ANLP

08 - HMMs (sequences, part II)

David Schlangen
University of Potsdam, MSc Cognitive Systems
Winter 2019 / 2020

where are we?

- words, and how to represent them
- sequences of words, modelled with n-gram models (generative model)
- classification of objects, with small label set; weighting of features of object
  (generative classifier: Naive Bayes; discriminative classifiers: logistic regression, SVMs)

recap classification

- feature function: extracts aspects of object that matter for classification. Turns object into vector.
- generative classifier: models joint of object and label ($P(x,y)$), uses Bayes' rule to get at $P(y|x)$
- discriminative classifier: directly learns $P(y|x)$
- learning as iterative adaptation of weights, to make output more than what is desired for given input
  - increase (decrease) weights for active input features, if it should have said yes (no) but didn't / if it didn't say yes (no) enough
  - difference in loss function: logistic is soft, SVM and perceptron use hinge losses (SVM with "safety distance")

recap classification

- multiclass classification: represent input separately for each class, concatenated into one vector (= learn different weights for each class)

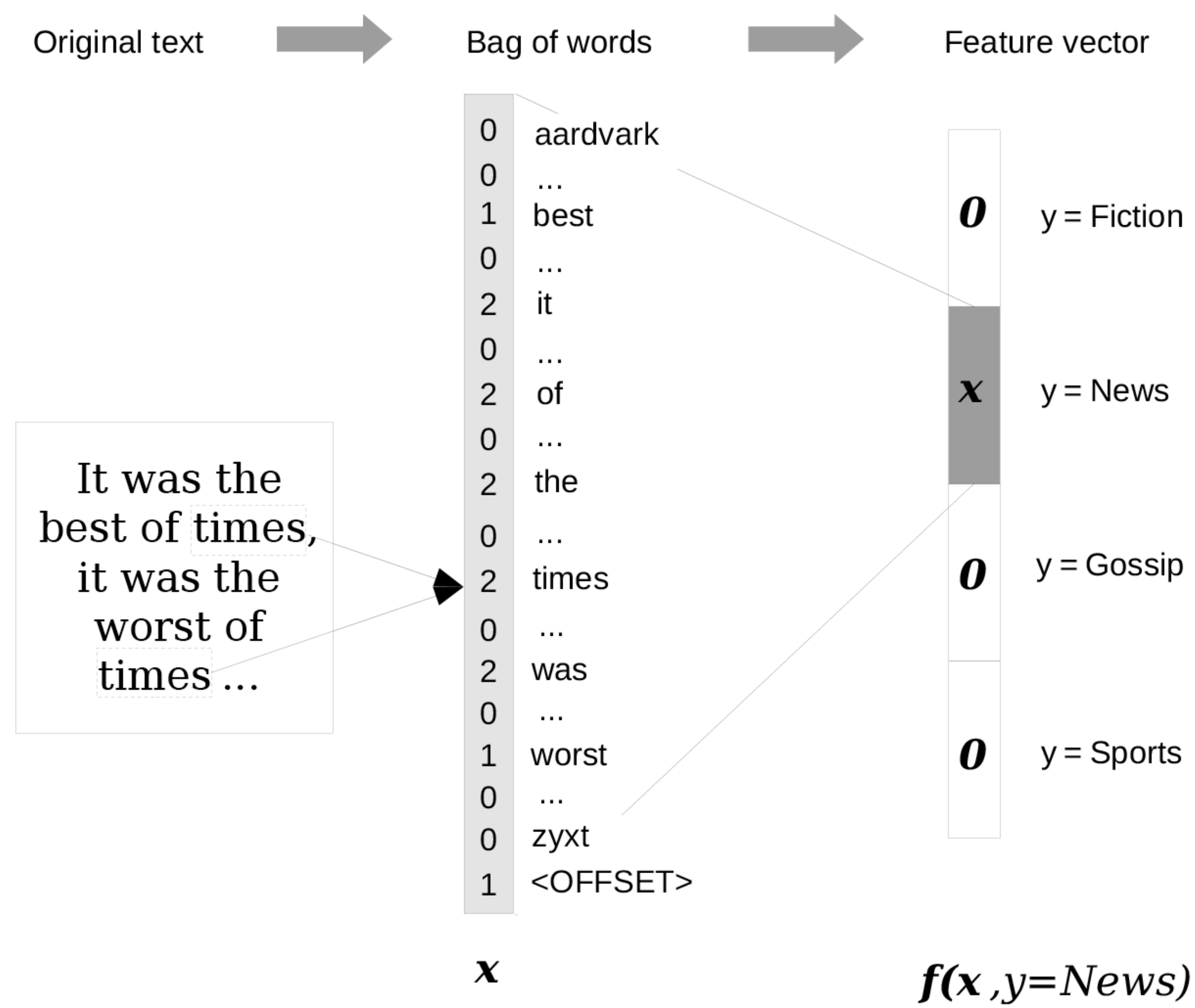


Figure 2.1
The bag-of-words and feature vector representations, for a hypothetical text classification task.

where are we?

- words, and how to represent them
- sequences of words, modelled with n-gram models (generative model)
- classification of objects, with small label set; weighting of features of object
  (generative classifier: Naive Bayes; discriminative classifiers: logistic regression, SVMs)
- now: classification of structured objects, with structured labels (generative: HMMs, discriminative: CRFs)

POS tagging

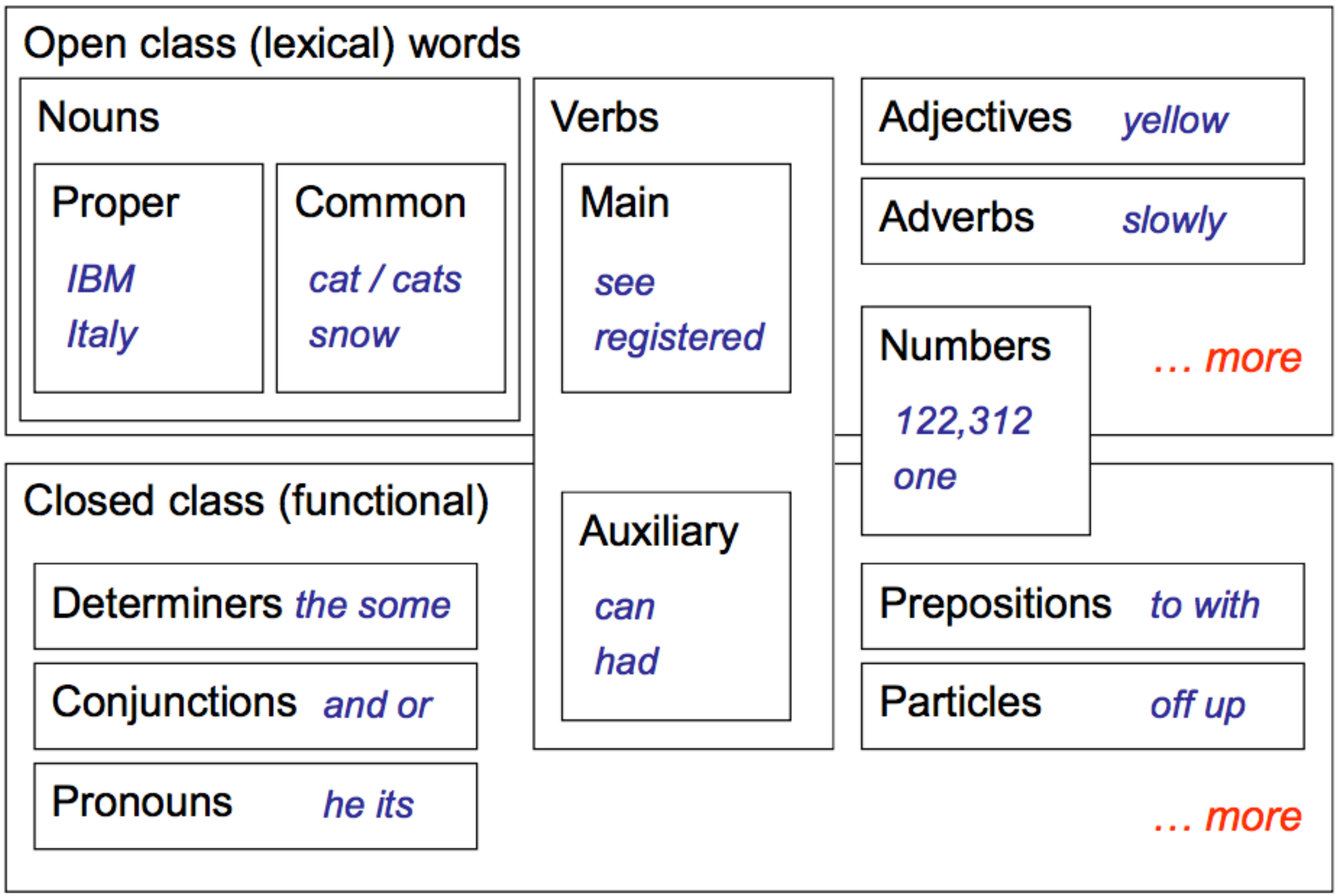
PP VBD DT NN IN NNS
I ate the spaghetti with chopsticks

I ate the spaghetti with meatballs

POS tagging as classification

- can we set this up as classification of full sequences, with many labels?
- represent sentence to tag as bag of words
- treat “PP-VBD-DT-NN-IN-NNS” as one label
- how many labels would we need?

