Sequence Models I

Wei Xu

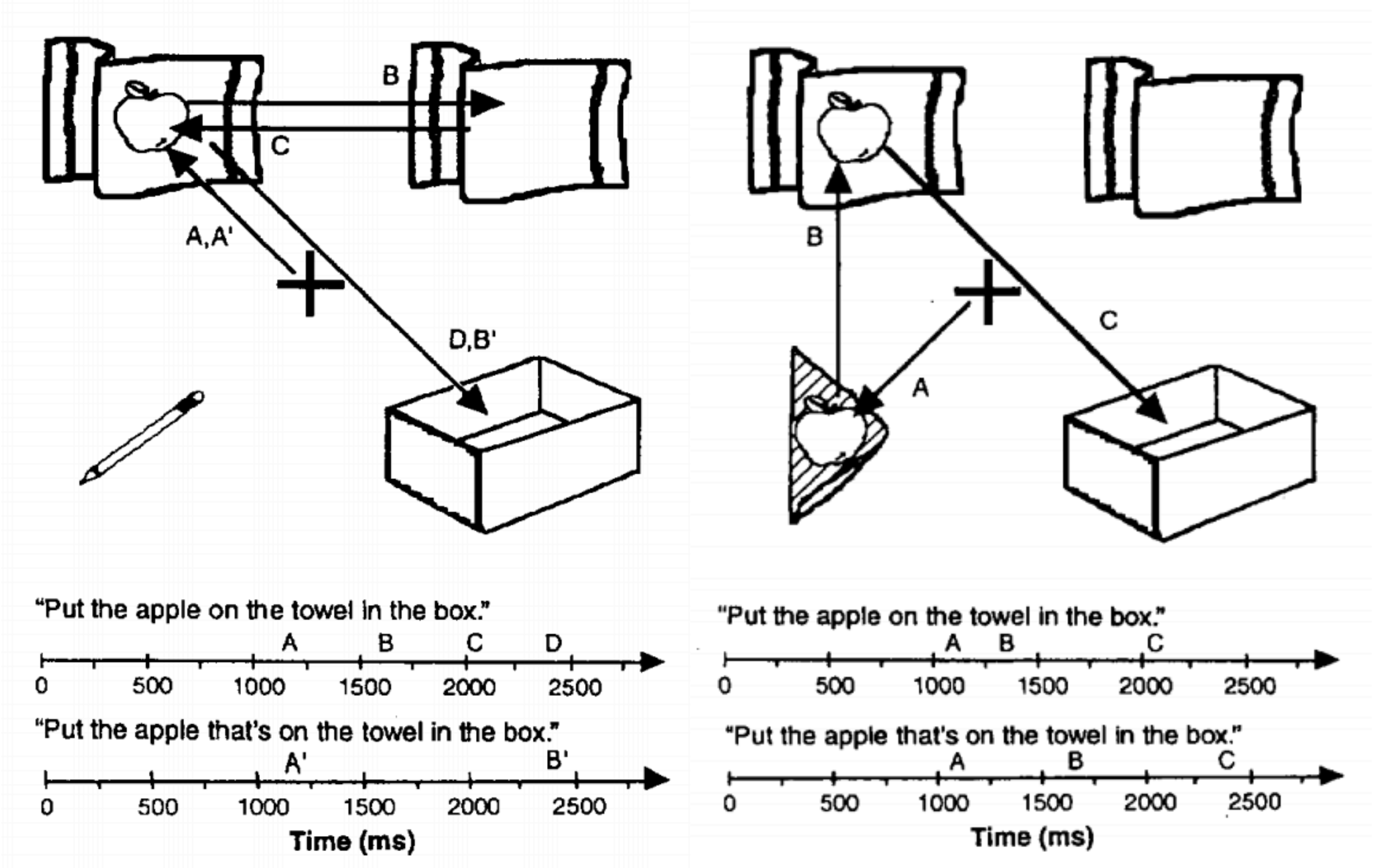
(many slides from Greg Durrett, Dan Klein, Vivek Srikumar, Chris Manning, Yoav Artzi)

This Lecture

- Sequence modeling
- HMMs for POS tagging
- HMM parameter estimation
- Viterbi, forward-backward
- Readings: Eisenstein 7.0-7.4, Jurafsky+Martin Chapter 17, Appendix A

Linguistic Structures

- Language is sequentially structured: interpreted in an online way



Tanenhaus et al. (1995)

POS Tagging

- What tags are out there?

Ghana ’s ambassador should have set up the big meeting in DC yesterday .

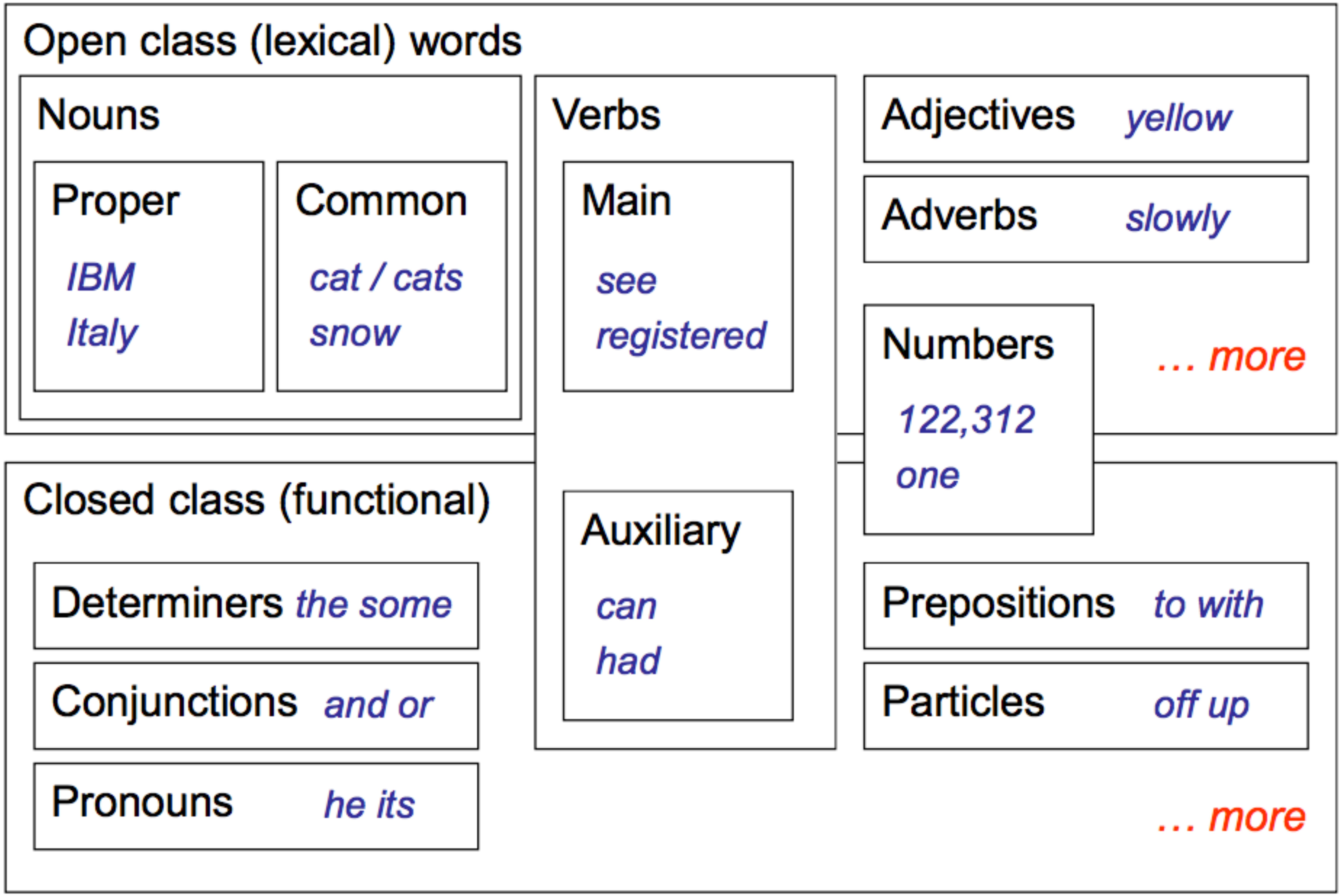
A demo —

POS Tagging

CC	conjunction, coordinating	and both but either or
CD	numeral, cardinal	mid-1890 nine-thirty 0.5 one
DT	determiner	a all an every no that the
EX	existential there	there
FW	foreign word	gemeinschaft hund ich jeux
IN	preposition or conjunction, subordinating	among whether out on by if
JJ	adjective or numeral, ordinal	third ill-mannered regrettable
JJR	adjective, comparative	braver cheaper taller
JJS	adjective, superlative	bravest cheapest tallest
MD	modal auxiliary	can may might will would
NN	noun, common, singular or mass	cabbage thermostat investment subhumanity
NNP	noun, proper, singular	Motown Cougar Yvette Liverpool
NNPS	noun, proper, plural	Americans Materials States
NNS	noun, common, plural	undergraduates bric-a-brac averages
POS	genitive marker	' 's
PRP	pronoun, personal	hers himself it we them
PRP$	pronoun, possessive	her his mine my our ours their thy your
RB	adverb	occasionally maddeningly adventurously
RBR	adverb, comparative	further gloomier heavier less-perfectly
RBS	adverb, superlative	best biggest nearest worst
RP	particle	aboard away back by on open through
TO	"to" as preposition or infinitive marker	to
UH	interjection	huh howdy uh whammo shucks heck
VB	verb, base form	ask bring fire see take
VBD	verb, past tense	pleaded swiped registered saw
VBG	verb, present participle or gerund	stirring focusing approaching erasing
VBN	verb, past participle	dilapidated imitated reunifed unsettled
VBP	verb, present tense, not 3rd person singular	twist appear comprise mold postpone
VBZ	verb, present tense, 3rd person singular	bases reconstructs marks uses
WDT	WH-determiner	that what whatever which whichever
WP	WH-pronoun	that what whatever which who whom
WP$	WH-pronoun, possessive	whose
WRB	Wh-adverb	however whenever where why

Slide credit: Yoav Artzi

POS Tagging



Slide credit: Dan Klein

POS Tagging

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

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.

