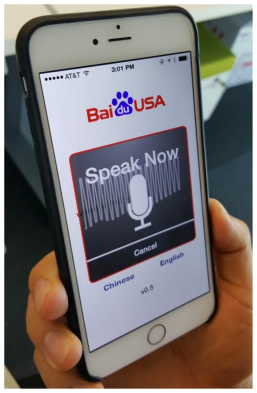
Speech Recognition and Deep Learning

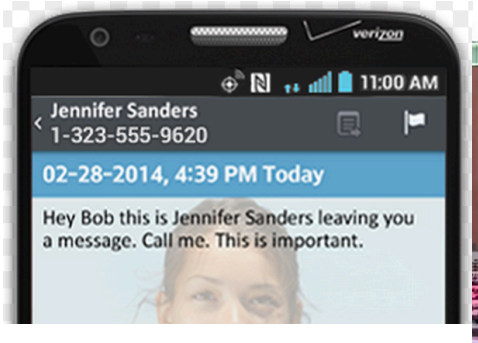
Adam Coates

Silicon Valley AI Lab

Speech recognition

- Important goal of AI research:
  - Lots of applications
    - Video/voice transcripts
    - Natural interface to services and devices

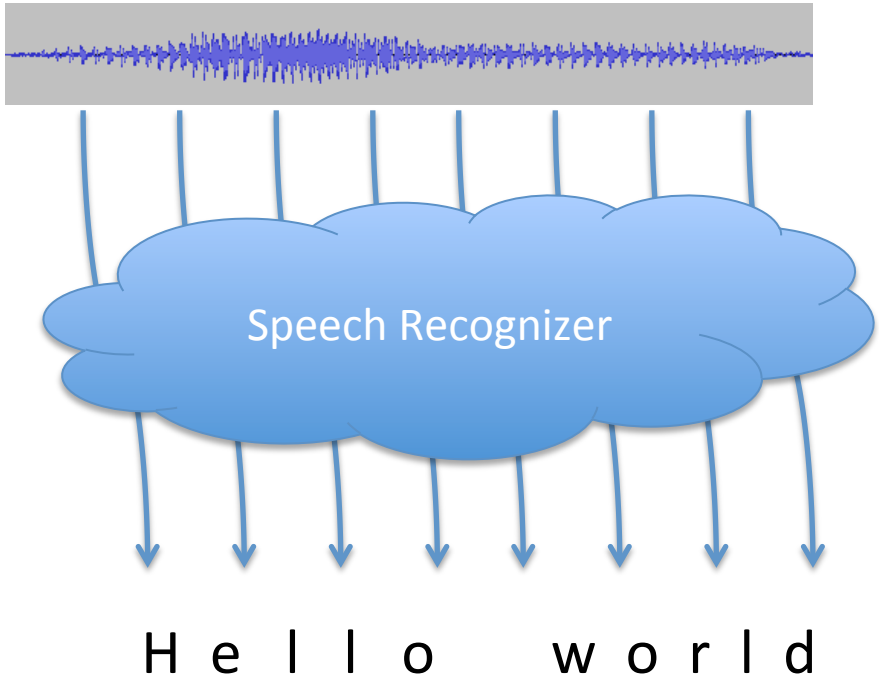




  - Transcription is often easy for people.
    - Historically really hard for machines.

Speech recognition

- High-level goal: given speech audio, generate a transcript.



Speech recognition

- Difficulty depends on many factors.
  - Type of speech:
    - Conversational versus read.
  - Variations in tempo, volume.
  - Natural speaker variation
  - Pronunciation and accents
  - Disfluency (repeated words, stuttering, uhms)
  - Environment: Signal to noise ratio; reverb.
  - Lombard effect
  - Large [likely superhuman] vocabulary.

- Very hard to engineer around all of these! Great place for DL to make a difference.

Outline

- Traditional speech models
  - Still dominant architecture behind state-of-the-art systems.
  - Commonly assumed throughout literature.

  Think of this as DL Survival School for speech.

- Deep Learning for speech recognition
  - Direct improvements on traditional method.
  - CTC and end-to-end learning.

TRADITIONAL SPEECH MODELS

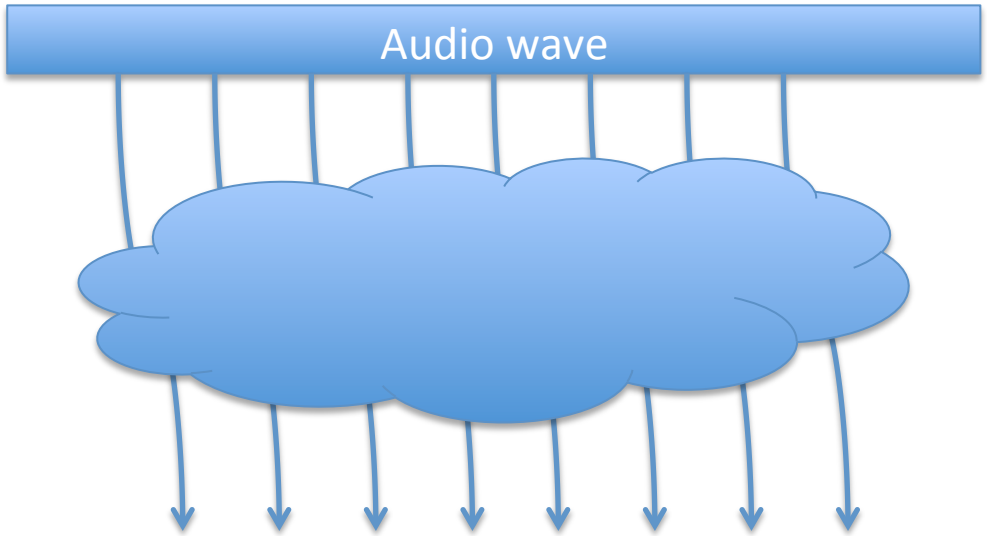
Basic pipeline

- Represents wide range of current practice.
  - Will gloss over some algorithmic details.
  - If DL community is successful, a lot may go away!

Basic pipeline

- Goal: given raw audio, convert to sequence of characters.

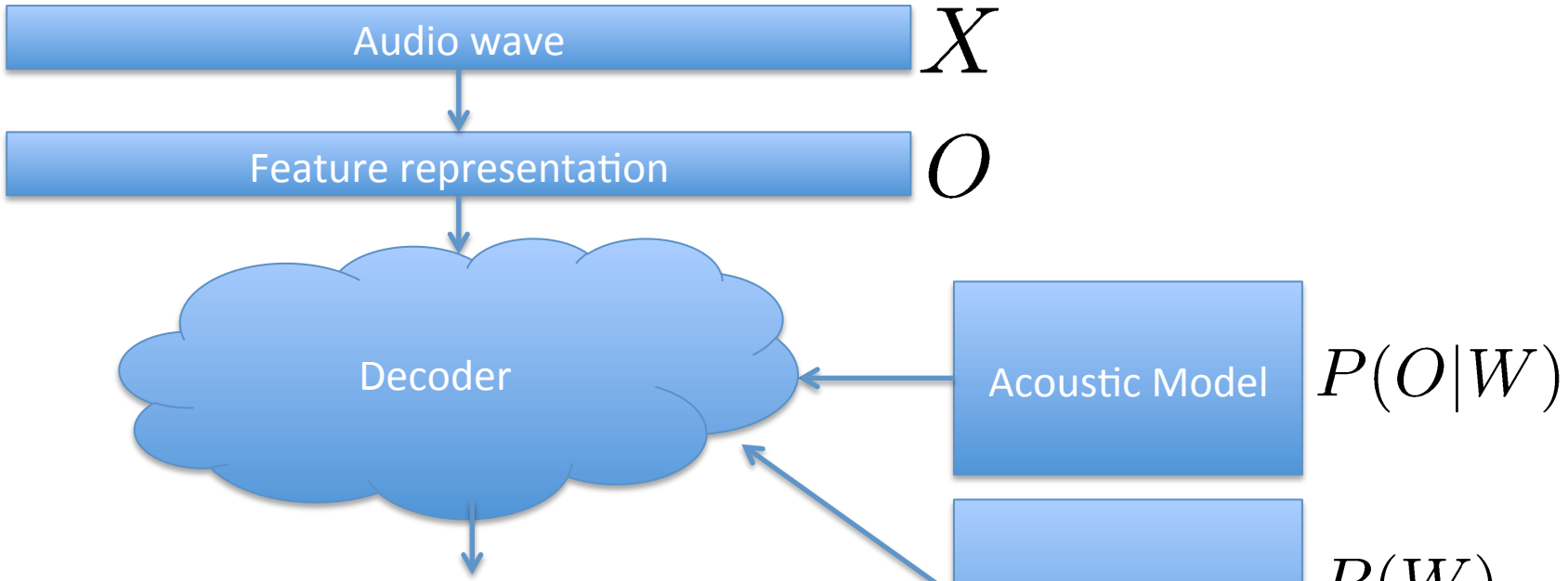
$$X = [x_1 x_2 \ldots]$$



$$W^* = [w_1 w_2] = \underset{W}{\operatorname{argmax}}\ P(W|X)$$

Basic pipeline

- In practice, systems factorize work into several components:



$$W^* = \operatorname*{argmax}_W P(W|X)$$
$$= \operatorname*{argmax}_W P(O|W)P(W)$$

Basic pipeline

- Usually represent words as sequence of "phonemes":

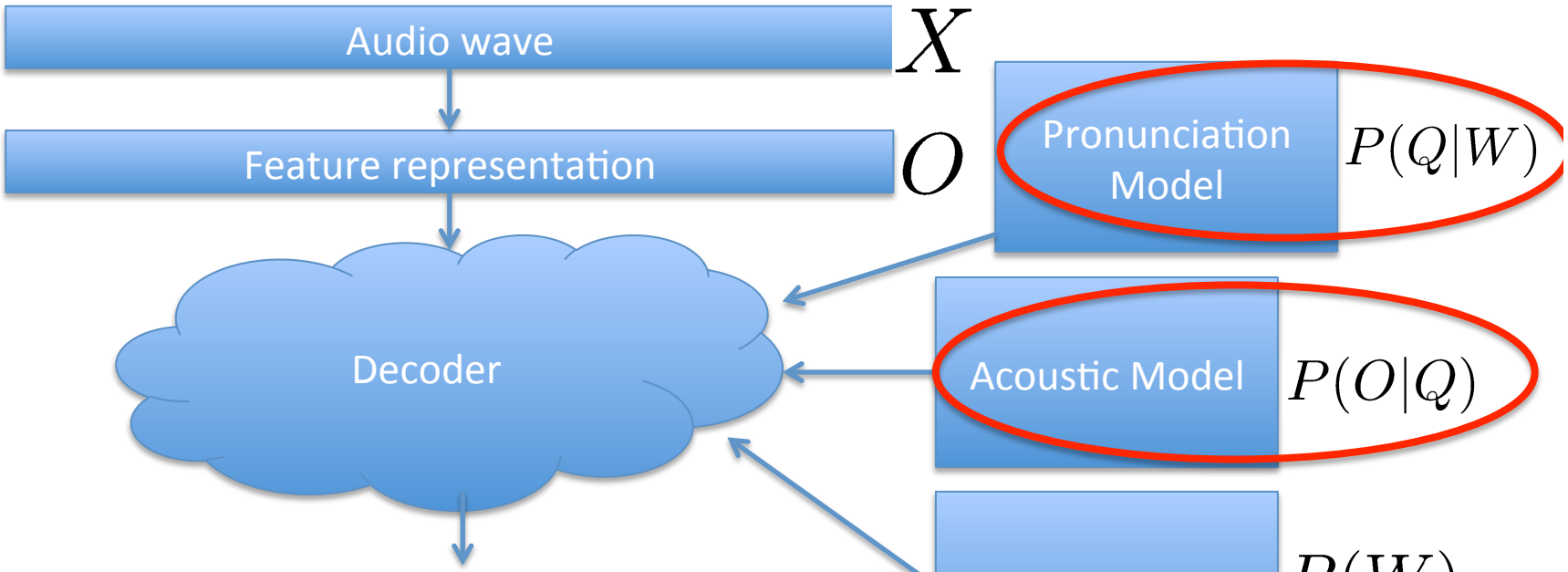
$$w_1 = \text{“hello”} = \text{[HH AH L OW]} = [q_1 q_2 q_3 q_4]$$

- Phonemes are the perceptually distinct units of sound that distinguish words.
  - Quite approximate… but sorta standardized-ish.
  - Some labeled corpora available (e.g., TIMIT)

	Phone Label	Example		Phone Label	Example		Phone Label	Example
1	iy	beet	22	ch	choke	43	en	button
2	ih	bit	23	b	bee	44	eng	Washington

Basic pipeline

- Traditional systems usually model phoneme sequences instead of words. This necessitates a dictionary or other model to translate.

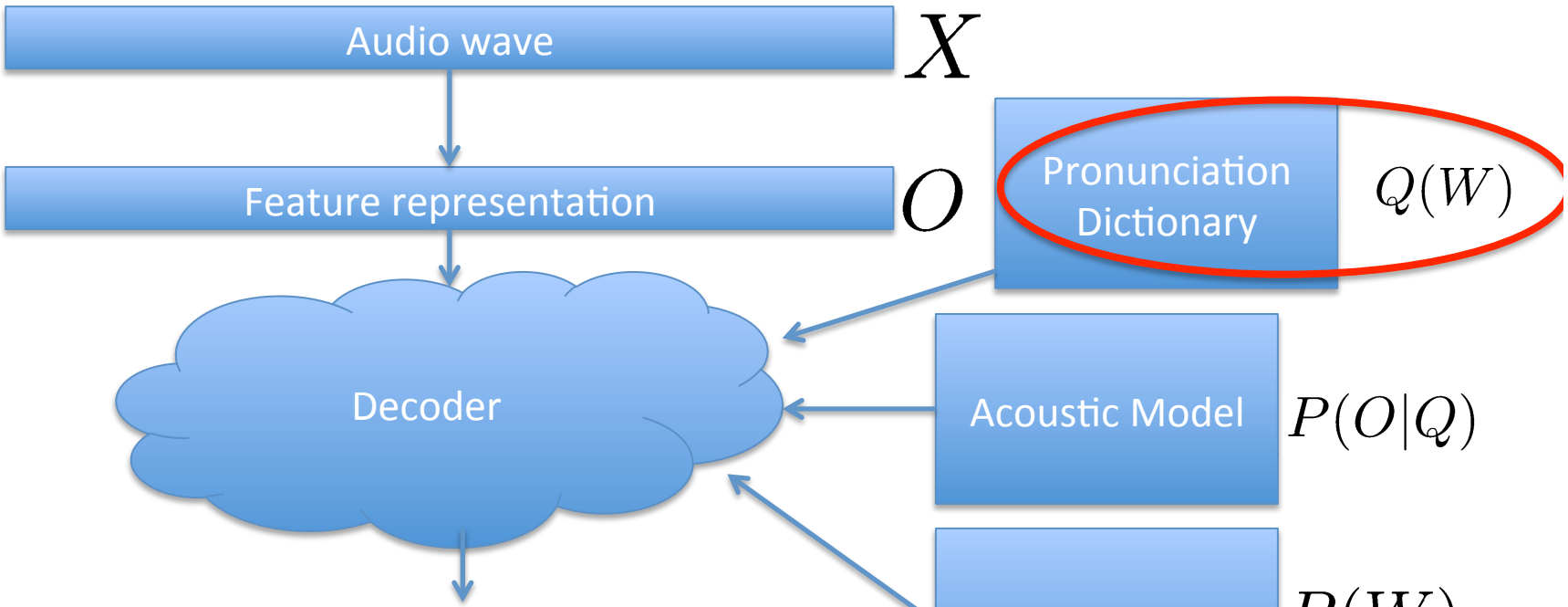


$$W^* = \underset{W}{\operatorname{argmax}}\ P(W|X)$$

$$= \underset{W}{\operatorname{argmax}} \sum_{Q} P(O|Q)P(Q|W)P(W)$$

Basic pipeline

- We'll just use a dictionary: only allow 1 pronunciation.



$$W^* = \underset{W}{\operatorname{argmax}}\ P(W|X)$$

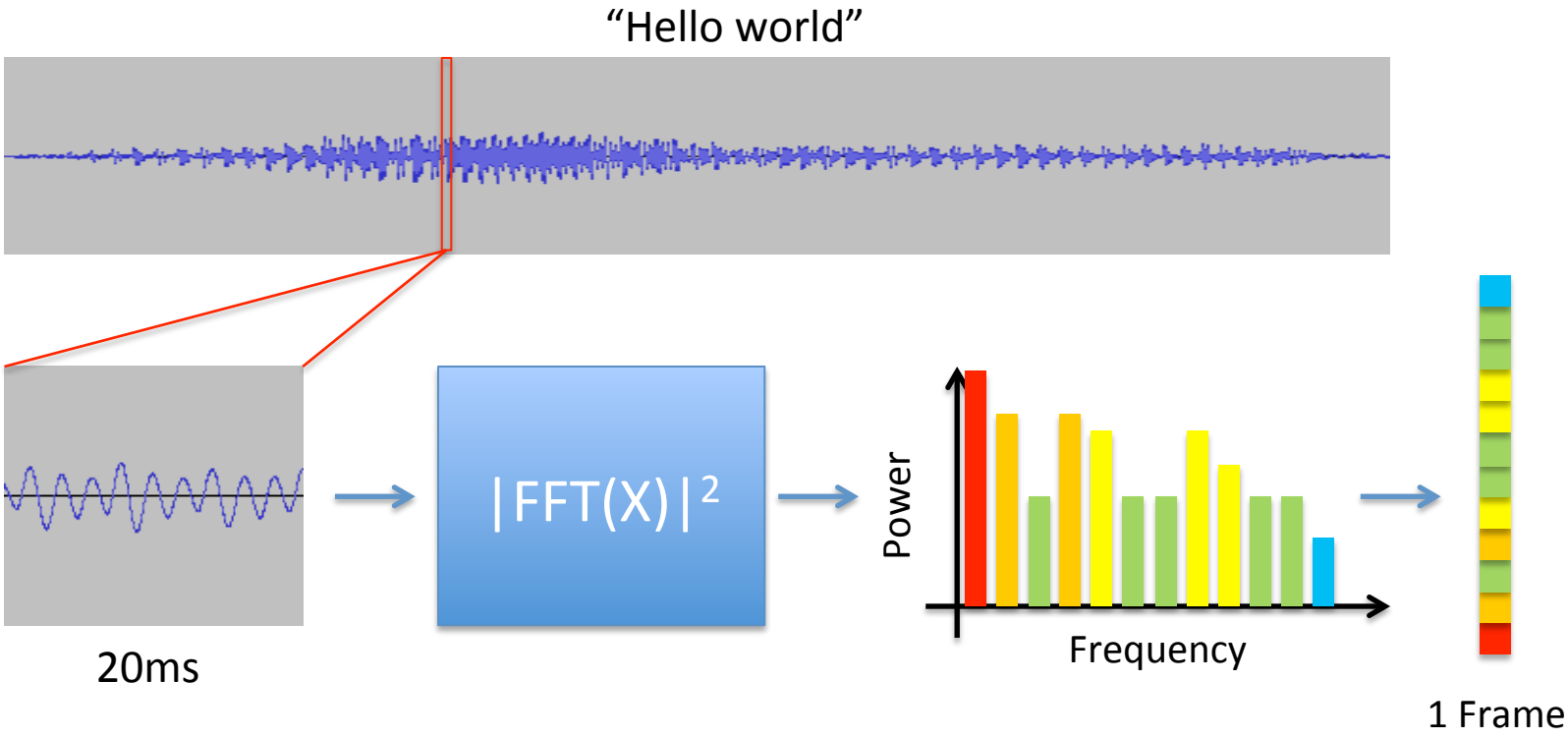
$$= \underset{W}{\operatorname{argmax}}\ P(O|Q(W))P(W)$$

