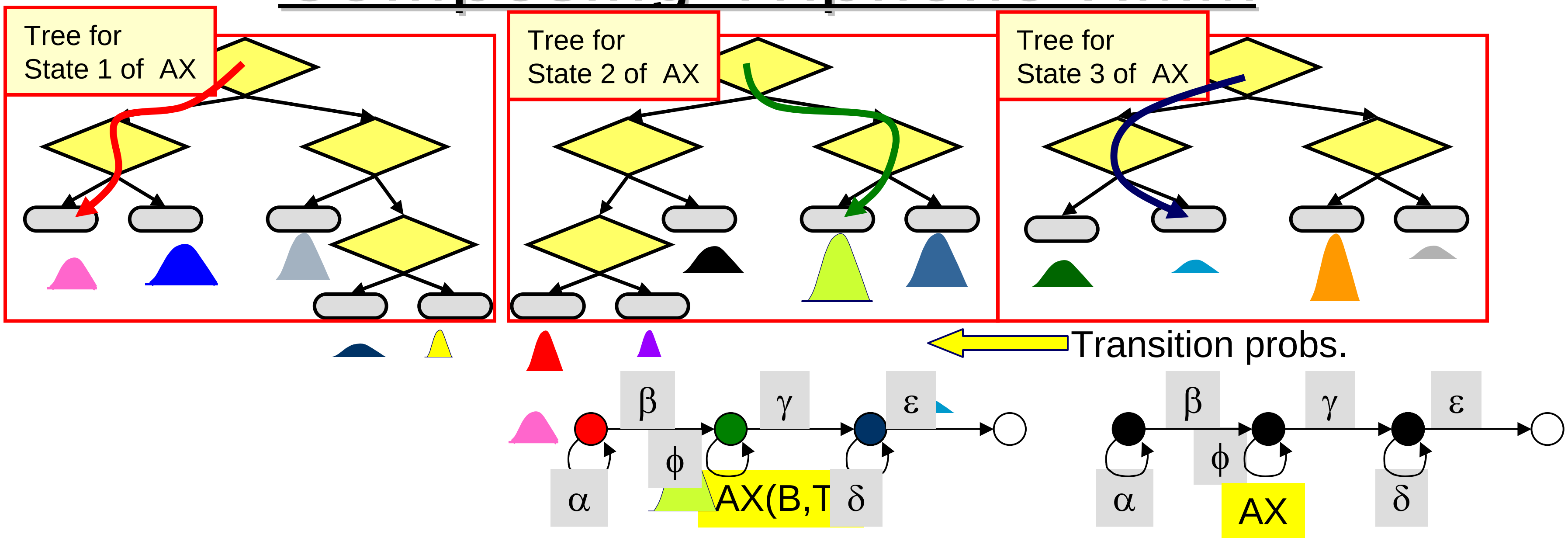
Composing Triphone HMM



- Pass each state of the triphone down its corresponding tree
- Select the state output distribution associated with the leaf it ends up at

- Finally select the transition matrix of the underlying base (context independent) phoneme
  - E.g. AX(B,T) uses the transition matrix of AX