What is this good for?

- Text-to-speech: *record*, *lead*
- Preprocessing step for syntactic parsers
- Domain-independent disambiguation for other tasks
- (Very) shallow information extraction
  - Identifying Subject-Verb-Object, action nouns, …

Sequence Models

- Input $\mathbf{x} = (x_1, ..., x_n)$ Output $\mathbf{y} = (y_1, ..., y_n)$
- POS tagging: ***x*** is a sequence of words, ***y*** is a sequence of tags
- Today: generative models P(***x, y***); discriminative models next time

Hidden Markov Models

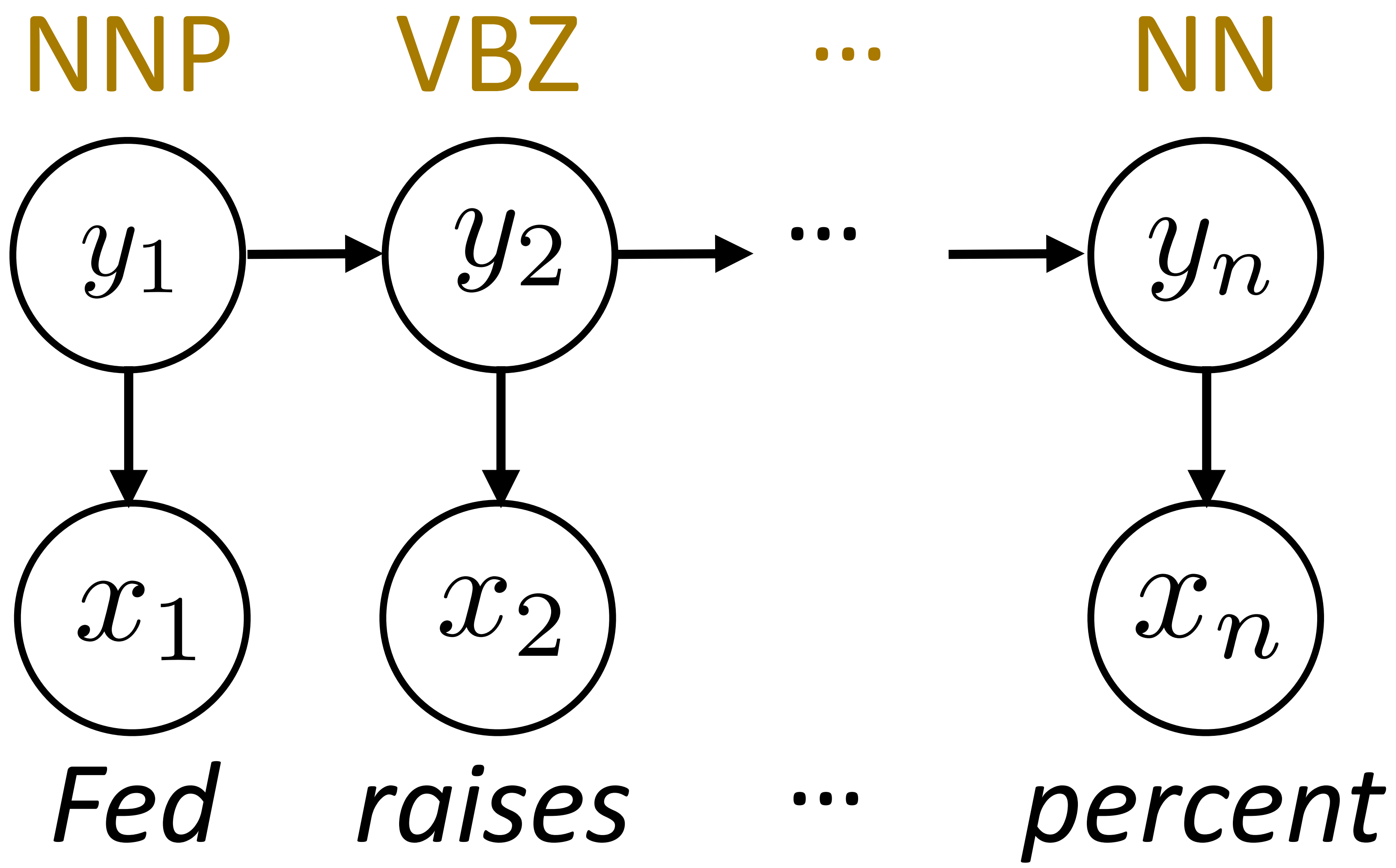
- Input $\mathbf{x} = (x_1, ..., x_n)$ Output $\mathbf{y} = (y_1, ..., y_n)$
- Model the sequence of *y* as a Markov process
- Markov property: future is conditionally independent of the past given the present

$$y_1 \to y_2 \to y_3 \qquad P(y_3 | y_1, y_2) = P(y_3 | y_2)$$

- Lots of mathematical theory about how Markov chains behave
- If *y* are tags, this roughly corresponds to assuming that the next tag only depends on the current tag, not anything before

Hidden Markov Models

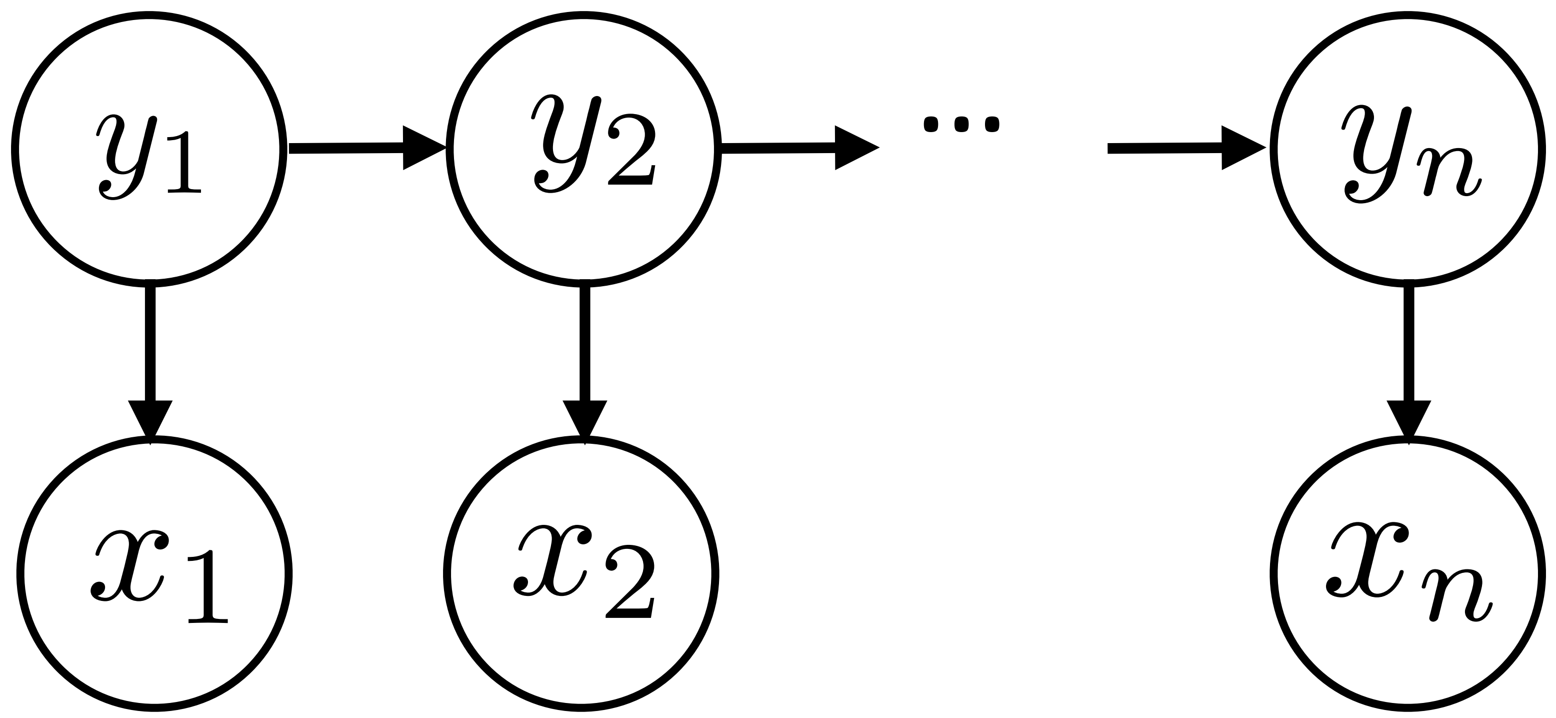
- Input $\mathbf{x} = (x_1, \ldots, x_n)$ Output $\mathbf{y} = (y_1, \ldots, y_n)$



- Each node (variable) is conditionally independent from its non-dependents given its parents.