POS Tagging



Slide credit: Dan Klein

Penn Treebank POS tags

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	“to”	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential ‘there’	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	*$*
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	*#*
PDT	Predeterminer	*all, both*	“	Left quote	(‘ or “)
POS	Possessive ending	*’s*	”	Right quote	(’ or ”)
PP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PP$	Possessive pronoun	*your, one’s*	)	Right parenthesis	( ],), }, >)
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	(: ; ... – -)
RP	Particle	*up, off*			

POS tagging as classification

- can we set this up as classification of full sequences, with many labels?
- represent sentence to tag as bag of words
- treat “PP-VBD-DT-NN-IN-NNS” as one label
- how many labels would we need?
- $|T|^n$, for tagset T and sequence of lenght n, for all n that we want to allow…

Multiclass POS tagging

- or maybe we just look at a single word and a bit of context?

- Classify *blocks* as one of 36 POS tags
- Example *x*: sentence with a word (in this case, *blocks*) highlighted
- Extract features with respect to this word:

the router *blocks* *the packets*

NNS
VBZ
NN
DT
...

$f(x, y=\text{VBZ})$ = I[curr_word=*blocks* & tag = VBZ],
I[prev_word=*router* & tag = VBZ]
I[next_word=*the* & tag = VBZ]
I[curr_suffix=*s* & tag = VBZ]

not saying that *the* is tagged as VBZ! saying that *the* follows the VBZ word

- That's a lot of features…
let's use specialised methods for sequence labeling!

POS tagging, what for?

Linguistic Structures

- Language is tree-structured

I ate the spaghetti with meatballs

I ate the spaghetti with chopsticks

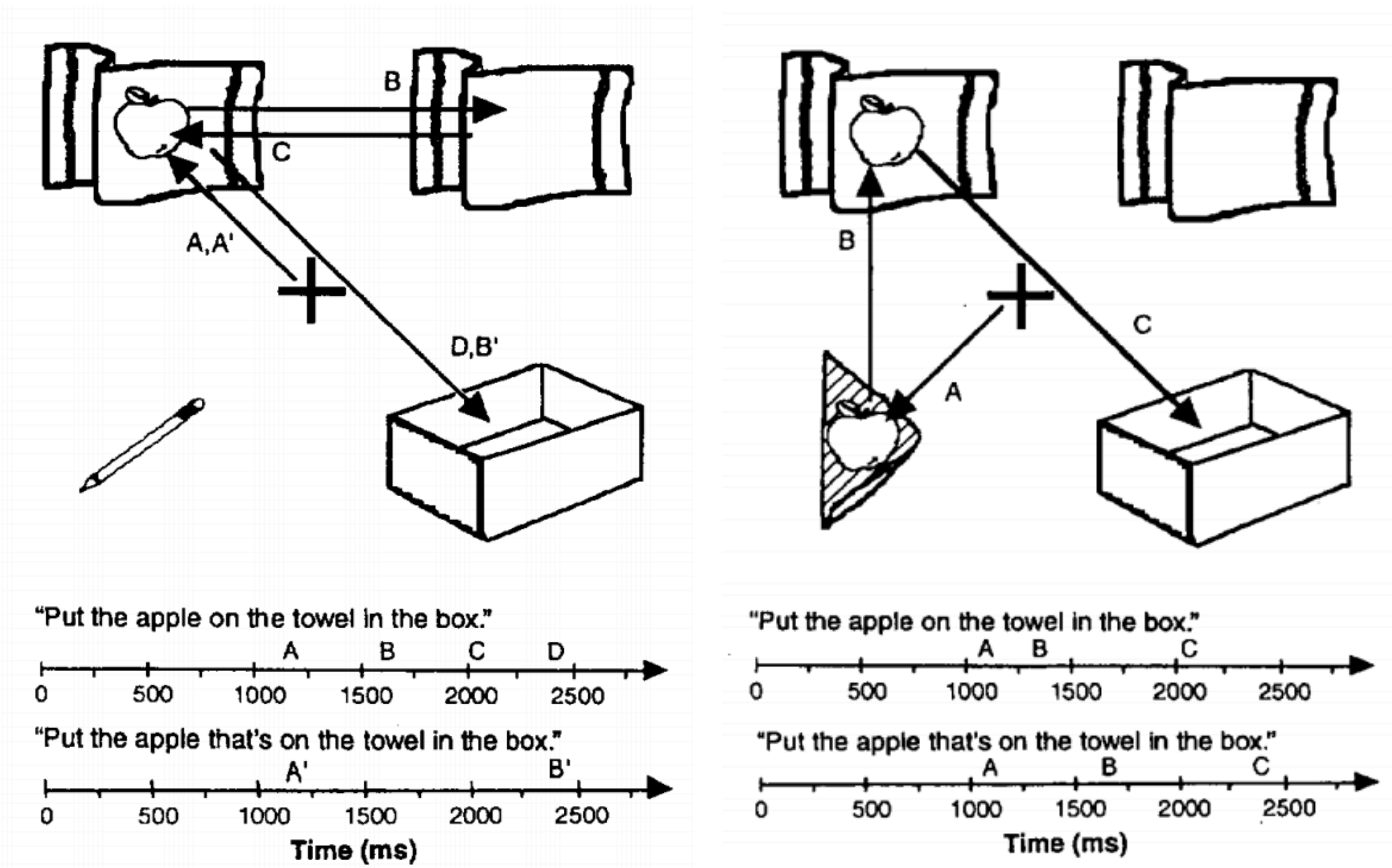
- Understanding syntax fundamentally requires trees
  — the sentences have the same shallow analysis

PRP	VBZ	DT	NN	IN	NNS
I	ate	the	spaghetti	with	chopsticks

PRP	VBZ	DT	NN	IN	NNS
I	ate	the	spaghetti	with	meatballs

Linguistic Structures

- Language is sequentially structured: interpreted in an online way



Tanenhaus et al. (1995)

POS Tagging

VBD VB
VBN VBZ VBP VBZ
NNP NNS NN NNS CD NN
Fed raises interest rates 0.5 percent

VBD VB
VBN VBZ VBP VBZ
NNP NNS NN NNS CD NN
Fed raises interest rates 0.5 percent

I hereby increase interest rates 0.5%

I'm 0.5% interested in the Fed's raises!

- Other paths are also plausible but even more semantically weird…
- What governs the correct choice? Word + context
- Word identity: most words have <=2 tags, many have one (*percent*, *the*)
- Context: nouns start sentences, nouns follow verbs, etc.

Let’s play a game

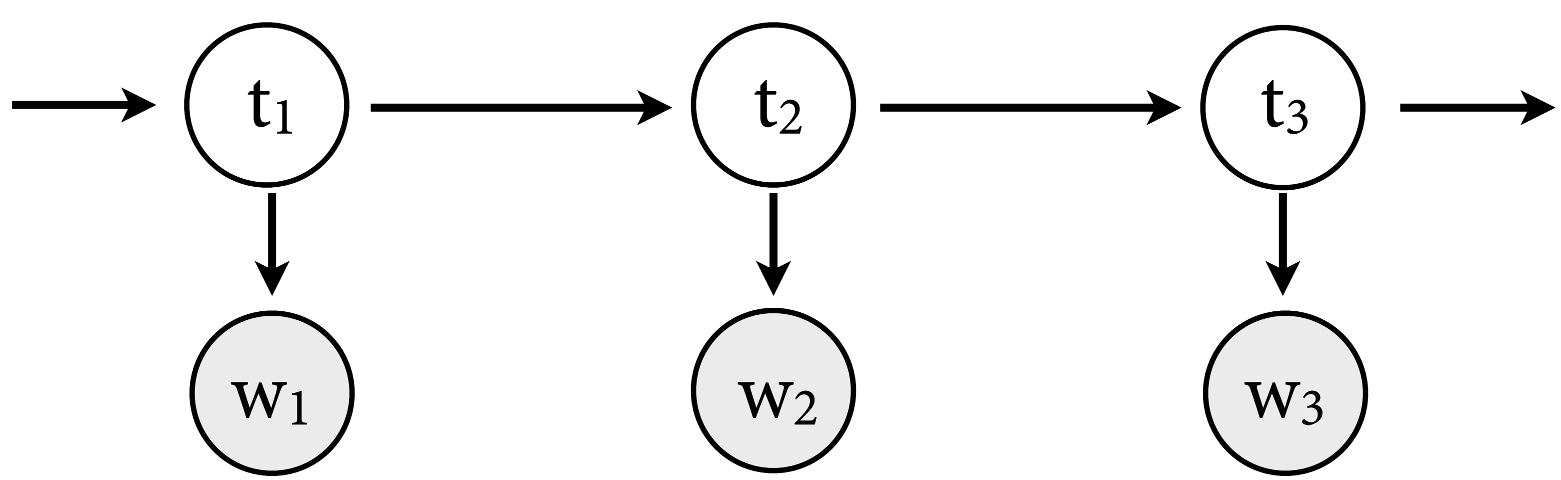
- I will write a sequence of part-of-speech tags and of words on the board.

- You take turns in giving me POS tags and words, and I will write them down.