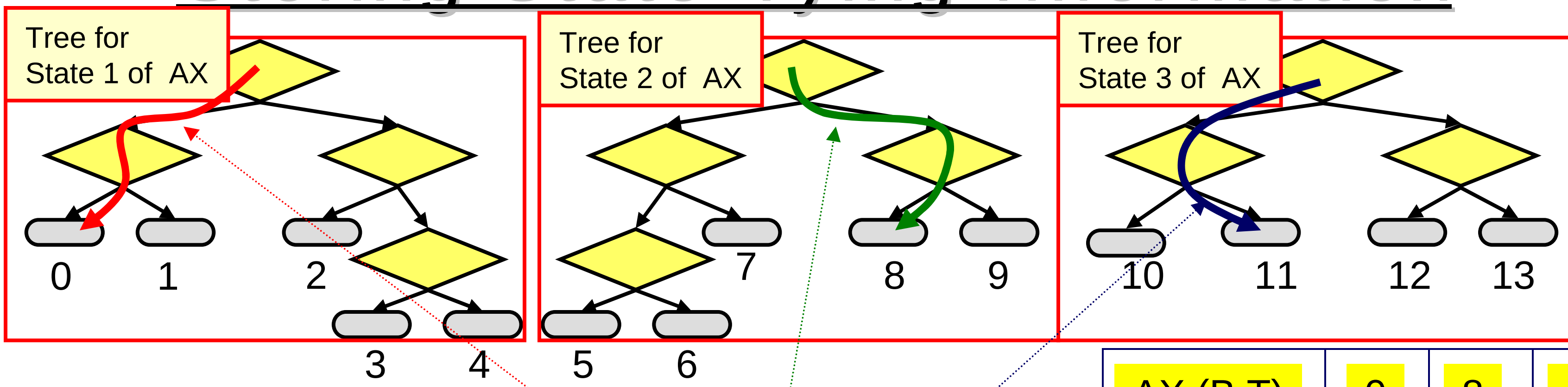
Composing Triphone HMM

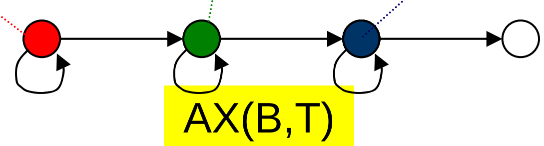


- Pass each state of the triphone down its corresponding tree
- Select the state output distribution associated with the leaf it ends up at

- Finally select the transition matrix of the underlying base (context independent) phoneme
  - E.g. AX(B,T) uses the transition matrix of AX

Storing State-Tying Information





AX (B,T)	0	8	11
AX (B,D)	0	9	13
Etc…			

- It is not necessary to identify the correct tied state using decision trees every time
- Decision tree leaves can be indexed
- The index of the leaves for each state of a triphone can be precomputed and stored in a table
- In the sphinx this table is called the “Model Definition File” (Mdef)
- The state output distribution to use with any state is identified by the index

How many tied states

- The total number of tied states is fixed
- I.e the total number of leaves on all the decision trees must be prespecified
- Tradeoff: More tied states result in better models
  - But only if they all have sufficient data
- The actual no. of tied states depends on the amount of training data
  - ~100 hours: 4000, ~2000 hours: 10000-20000
- Definition: the “tied” state output distributions are referred to as “senones” in the sphinx
- There are as many senones as the total no. of leaves in all pruned decision trees

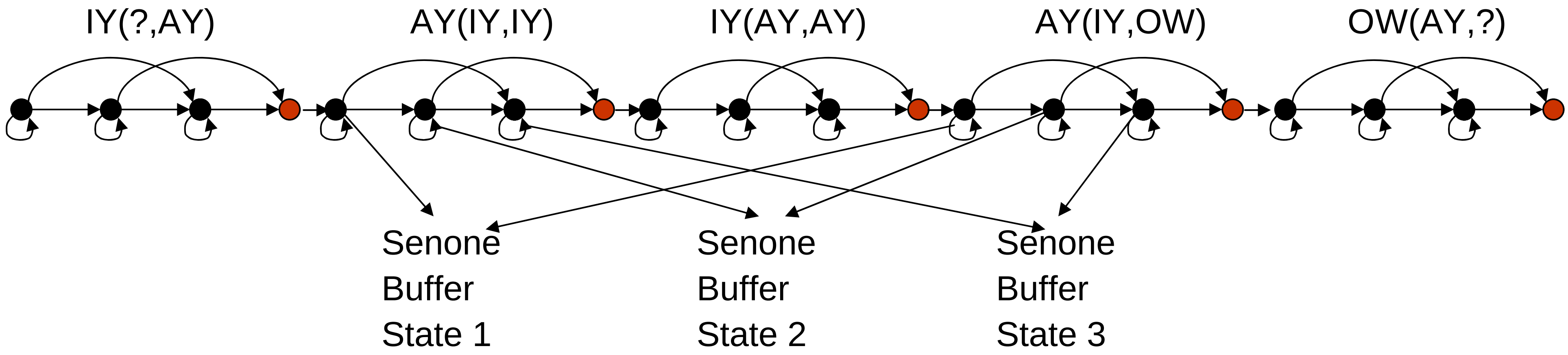
How many Gaussians per tied state

- The number of Gaussians per tied state also depends on the amount of training data

- More Gaussians is better, but only if we have enough data.
  - 200 hours of training: 4000-6000 tied states with 16-32 Gaussians/state
  - 2000 hours: 10000-20000 tied states with 32-64 Gaussians per state