Hidden Markov Models

- Input $\mathbf{x} = (x_1, ..., x_n)$ Output $\mathbf{y} = (y_1, ..., y_n)$

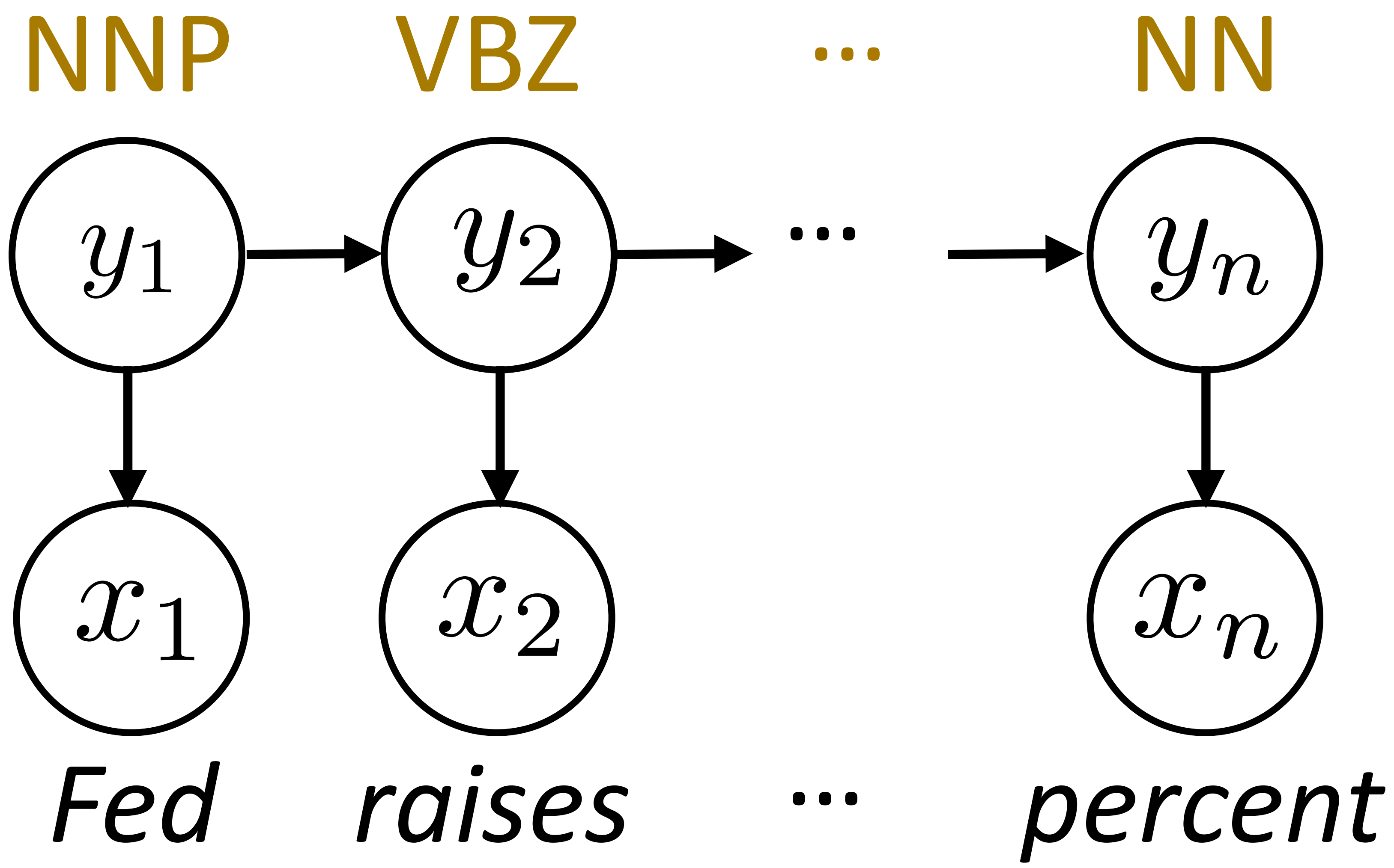


$$P(\mathbf{y}, \mathbf{x}) = \underbrace{P(y_1)}_{\text{Initial distribution}} \underbrace{\prod_{i=2}^{n} P(y_i|y_{i-1})}_{\text{Transition probabilities}} \underbrace{\prod_{i=1}^{n} P(x_i|y_i)}_{\text{Emission probabilities}}$$

- Each node (variable) is conditionally independent from its non-dependents given its parents.
- Observation (*x*) depends only on current state (*y*)

Hidden Markov Models

- Input $\mathbf{x} = (x_1, ..., x_n)$ Output $\mathbf{y} = (y_1, ..., y_n)$



$$P(\mathbf{y}, \mathbf{x}) = \underbrace{P(y_1)}_{\text{Initial distribution}} \underbrace{\prod_{i=2}^{n} P(y_i | y_{i-1})}_{\text{Transition probabilities}} \underbrace{\prod_{i=1}^{n} P(x_i | y_i)}_{\text{Emission probabilities}}$$

- Initial distribution: |T| x 1 vector (distribution over initial states)
- Emission probabilities: |T| x |V| matrix (distribution over words per tag)
- Transition probabilities: |T| x |T| matrix (distribution over next tags per tag)

Transitions in POS Tagging

- Dynamics model $P(y_1)\prod_{i=2}^{n}P(y_i|y_{i-1})$

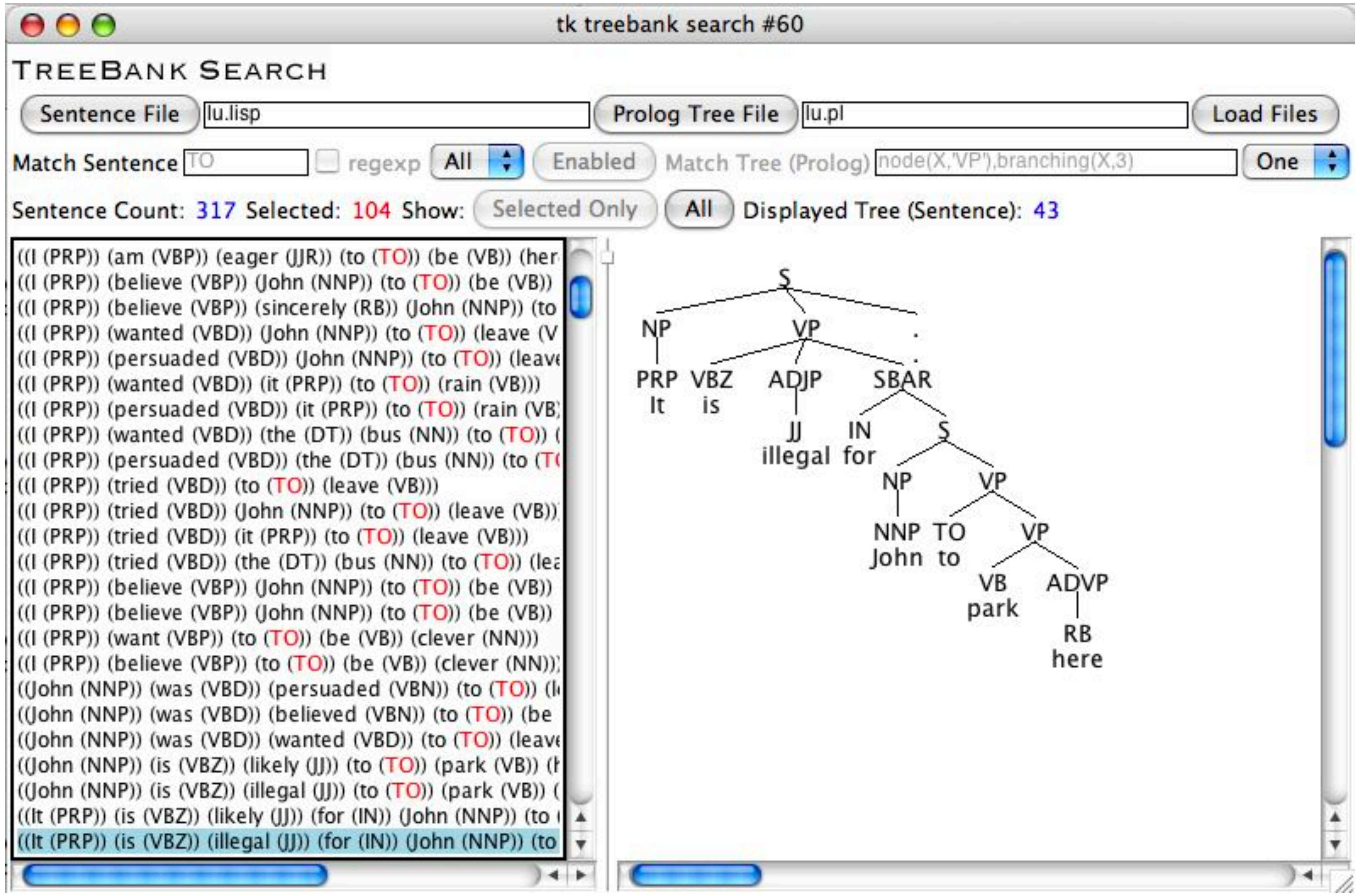
VBD		VB					
VBN	VBZ	VBP	VBZ				
NNP	NNS	NN	NNS	CD	NN		.
Fed	*raises*	*interest*	*rates*	*0.5*	*percent*		*.*

NNP - proper noun, singular
VBZ - verb, 3rd ps. sing. present
NN - noun, singular or mass

- $P(y_1 = \mathrm{NNP})$ likely because start of sentence
- $P(y_2 = \mathrm{VBZ}|y_1 = \mathrm{NNP})$ likely because verb often follows noun
- $P(y_3 = \mathrm{NN}|y_2 = \mathrm{VBZ})$ direct object follows verb, other verb rarely follows past tense verb (main verbs can follow modals though!)