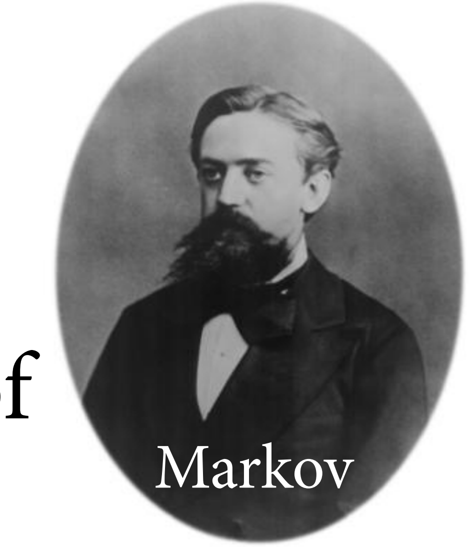
Hidden Markov Models

- Previous generative story: generate words at random from n-gram model $P(w_n \mid w_1, \ldots, w_{n-1})$.

- Replace with new generative story:
  - Language is generated by a two-step process.
  - First, generate sequence of hidden POS tags $t_1, \ldots, t_T$ tag by tag, left to right from bigram model $P(t_i \mid t_{i-1})$.
  - Independently, generate an observable word w_i from each t_i, at random from model $P(w_i \mid t_i)$.



Hidden Markov Models



- A *Hidden Markov Model* is 5-tuple consisting of
  - finite set $Q = \{q_1, \ldots, q_N\}$ of *states* (= POS tags)
  - finite set O of possible *observations* (= words)
  - *transition probabilities* $a_{ij} = P(X_{t+1} = q_j \mid X_t = q_i)$ $\quad \sum_{j=1}^{N} a_{ij} = 1$
  - *initial probabilities* $c_i = P(X_1 = q_i)$ $\quad \sum_{i=1}^{N} c_i = 1$
  - *emission probabilities* $b_i(o) = P(Y_t = o \mid X_t = q_i)$ $\quad \sum_{o \in O} b_i(o) = 1$

- The HMM describes two coupled random processes:
  - event $X_t = q_i$: At time t, HMM is in state q_i.
  - event $Y_t = o$: At time t, HMM emits observation o.

Question 1: Language modeling

- Given an HMM and a string $w_1, \ldots, w_T$, what is the likelihood $P(w_1 \ldots w_T)$?

- We can compute $P(w_1 \ldots w_T)$ efficiently with the *forward algorithm.*

DT	NN	VBD	NNS	IN	DT	NN	
The	representative	put	chairs	on	the	table.	p_1

DT	JJ	NN	VBZ	IN	DT	NN	
The	representative	put	chairs	on	the	table.	p_2

$P(\text{sentence}) = p_1 + p_2$

Question 2: Tagging (aka Decoding)

- Given an HMM and an observed string $w_1, \ldots, w_T$, what is the most likely sequence of hidden tags $t_1, \ldots, t_T$?

- We can compute $\arg\max_{t_1,\ldots,t_T} P(t_1, w_1, \ldots, t_T, w_T)$

  efficiently with the *Viterbi algorithm*.

DT	NN	VBD	NNS	IN	DT	NN	
The	representative	put	chairs	on	the	table.	p_1

DT	JJ	NN	VBZ	IN	DT	NN	
The	representative	put	chairs	on	the	table.	p_2

Question 3a: Supervised learning

- Given a set of POS tags and *annotated* training data $(w_1,t_1), \ldots, (w_T,t_T)$, compute parameters for HMM that maximize likelihood of training data.

- Do it efficiently with maximum likelihood training plus smoothing.

DT	NN	VBD	NNS	IN	DT	NN
The	representative	put	chairs	on	the	table.

NNP	VBZ	VBN	TO	VB	NR
Secretariat	is	expected	to	race	tomorrow.

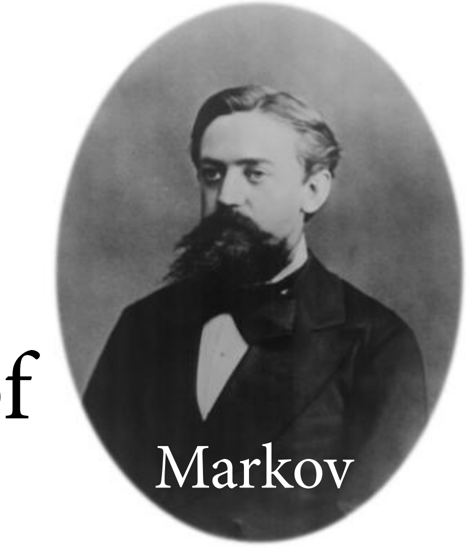
Question 3b: Unsupervised learning

- Given a set of POS tags and *unannotated* training data $w_1, \ldots, w_T$, compute parameters for HMM that maximize likelihood of training data.

- Do it with the *forward-backward algorithm* (an instance of *Expectation Maximization*).

The representative put chairs on the table.
Secretariat is expected to race tomorrow.

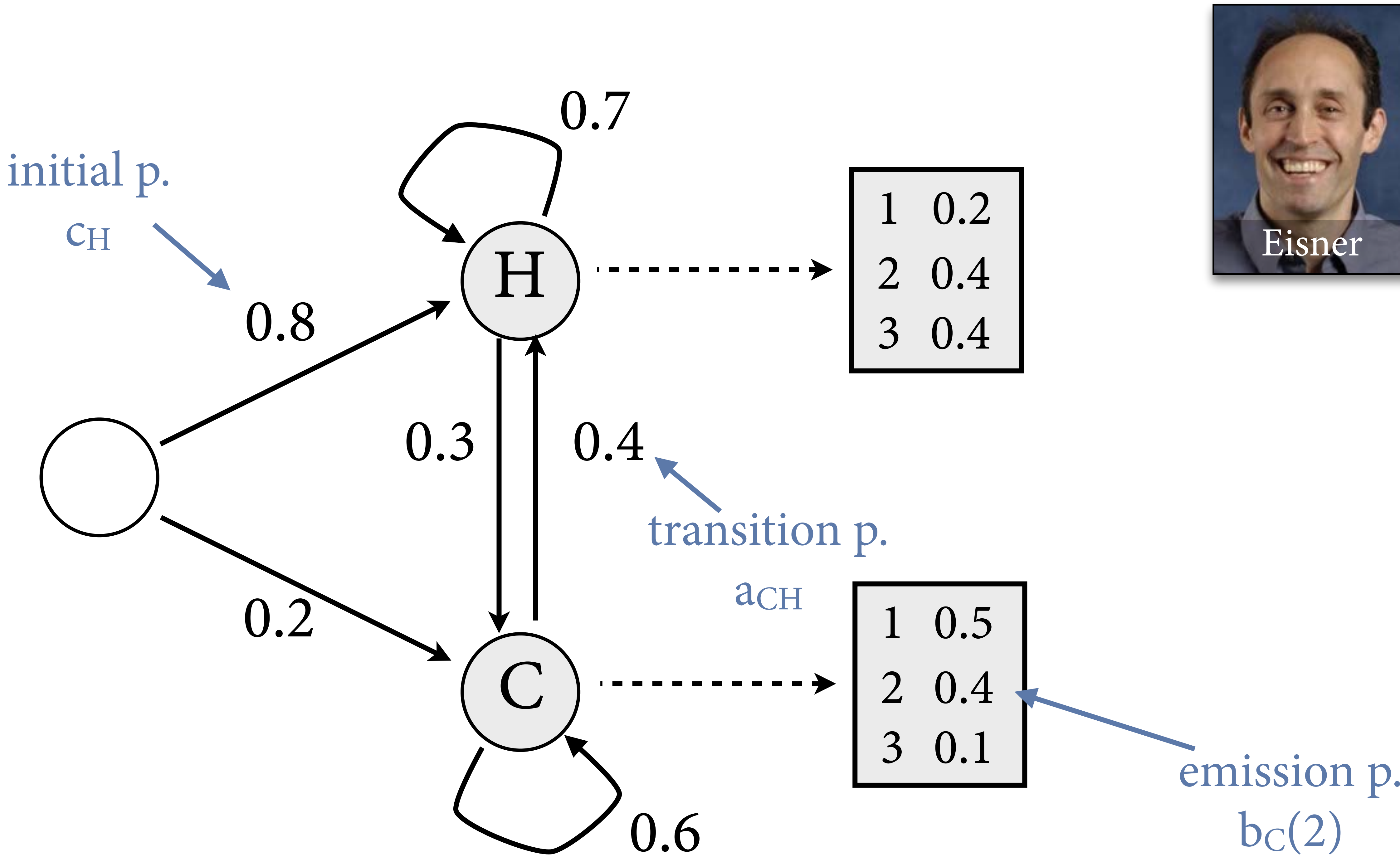
Hidden Markov Models



- A *Hidden Markov Model* is 5-tuple consisting of
  - finite set $Q = \{q_1, \ldots, q_N\}$ of *states* (= POS tags)
  - finite set O of possible *observations* (= words)
  - *transition probabilities* $a_{ij} = P(X_{t+1} = q_j \mid X_t = q_i)$ $\quad \sum_{j=1}^{N} a_{ij} = 1$
  - *initial probabilities* $c_i = P(X_1 = q_i)$ $\quad \sum_{i=1}^{N} c_i = 1$
  - *emission probabilities* $b_i(o) = P(Y_t = o \mid X_t = q_i)$ $\quad \sum_{o \in O} b_i(o) = 1$

- The HMM describes two coupled random processes:
  - event $X_t = q_i$: At time t, HMM is in state q_i.
  - event $Y_t = o$: At time t, HMM emits observation o.