Features

- As with most ML tasks, first want to convert raw input into more convenient features.
  - Spectrograms
  - MFCC (Mel Frequency Cepstral Coefficients)
  - PLP, RASTA [Hermansky, 1990; 1994]
  - “Delta” features

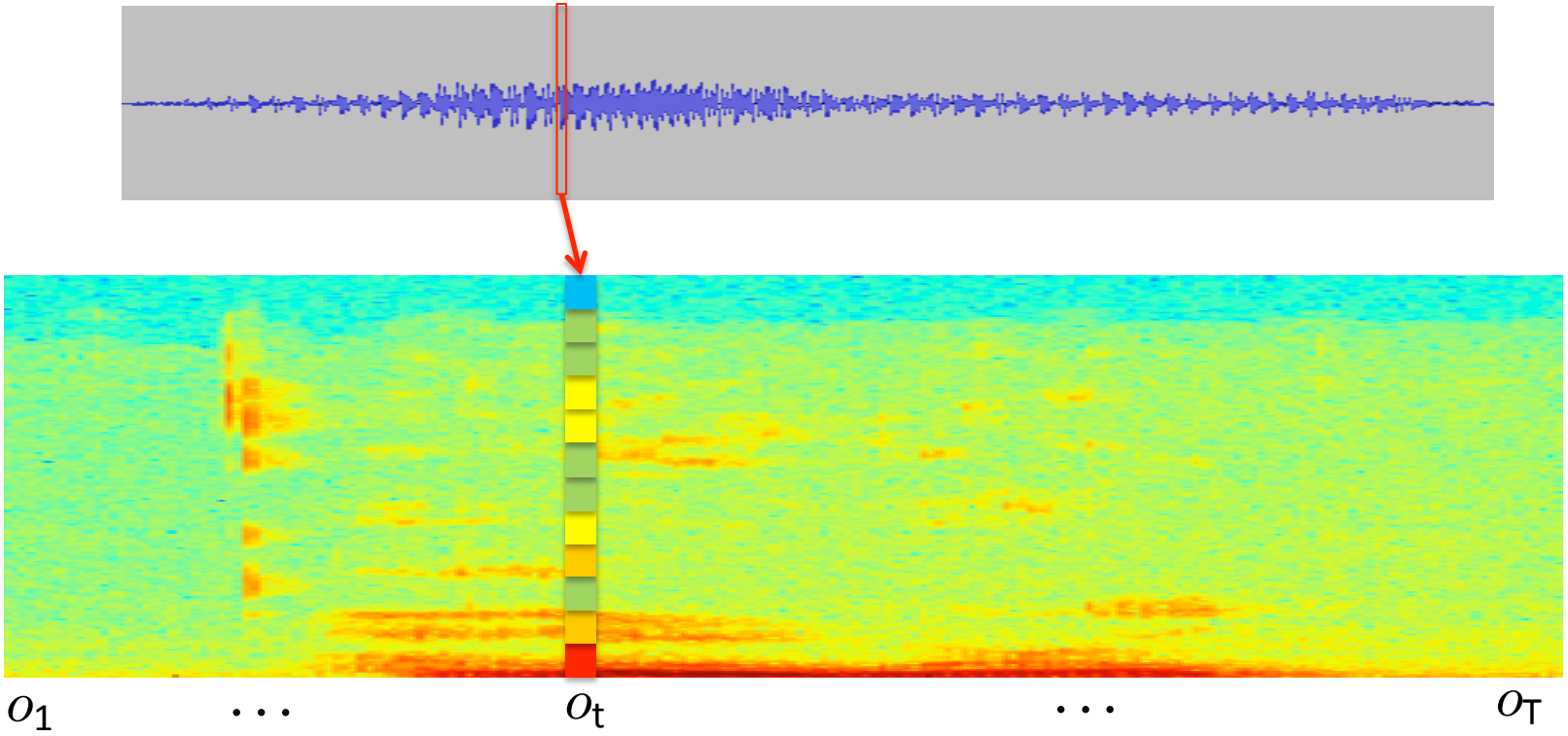
Example: Spectrogram

- Take a small window (e.g., 20ms) of waveform.
  - Compute FFT and take magnitude. (i.e., power)
  - Describes frequency content in local window.



Example: Spectrogram

- Concatenate frames from adjacent windows to form “spectrogram”.



BACK TO MODELING

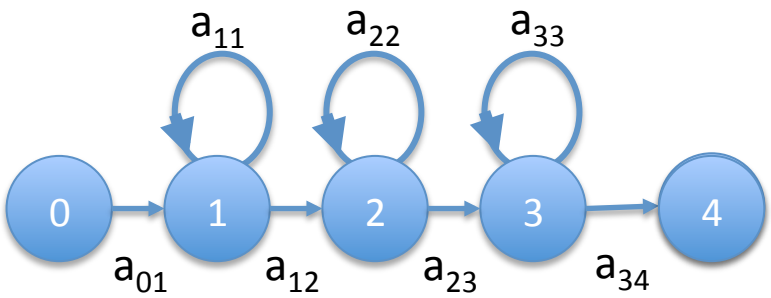
Acoustic model

- We need a model of P(O|Q): a generative model of features (e.g., spectrogram) given phoneme sequence Q.

- Start with a simpler case: a single phoneme q.
- Model sequence of observations generated while speaking q using HMM.

Modeling 1 phoneme

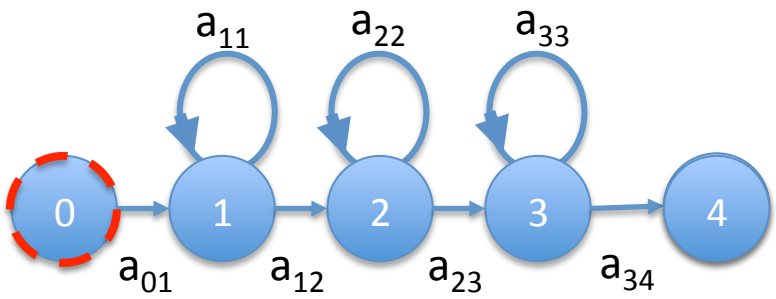
- Use an HMM with simple "left-to-right" state structure.
  - Think of generating process as a state machine.
    - Start in state 0 at t=0.
    - At each time step: Jump from state s_t=i to state s_{t+1}=j with probability a_{ij}
    - After each jump generate a frame according to $P(o_t|s_t)$.
      - E.g., use $P(o_t|s_t=j)$ = Gaussian(μ_j, Σ_j)



[Gales & Young, 2008]

Modeling 1 phoneme

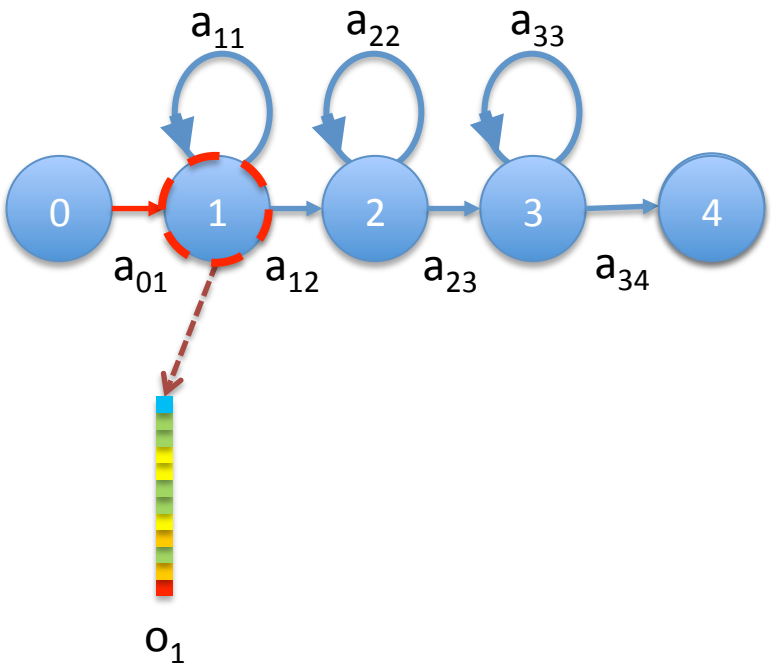
- Use an HMM with simple "left-to-right" state structure.
  - Think of generating process as a state machine.
    - Start in state 0 at t=0.
    - At each time step: Jump from state s_t=i to state s_{t+1}=j with probability a_{ij}
    - After each jump generate a frame according to $P(o_t|s_t)$.
      - E.g., use $P(o_t|s_t=j)$ = Gaussian(μ_j, Σ_j)



[Gales & Young, 2008]

Modeling 1 phoneme

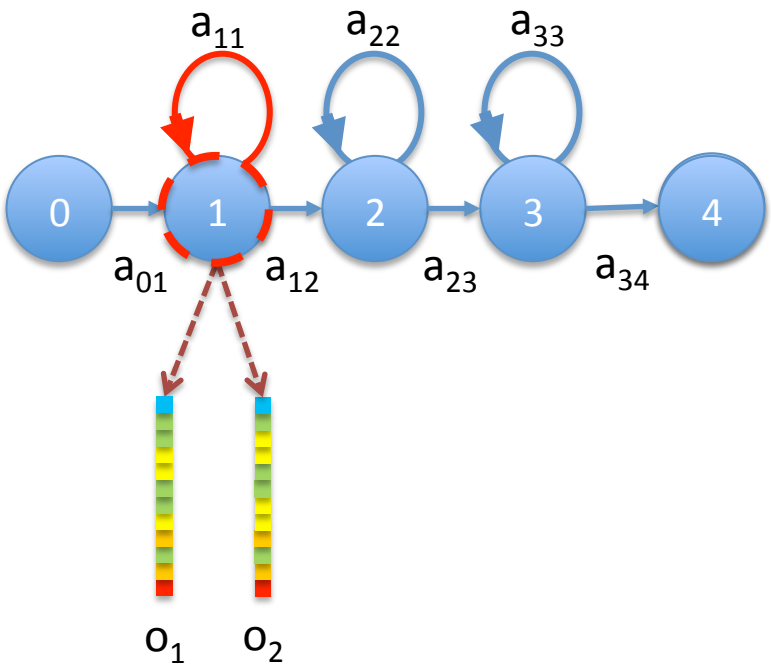
- Use an HMM with simple "left-to-right" state structure.
  - Think of generating process as a state machine.
    - Start in state 0 at t=0.
    - At each time step: Jump from state s_t=i to state s_{t+1}=j with probability a_{ij}
    - After each jump generate a frame according to $P(o_t|s_t)$.
      - E.g., use $P(o_t|s_t=j)$ = Gaussian(μ_j, Σ_j)



[Gales & Young, 2008]

Modeling 1 phoneme

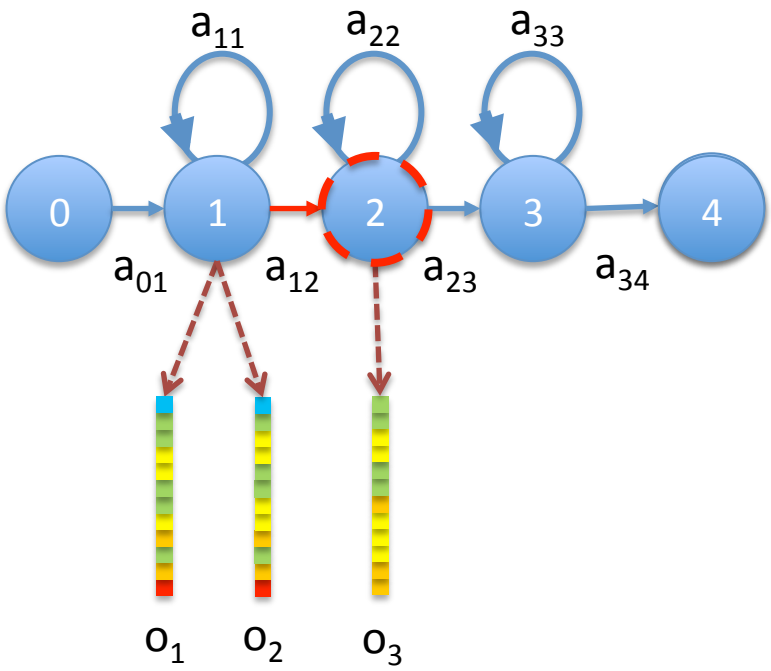
- Use an HMM with simple "left-to-right" state structure.
  - Think of generating process as a state machine.
    - Start in state 0 at t=0.
    - At each time step: Jump from state s_t=i to state s_{t+1}=j with probability a_{ij}
    - After each jump generate a frame according to $P(o_t|s_t)$.
      - E.g., use $P(o_t|s_t=j)$ = Gaussian(μ_j, Σ_j)



[Gales & Young, 2008]

Modeling 1 phoneme

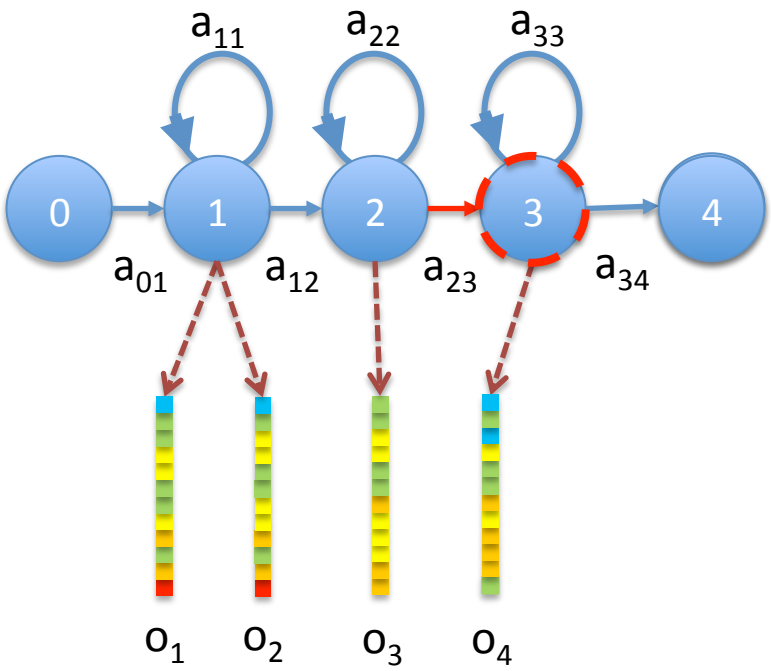
- Use an HMM with simple "left-to-right" state structure.
  - Think of generating process as a state machine.
    - Start in state 0 at t=0.
    - At each time step: Jump from state s_t=i to state s_{t+1}=j with probability a_{ij}
    - After each jump generate a frame according to $P(o_t|s_t)$.
      - E.g., use $P(o_t|s_t=j)$ = Gaussian(μ_j, Σ_j)



[Gales & Young, 2008]

Modeling 1 phoneme

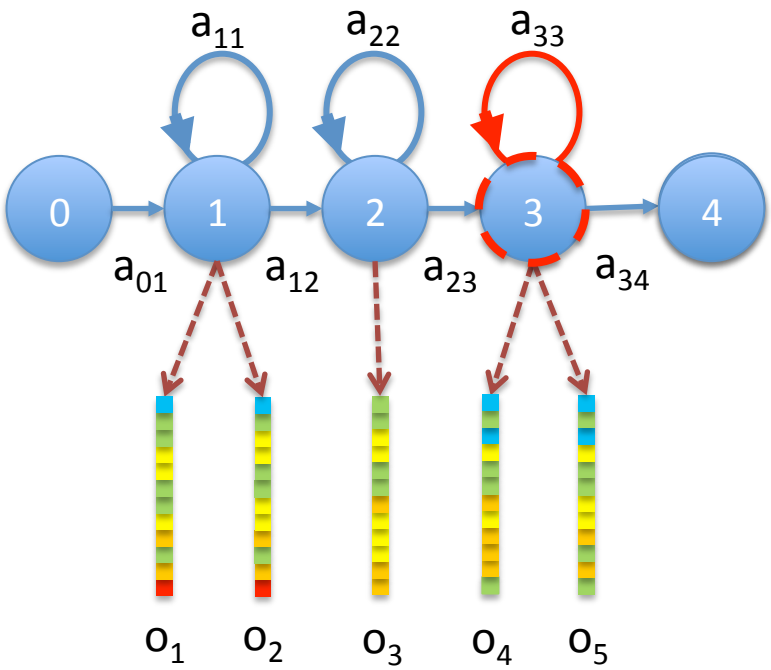
- Use an HMM with simple "left-to-right" state structure.
  - Think of generating process as a state machine.
    - Start in state 0 at t=0.
    - At each time step: Jump from state s_t=i to state s_{t+1}=j with probability a_{ij}
    - After each jump generate a frame according to $P(o_t|s_t)$.
      - E.g., use $P(o_t|s_t=j)$ = Gaussian(μ_j, Σ_j)



[Gales & Young, 2008]

Modeling 1 phoneme

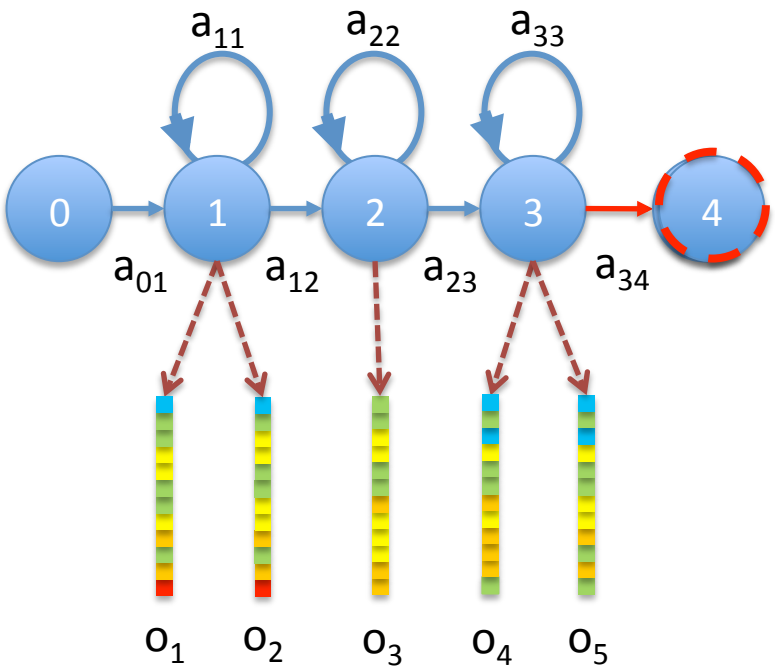
- Use an HMM with simple "left-to-right" state structure.
  - Think of generating process as a state machine.
    - Start in state 0 at t=0.
    - At each time step: Jump from state s_t=i to state s_{t+1}=j with probability a_{ij}
    - After each jump generate a frame according to $P(o_t|s_t)$.
      - E.g., use $P(o_t|s_t=j)$ = Gaussian(μ_j, Σ_j)



[Gales & Young, 2008]

Modeling 1 phoneme

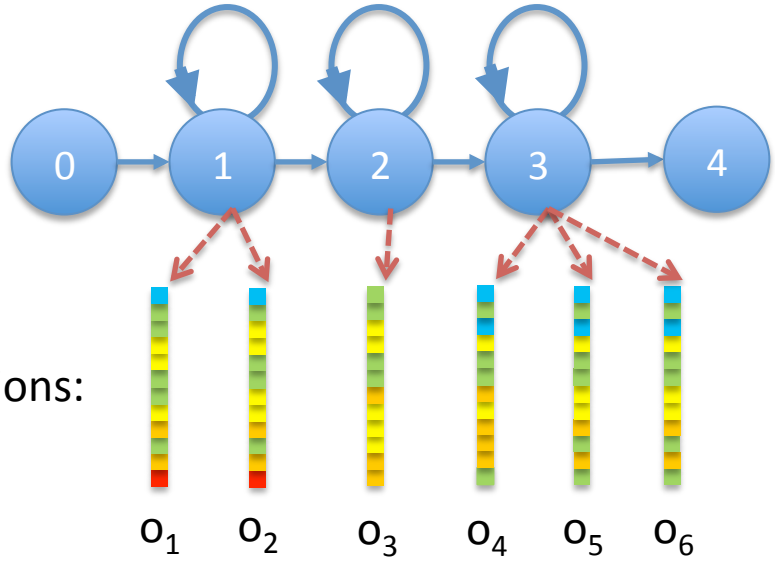
- Use an HMM with simple "left-to-right" state structure.
  - Think of generating process as a state machine.
    - Start in state 0 at t=0.
    - At each time step: Jump from state s_t=i to state s_{t+1}=j with probability a_{ij}
    - After each jump generate a frame according to $P(o_t|s_t)$.
      - E.g., use $P(o_t|s_t=j)$ = Gaussian(μ_j, Σ_j)



[Gales & Young, 2008]

Inference with 1 phoneme

- We now have HMM with parameters $\{a_{ij}, \mu_j, \Sigma_j\}$
  - *Given* an observation sequence, $o_1\ o_2\ \ldots.\ o_T$ we can*:*
    - Find most likely sequence of internal states $s_1\ s_2\ \ldots\ s_T$ that generated O.
      - Viterbi algorithm.