Penn Treebank

- Developed 1988 — 1994;
- manually annotated with Part-of-Speech tags and syntactic structure
- Wall Street Journal, Brown, and Switchboard Corpus (>2m words)



Training HMMs

- Similar to Naive Bayes estimation: maximum likelihood solution = normalized counts (with smoothing) read off supervised data

- Transitions
  - Count up all pairs (y_i, y_{i+1}) in the training data
  - Count up occurrences of what tag T can transition to
  - Normalize to get a distribution for P (next tag | T)
  - Need to smooth (omitting details here)

- Emissions: similar scheme, but trickier smoothing

Estimating Transitions

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

- Similar to Naive Bayes estimation: maximum likelihood solution = normalized counts (with smoothing) read off supervised data
- P(tag | NN) = (0.5 ., 0.5 NNS)
- How to smooth?
- One method: smooth with unigram distribution over tags

$$P(\text{tag}|\text{tag}_{-1}) = (1-\lambda)\hat{P}(\text{tag}|\text{tag}_{-1}) + \lambda\hat{P}(\text{tag})$$

$\hat{P}$ = empirical distribution (read off from data)

Emissions in POS Tagging

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

- Emissions P(*x* | *y*) capture the distribution of words occurring with a given tag
- P(word | NN) = (0.05 *person*, 0.04 *official*, 0.03 *interest*, 0.03 *percent* …)
- When you compute the posterior for a given word's tags, the distribution favors tags that are more likely to generate that word
- How should we smooth this?

Estimating Emissions

NNP VBZ NN NNS CD NN

Fed raises interest rates 0.5 percent

- P(word | NN) = (0.5 *interest*, 0.5 *percent*) — hard to smooth!
- Can interpolate with distribution looking at word shape P(word shape | tag) (e.g., P(capitalized word of len >= 8 | tag))
- Alternative: use Bayes' rule

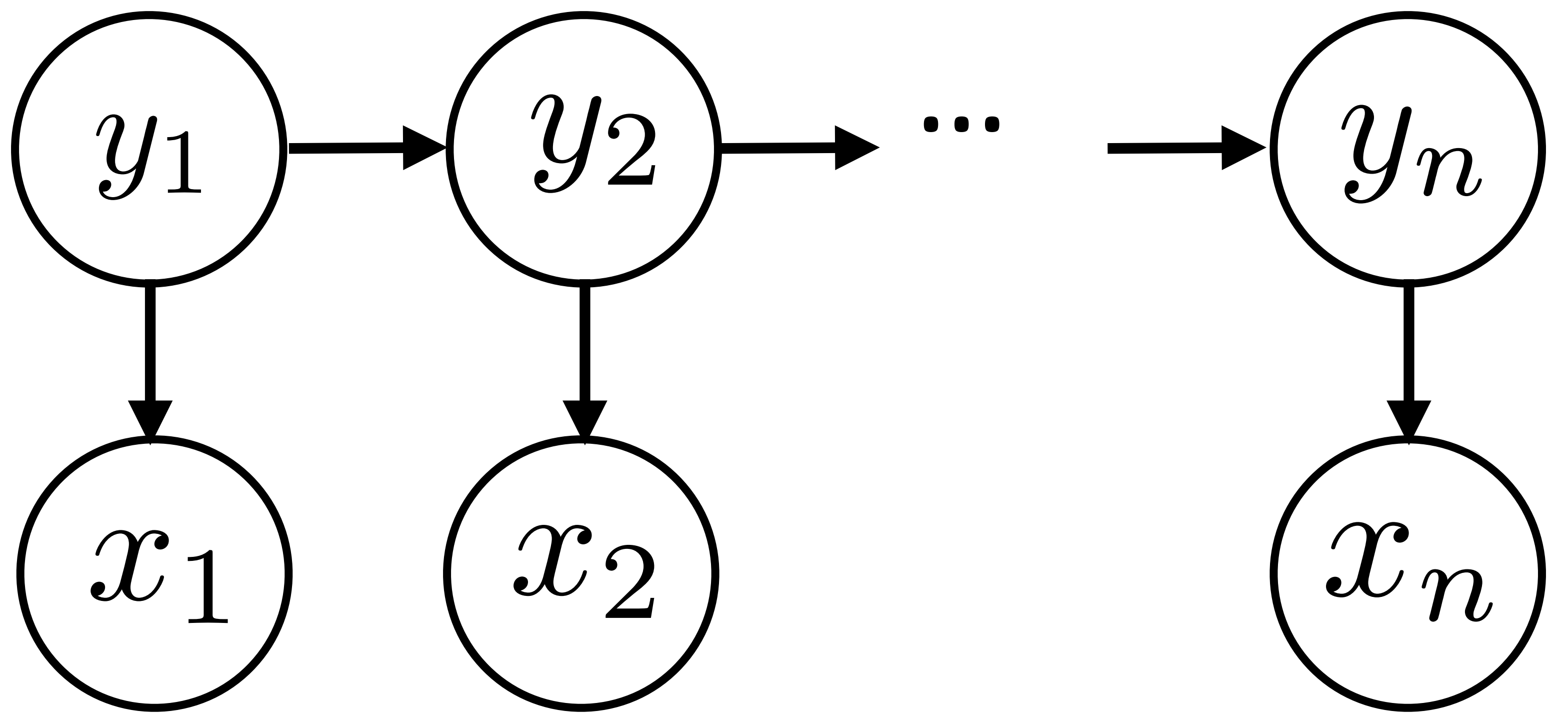
$$P(\text{word}|\text{tag}) = \frac{P(\text{tag}|\text{word})P(\text{word})}{P(\text{tag})}$$

  - Fancy techniques from language modeling, e.g. look at type fertility — P(tag|word) is flatter for some kinds of words than for others
- P(word|tag) can be a log-linear model — we'll see in a few lectures

TnT tagger – https://arxiv.org/pdf/cs/0003055.pdf

Inference in HMMs

- Input $\mathbf{x} = (x_1, ..., x_n)$ Output $\mathbf{y} = (y_1, ..., y_n)$



$$P(\mathbf{y}, \mathbf{x}) = P(y_1) \prod_{i=2}^{n} \boxed{P(y_i|y_{i-1})} \prod_{i=1}^{n} P(x_i|y_i)$$

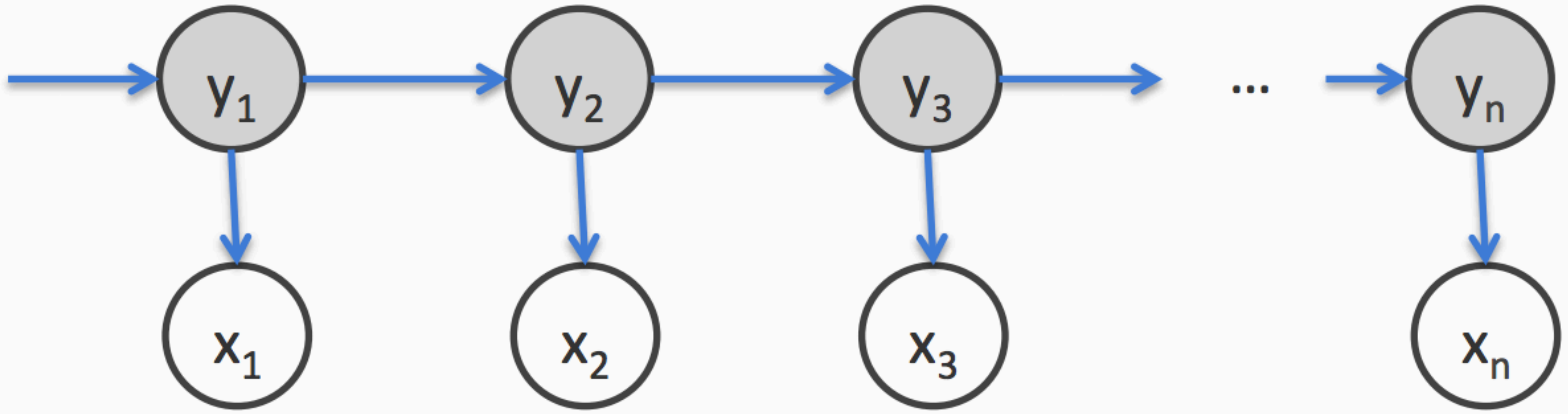
- Inference problem: $\text{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = \text{argmax}_{\mathbf{y}} \dfrac{P(\mathbf{y}, \mathbf{x})}{\cancel{P(\mathbf{x})}}$
- Exponentially many possible ***y*** here!
- Solution: **dynamic programming** (possible because of Markov structure!)
  - Many neural sequence models depend on entire previous tag sequence, need to use approximations like beam search

Viterbi Algorithm

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

$$\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

Transition probabilities Emission probabilities Initial probability



slide credit: Vivek Srikumar

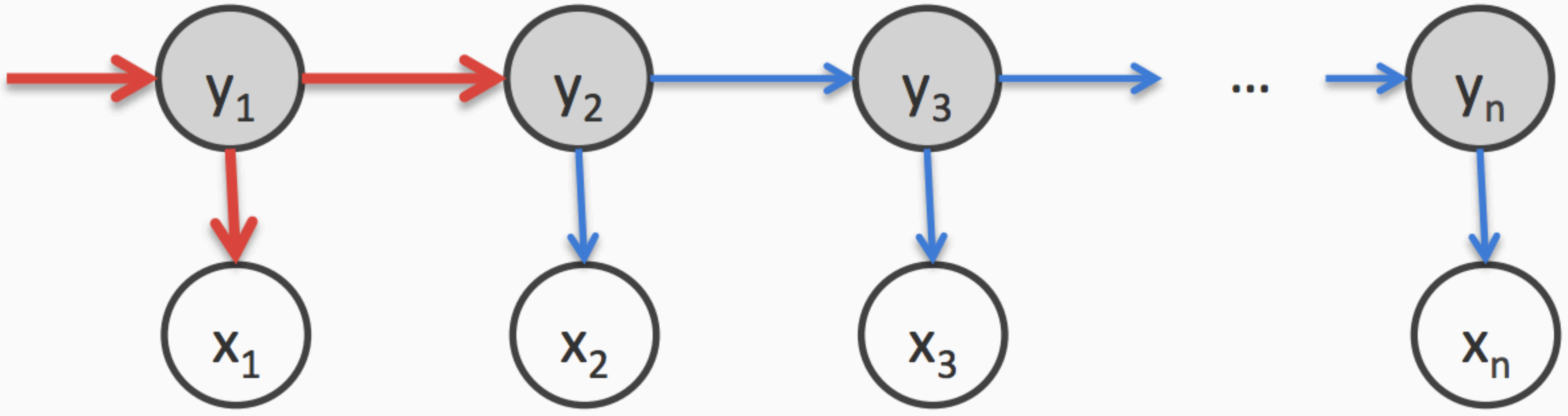
Viterbi Algorithm

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

$$\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

The only terms that depend on y_1



slide credit: Vivek Srikumar

Viterbi Algorithm

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

$$\max_{y_1,y_2,\cdots,y_n} P(y_n|y_{n-1})P(x_n|y_n)\cdots P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