- More Gaussians or more tied states?
  - Both increasing the number of Gaussians and increasing the no. of tied states needs more data
  - Tradeoff: for a given amount of data we could have either more Gaussians or more tied states
  - Having fewer tied states and more Gaussians per tied state has its advantages

How training happens



HMM for EIEIO IY AY IY AY O

Assuming triphones AY(IY,IY) and AY(IY,OW) have common tied states for all states

- When training models, we directly compute tied-state (senone) distributions from the data
  - Senone distributions and context-independent phoneme transition matrices are used to compose the HMM for the utterance
  - Contributions of data from the HMM states go directly to updating senone distributions without referring to an intermediate triphone model

Overall Process of Training Senone Models

- The overall process is required to go through a sequence of steps:

1. Train CI models
2. Train "untied" CD models
   - Initialized by CI models
3. Train and prune decision trees
   - Build State-tying Tables
4. Train Senone Distributions
   - Initialized by the corresponding state output distributions of the CI phoneme

Initialization and Gaussian Splitting

- All senone distributions begin as Gaussians
  - These are initialized with the Gaussian state output distributions of the corresponding state of the corresponding phoneme
  - E.g. The distributions of all tied states from the decision tree for the first state of “AA” are initialized with the distribution of the first state of “AA”

- Training is performed over all training data until all senone distributions (and transition matrices) have convereged

- If the senone distributions do not have the desired number of Gaussians yet, split one or more Gaussian and return to previous step
  - At splitting we are effectively re-initializing the training for models with N+K Gaussians
    - N = no. of Gaussians in senone distribution before splitting; K = no. of Gaussians split

Not all models share states

- All triphones utilize tied-state distributions
- The trainer also simultaneously trains context-independent phonemes
  - These do not use tied-states – each state of each phoneme has its own unique distribution
- The speech recognizer also includes models for silences and other noises that may be transcribed in the training data
- The spectra for these do not vary with context
  - Silence looks like silence regardless of what precedes or follows it
- For these sounds, only context-independent models are trained
  - States are not tied

Silence as a context

- Although silence itself does not change with the adjacent sounds (i.e. it is not "context-dependent"), it can *affect* adjacent sounds
- A phoneme that begins after a silence has initial spectral trajectories that are different from trajectories observed in other contexts
- As a result silences form valid triphonetic contexts
  - E.g. Triphones such as DH(SIL, AX) are distinctly marked
    - E.g. the word "THE" at the beginning of a sentence following a pause
- It is not *silence* per-se that is the context; it is the fact that the sound was the first one uttered
  - And the "SIL" context represents the effect of the articulatory effort in starting off with that sound
- As a result, any time speech begins, the first phoneme is marked as having SIL as a left context
  - Regardless of whether the background is really silent or not
- SIL also similarly forms the right context of the final triphone before a pause

Pauses, Silences, Pronunciation Markers

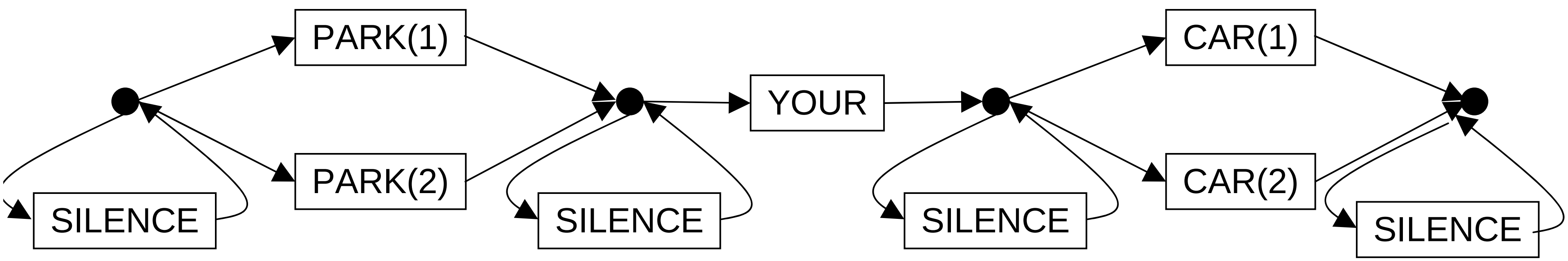
- Pauses and silences are usually not marked on transcriptions
  - Especially short pauses
- Pauses must be introduced automatically.