Viterbi output:

s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7
0	1	1	2	3	3	3	4

[Gales & Young, 2008]

Inference with 1 phoneme

- We now have HMM with parameters $\{a_{ij}, \mu_j, \Sigma_j\}$
  - *Given* an observation sequence, o_1 o_2 …. o_T we can*:*
    - Find most likely sequence of internal states s_1 s_2 … s_T that generated O.
      - Viterbi algorithm.
    - Also: Compute likelihood of observations P(O|q) by summing over all possible state sequences in the HMM for q.

$$P(O|q) = \sum_S P(O|S)P(S|q)$$

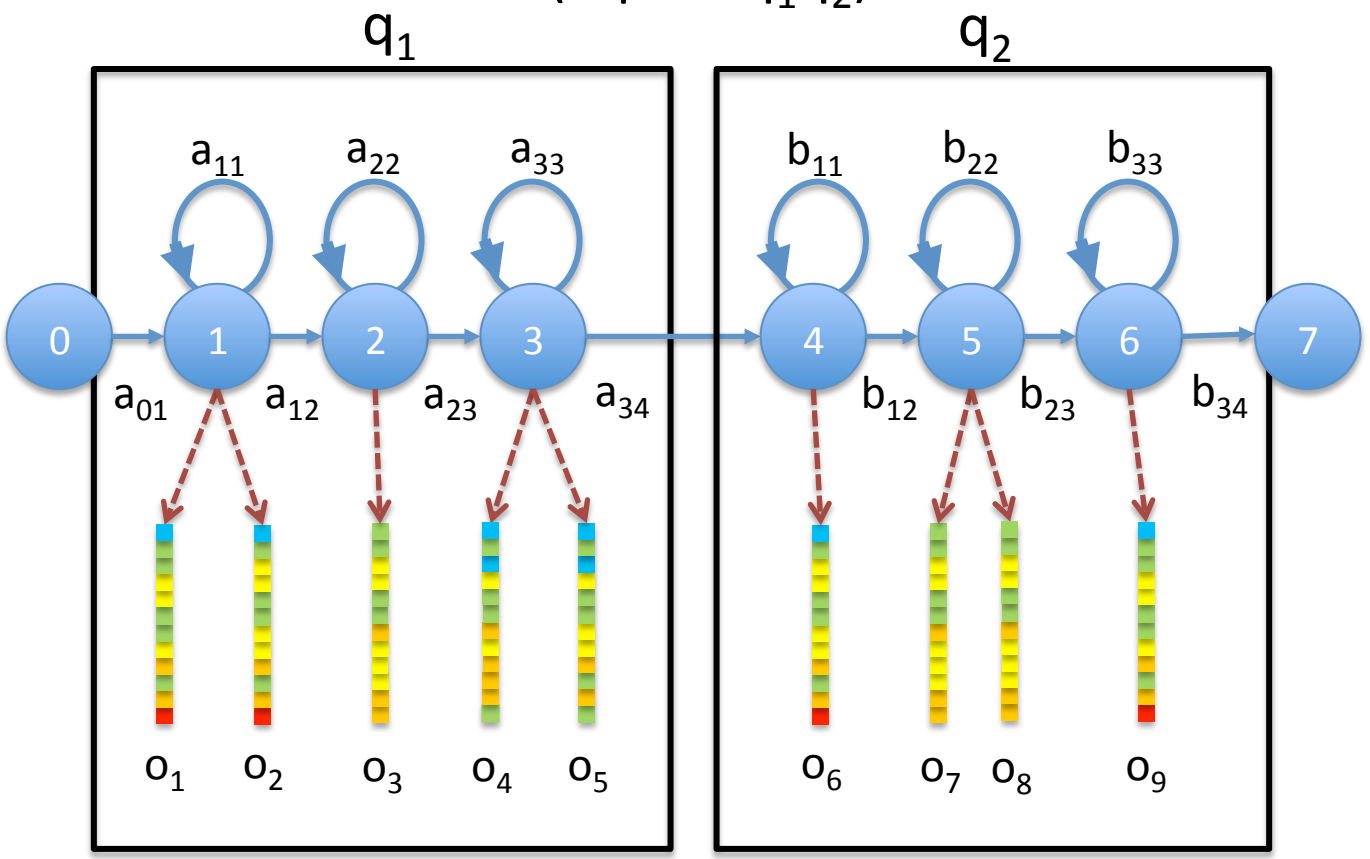
    - Solved with forward-backward algorithm. (This is the acoustic model likelihood we wanted!)

[Gales & Young, 2008]

Modeling a word

- Given a phoneme sequence (word) we can "construct" a word-level HMM by stringing state machines together.

$$P(O|Q = q_1 q_2)$$



[Gales & Young, 2008]

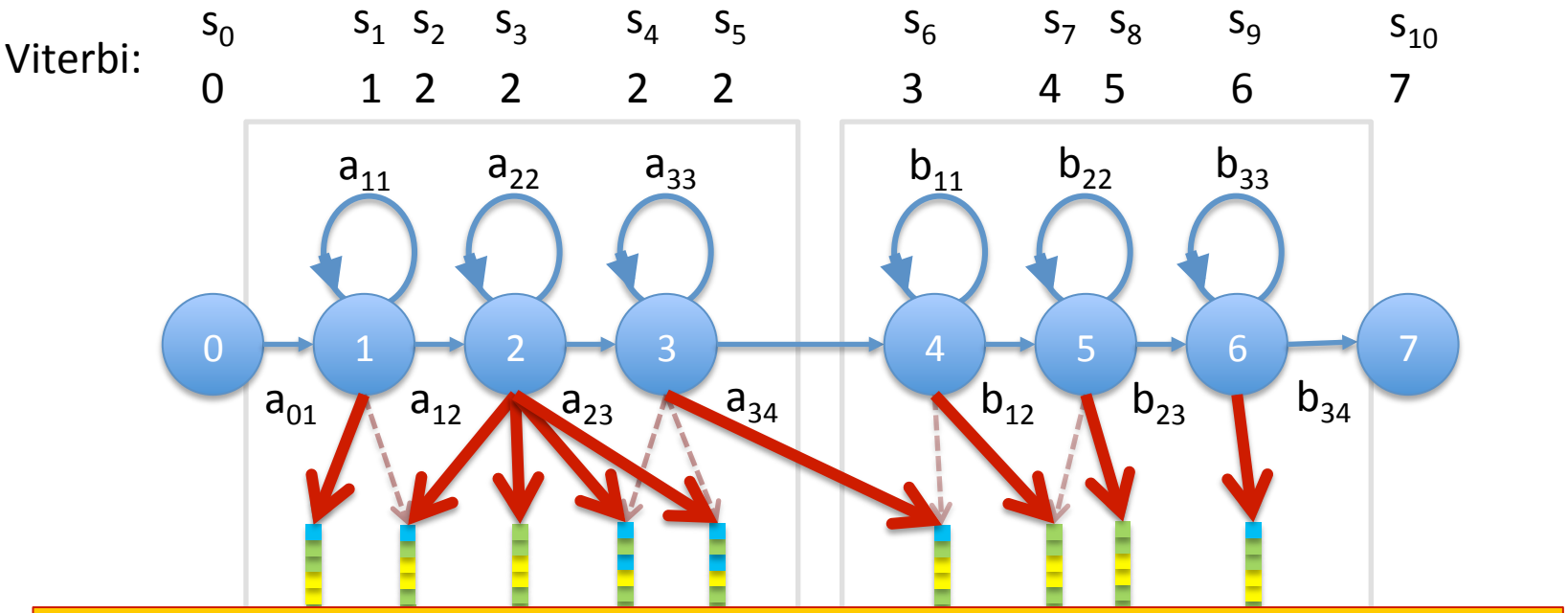
Training from sentences

- Sentence is just a sequence of word models.
  1. Convert sentence into sequence of phonemes.
  2. Define HMM by composing phoneme models.

- Training:
  - We have a fixed HMM structure defined by sentence.
  - We have observations (training data) generated by the HMM.
  - Use Expectation-Maximization (EM; aka Baum-Welch)
  - E-step: Inference to find hidden state posterior [P(s|O)]
    - Use forward-backward.
  - M-step: Update parameters to maximize likelihood of O.

Training from sentences

- Sadly, EM is not guaranteed to give us a good answer. What can go wrong?
- Illustration: imagine E-step just computes most likely state sequence.



This corresponds to an *alignment* between observations and phonemes.
If we get it wrong, parameter update might be poor.
E.g., tune observation models for wrong phoneme.

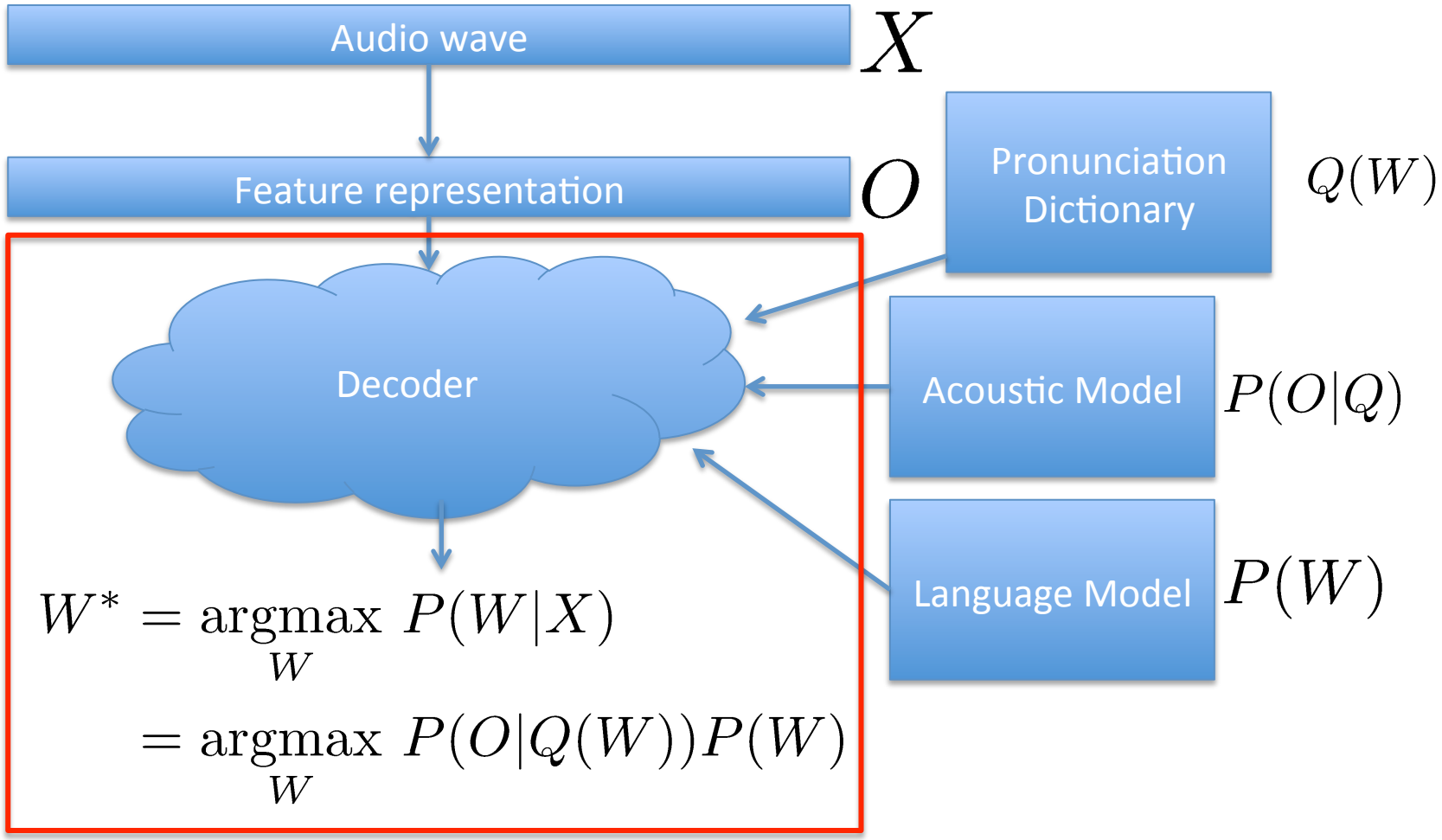
Obstacles…

- Lots of tricks to get this to work well.
  - E.g., initialize observation models by pre-training from small corpus of annotated data.

Language modeling

- In addition to acoustic model, need LM: $P(W)$
  - Many options, but a few desiderata are important:
    - Reasonably fast to query.
      - Used inside decoder.
    - Ability to train on huge corpora.
      - Make up for relative paucity of speech data.
    - Ability to train *quickly.*
      - Production systems often want to deal with shifting/trending vocabulary.
- Very common default: N-gram model.
  - $P(w_k \mid w_{k-1}, w_{k-2}, \ldots, w_{k-N+1})$
    - Lots of smoothing tricks to be used with large N.
    - See, e.g., [Jurafsky & Martin, 2000] for intro.

Putting it together



Decoder

- Basic problem: search for sequence of words W = w_1 w_2 … w_K to maximize P(W|X).

$$W^* = \underset{W}{\operatorname{argmax}}\ P(W|X)$$
$$= \underset{W}{\operatorname{argmax}}\ P(O|Q(W))P(W)$$

Decoder

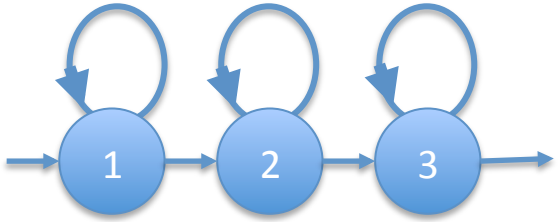
- Basic problem: search for sequence of words W = w_1 w_2 … w_K to maximize P(W|X).

$$
\begin{aligned}
W^* &= \underset{W}{\operatorname{argmax}}\ P(W|X) \\
&= \underset{W}{\operatorname{argmax}}\ P(O|Q(W))P(W) \\
&= \underset{W}{\operatorname{argmax}}\ \sum_S P(O|S)P(S|Q(W))P(W)
\end{aligned}
$$

  - Many strategies to do this. Often complex.
  - Here: simplify the problem to illustrate idea.
    - Only look for most likely state sequence S.
    - Note: if we fix a choice of S, this gives us Q and W.

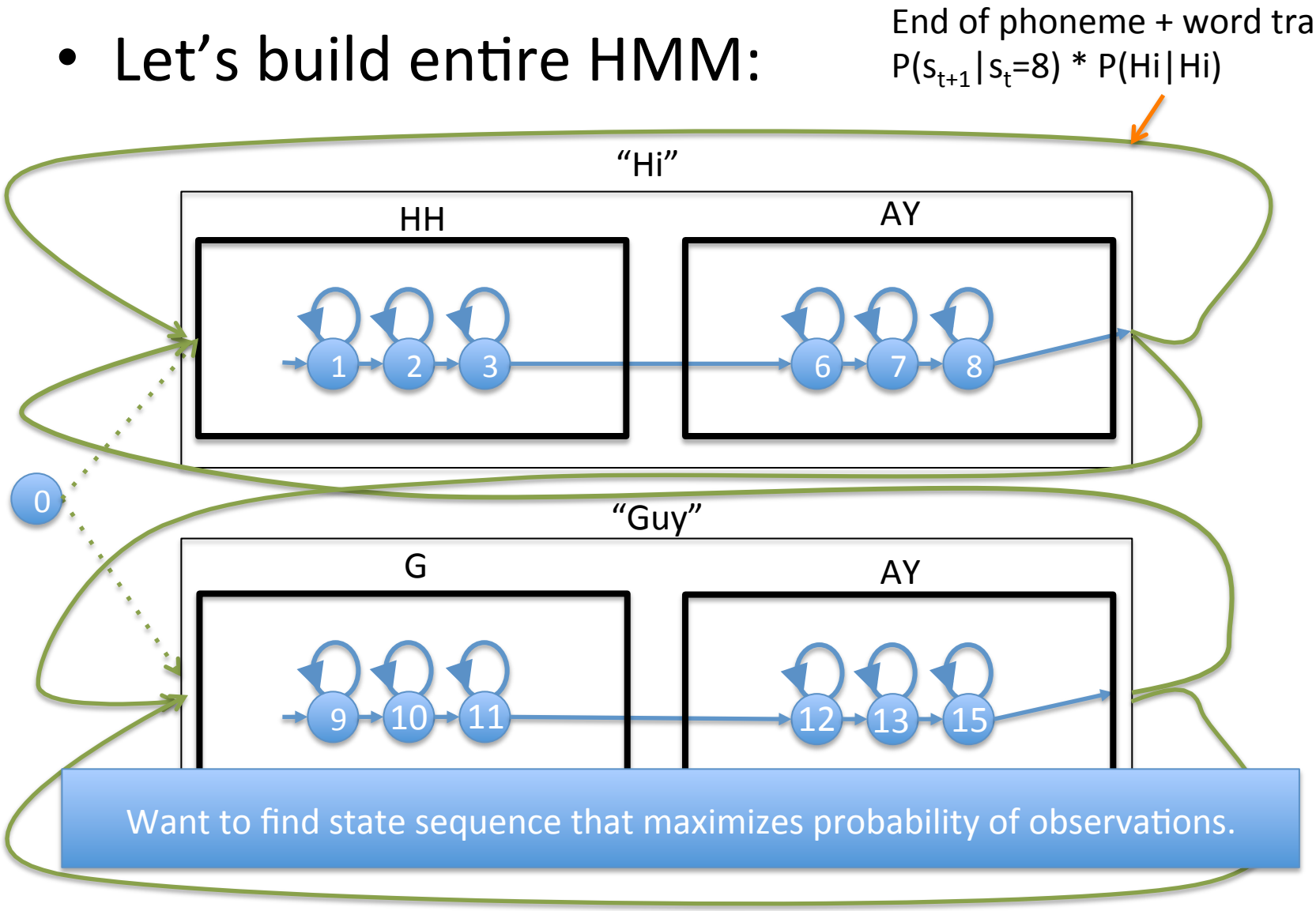
Decoder

- Simple problem: two word vocabulary.
  - "Hi" [HH AY] or "Guy" [G AY].
  - Language model:
    - P(Guy|Hi) = 0.9; P(Hi|Hi) = 0.1
    - P(Hi|Guy) = 0.5; P(Guy|Guy) = 0.5
  - HMM acoustic models like earlier.