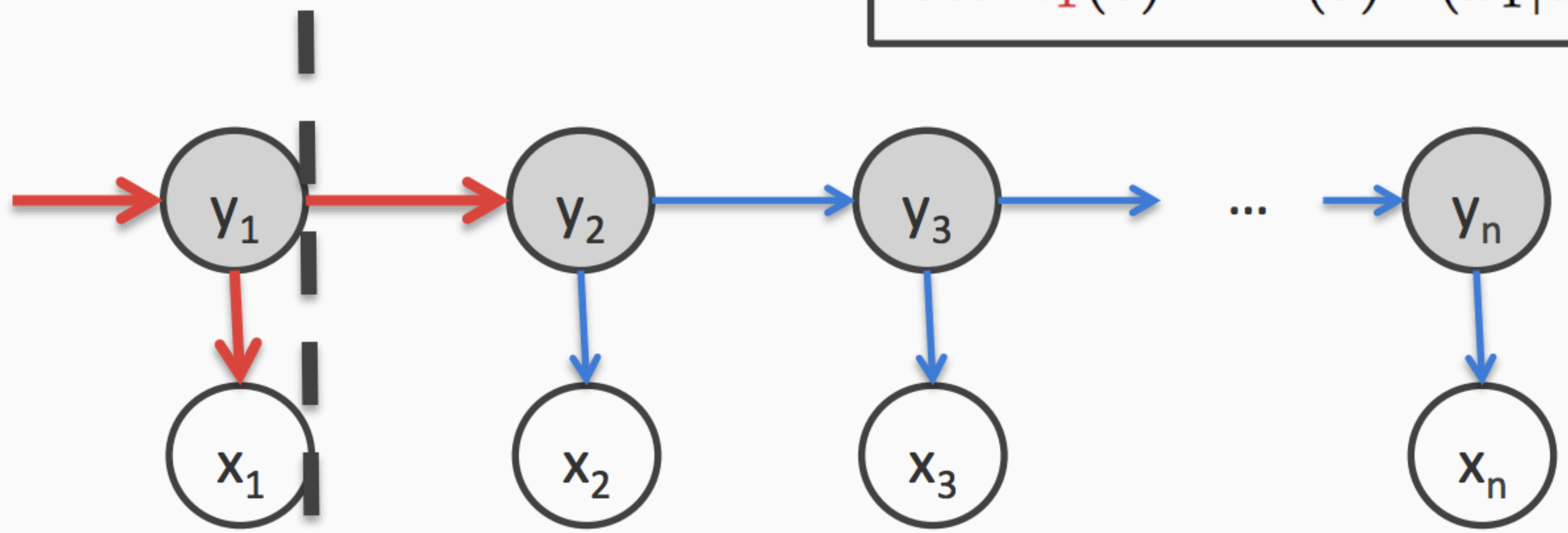
$$= \max_{y_2,\cdots,y_n} P(y_n|y_{n-1})P(x_n|y_n)\cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

$$= \max_{y_2,\cdots,y_n} P(y_n|y_{n-1})P(x_n|y_n)\cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)\text{score}_1(y_1)$$

best (partial) score for
a sequence ending in state *s*

Abstract away the score for all decisions till here into score

$$\boxed{\text{score}_1(s) = P(s)P(x_1|s)}$$



slide credit: Vivek Srikumar

Viterbi Algorithm

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

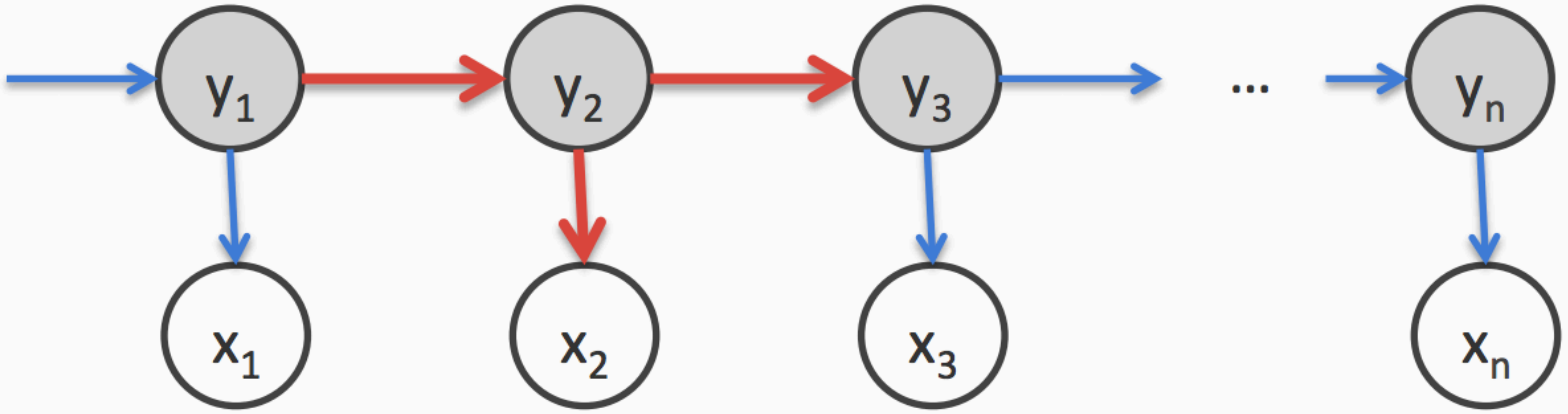
$$\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)\mathrm{score}_1(y_1)$$

$$= \max_{y_3, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2)P(x_3|y_3) \max_{y_1} P(y_2|y_1)P(x_2|y_2)\mathrm{score}_1(y_1)$$

Only terms that depend on y_2



slide credit: Vivek Srikumar

Viterbi Algorithm

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

$$\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

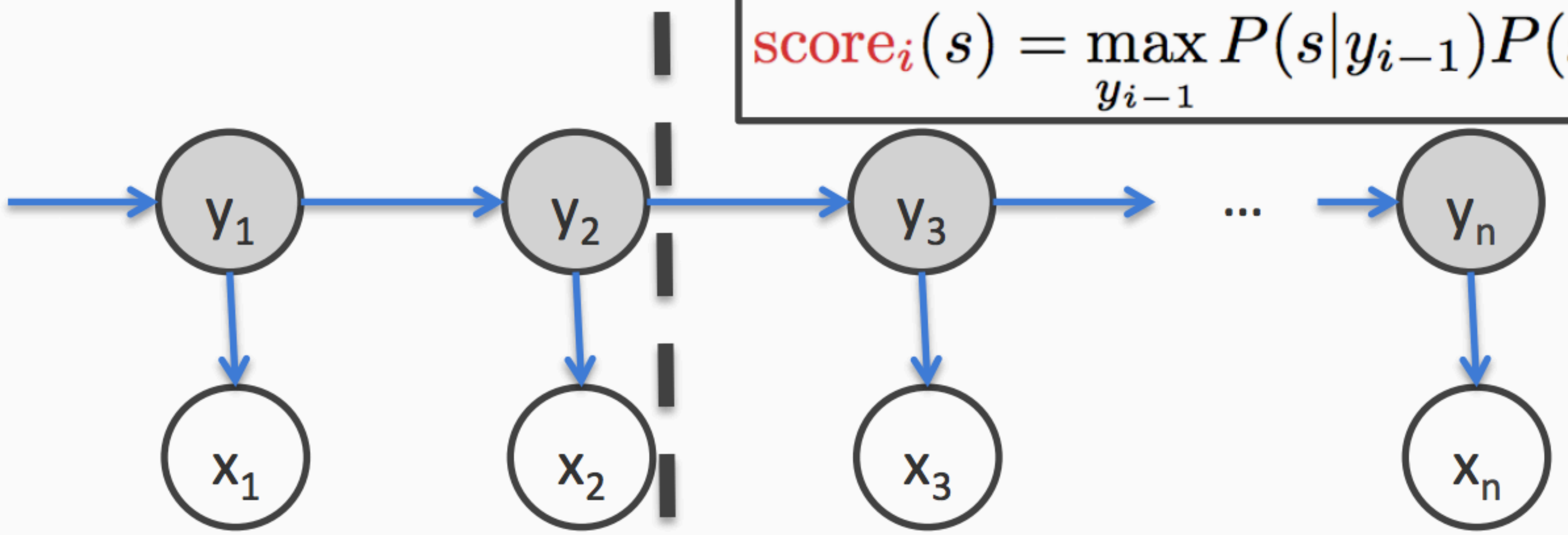
$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)\text{score}_1(y_1)$$

$$= \max_{y_3, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2)P(x_3|y_3) \max_{y_1} P(y_2|y_1)P(x_2|y_2)\text{score}_1(y_1)$$

$$= \max_{y_3, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2)P(x_3|y_3)\text{score}_2(y_2)$$

$$\boxed{\text{score}_i(s) = \max_{y_{i-1}} P(s|y_{i-1})P(x_i|s)\text{score}_{i-1}(y_{i-1})}$$