Example: Eisner's Ice Cream



States represent weather on a given day: Hot, Cold

Outputs represent number of ice creams Jason eats that day

HMMs as joint models of x, y

- Coupled random processes of HMM directly give us model for *joint* probability P(x, y) where
  - $y = y_1 \ldots y_T$ sequence of observations
  - $x = x_1 \ldots x_T$ sequence of hidden states

- Defined as follows:

$$
\begin{aligned}
P(x, y) &= P(x) \cdot P(y \mid x) \\
&= P(X_1 = x_1) \cdot \prod_{t=2}^{T} P(X_t = x_t \mid X_1 = x_1, \ldots, X_{t-1} = x_{t-1}) \\
&\quad \cdot \prod_{t=1}^{T} P(Y_t = y_t \mid Y_1 = y_1, \ldots, Y_{t-1} = y_{t-1}, x) \\
&= P(X_1 = x_1) \cdot \prod_{t=2}^{T} P(X_t = x_t \mid X_{t-1} = x_{t-1}) \cdot \prod_{t=1}^{T} P(Y_t = y_t \mid X_t = x_t) \\
&= c_{x_1} \cdot \prod_{t=2}^{T} a_{x_{t-1} x_t} \cdot \prod_{t=1}^{T} b_{x_t}(y_t)
\end{aligned}
$$

Question 2: Tagging

- Given observations $y_1, \ldots, y_T$ (# ice creams), what is the most probable sequence $x_1, \ldots, x_T$ of hidden states (temperatures)?

- Maximum probability:

$$\max_{x_1,\ldots,x_T} P(x_1, \ldots, x_T \mid y_1, \ldots, y_T)$$

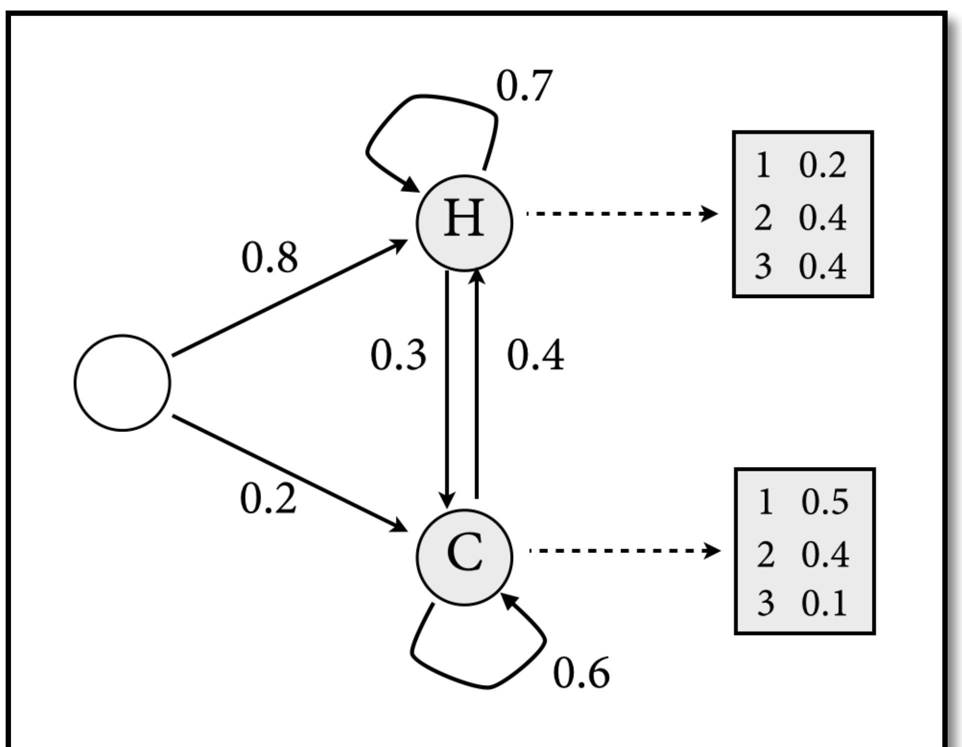
- We are primarily interested in arg max:

$$\begin{aligned}
&\arg\max_{x_1,\ldots,x_T} P(x_1, \ldots, x_T \mid y_1, \ldots, y_T) \\
&= \arg\max_{x_1,\ldots,x_T} \frac{P(x_1, \ldots, x_T, y_1, \ldots, y_T)}{P(y_1, \ldots, y_T)} \\
&= \arg\max_{x_1,\ldots,x_T} P(x_1, \ldots, x_T, y_1, \ldots, y_T)
\end{aligned}$$

Naive approach

- Let's say Jason ate 3, 1, 3 ice creams. What was the most likely weather on these three days?

- Compute max $P(x_1, 3, x_2, 1, x_3, 3)$ by maximizing over all possible state sequences.



$P(x_1, 3, x_2, 1, x_3, 3)$ is max of

P(H,3,H,1,H,3)	0.013
P(H,3,H,1,C,3)	0.001
P(H,3,C,1,H,3)	0.008
...	. . .
P(C,3,C,1,C,3)	0.0004
=	0.013

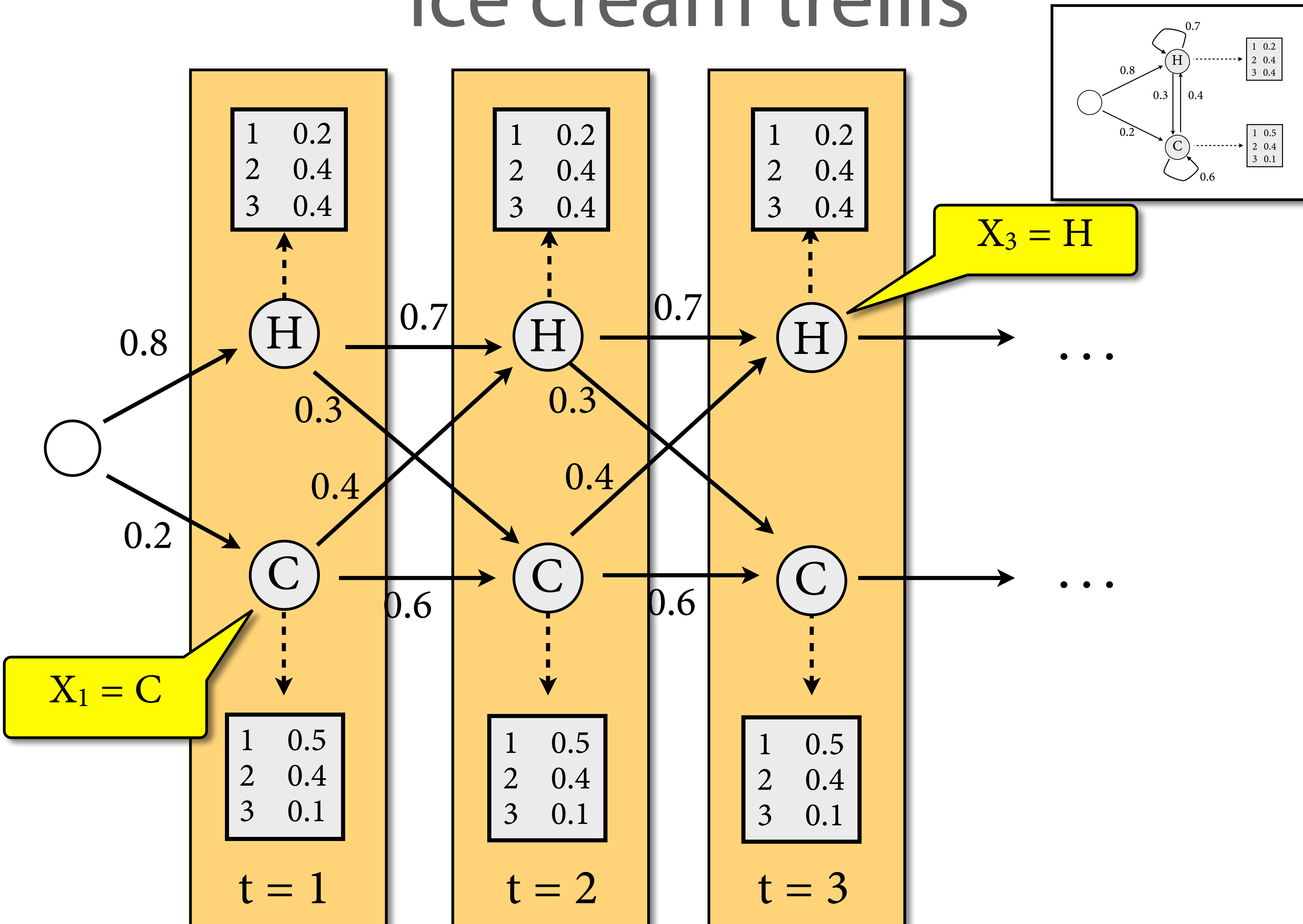
Too expensive

- Naive approach maximizes over exponential set of terms. This is too slow for practical use.