Decoder visualization

- Let's build entire HMM:



Decoder

- Finding most likely sequence is easy with Viterbi!
  - Main issues:
    - Not practical for big problems.
    - We chose a bigram language model! Bigger N-gram would violate the Markov assumption.
      - Dynamic programming no longer works. :-(
- In practice: use general search formalism.
  - E.g., Beam search.

Decoder

- Beam search:
  - Keep a list of top N candidate partial state sequences.
  - Propose extensions (next state) for each candidate.
  - If we had entire state sequence, likelihood is:

$$\log P(O|S) + \log P(S|Q(W)) + \log(P(W)) =$$

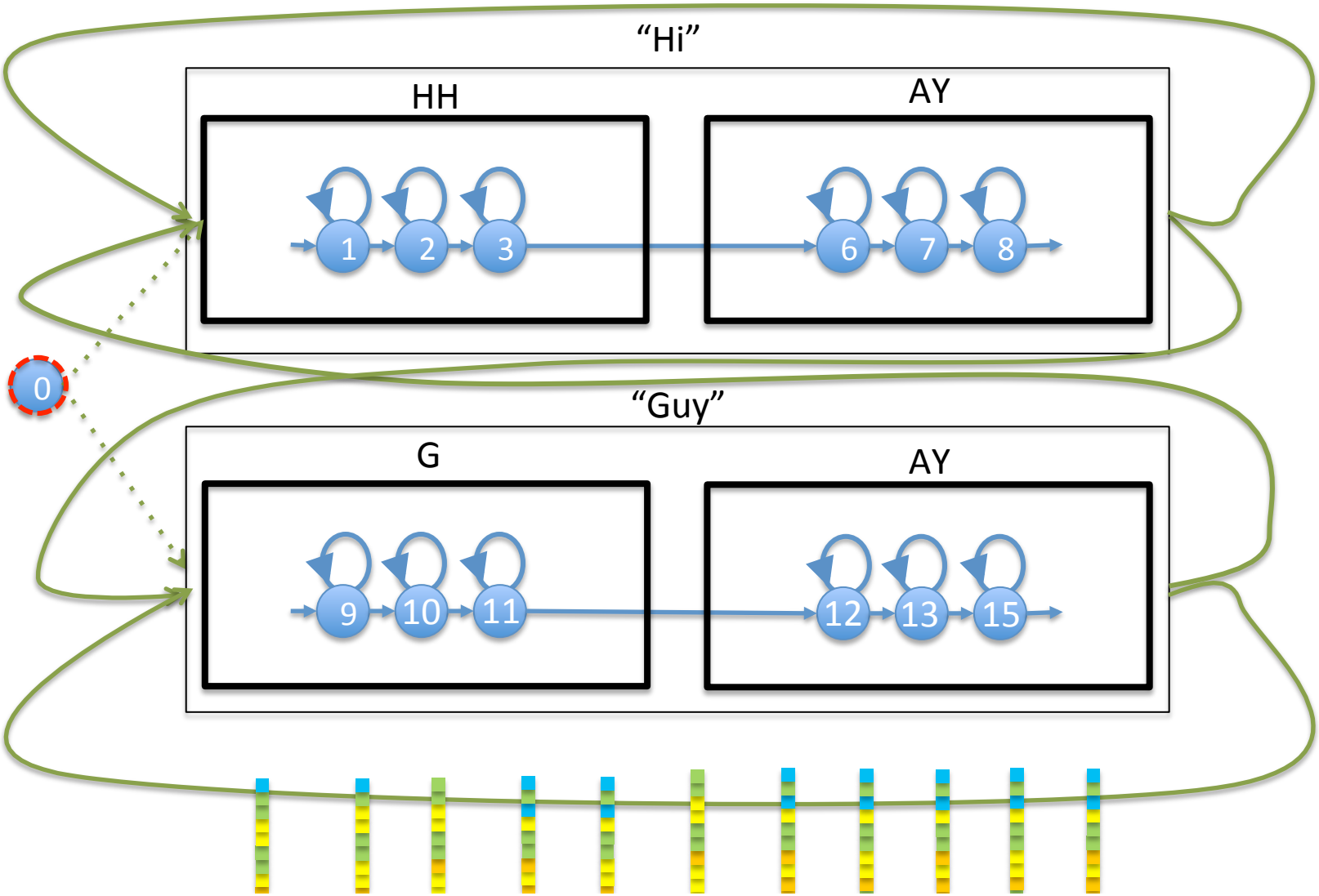
$$\sum_t \log P(O_t|S_t) + \log P(S_t|S_{t-1}) + \sum_k \log P(W_k|W_{k-1})$$

  - During search, keep track of partial sum:

$$\underbrace{\sum_{t=1}^{t'} \log P(O_t|S_t) + \log P(S_t|S_{t-1})}_{\text{Observations and state transitions so far.}} + \underbrace{\sum_{k=1}^{K'} \log P(W_k|W_{k-1})}_{\text{Words in sequence so far.}}$$

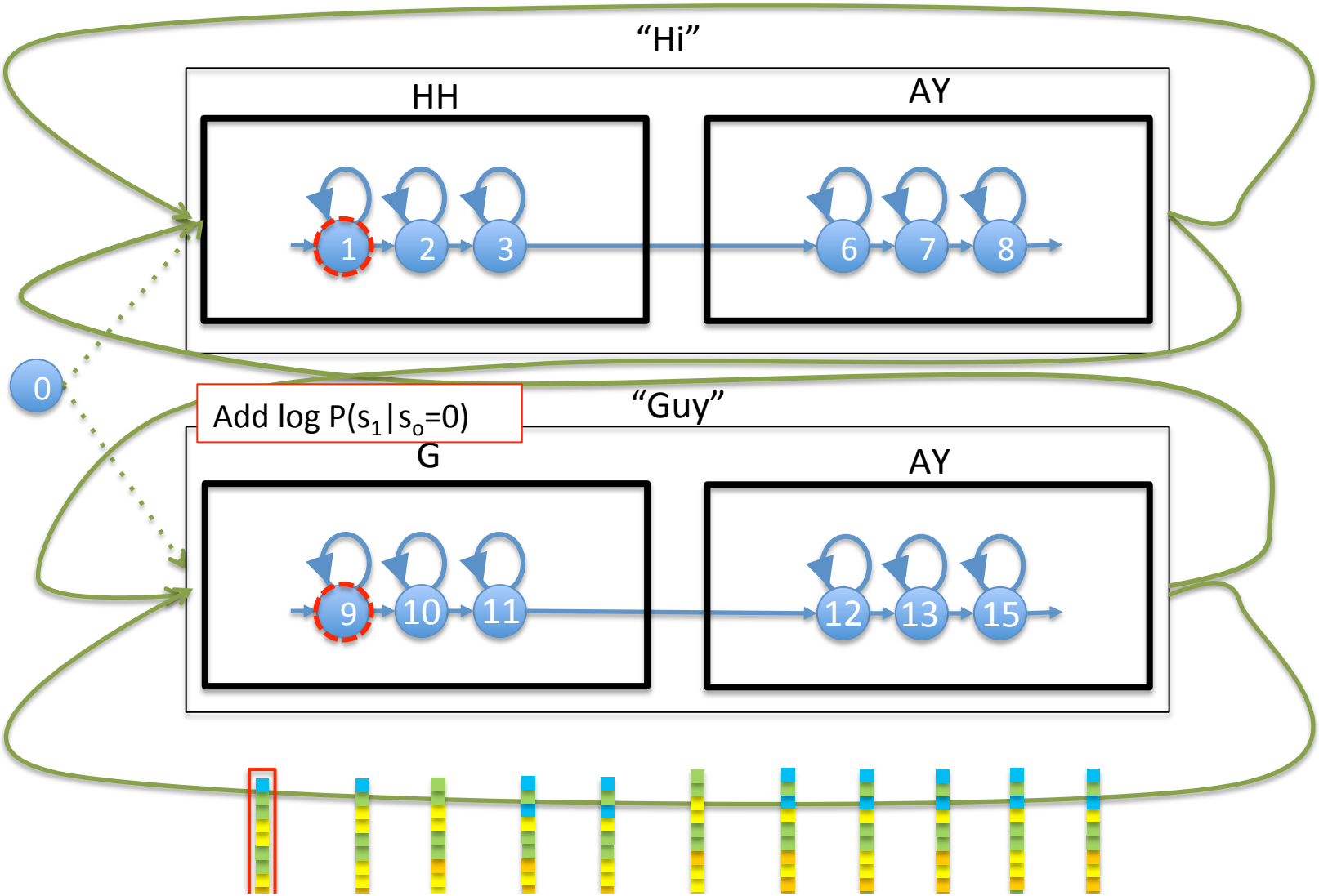
Decoder visualization

Example Beam search: keep top 2 paths; accumulate likelihood of O as we go.



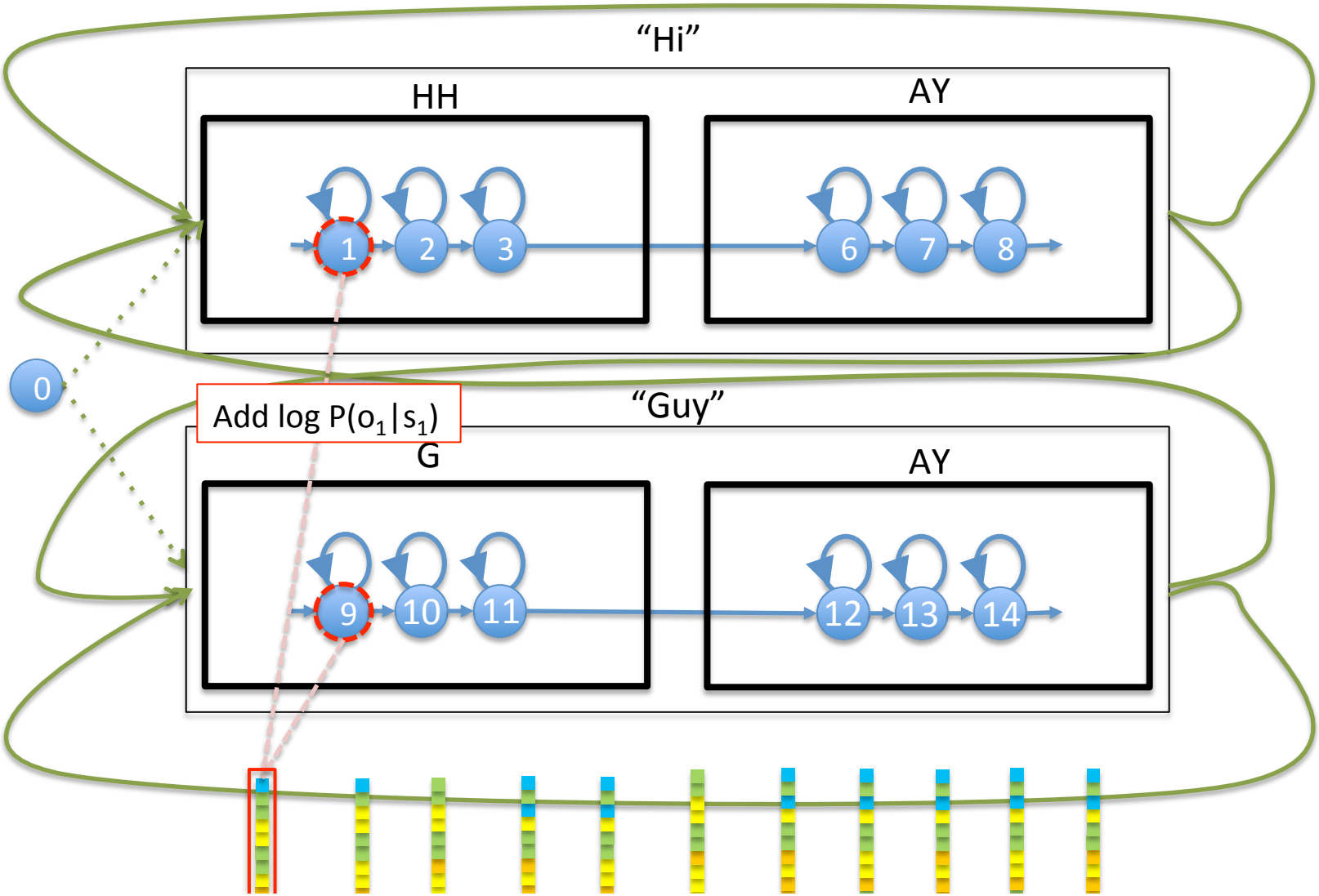
Decoder visualization

Example Beam search: keep top 2 paths; accumulate likelihood of O as we go.



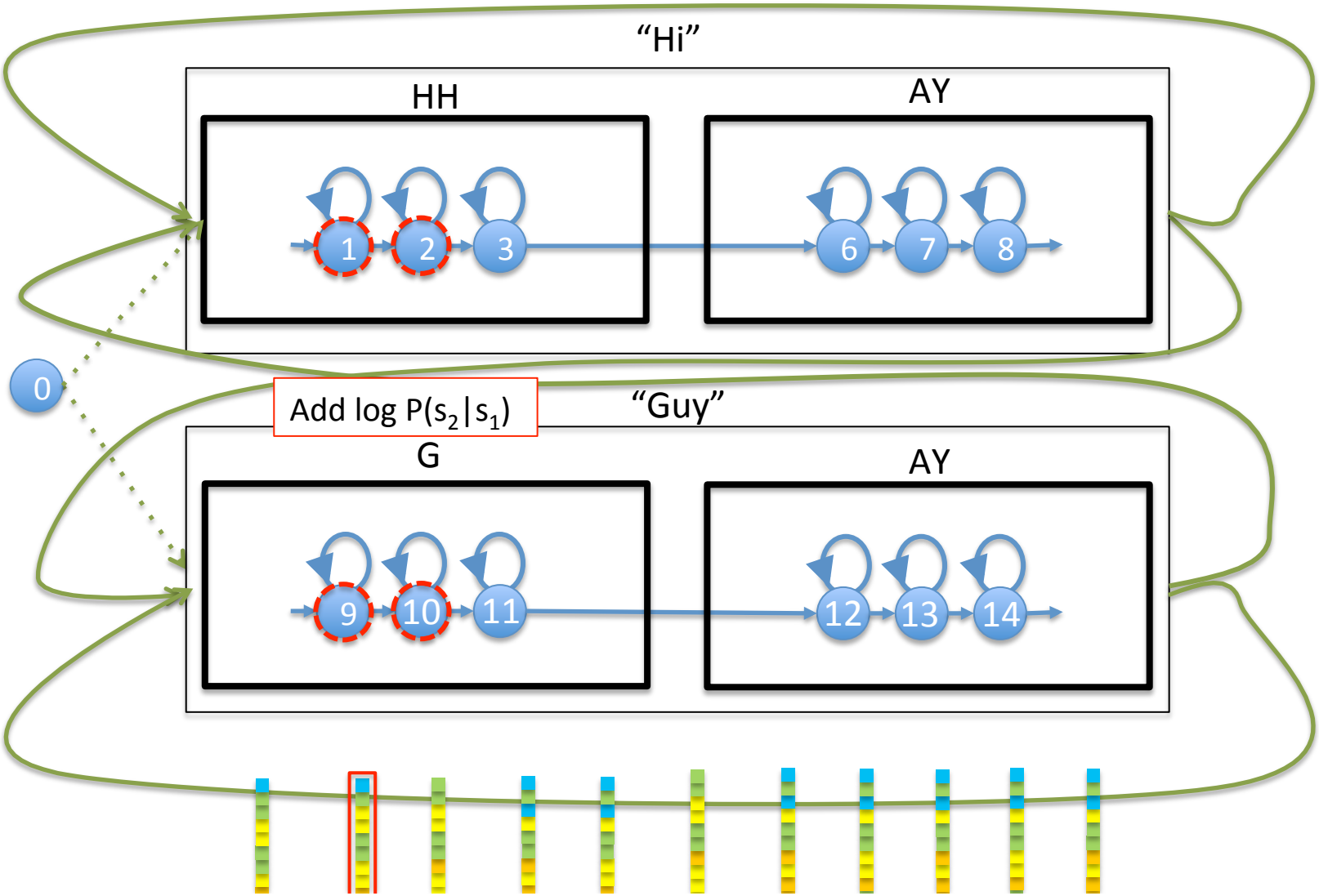
Decoder visualization

Example Beam search: keep top 2 paths; accumulate likelihood of O as we go.



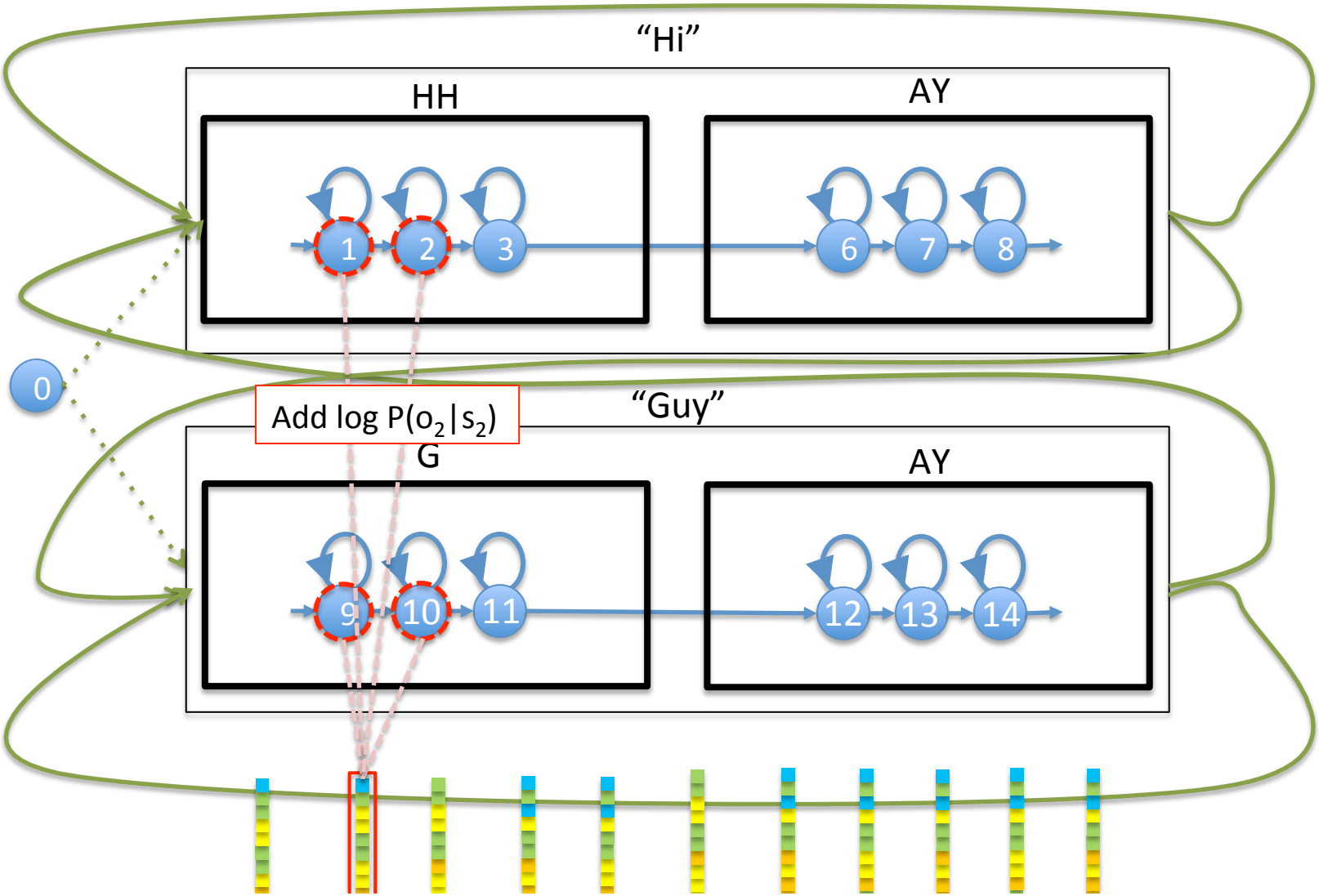
Decoder visualization

Example Beam search: keep top 2 paths; accumulate likelihood of O as we go.