Abstract away the score for all decisions till here into score

slide credit: Vivek Srikumar

Viterbi Algorithm

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

$$\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

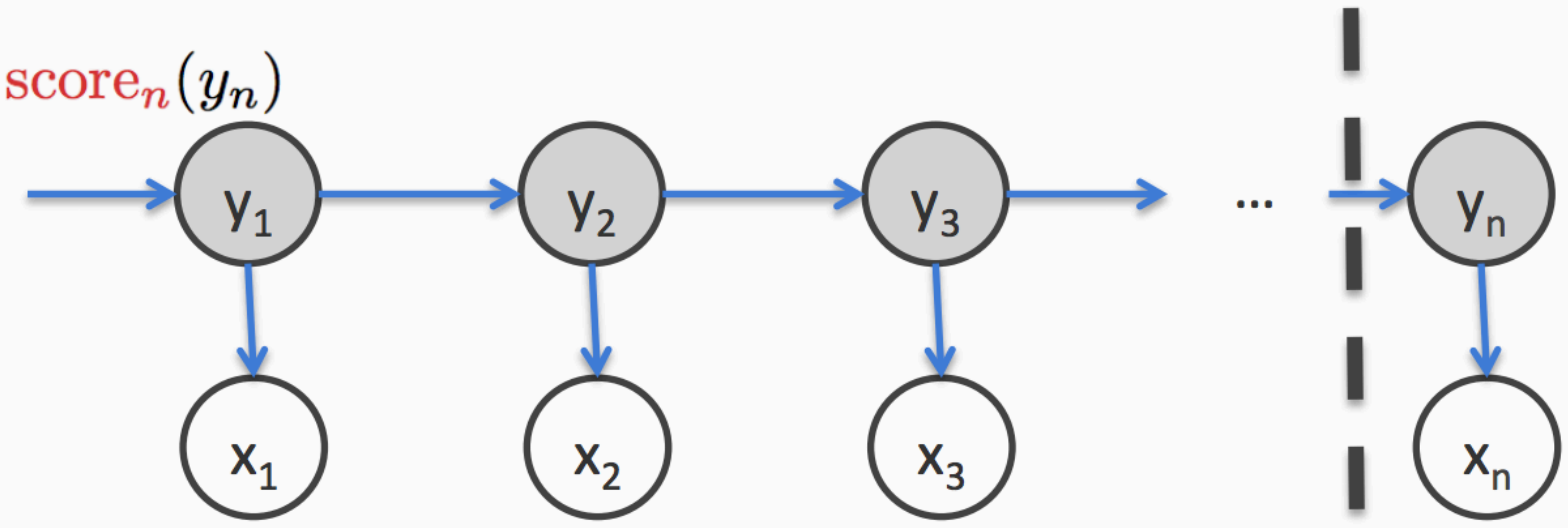
$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)\mathrm{score}_1(y_1)$$

$$= \max_{y_3, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2)P(x_3|y_3) \max_{y_1} P(y_2|y_1)P(x_2|y_2)\mathrm{score}_1(y_1)$$

$$= \max_{y_3, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2)P(x_3|y_3)\mathrm{score}_2(y_2)$$

$$\vdots$$

$$= \max_{y_n} \mathrm{score}_n(y_n)$$



Abstract away the score for all decisions till here into score

slide credit: Vivek Srikumar

Viterbi Algorithm

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

$$\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots P(y_2|y_1) P(x_2|y_2) P(y_1) P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1) P(x_2|y_2) P(y_1) P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1) P(x_2|y_2) \mathrm{score}_1(y_1)$$

$$= \max_{y_3, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2) P(x_3|y_3) \max_{y_1} P(y_2|y_1) P(x_2|y_2) \mathrm{score}_1(y_1)$$

$$= \max_{y_3, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2) P(x_3|y_3) \mathrm{score}_2(y_2)$$

$$\vdots$$

$$= \max_{y_n} \mathrm{score}_n(y_n)$$

$$\mathrm{score}_1(s) = P(s) P(x_1|s)$$

$$\mathrm{score}_i(s) = \max_{y_{i-1}} P(s|y_{i-1}) P(x_i|s) \mathrm{score}_{i-1}(y_{i-1})$$

slide credit: Vivek Srikumar

Viterbi Algorithm

1. Initial: For each state s, calculate

$$\text{score}_1(s) = P(s)P(x_1|s) = \pi_s B_{x_1,s}$$

2. Recurrence: For i = 2 to n, for every state s, calculate

$$\text{score}_i(s) = \max_{y_{i-1}} P(s|y_{i-1})P(x_i|s)\text{score}_{i-1}(y_{i-1})$$

$$= \max_{y_{i-1}} A_{y_{i-1},s} B_{s,x_i} \text{score}_{i-1}(y_{i-1})$$

3. Final state: calculate

$$\max_{\mathbf{y}} P(\mathbf{y}, \mathbf{x}|\pi, A, B) = \max_{s} \text{score}_n(s)$$

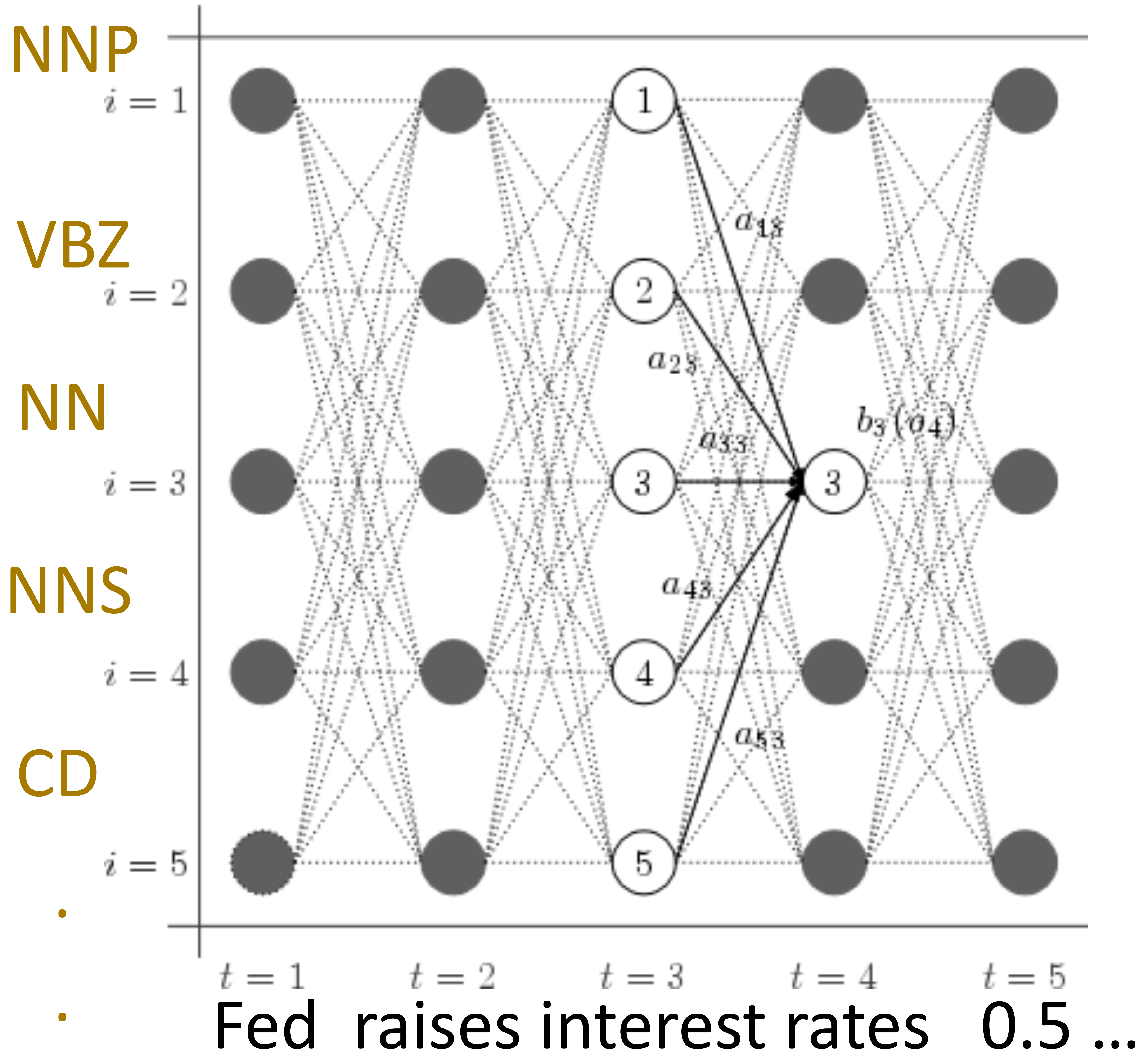
π: Initial probabilities
A: Transitions
B: Emissions