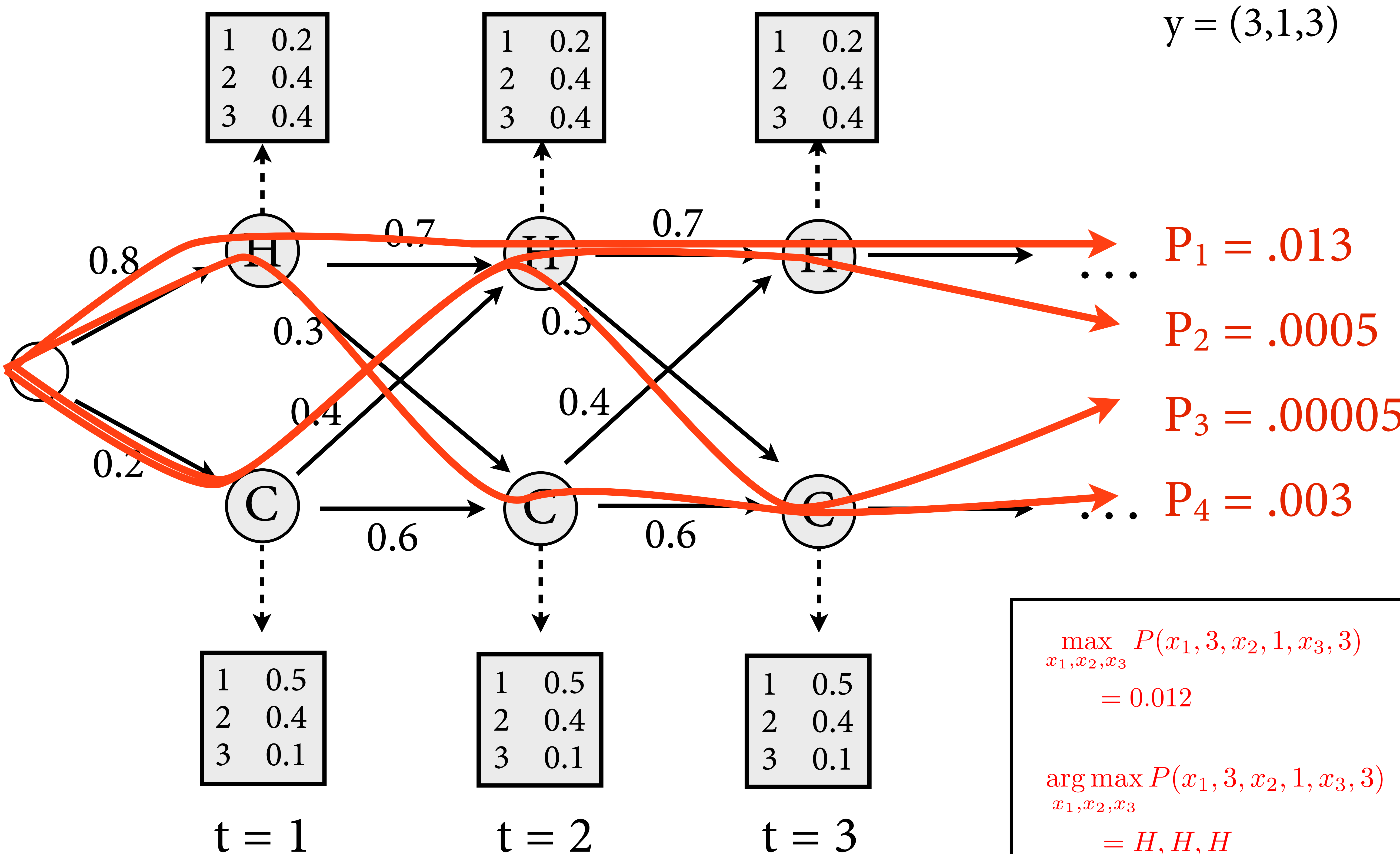
- Visualize this in *trellis:* unfolding of HMM
  - one column for each time point t, represents X_t
  - each column contains a copy of each state of HMM
  - edges from t to t+1 = transitions of HMM

- Each path through trellis represents one state sequence.
  - So computation of max $P(x,y)$ = max over all paths.

Ice cream trellis



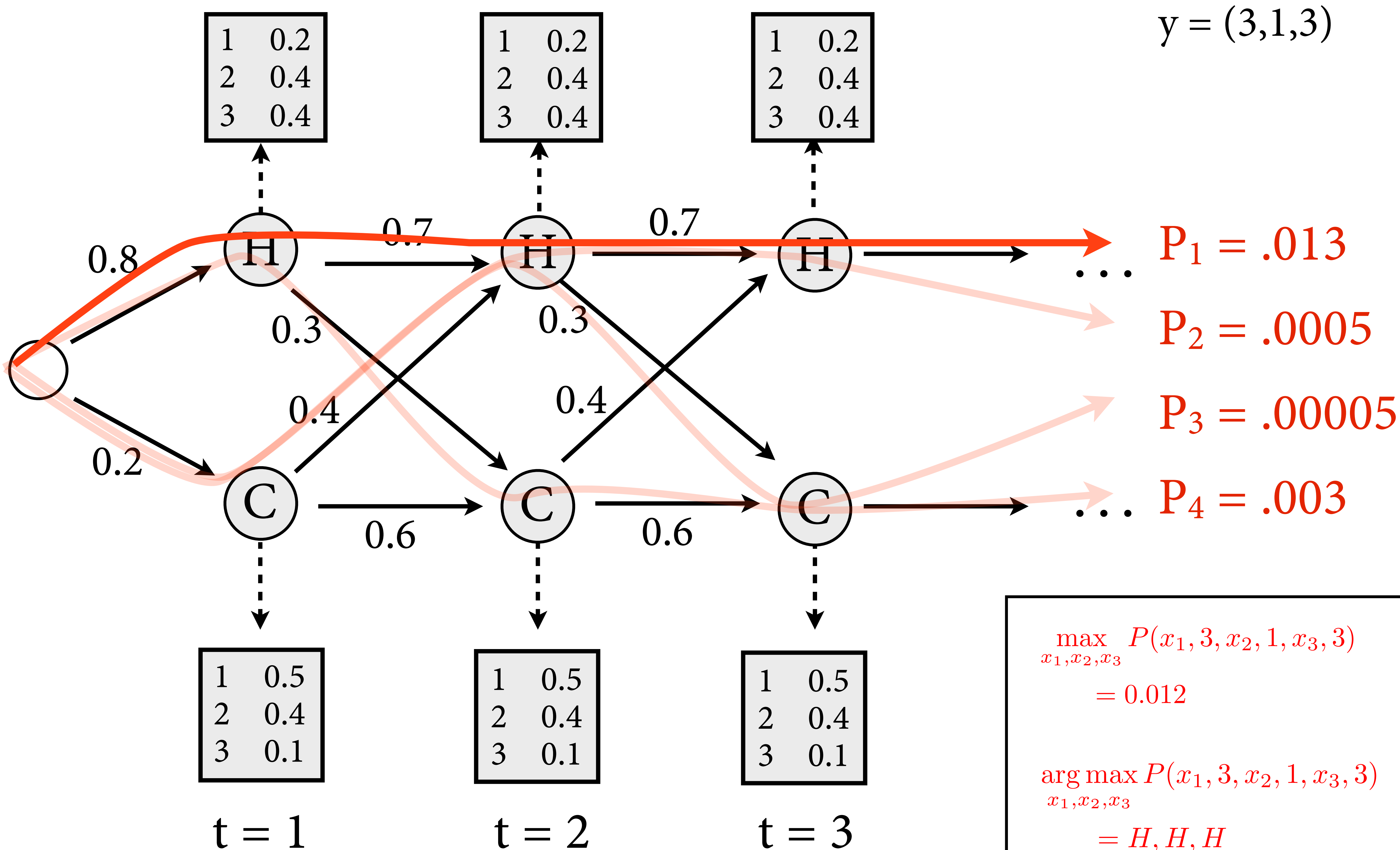
Ice cream trellis



$$\max_{x_1,x_2,x_3} P(x_1, 3, x_2, 1, x_3, 3) = 0.012$$

$$\arg\max_{x_1,x_2,x_3} P(x_1, 3, x_2, 1, x_3, 3) = H, H, H$$

Ice cream trellis



$$\max_{x_1, x_2, x_3} P(x_1, 3, x_2, 1, x_3, 3)$$
$$= 0.012$$

$$\arg\max_{x_1, x_2, x_3} P(x_1, 3, x_2, 1, x_3, 3)$$
$$= H, H, H$$

The Viterbi Algorithm

- Because of statistical independencies, can decompose joint probability:

$$\begin{aligned} P(x_1, y_1, \ldots, x_t, y_t) &= P(y_t \mid x_1, \ldots, x_t, y_1, \ldots, y_{t-1}) \cdot P(x_t \mid x_1, \ldots, x_{t-1}, y_1, \ldots, y_{t-1}) \\ &\quad \cdot P(x_1, y_1, \ldots, x_{t-1}, y_{t-1}) \\ &= P(y_t \mid x_t) \cdot P(x_t \mid x_{t-1}) \cdot P(x_1, y_1, \ldots, x_{t-1}, y_{t-1}) \end{aligned}$$

- Thus, maximum has recursive structure:

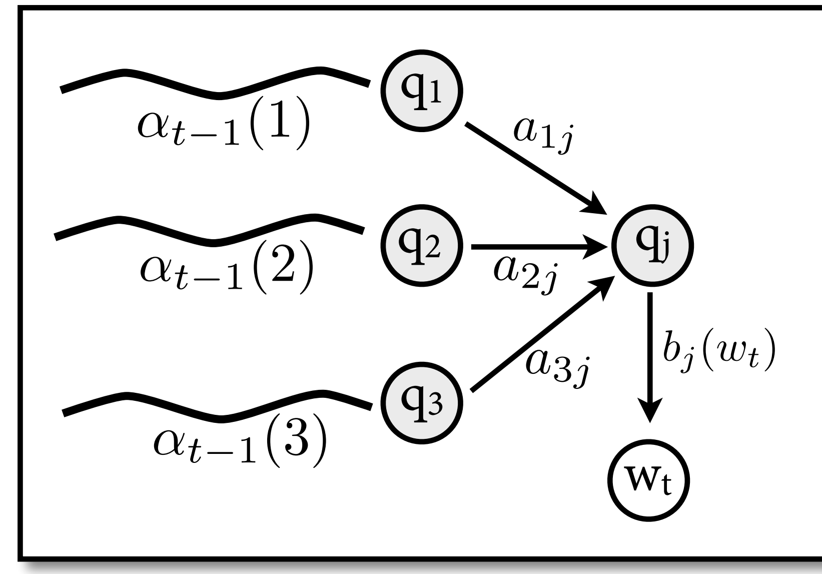
$$\begin{aligned} V_t(j) &= \max_{x_1, \ldots, x_{t-1}} P(y_1, \ldots, y_t, x_1, \ldots, x_{t-1}, X_t = q_j) \\ &= \max_{x_1, \ldots, x_{t-1}} P(y_t \mid X_t = q_j) \cdot P(X_t = q_j \mid x_{t-1}) \cdot P(y_1, \ldots, y_{t-1}, x_1, \ldots, x_{t-1}) \\ &= \max_{x_{t-1}} P(y_t \mid X_t = q_j) \cdot P(X_t = q_j \mid x_{t-1}) \cdot (\max_{x_1, \ldots, x_{t-2}} P(y_1, \ldots, y_{t-1}, x_1, \ldots, x_{t-1})) \\ &= \max_{i} P(y_t \mid X_t = q_j) \cdot P(X_t = q_j \mid X_{t-1} = q_i) \cdot (\max_{x_1, \ldots, x_{t-2}} P(y_1, \ldots, y_{t-1}, x_1, \ldots, X_{t-1} = q_i)) \\ &= \max_{i} P(y_t \mid X_t = q_j) \cdot P(X_t = q_j \mid X_{t-1} = q_i) \cdot V_{t-1}(i) \; \boxed{= \max_{i=1}^{N} V_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)} \end{aligned}$$

The Viterbi Algorithm

$$V_t(j) = \max_{x_1,\ldots,x_{t-1}} P(y_1,\ldots,y_t,x_1,\ldots,x_{t-1},X_t = q_j)$$

- Base case, t = 1:

$$V_1(j) = b_j(y_1) \cdot a_{0j}$$



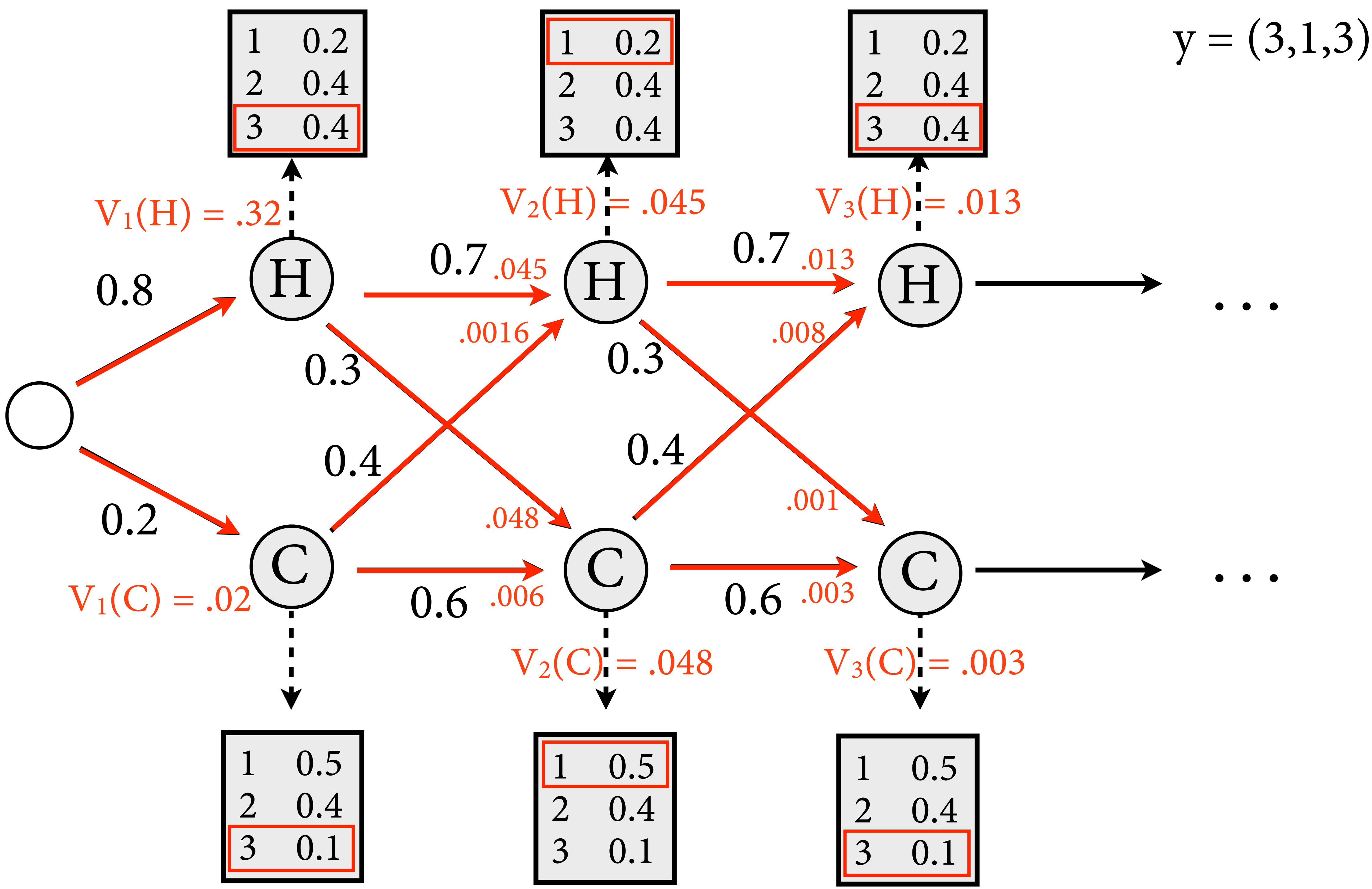
- Inductive case, for t = 2, …, T:

$$V_t(j) = \max_{i=1}^{N} V_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)$$

- Once we have calculated all V values, we can easily calculate prob of best path:

$$\max_{x_1,\ldots,x_T} P(x_1,y_1,\ldots,x_T,y_T) = \max_{q \in Q} V_T(q)$$

Viterbi Algorithm: Example



$$V_t(j) = \max_{x_1,\dots,x_{t-1}} P(y_1,\dots,y_t,x_1,\dots,x_{t-1},X_t=q_j)$$

$$V_t(j) = \max_{i=1}^{N} V_{t-1}(i)\cdot a_{ij}\cdot b_j(y_t)$$

Backpointers

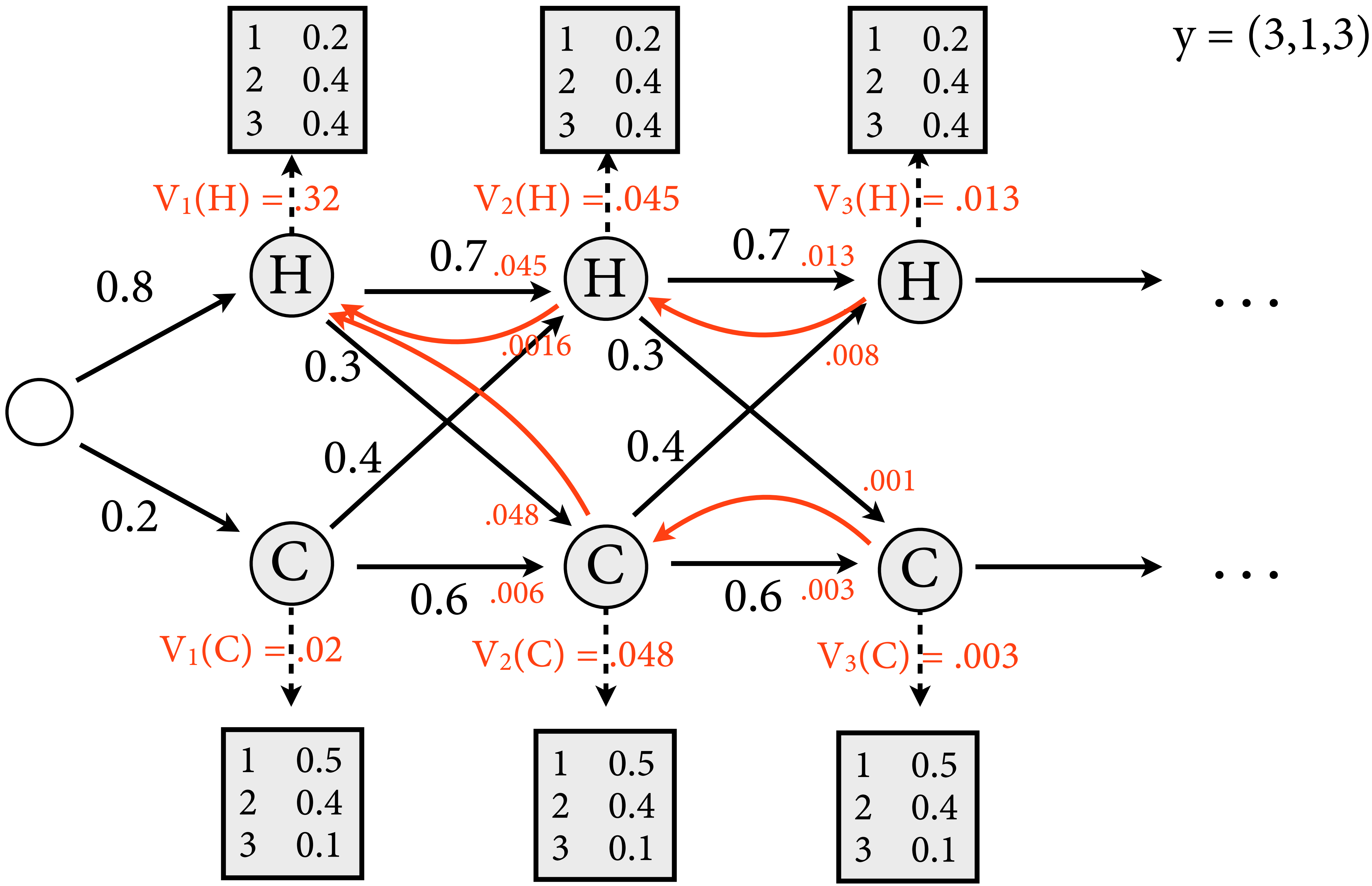
- In the end, we need to reconstruct the sequence of states $x_1, \ldots, x_T$ with max probability.