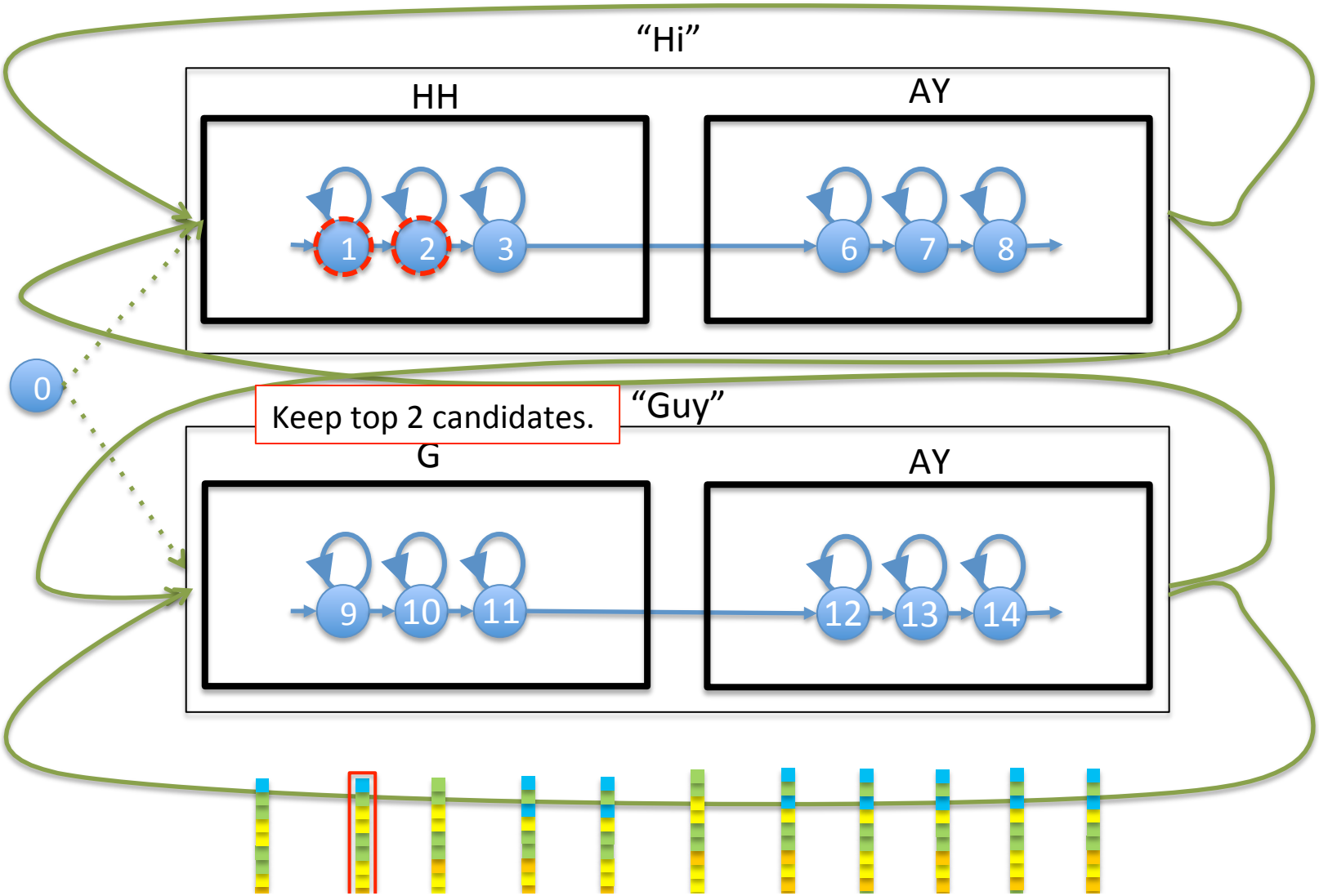
Decoder visualization

Example Beam search: keep top 2 paths; accumulate likelihood of O as we go.



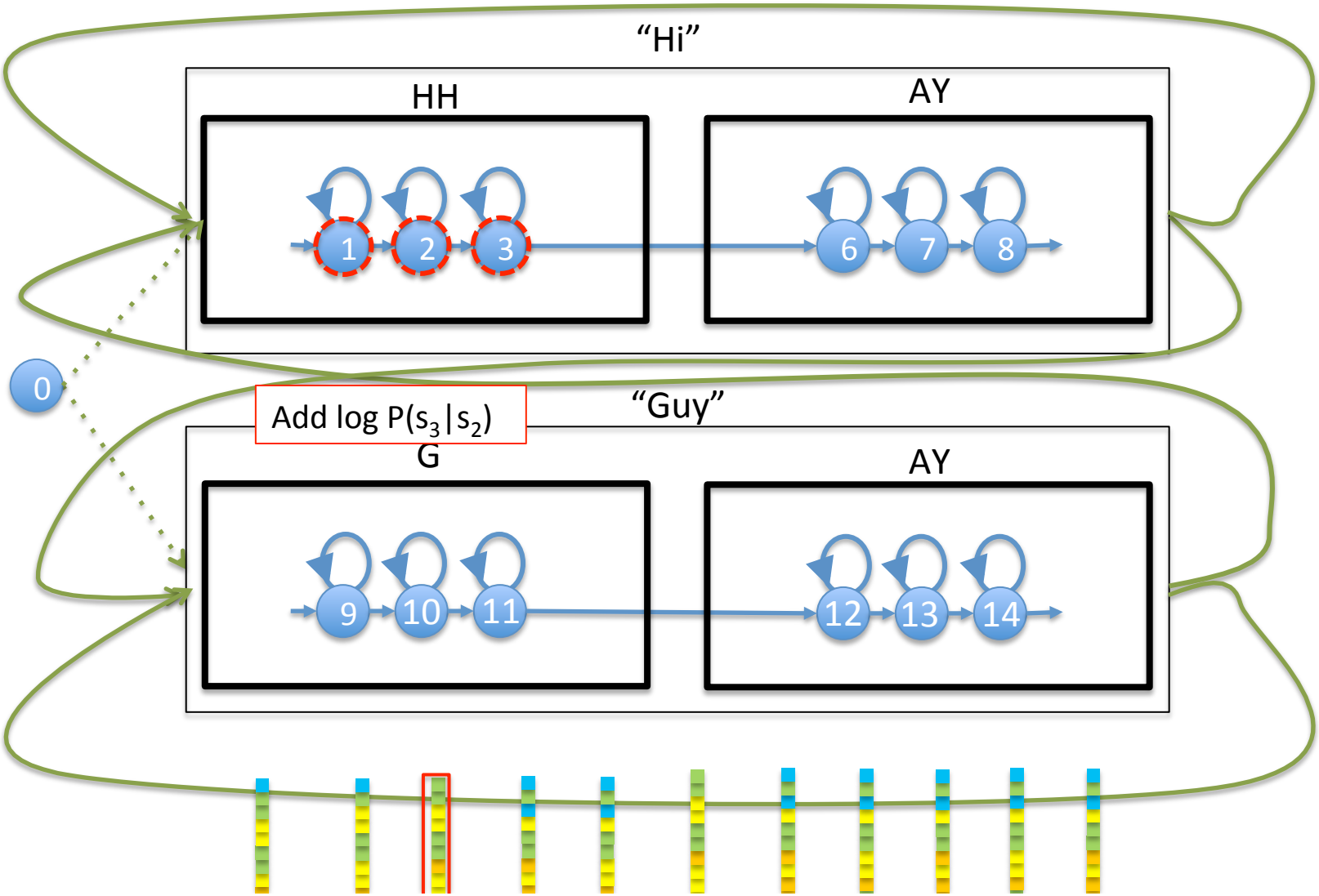
Decoder visualization

Example Beam search: keep top 2 paths; accumulate likelihood of O as we go.



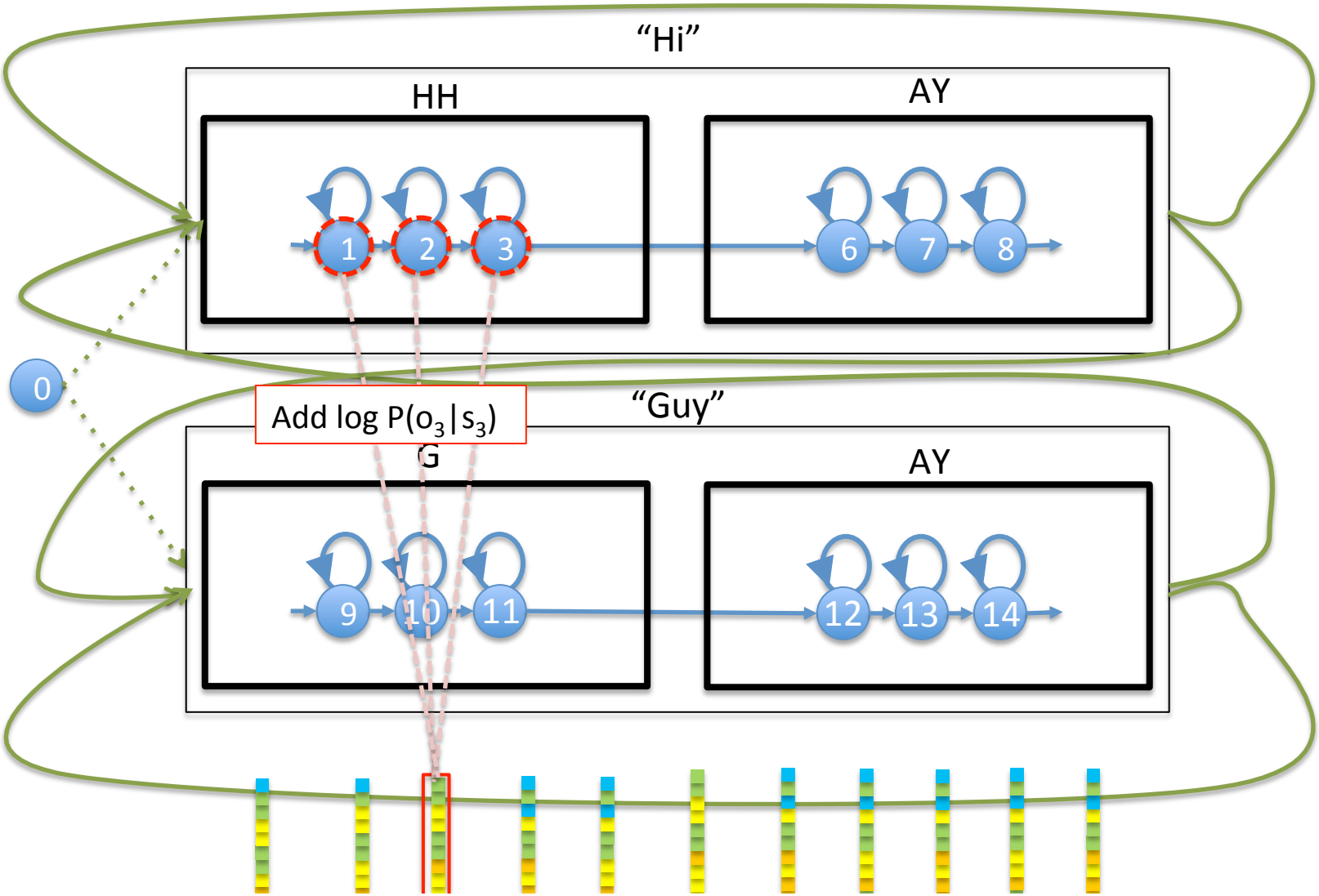
Decoder visualization

Example Beam search: keep top 2 paths; accumulate likelihood of O as we go.



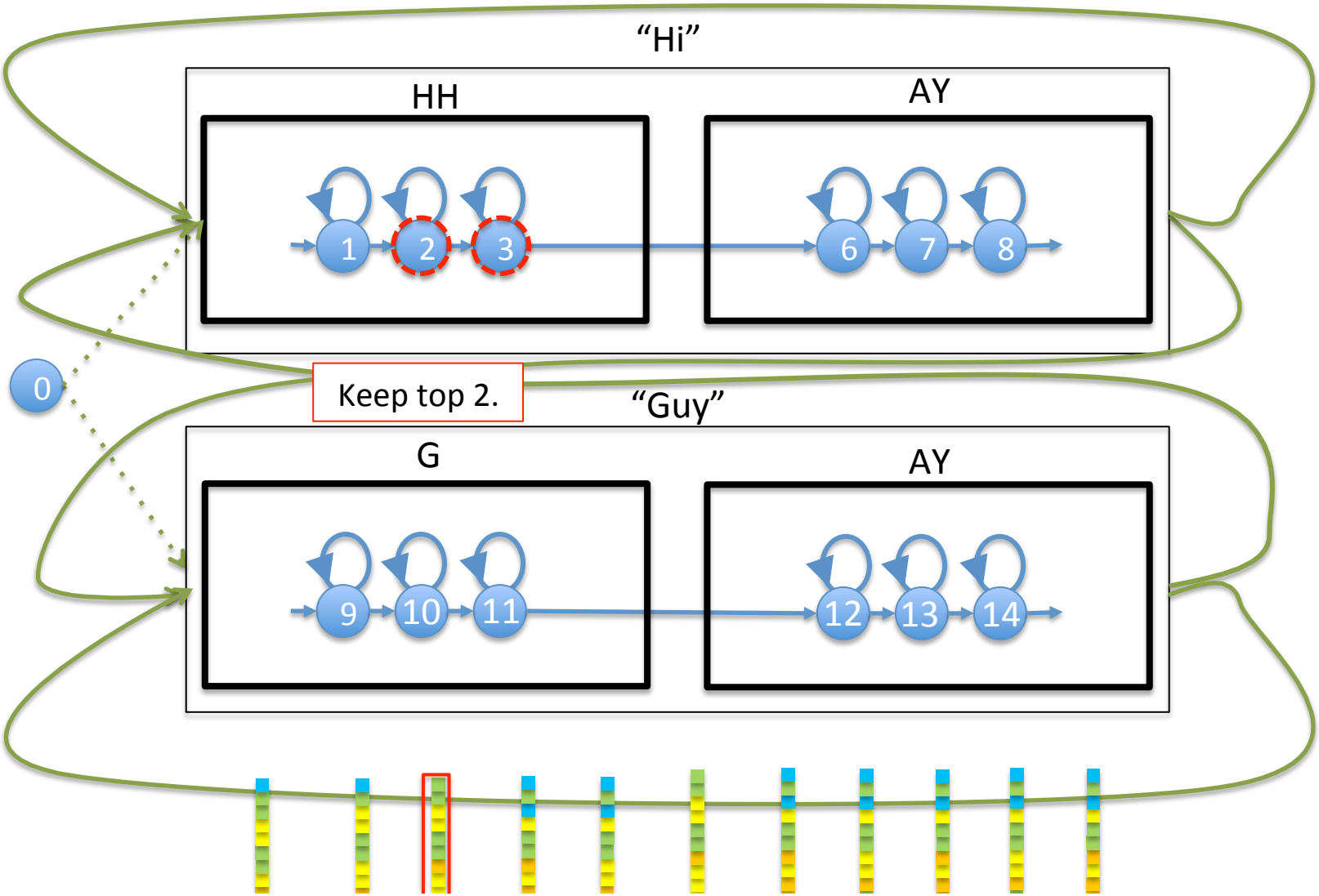
Decoder visualization

Example Beam search: keep top 2 paths; accumulate likelihood of O as we go.



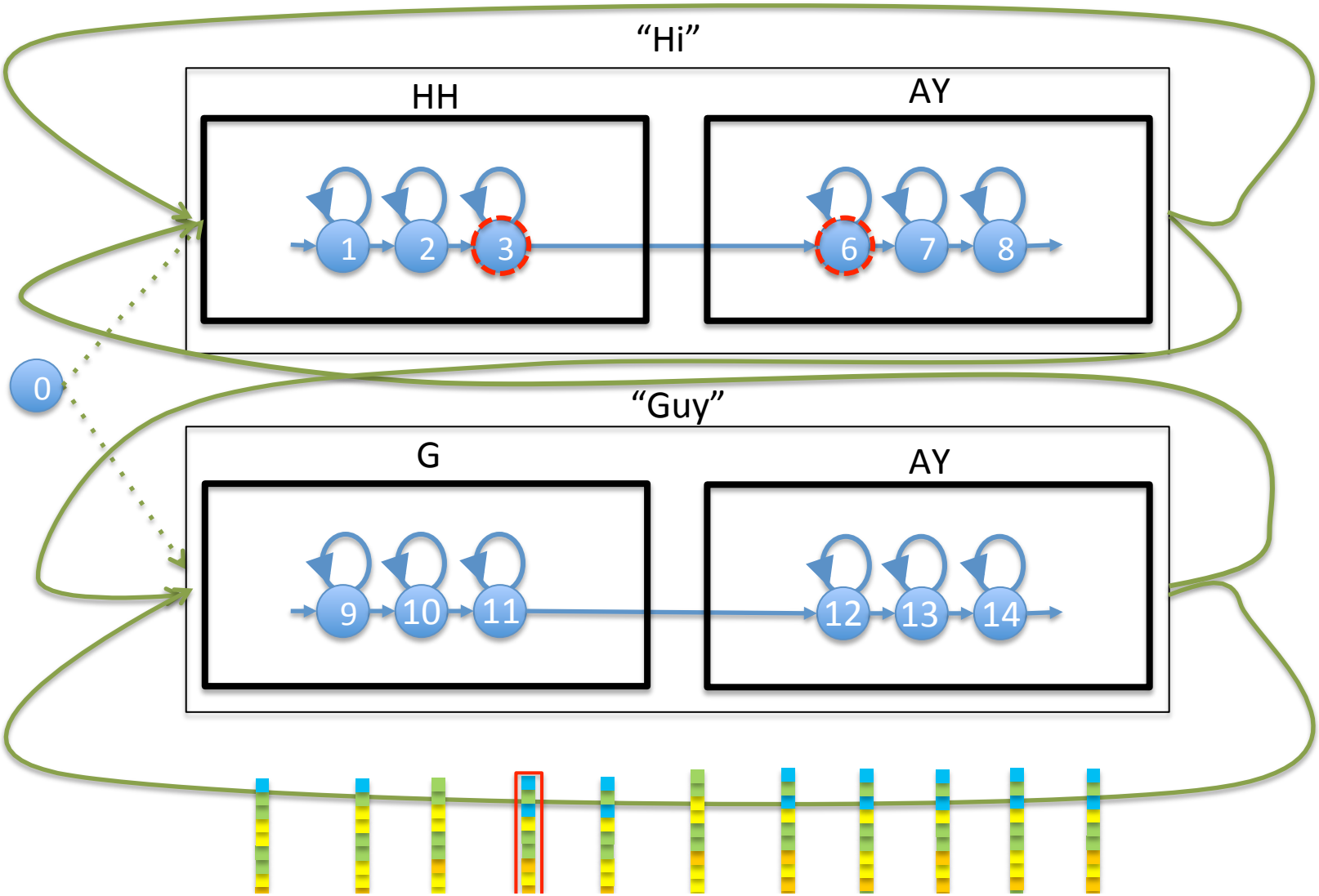
Decoder visualization

Example Beam search: keep top 2 paths; accumulate likelihood of O as we go.