This only calculates the max. To get final answer (*argmax*),

- keep track of which state corresponds to the max at each step
- build the answer using these back pointers

slide credit: Vivek Srikumar

Viterbi Algorithm

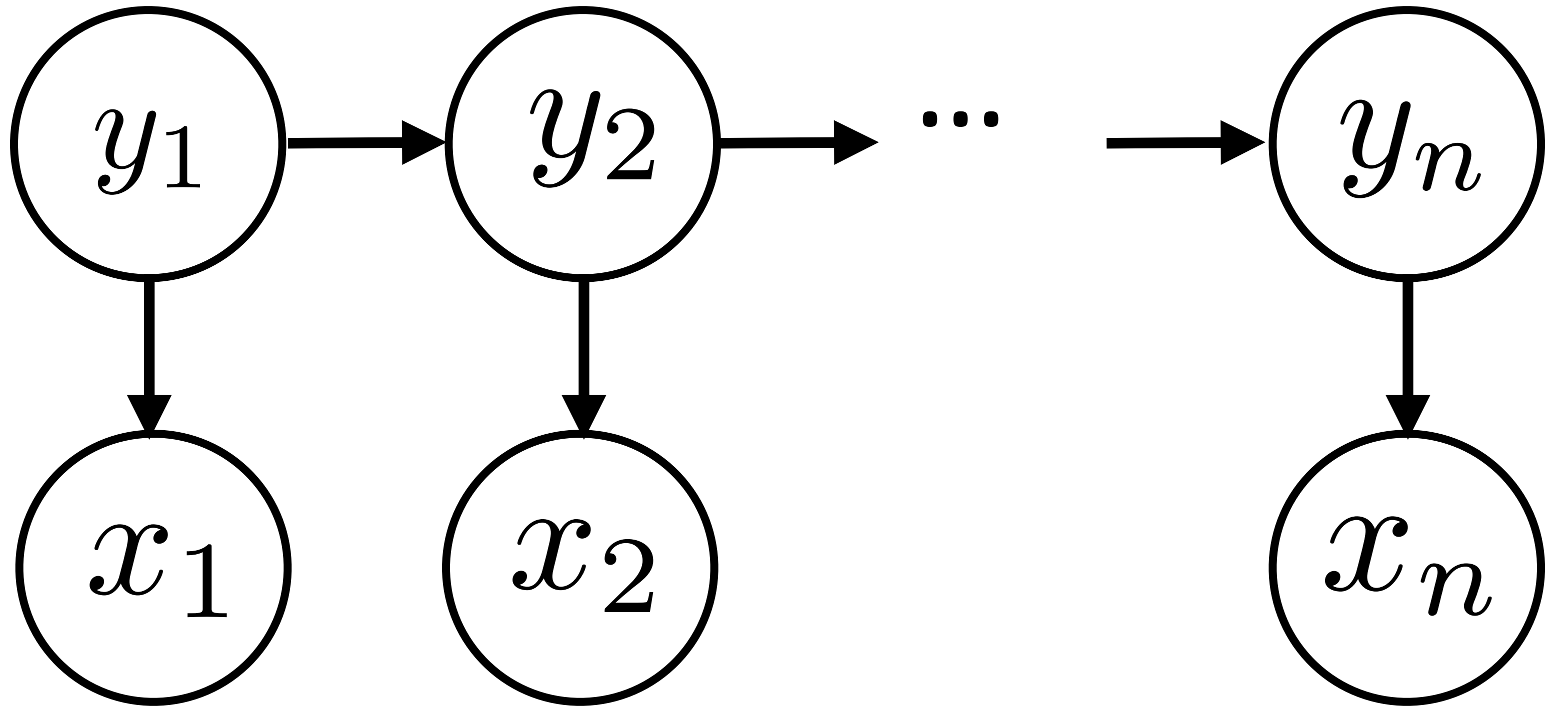


- "Think about" all possible immediate prior state values. Everything before that has already been accounted for by earlier stages.
- Compute scores for next step (score of optimal tag sequence ending with tag *i* at the *t*-th step/word).

slide credit: Dan Klein

Summary: HMMs

- Input $\mathbf{x} = (x_1, ..., x_n)$ Output $\mathbf{y} = (y_1, ..., y_n)$



$$P(\mathbf{y}, \mathbf{x}) = P(y_1) \prod_{i=2}^{n} P(y_i | y_{i-1}) \prod_{i=1}^{n} P(x_i | y_i)$$

- Training: maximum likelihood estimation (with smoothing)
- Inference problem: $\text{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = \text{argmax}_{\mathbf{y}} \frac{P(\mathbf{y}, \mathbf{x})}{\cancel{P(\mathbf{x})}}$
- Viterbi: $\text{score}_i(s) = \max_{y_{i-1}} P(s|y_{i-1}) P(x_i|s) \text{score}_{i-1}(y_{i-1})$

Andrew Viterbi, 1967

HMM POS Tagging

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

- Normal HMM "bigram" model: y_1 = NNP, y_2 = VBZ, …
- Trigram model: y_1 = (<S>, NNP), y_2 = (NNP, VBZ), …
- Probabilities now looks like:

With more context!

  - P((NNP, VBZ) | (<S>, NNP)) — verb is occurring two words after <S>
  - P((VBZ, NN) | (NNP, VBZ)) — Noun-verb-noun S-V-O

- Tradeoff between model capacity and data size — trigrams are a "sweet spot" for POS tagging

HMM POS Tagging

- Dataset: Penn Treebank English Corpus (44 POS tags)
- 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
- MaxEnt tagger (Toutanova + Manning 2000): 96.9% / 87.0% on unks
- State-of-the-art (BiLSTM-CRFs, BERT): 97.5% / 89%+ on unks

Errors

gold label

	JJ	NN	NNP	NNPS	RB	RP	IN	VB	VBD	VBN	VBP	Total
JJ	0	**177**	**56**	0	**61**	2	5	10	15	**108**	0	488
NN	**244**	0	**103**	0	12	1	1	29	5	6	19	525
NNP	**107**	**106**	0	**132**	5	0	7	5	1	2	0	427
NNPS	1	0	**110**	0	0	0	0	0	0	0	0	142
RB	**72**	21	7	0	0	16	**138**	1	0	0	0	295
RP	0	0	0	0	**39**	0	**65**	0	0	0	0	104
IN	11	0	1	0	**169**	**103**	0	1	0	0	0	323
VB	17	**64**	9	0	2	0	1	0	4	7	**85**	189
VBD	10	5	3	0	0	0	0	3	0	**143**	2	166
VBN	**101**	3	3	0	0	0	0	3	**108**	0	1	221
VBP	5	34	3	1	1	0	2	**49**	6	3	0	104
Total	626	536	348	144	317	122	279	102	140	269	108	3651

Particle / Preposition or Subordinating Conjunction

Verb Past Tense / Verb Past Participles

JJ/NN NN
official knowledge

VBD RP/IN DT NN
made up the story

RB VBD/VBN NNS
recently sold shares

(NN NN: *tax cut, art gallery, …*)

Slide credit: Dan Klein / Toutanova + Manning (2000)

https://sites.google.com/site/partofspeechhelp/home/in_rp

Remaining Errors

- Lexicon gap (word not seen with that tag in training): 4.5% of errors
- Unknown word: 4.5%
- Could get right: 16% (many of these involve parsing!)
- Difficult linguistics: 20%

VBD / VBP? (past or present?)

They set up absurd situations, detached from reality

- Underspecified / unclear, gold standard inconsistent / wrong: **58%**

adjective or verbal participle? JJ / VBN?

a $ 10 million fourth-quarter charge against discontinued operations

Manning 2011 "Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics?"

POS with Feedforward Networks

- Part-of-speech tagging with FFNNs

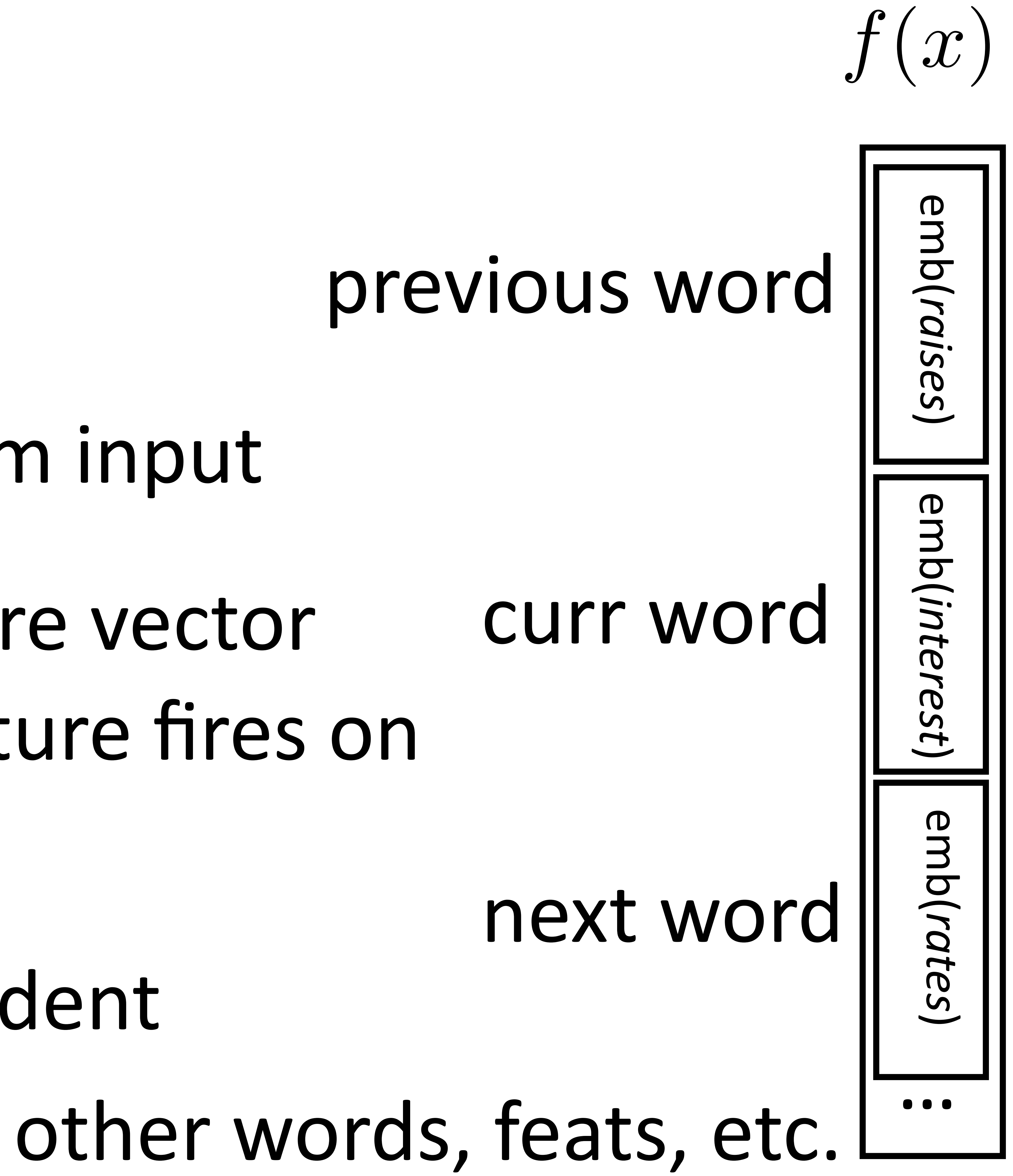
??