- For each t, j: remember the value of i for which the maximum was achieved in *backpointer* $bp_t(j)$.

$$V_t(j) = \max_{i=1}^{N} V_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)$$

- Then just follow backpointers from right to left.

Viterbi Algorithm: Example



$V_t(j) = \max_{x_1,\ldots,x_{t-1}} P(y_1,\ldots,y_t,x_1,\ldots,x_{t-1},X_t = q_j)$

$V_t(j) = \max_{i=1}^{N} V_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)$

Question 1: Likelihood, P(y)

- How likely is it that Jason Eisner ate 3 ice creams on day 1, 1 ice cream on day 2, 3 ice creams on day 3?

- Want to compute: P(3, 1, 3).

- Same problem as with max:
  - Output 3, 1, 3 can be emitted by many different state sequences.
  - Obtain by marginalization:

$$P(3,1,3) = \sum_{x_1,x_2,x_3 \in Q} P(x_1,3,x_2,1,x_3,3)$$

  - Naive computation is far too slow.

The Forward Algorithm

- Key idea: *Forward probability* $\alpha_t(j)$ that HMM outputs $y_1, \ldots, y_t$ and then ends in $X_t = q_j$.

$$\begin{aligned}\alpha_t(j) &= P(y_1, \ldots, y_t, X_t = q_j) \\ &= \sum_{x_1, \ldots, x_{t-1}} P(y_1, \ldots, y_t, X_1 = x_1, \ldots, X_{t-1} = x_{t-1}, X_t = q_j)\end{aligned}$$

- From this, can compute easily

$$P(y_1, \ldots, y_T) = \sum_{q \in Q} \alpha_T(q)$$

The Forward Algorithm

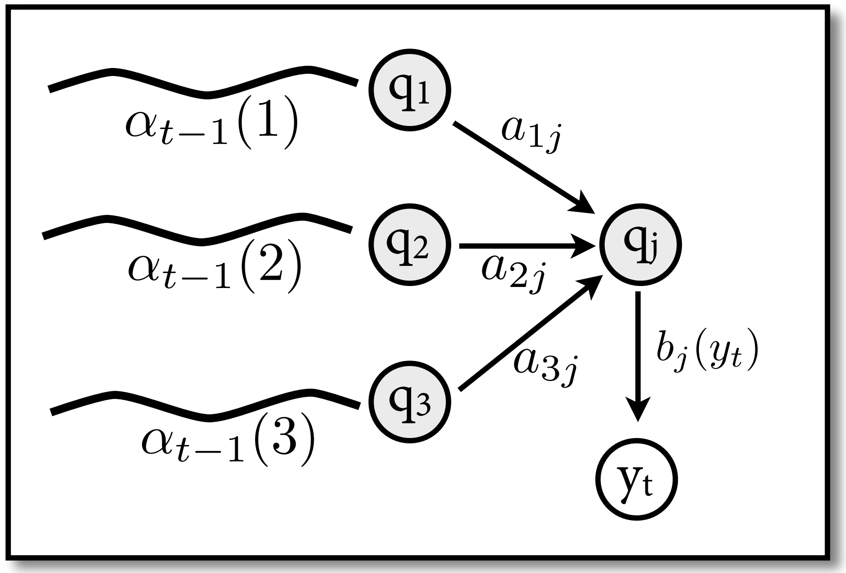
$$\alpha_t(j) = P(y_1, \ldots, y_t, X_t = q_j)$$

- Base case, t = 1:

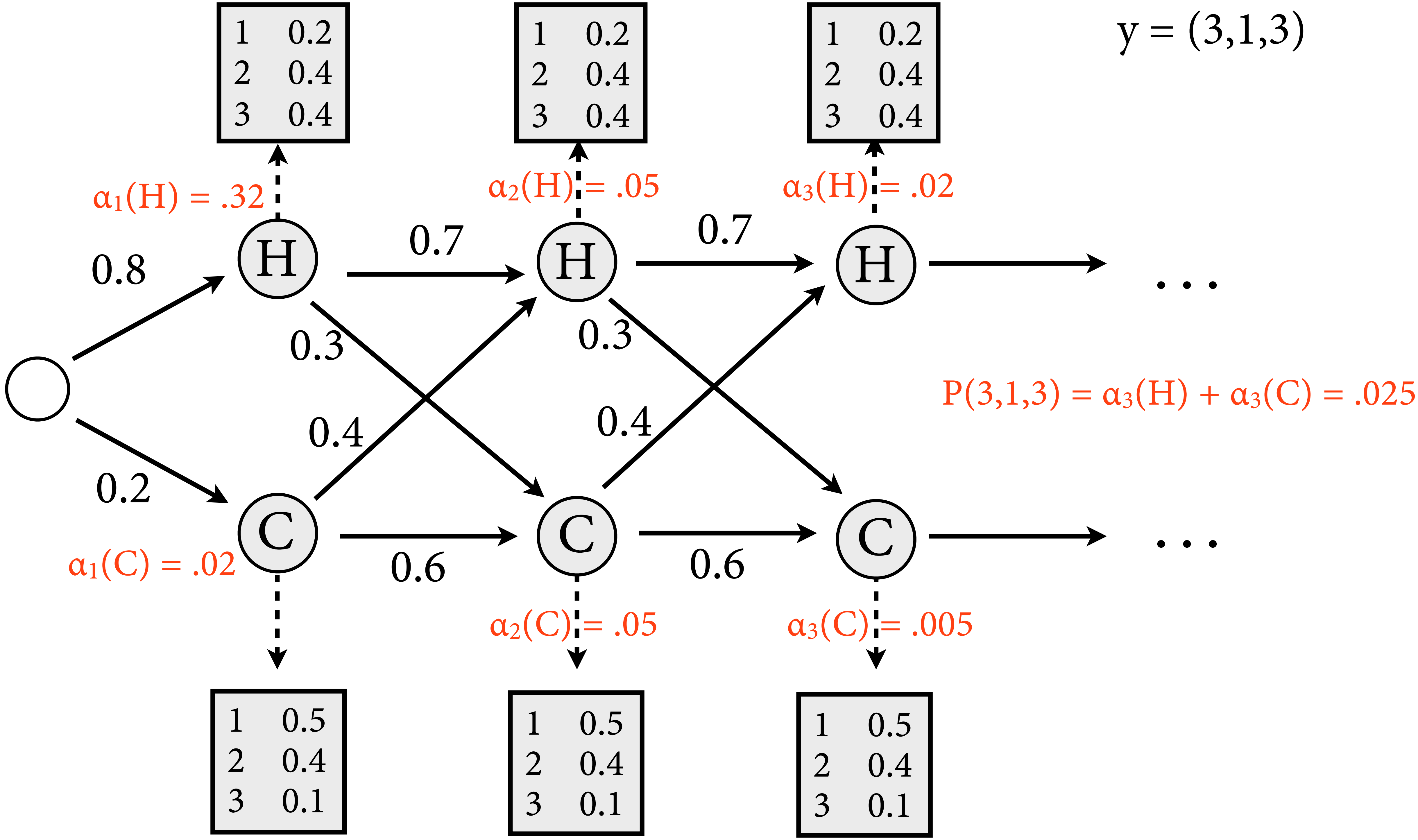
$$\alpha_1(j) = P(y_1, X_1 = q_j) = b_j(y_1) \cdot a_{0j}$$

- Inductive case, compute for t = 2, …, T:

$$\begin{aligned}\alpha_t(j) &= P(y_1, \ldots, y_t, X_t = q_j) \\ &= \sum_{i=1}^{N} P(y_1, \ldots, y_{t-1}, X_{t-1} = q_i) \cdot P(X_t = q_j \mid X_{t-1} = q_i) \cdot P(y_t \mid X_t = q_j) \\ &= \sum_{i=1}^{N} \alpha_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)\end{aligned}$$



P(3,1,3) with Forward



$\alpha_t(j) = P(y_1, \ldots, y_t, X_t = q_j)$ $\alpha_1(j) = b_j(y_1) \cdot a_{0j}$ $\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)$

Runtime

- Forward and Viterbi have the same runtime, dominated by inductive step:

$$V_t(j) = \max_{i=1}^{N} V_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)$$

- Compute N values for $V_t(j)$. Each computation step requires iteration over N predecessor states.