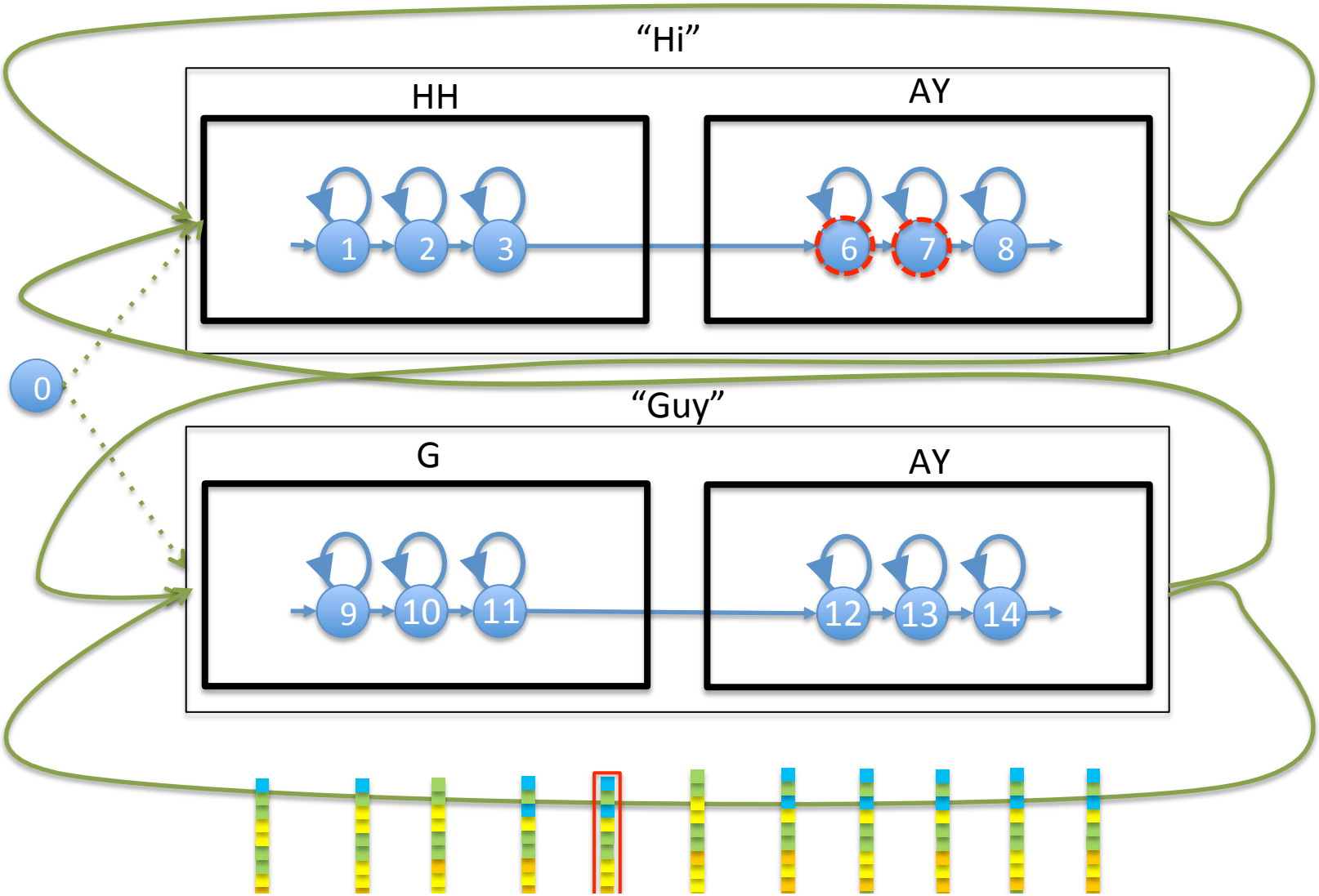
Decoder visualization

Example Beam search: keep top 2 paths; accumulate likelihood of O as we go.



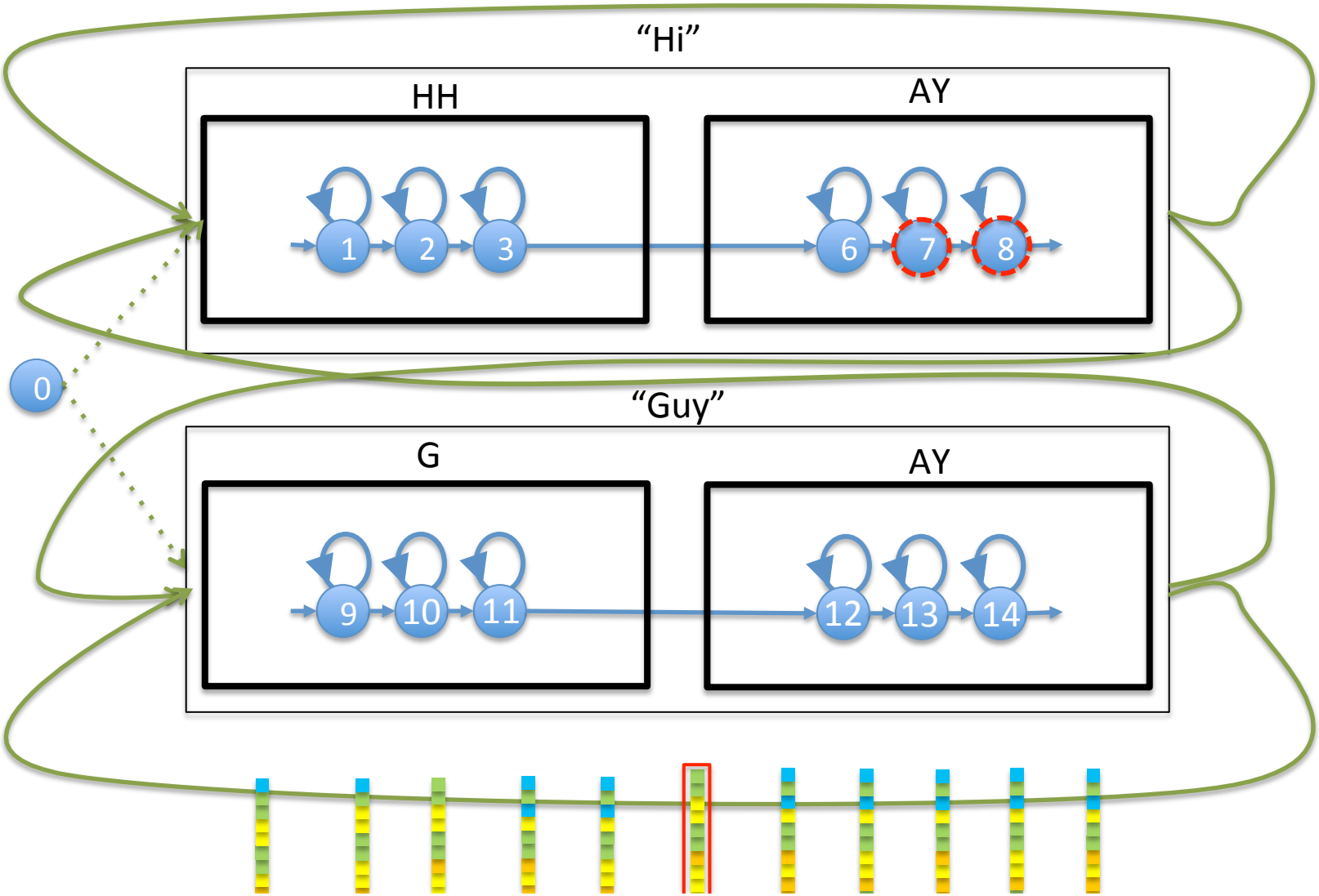
Decoder visualization

Example Beam search: keep top 2 paths; accumulate likelihood of O as we go.



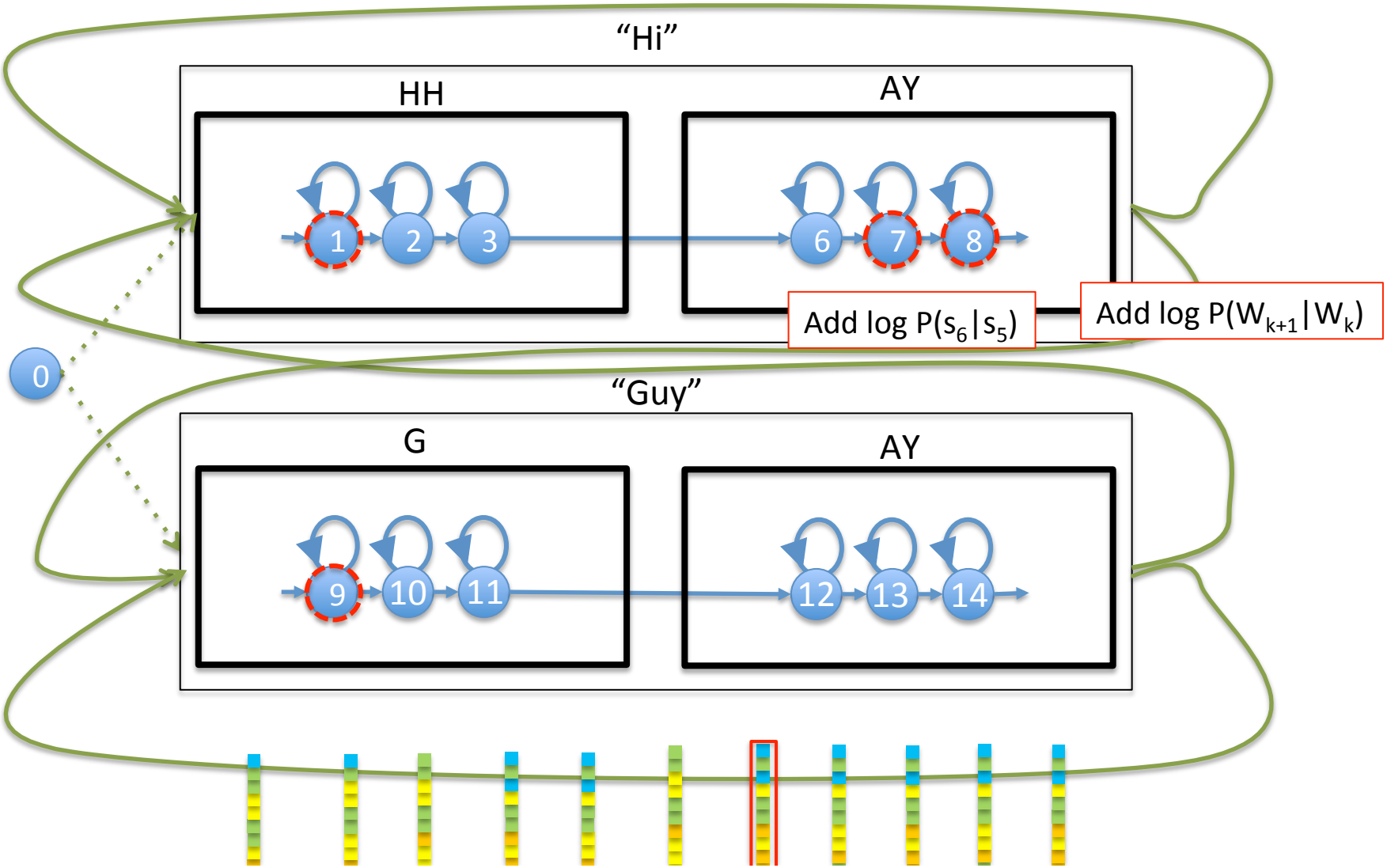
Decoder visualization

Example Beam search: keep top 2 paths; accumulate likelihood of O as we go.



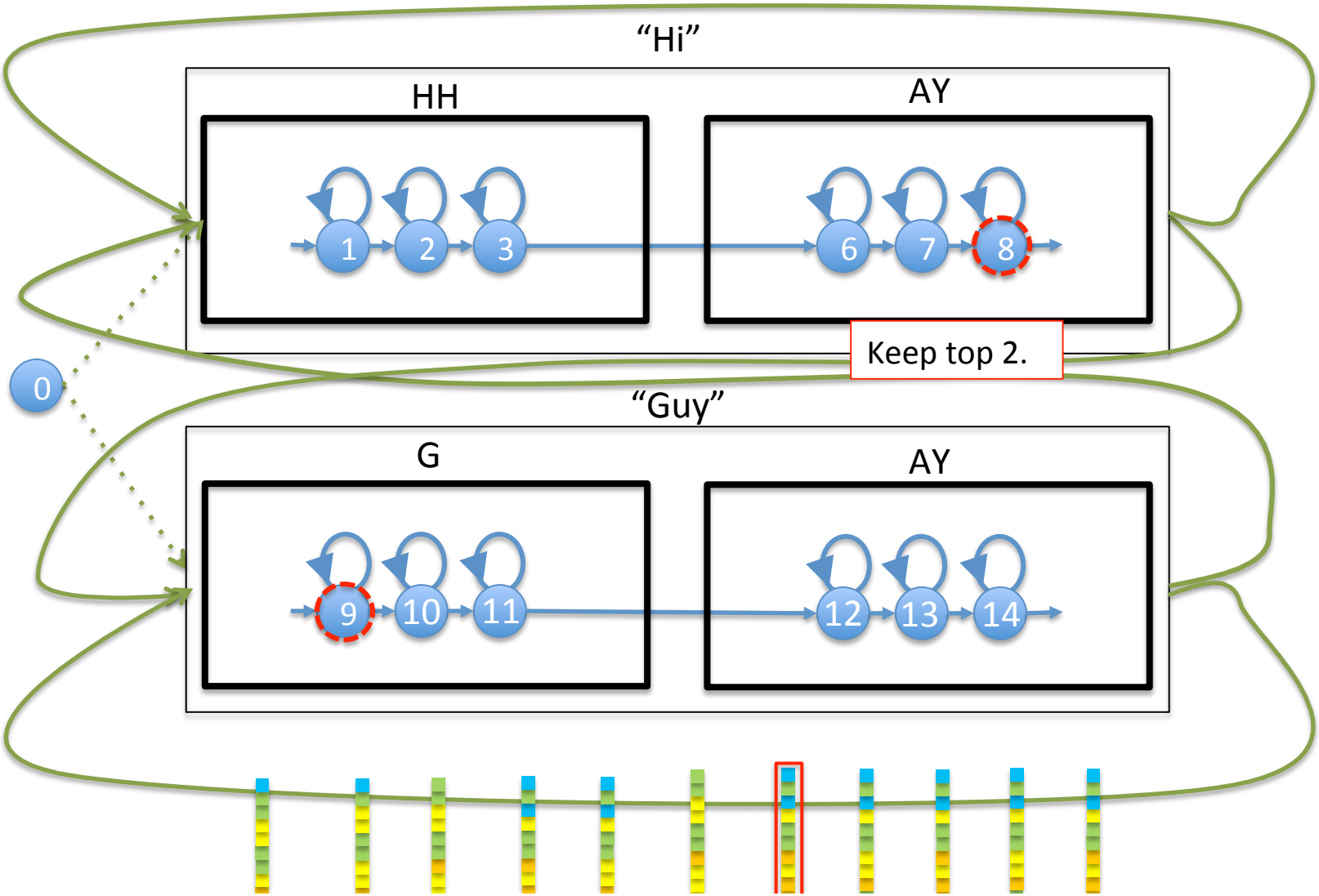
Decoder visualization

Example Beam search: keep top 2 paths; accumulate likelihood of O as we go.



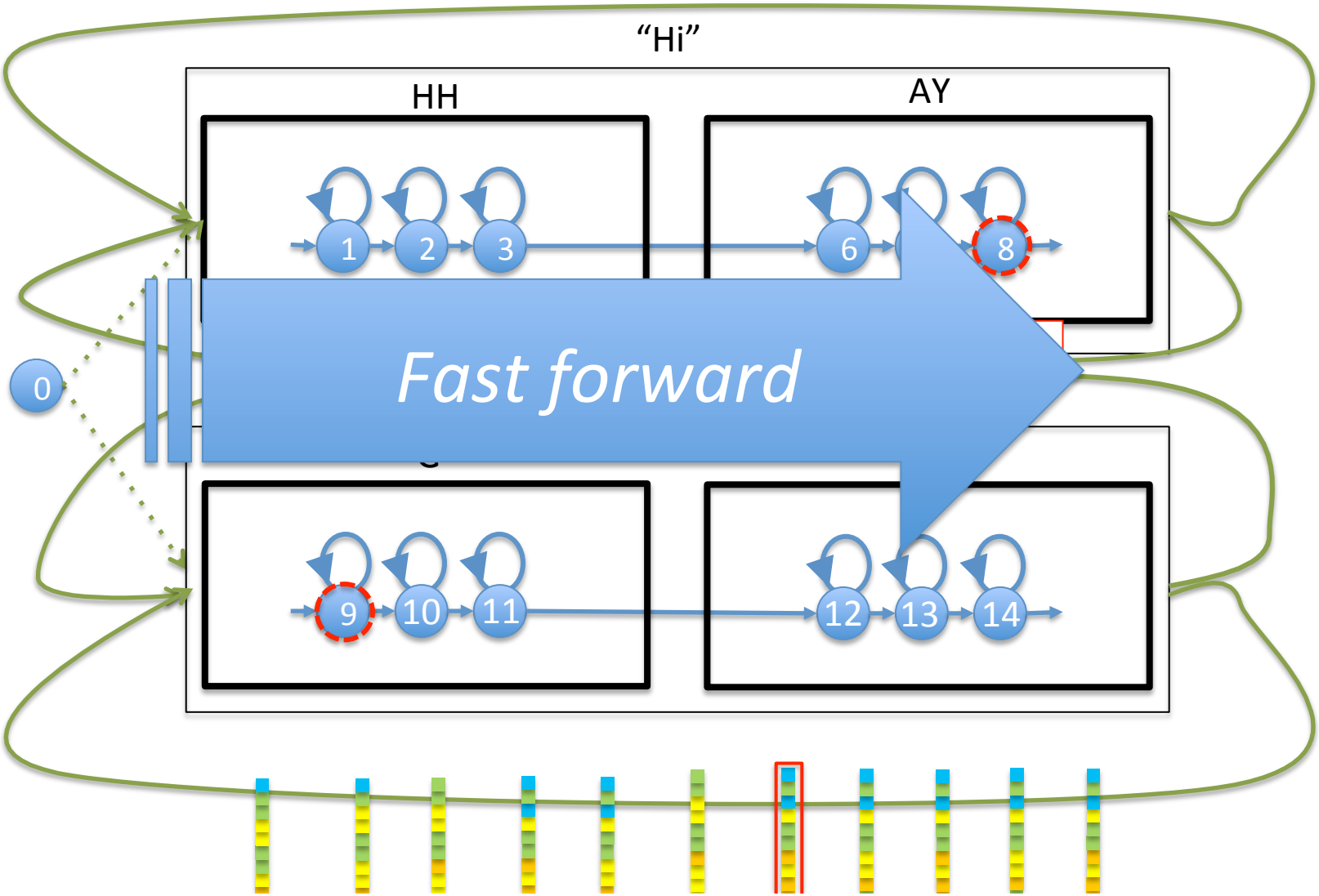
Decoder visualization

Example Beam search: keep top 2 paths; accumulate likelihood of O as we go.