- Words may be pronounced in different ways
  - Read: R IY D or R EH D?

- The specific pronunciation is usually not indicated on the transcripts
  - Must be deduced automatically

- Pauses and identity of pronunciation variants can be discovered through “forced alignment”

Forced Alignment



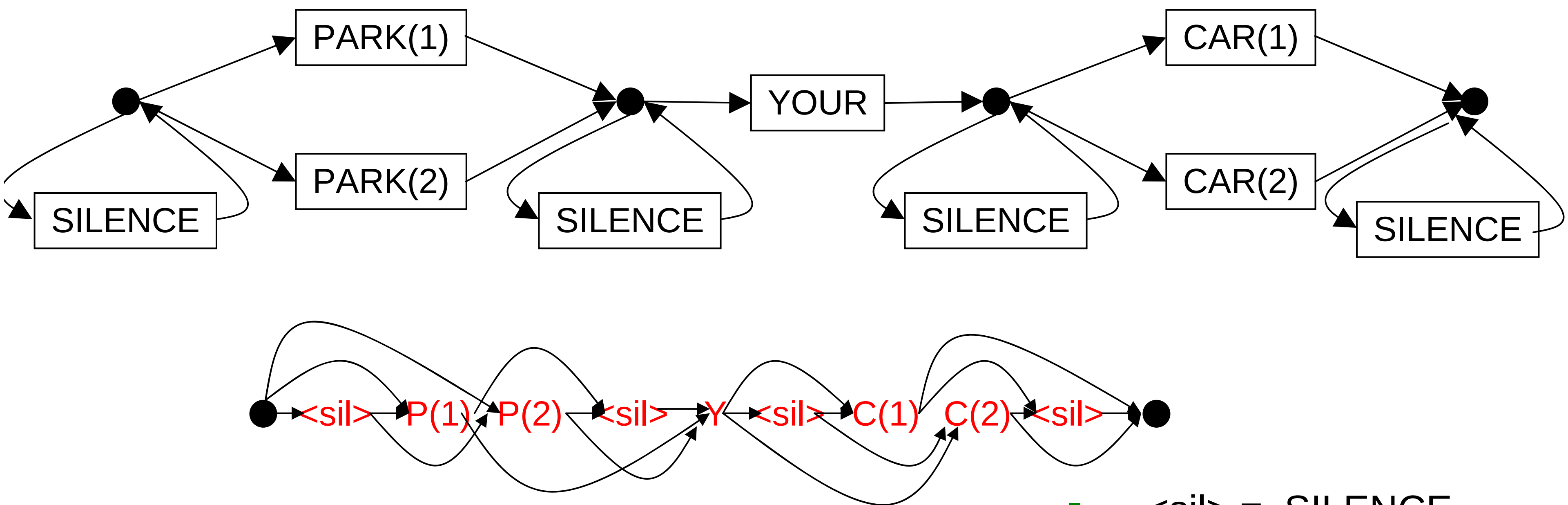
HMM for "Park Your Car" with optional silences between words

Rectangles actually represent entire HMMs (simplified illustration)