- Total runtime is $O(N^2 \cdot T)$, i.e.
  - linear in sentence length
  - quadratic in size of tag set

Summary

- Hidden Markov Models popular model for POS tagging (and other applications, see later).

- Two coupled random processes:
  - bigram model for hidden states (“Markov Chaing”)
  - model for producing observable output from each state

- Efficient algorithms for common problems:
  - Likelihood computation: Forward algorithm
  - Best state sequence: Viterbi algorithm.

Question 3a: Supervised learning

- Given a set of POS tags and *annotated* training data (w_1,t_1), …, (w_T,t_T), compute parameters for HMM that maximize likelihood of training data.

DT	NN	VBD	NNS	IN	DT	NN
The	representative	put	chairs	on	the	table.

NNP	VBZ	VBN	TO	VB	NR
Secretariat	is	expected	to	race	tomorrow.

Maximum likelihood training

- Estimate bigram model for state sequence:

$$a_{ij} = \frac{C(X_t = q_i, X_{t+1} = q_j)}{C(X_t = q_i)} \qquad c_j = \frac{\#\ \text{sentences with}\ X_1 = q_j}{\#\ \text{sentences}}$$

- ML estimate for emission probabilities:

$$b_i(o) = \frac{C(X_t = q_i, Y_t = o)}{C(X_t = q_i)}$$

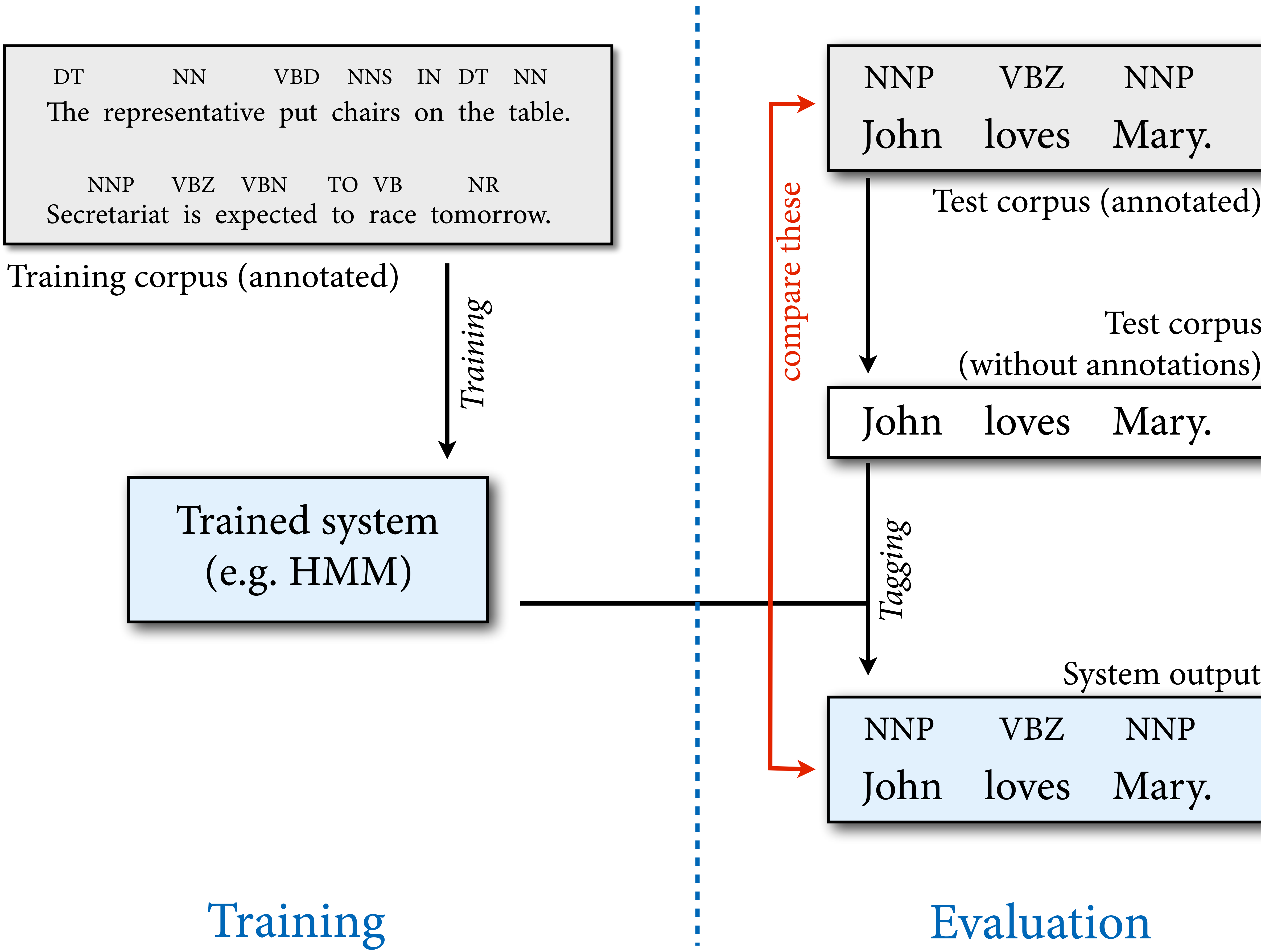
- Apply smoothing as you would for ordinary n-gram models (increase all counts C by one).

Evaluation

- How do you know how well your tagger works?

- Run it on *test data* and evaluate *accuracy*.
    - Test data: Really important to evaluate on unseen sentences to get a fair picture of how well tagger generalizes.
    - Accuracy: Measure percentage of correctly predicted POS tags.

Evaluation on test data



Question 3b: Unsupervised learning

- Given a set of POS tags and *unannotated* training data $w_1, \ldots, w_T$, compute parameters for HMM that maximize likelihood of training data.

- Relevant because annotated data is expensive to obtain, but raw text is really cheap.

The representative put chairs on the table.
Secretariat is expected to race today.

doesn't work very well for POS tagging
can be done using Expectation/Maximization
will skip that here

Trigram Taggers

NNP VBZ NN NNS CD NN
Fed raises interest rates 0.5 percent

- Trigram model: y_1 = (<S>, NNP), y_2 = (NNP, VBZ), …
- P((VBZ, NN) | (NNP, VBZ)) — more context! Noun-verb-noun S-V-O
- Tradeoff between model capacity and data size — trigrams are a "sweet spot" for POS tagging

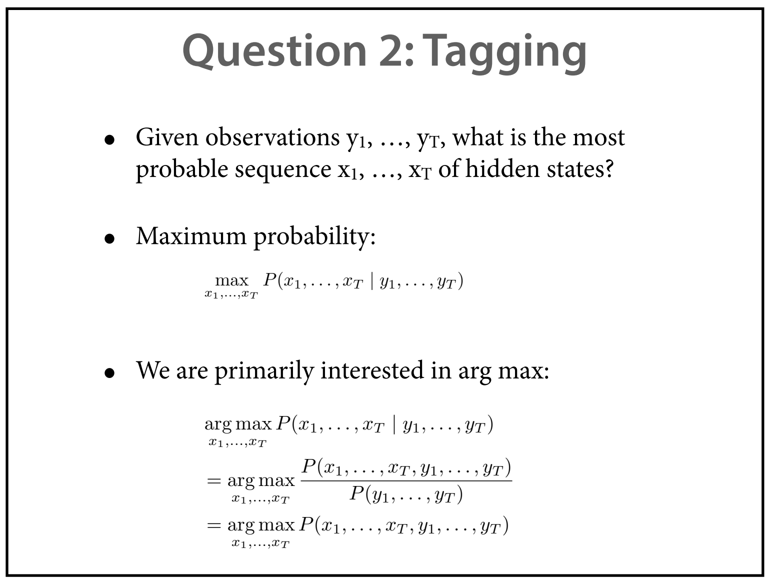
HMM POS Tagging

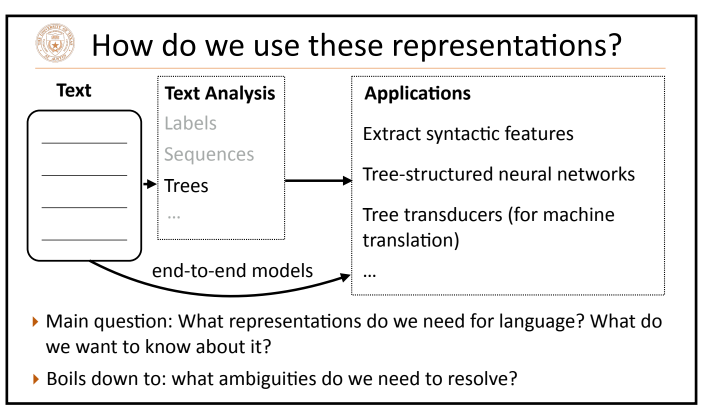
- Baseline: assign each word its most frequent tag: ~90% accuracy
- Trigram HMM: ~95% accuracy / 55% on unknown words
- TnT tagger (Brants 1998, tuned HMM): 96.2% accuracy / 86.0% on unks
- State-of-the-art (BiLSTM-CRFs): 97.5% / 89%+

Slide credit: Dan Klein

slide credits

slides that look like this





come from

earlier editions of this class (ANLP), given by Alexander Koller

CS388 given by Greg Durrett at U Texas, Austin

and their use is gratefully acknowledged. I try to make any modifications obvious, but if there are errors on a slide, assume that I added them.