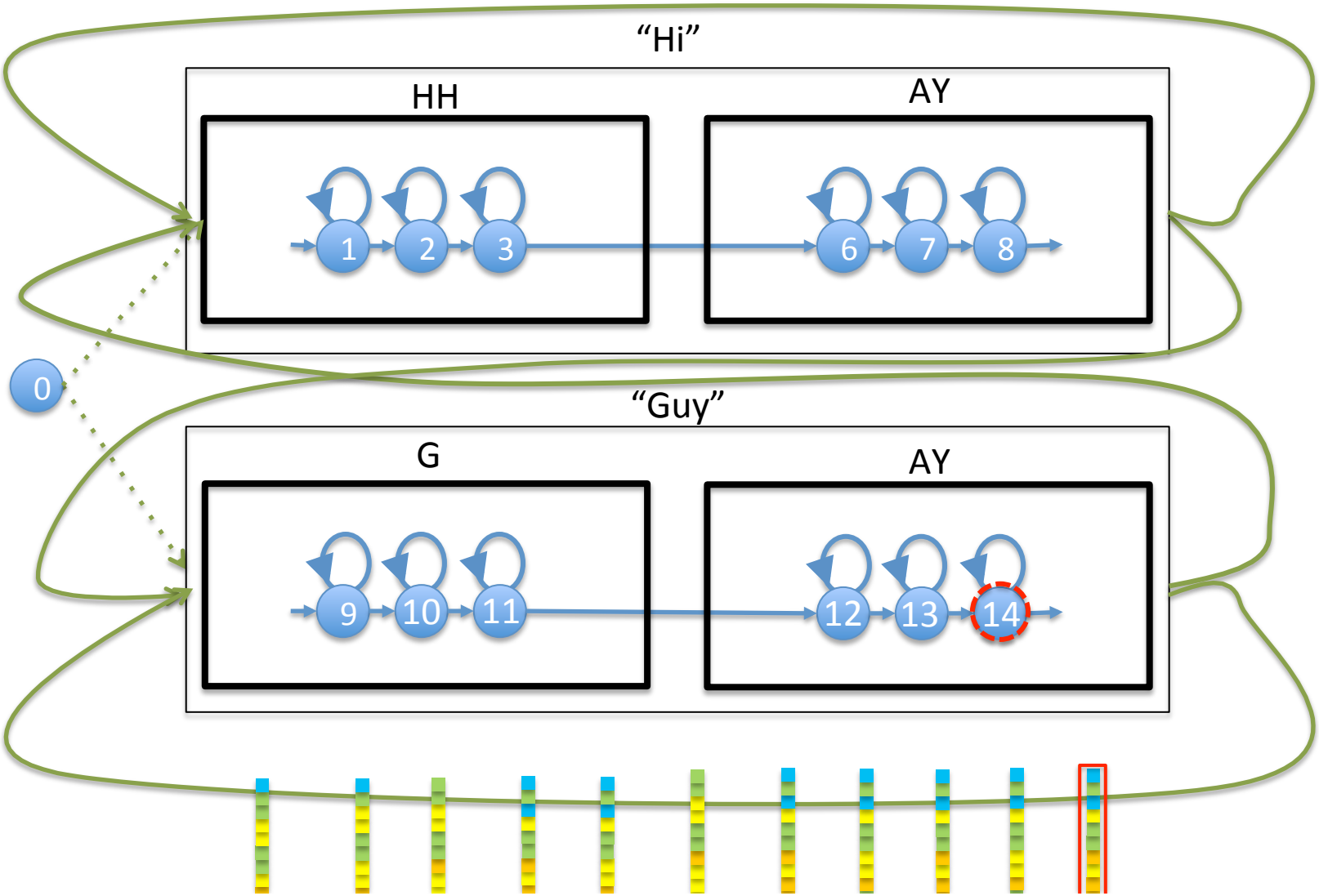
Decoder visualization

Example Beam search: keep top 2 paths; accumulate likelihood of O as we go.



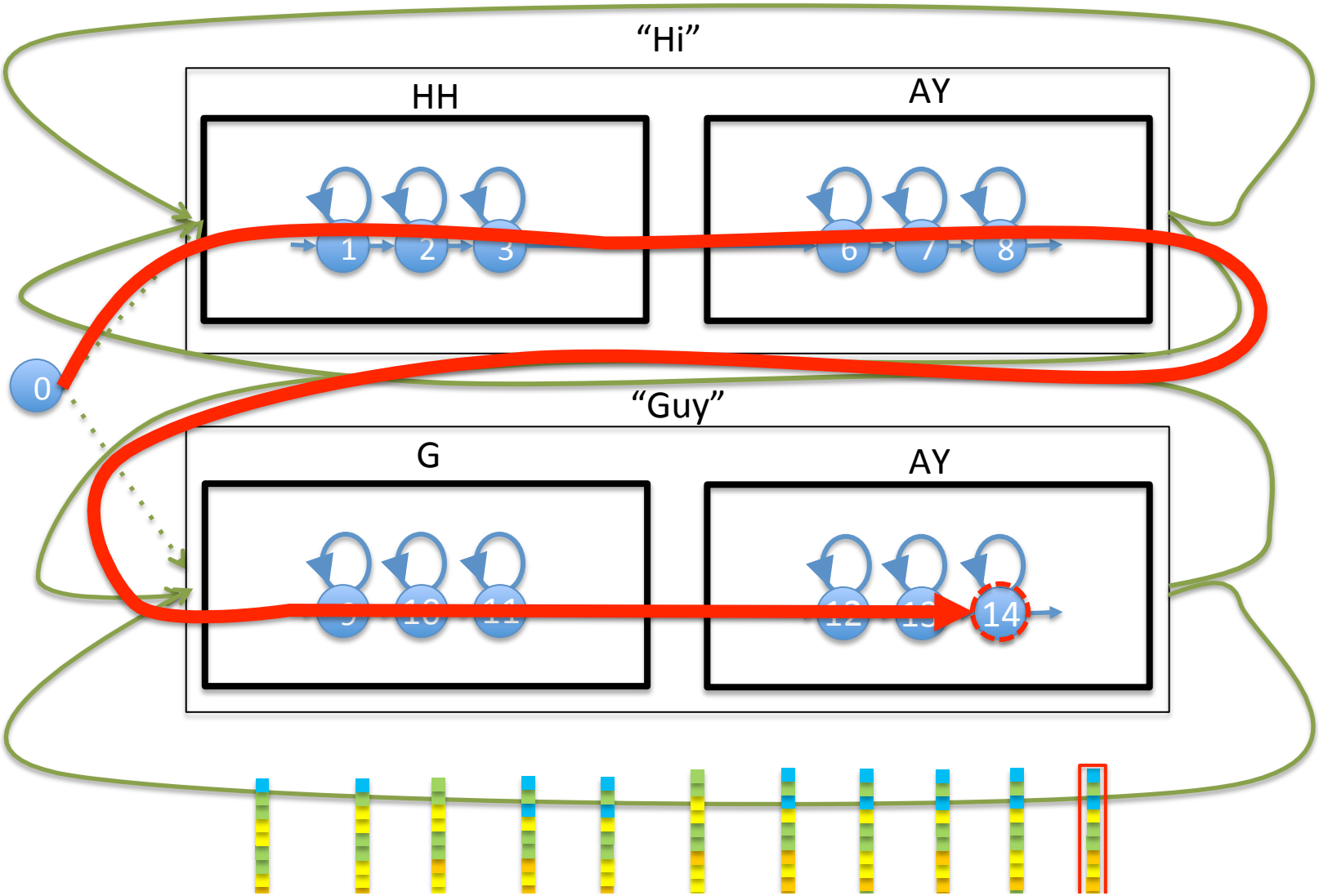
Decoder visualization

Example Beam search: keep top 2 paths; accumulate likelihood of O as we go.



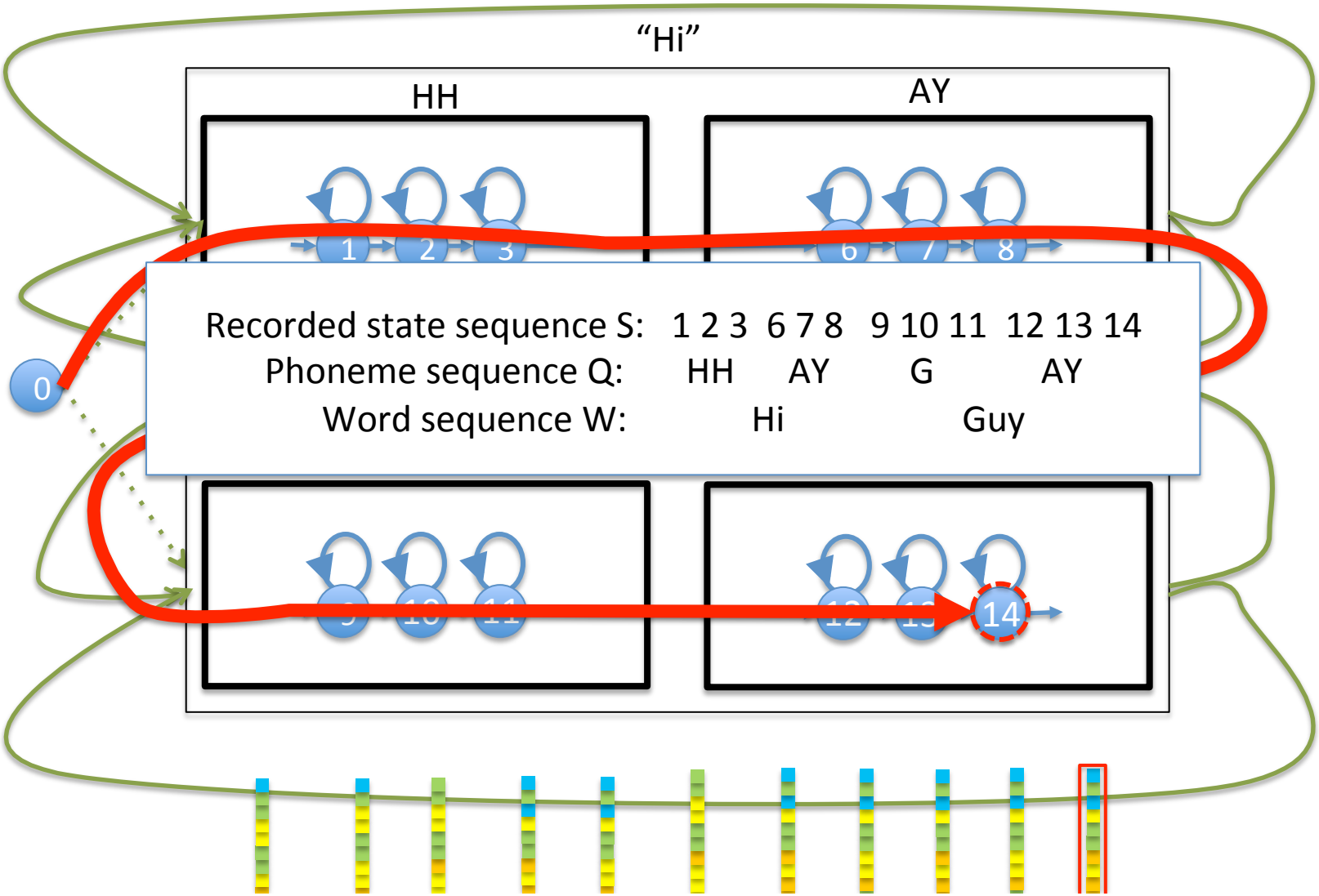
Decoder visualization

Example Beam search: keep top 2 paths; accumulate likelihood of O as we go.



Decoder visualization

Example Beam search: keep top 2 paths; accumulate likelihood of O as we go.



Is that all?

- No.
  - Highly simplified model here, but with all the major moving pieces.

- More components of real systems:
  - Phoneme models → Triphones (HH-AH-LL)
  - Normalization and noise filtering.
  - Speaker adaptation
    - "Vocal Tract Length Normalization"

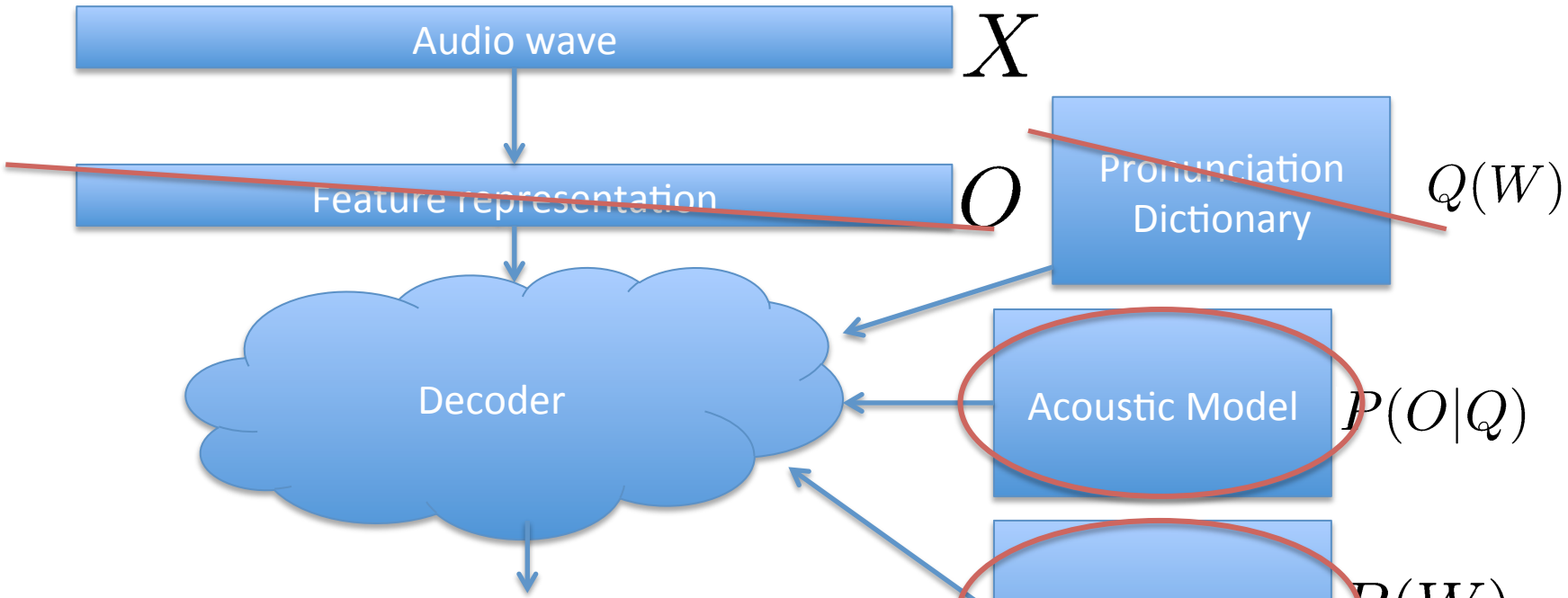
….

DEEP LEARNING!

Where can DL help?

Basic pipeline

- We'll just use a dictionary: only allow 1 pronunciation.

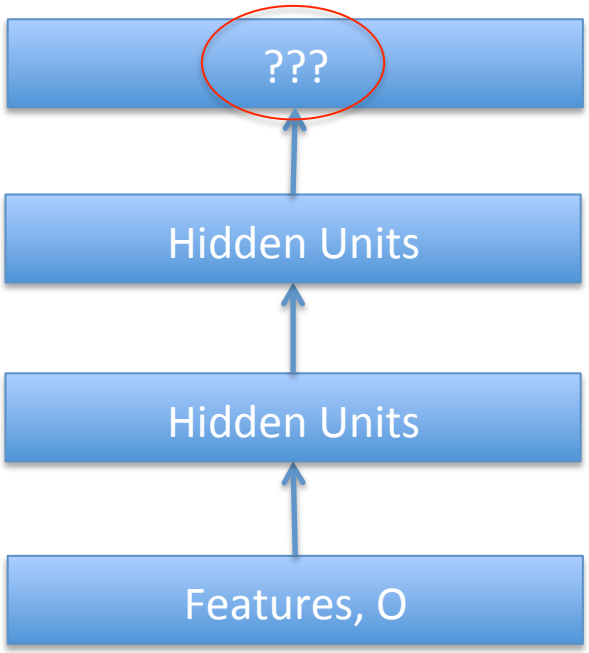


$$W^* = \underset{W}{\operatorname{argmax}}\ P(W|X)$$

$$= \underset{W}{\operatorname{argmax}}\ P(O|Q(W))P(W)$$

DNN acoustic models

- One classic improvement: HMM+DNN
  - Basic idea: Enhance P(O|Q) with neural network.
  - Still re-use HMM machinery to model sequences, words, etc.
    - So usually only aim to replace P(O|S)



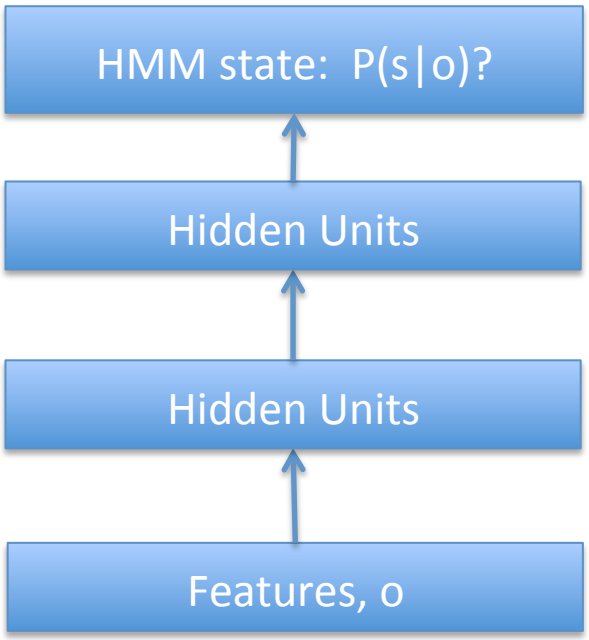
Usually we work with DNNs that are trained for a discriminative task:

Take in O, make a prediction.

But here trying to plug into generative model.

DNN acoustic models

- Clearly: DNN useful to model P(s|o) if we know the target for s.



Discrete label s → Predict with softmax neurons

$$a = Wh + b$$

$$y_i = P(s_i|o) = \exp(a_i) / \sum_j \exp(a_j)$$

Sigmoid or ReLu units.

Speech frame

DNN acoustic models

- Where do we get targets for P(s|o)?
  - Use standard pipeline to find most likely state sequence, S, for training utterance input, O.
    - Recall: We have word labels, so this is just "alignment".
    - Hack up into training pairs s_t and o_t for DNN.
  - Use a small carefully annotated training set to train DNN (bootstrap), re-run alignment, retrain.
  - Train to predict phonemes directly: P(q|o).
    - Phoneme-annotated data (bootstrap) is more plentiful.
    - Can rework HMMs so that emission/observation from "hidden" state is phoneme itself.

DNN acoustic models

- But we want observation model P(o|s) to integrate into HMM framework.
  - Bayes rule:

$$P(o|s) = P(s|o)P(o)/P(s)$$

$$P(o|s) \propto P(s|o)/P(s)$$

Introduces harmless constant into recognizer since *o* is given.

- Thus, normalize output of DNN by prior probability of state.
  - Just take empirical frequency of state in training data.
  - If you're getting great frame accuracy but poor word accuracy, this can be culprit. Especially when labels are skewed.