Note: "Park" and "Car" are pronounced differently in Boston than elsewhere

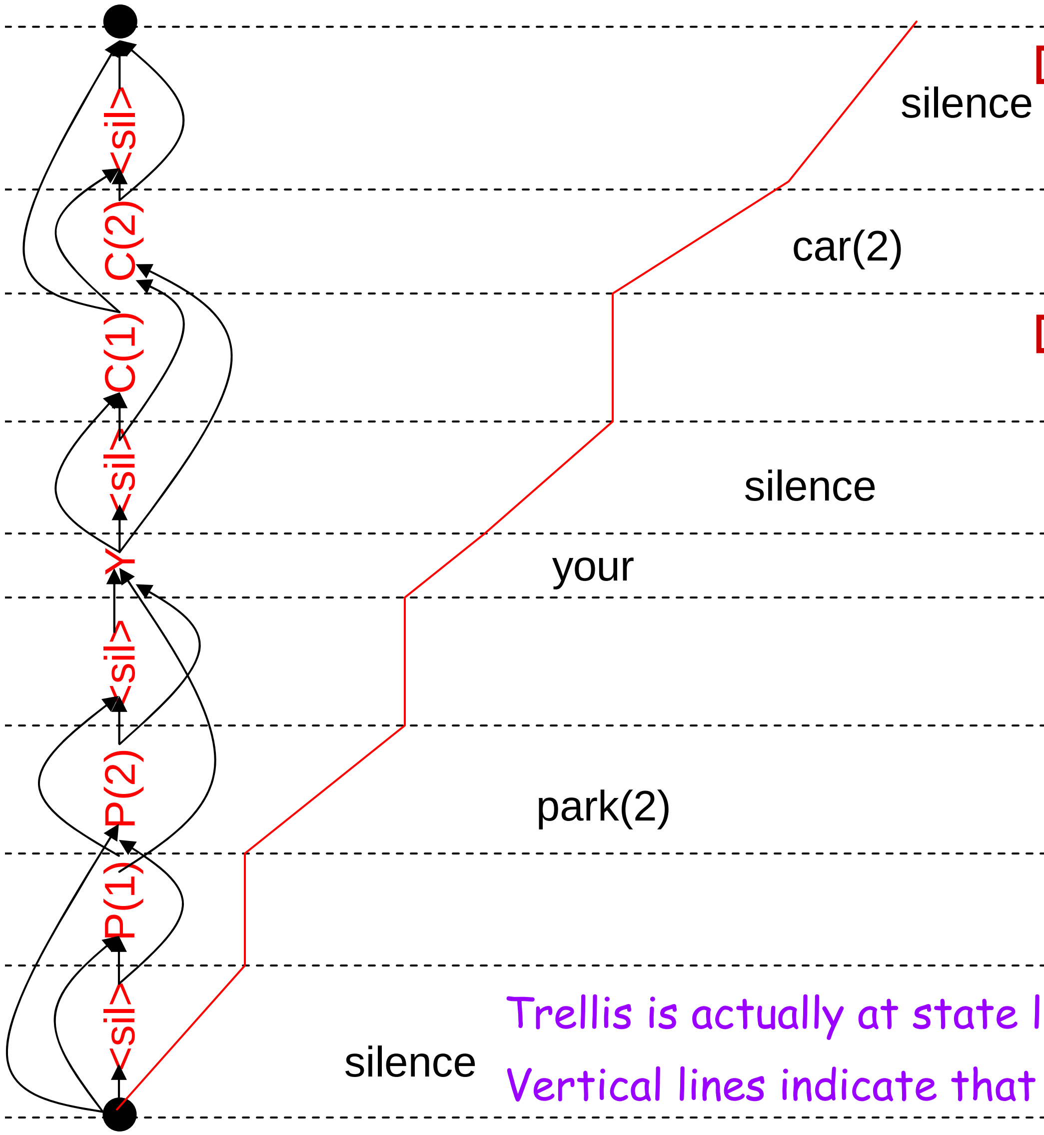
- For each sentence in the training data, compose an HMM as shown
  - "Optional" silences between words
  - All pronunciations for a word are included as parallel paths

A short(er) hand illustration



- <sil> = SILENCE
- P(1) = PARK(1)
- P(2) = PARK(2)
- Y = YOUR
- C(1) = CAR(1)
- C(2) = CAR(2)

Forced Alignment



Trellis is actually at state level

Vertical lines indicate that a "skip" transition has been followed

- A viterbi algorithm can then be used to obtain a state segmentation
- This also identifies the locations of pauses and specific pronunciation variant
  - E.g the state sequence may identify, for <sil> PARK(2) YOUR <sil> CAR(2) <sil>
    - Clearly a Bostonian

Forced Alignment Requires Existing Models

- In order to perform forced alignment to identify pauses and pronunciation tags, we need existing acoustic models
  - Which we will not have at the outset

- Solution:
  - Train a *preliminary* set of models with no pauses or pronunciation tags marked in the transcript
  - We *will* however need some initial guess to the location of silences
    - Or we will not be able to train models for them

  - A good guess: There is typically silence at the beginning and end of utterances
  - Mark silences at the beginning and end of utterances when training preliminary models
    - E.g. <SIL> A GOOD CIGAR IS A SMOKE <SIL>

  - Forced align with preliminary models for updated tagged transcripts
  - Retrain acoustic models with modified transcripts

Building tied-state Models

- Sphinxtrain exercise
- Note Sphinx restriction: No. of Gaussians per state same for all states