Early wins for DNN models

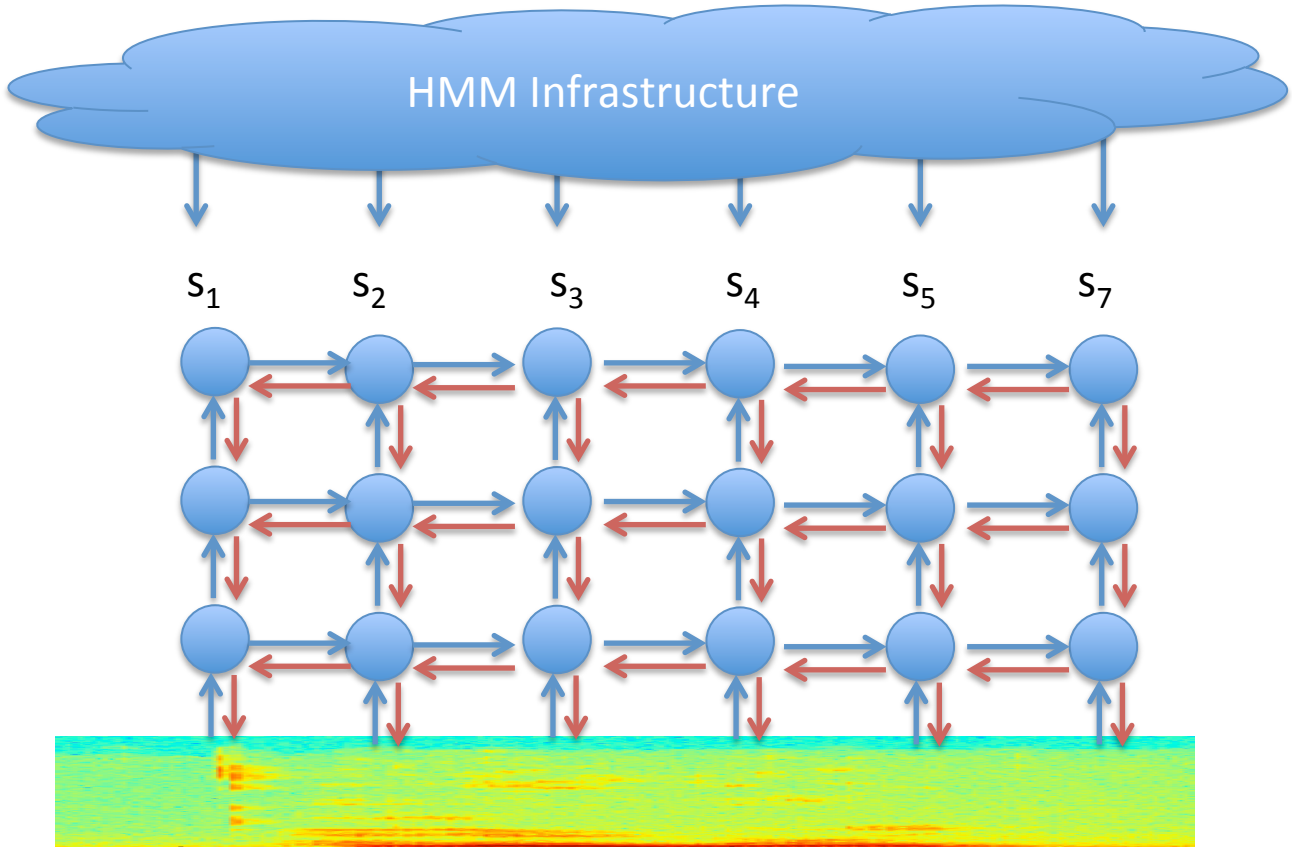
- From Dahl et al., ICASSP 2011:

DBN-HMM	5	from DBN-HMM	Triphone Senones	yes	71.8%	69.6%
ML GMM-HMM baseline					62.9%	60.4%
MMI GMM-HMM baseline					65.1%	62.8%
MPE GMM-HMM baseline					65.5%	63.8%
ML GMM-HMM baseline 2100 hours of training data (transcription is 90% accurate)					-	62.9% [13]

- ~10% relative improvement with DBN.
  - Later results improve with ReLu and Dropout.

More powerful acoustic models

- Can replace DNN with more powerful networks.
  - E.g., use large context, or recurrent network (RNN).



Rescoring

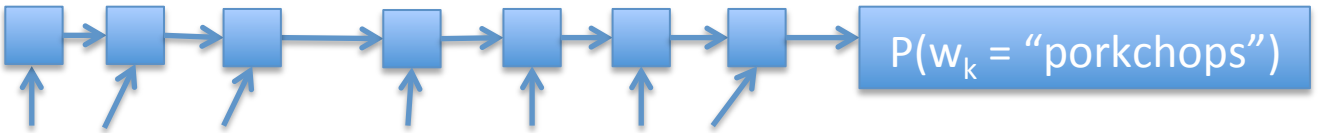
- Another place to plug in better algorithms:

  Systems usually produce N-best list.

  Use fancier algorithms to “rescore” (pick best)

Rescoring with Neural LM

- Example: train neural language model and rescore word transcriptions.
  - Cheap to evaluate $P(w_k | w_{k-1}, w_{k-2}, \ldots, w_1)$ NLM on many sentences.
  - In practice, often combine with N-gram trained from big corpora.



1. (-25.45) I'm a connoisseur looking for wine and porkchops. -24.45
2. (-26.32) I'm a connoisseur looking for wine and port shops. -23.45
3. …
4. …
5. …

TRAINING FROM UNSEGMENTED DATA WITH CTC

Complexity

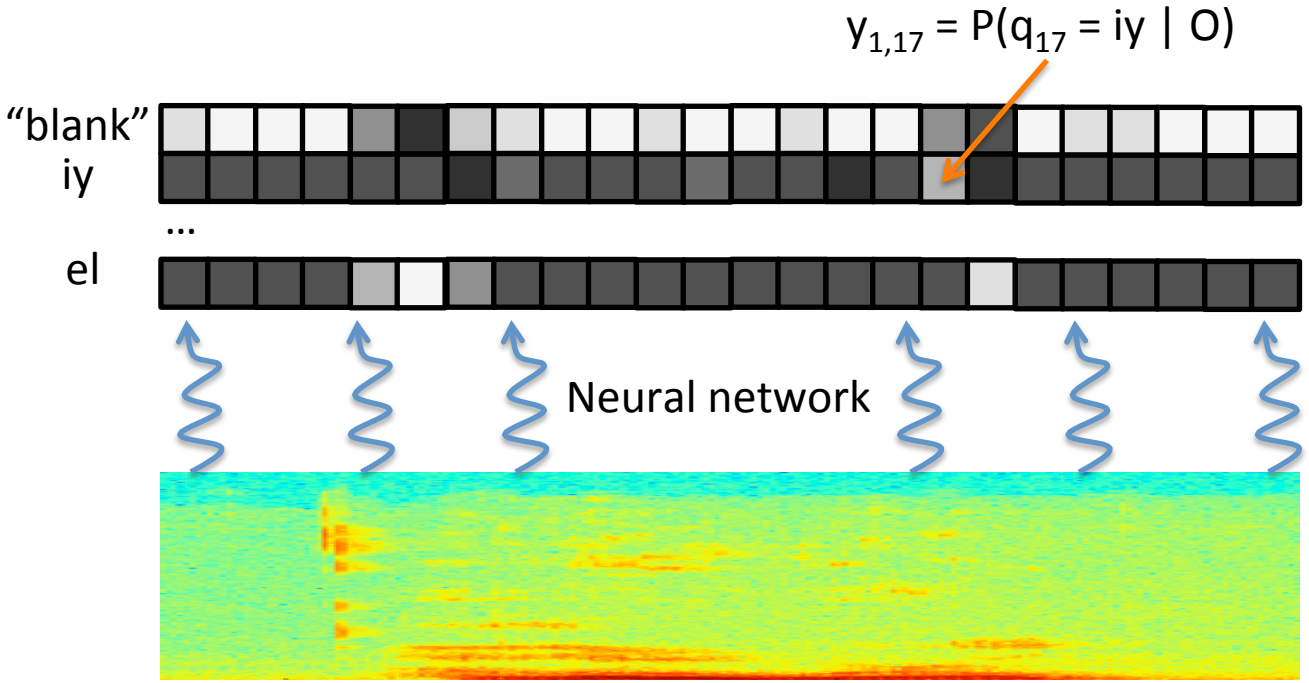
- Alignment and bootstrapping makes training cumbersome and error prone.

- What if we could train acoustic model without alignment step?
  - One proposal by Graves et al., ICML 2006:

  "Connectionist Temporal Classification" (CTC)

Network setup

- We create a neural network that outputs sequence of “probability vectors” $y_t = P(q_t|O)$ of same length as input.
- Assume that $P(Q|O) = \prod_t P(q_t|O)$.
- Allow q to take “blank” value so that Q can be same length as O.



Problem

- We don't know phoneme alignment with input, so can't train supervised directly.
  - Previously, we solved this by letting EM "guess" the alignment iteratively.
  - We want alignment to be *irrelevant*.

- Solution idea: introduce an operation that makes the transcription from P(q|o) "invariant" to misalignment.

Collapsing operator

- Suppose we start with decent predictions $y_t = P(q_t|O)$ from a neural network.
- Consider a string sampled from this distribution:

_ _ _ _ HH HH HH _ _ _ _ AH AH _ _ _ L L L _ _ OW _ _ _

- Make true "transcription" invariant by removing repeats, then blanks:

HH AH L OW == "Hello"

- Under this operation, these also map to "Hello":

HH HH _ _ _ _ _ AH AH _ _ _ L L L _ _ OW _ _ _ _ _ _ _ _ _

_ _HH AH _ _ _ L L L L L L L L _ _ OW OW OW _ _ _ _ _ _

Likelihood of a sequence

- Want to compute likelihood of a label sequence:

$$q_1\ q_2\ q_3\ q_4 = \text{HH}\quad \text{AH}\quad \text{L}\quad \text{OW}$$

- Sum over all possible transcriptions that collapse to the label string:

$$\begin{aligned} P(q_1\ q_2\ q_3\ q_4 | O) = P(&\ \_\ \_\ \_\ \_\ \text{HH HH HH}\ \_\ \_\ \_\ \_\ \text{AH AH}\ \_\ \_\ \_\ \text{L L L}\ \_\ \_\ \text{OW}\ \_\ \_\ \_\ ) \\ + P(&\ \text{HH HH}\ \_\ \_\ \_\ \_\ \_\ \text{AH AH}\ \_\ \_\ \_\ \text{L L L}\ \_\ \_\ \text{OW}\ \_\ \_\ \_\ \_\ \_\ \_\ \_\ \_\ ) \\ + P(&\ \_\ \_\text{HH AH}\ \_\ \_\ \_\ \text{L L L L L L L L L}\ \_\ \_\ \text{OW OW OW}\ \_\ \_\ \_\ \_\ \_\ \_\ ) \\ + &\ \ \ldots \end{aligned}$$

For fixed Q, Graves et al. give a forward-backward algorithm to compute this summation!

Training

- We want to do gradient ascent to maximize likelihood of a training label. We need:

$$\nabla_\theta P(Q|O) =$$
$$\nabla_\theta P(q_1 q_2 \ldots q_K|O) =$$

Training

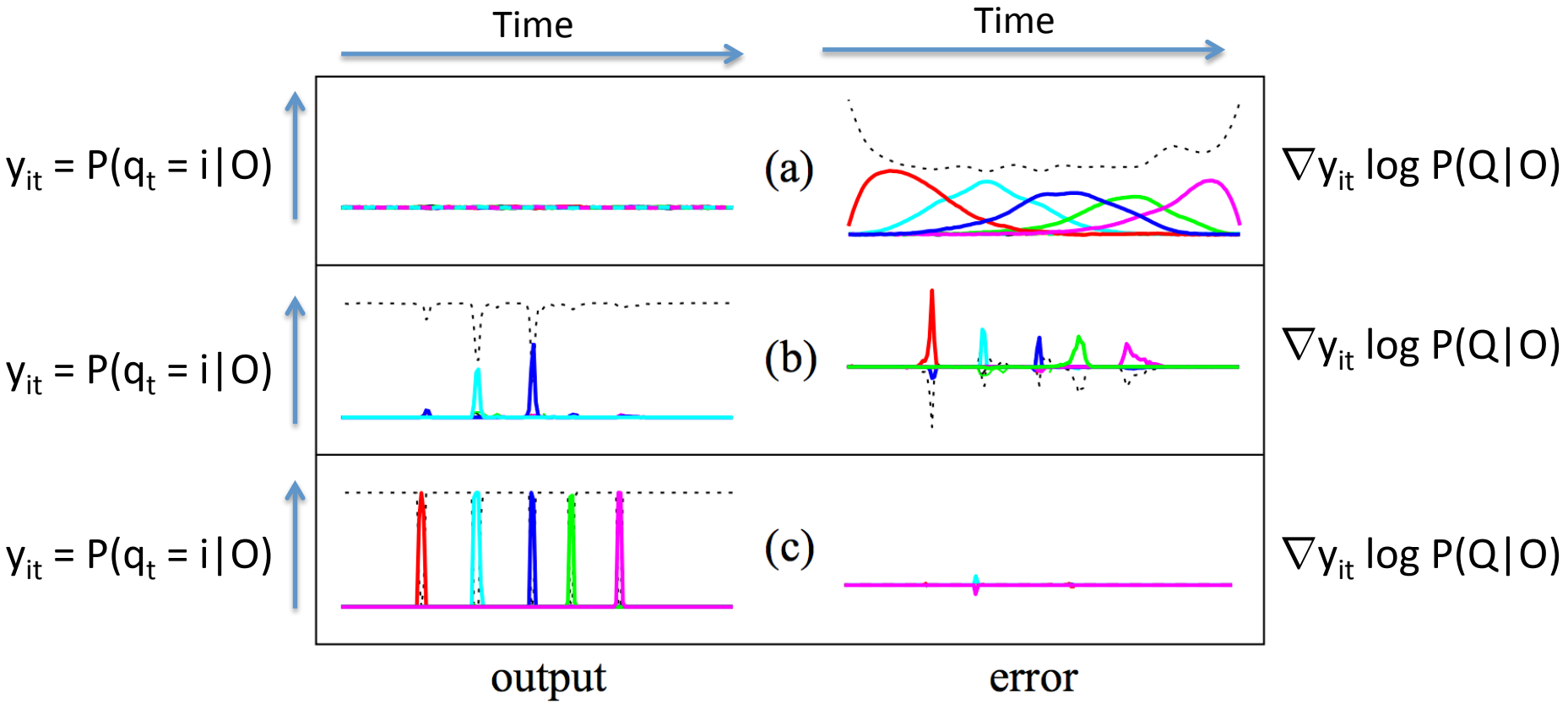
- We want to do gradient ascent to maximize likelihood of a training label. We need:

$$\nabla_\theta P(Q|O) =$$
$$\nabla_\theta P(q_1 q_2 \ldots q_K|O) =$$
$$\nabla_\theta \sum_{\hat{Q}:collapse(\hat{Q})=Q} P_{net}(\hat{q}_1 \hat{q}_2 \ldots \hat{q}_T|O)$$

Luckily, output of forward-backward algorithm can be used to compute gradient, including summation.

Training

- What happens?



[From Graves et al., 2006]

Decoding

- Given outputs, we still need to find most likely sequence and convert to words.
  - I.e., want to compute:

$$\begin{aligned} & \underset{Q}{\operatorname{argmax}}\ P(Q|O)) \\ = & \underset{Q}{\operatorname{argmax}} \sum_{\hat{Q}:collapse(\hat{Q})=Q} P_{net}(\hat{q}_1\hat{q}_2\ldots\hat{q}_T|O) \end{aligned}$$

Decoding

- Quick and dirty solution:

$$\begin{aligned} &\operatorname*{argmax}_{Q} P(Q|O)) \\ &\approx \text{collapse}(\operatorname*{argmax}_{\hat{Q}} P_{net}(\hat{q}_1 \hat{q}_2 \ldots \hat{q}_T|O)) \end{aligned}$$

- Not guaranteed to be best sequence, but useful sanity-check.

Decoding

- Alternatively, resort to beam search over Q, as with traditional systems.
  - Can also incorporate LM score at this point as with traditional systems.
    - See, Hannun, Maas, Jurafsky & Ng, 2014.
- Or: don't bother to decode and just use P(Q|O) to rescore N-best from traditional baseline.
  - See, e.g., Graves & Jaitly, 2014.

End-to-end learning

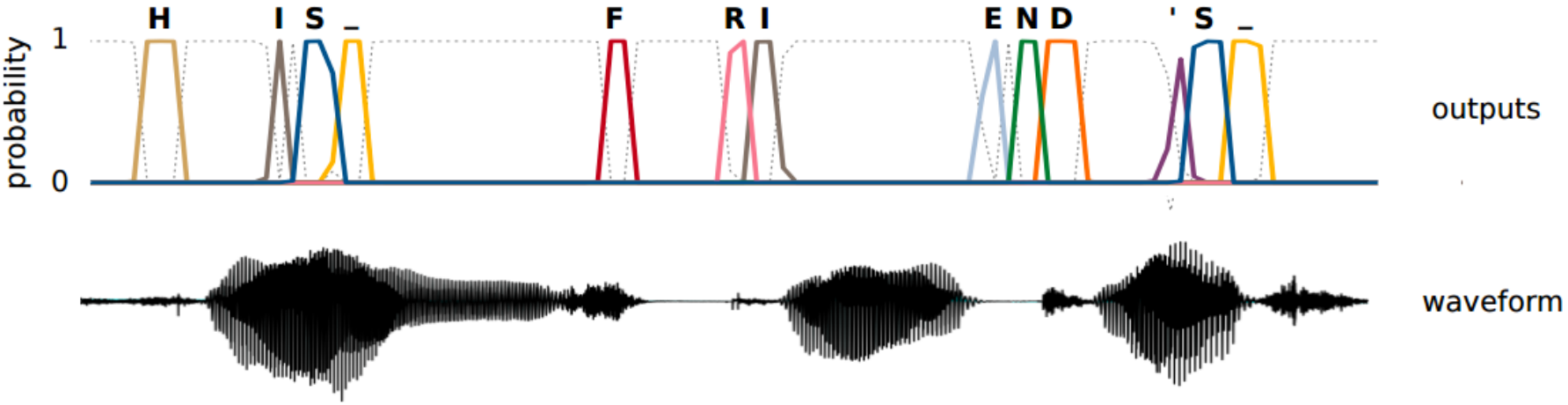
- *No fundamental reason we must use phonemes.*

➢ Jettison HMM infrastructure for transcribing phonemes/words, and use CTC to transcribe directly to characters/graphemes.
  - Let neural network (RNN) do all the work.
    - See, e.g., Graves & Jaitly, ICML 2014.
  - Still probably want LM.

- No major changes to training algorithm!
  - But needs a lot of training data / large models to compete with traditional systems.
  - Yet *much* simpler to build (Hannun et al., 2014).

End-to-end learning

- Graves & Jaitly, 2014:



- Caveat: character transcription leads to "hearing errors" cropping up.

target: TO ILLUSTRATE THE POINT
output: TWO ALSTRAIT THE POINT

- These can be hard to fix with language model because words look very different though sound the same.

Example transcriptions

- End-to-end networks can still work well on their own, but LM is still needed.

Max Decoding:	LM Decoding:
what is the weather like in bostin right now	what is the weather like in boston right now
prime miniter nerenr modi	prime minister narendra modi
arther n tickets for the game	are there any tickets for the game

From Hannun et al., 2014.

Conclusion

- Traditional HMM-DNN hybrid speech system still very common in the wild.
  - Multiple places for DL to plug in and make improvements.

- More recent trend: replace with more end-to-end DL approach.
  - Speech works *significantly* better today due to DL.
  - Next wave of DL systems should be even better as end-to-end methods supplant engineering.

Thank you

Special thanks to Awni Hannun for his help checking these slides.

References

- Gales and Young. “The Application of Hidden Markov Models in Speech Recognition” Foundations and Trends in Signal Processing, 2008.
- Jurafsky and Martin. “Speech and Language Processing”. Prentice Hall, 2000.
- Bourlard and Morgan. “CONNECTIONIST SPEECH RECOGNITION: A Hybrid Approach”. Kluwer Publishing, 1994.
- A Graves, S Fernández, F Gomez, J Schmidhuber. “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks.” ICML, 2006.
- Dahl, Yu, Deng, Acero, “Large Vocabulary Continuous Speech Recognition with Context-Dependent DBN-HMMs”. ICASSP, 2011
- Hannun, Maas, Jurafsky, Ng. “First-Pass Large Vocabulary Continuous Speech Recognition using Bi-Directional Recurrent DNNs” ArXiv:1408.2873
- Hannun, et al. “Deep Speech: Scaling up end-to-end speech recognition”. ArXiv:1412.5567
- H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech", J. Acoust. Soc. Am., vol. 87, no. 4, pp. 1738-1752, Apr. 1990.
- H. Hermansky and N. Morgan, "RASTA processing of speech", IEEE Trans. on Speech and Audio Proc., vol. 2, no. 4, pp. 578-589, Oct. 1994.
- H. Schwenk, “Continuous space language models”, 2007.