Fed raises ***interest*** *rates in order to …*

- Word embeddings for each word form input

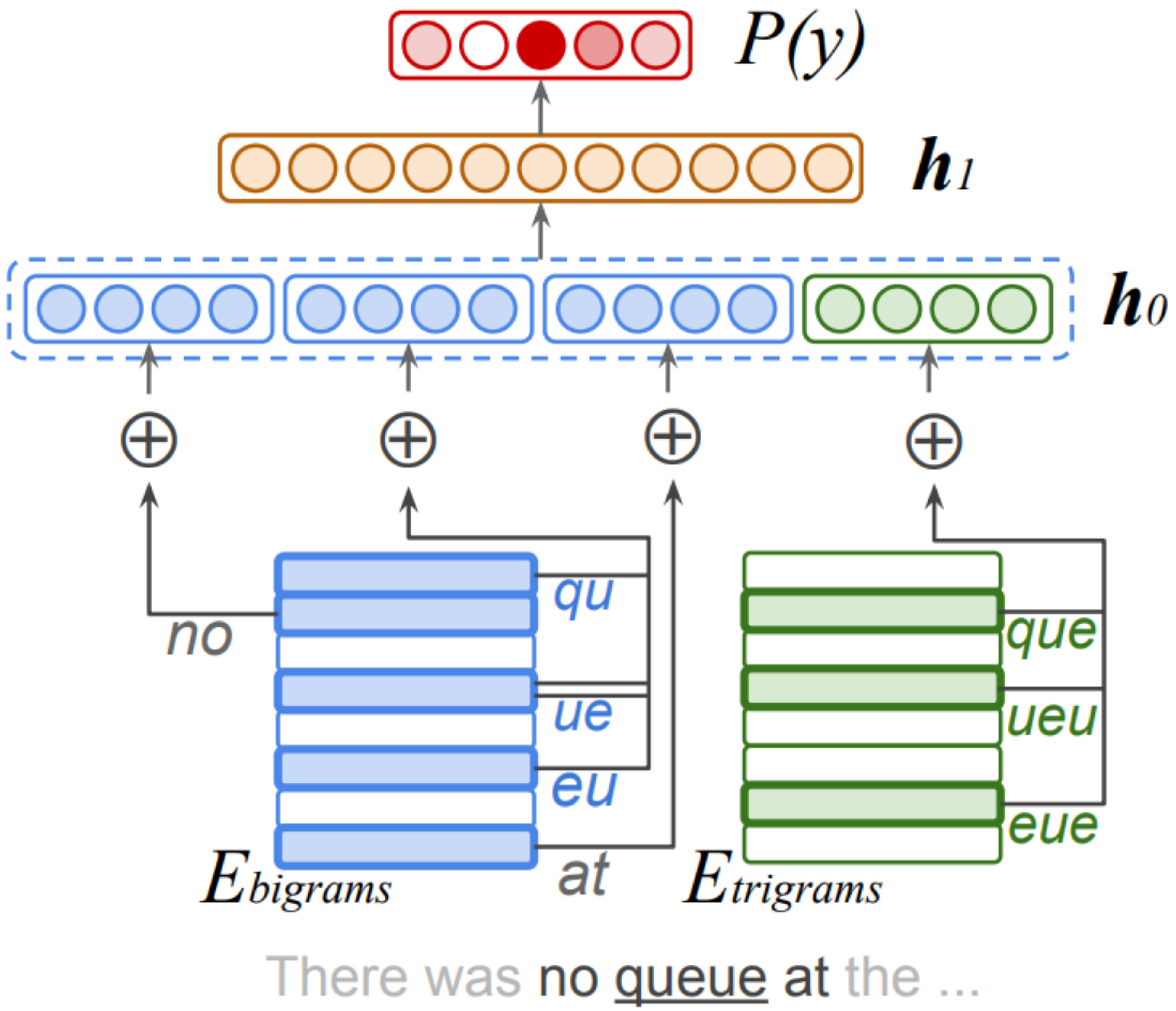
- ~1000 features here — smaller feature vector than in sparse models, but every feature fires on every example

- Weight matrix learns position-dependent processing of the words



Botha et al. (2017)

POS with Feedforward Networks



- Hidden layer mixes these different signals and learns feature conjunctions

Botha et al. (2017)

POS with Feedforward Networks

- Multilingual tagging results:

Model	Acc.	Wts.	MB	Ops.
Gillick et al. (2016)	95.06	900k	-	6.63m
Small FF	94.76	241k	0.6	0.27m
+Clusters	95.56	261k	1.0	0.31m
$\frac{1}{2}$ Dim.	95.39	143k	0.7	0.18m

- Gillick et al. (2016) used LSTMs; this is smaller, faster, and better

Botha et al. (2017)

Other Languages

sentence:	The	oboist	Heinz	Holliger	has	taken	a	hard	line	about	the	problems	.
original:	DT	NN	NNP	NNP	VBZ	VBN	DT	JJ	NN	IN	DT	NNS	.
universal:	DET	NOUN	NOUN	NOUN	VERB	VERB	DET	ADJ	NOUN	ADP	DET	NOUN	.

Figure 1: Example English sentence with its language specific and corresponding universal POS tags.

Language	Source	# Tags	O/O	U/U	O/U
Arabic	PADT/CoNLL07 (Hajič et al., 2004)	21	96.1	96.9	97.0
Basque	Basque3LB/CoNLL07 (Aduriz et al., 2003)	64	89.3	93.7	93.7
Bulgarian	BTB/CoNLL06 (Simov et al., 2002)	54	95.7	97.5	97.8
Catalan	CESS-ECE/CoNLL07 (Martí et al., 2007)	54	98.5	98.2	98.8
Chinese	Penn ChineseTreebank 6.0 (Palmer et al., 2007)	34	91.7	93.4	94.1
Chinese	Sinica/CoNLL07 (Chen et al., 2003)	294	87.5	91.8	92.6
Czech	PDT/CoNLL07 (Böhmová et al., 2003)	63	99.1	99.1	99.1
Danish	DDT/CoNLL06 (Kromann et al., 2003)	25	96.2	96.4	96.9
Dutch	Alpino/CoNLL06 (Van der Beek et al., 2002)	12	93.0	95.0	95.0
English	PennTreebank (Marcus et al., 1993)	45	96.7	96.8	97.7
French	FrenchTreebank (Abeillé et al., 2003)	30	96.6	96.7	97.3
German	Tiger/CoNLL06 (Brants et al., 2002)	54	97.9	98.1	98.8
German	Negra (Skut et al., 1997)	54	96.9	97.9	98.6
Greek	GDT/CoNLL07 (Prokopidis et al., 2005)	38	97.2	97.5	97.8
Hungarian	Szeged/CoNLL07 (Csendes et al., 2005)	43	94.5	95.6	95.8
Italian	ISST/CoNLL07 (Montemagni et al., 2003)	28	94.9	95.8	95.8
Japanese	Verbmobil/CoNLL06 (Kawata and Bartels, 2000)	80	98.3	98.0	99.1
Japanese	Kyoto4.0 (Kurohashi and Nagao, 1997)	42	97.4	98.7	99.3
Korean	Sejong (http://www.sejong.or.kr)	187	96.5	97.5	98.4
Portuguese	Floresta Sintá(c)tica/CoNLL06 (Afonso et al., 2002)	22	96.9	96.8	97.4
Russian	SynTagRus-RNC (Boguslavsky et al., 2002)	11	96.8	96.8	96.8
Slovene	SDT/CoNLL06 (Džeroski et al., 2006)	29	94.7	94.6	95.3
Spanish	Ancora-Cast3LB/CoNLL06 (Civit and Martí, 2004)	47	96.3	96.3	96.9
Swedish	Talbanken05/CoNLL06 (Nivre et al., 2006)	41	93.6	94.7	95.1
Turkish	METU-Sabanci/CoNLL07 (Oflazer et al., 2003)	31	87.5	89.1	90.2

Petrov et al. (2012)

Other Languages

Language	CRF+	CRF	BTS	BTS*
Bulgarian	97.97	97.00	97.84	97.02
Czech	98.38	98.00	98.50	98.44
Danish	95.93	95.06	95.52	92.45
German	93.08	91.99	92.87	92.34
Greek	97.72	97.21	97.39	96.64
English	95.11	94.51	93.87	94.00
Spanish	96.08	95.03	95.80	95.26
Farsi	96.59	96.25	96.82	96.76
Finnish	94.34	92.82	95.48	96.05
French	96.00	95.93	95.75	95.17
Indonesian	92.84	92.71	92.85	91.03
Italian	97.70	97.61	97.56	97.40
Swedish	96.81	96.15	95.57	93.17
AVERAGE	96.04	95.41	95.85	95.06

Byte-to-Span

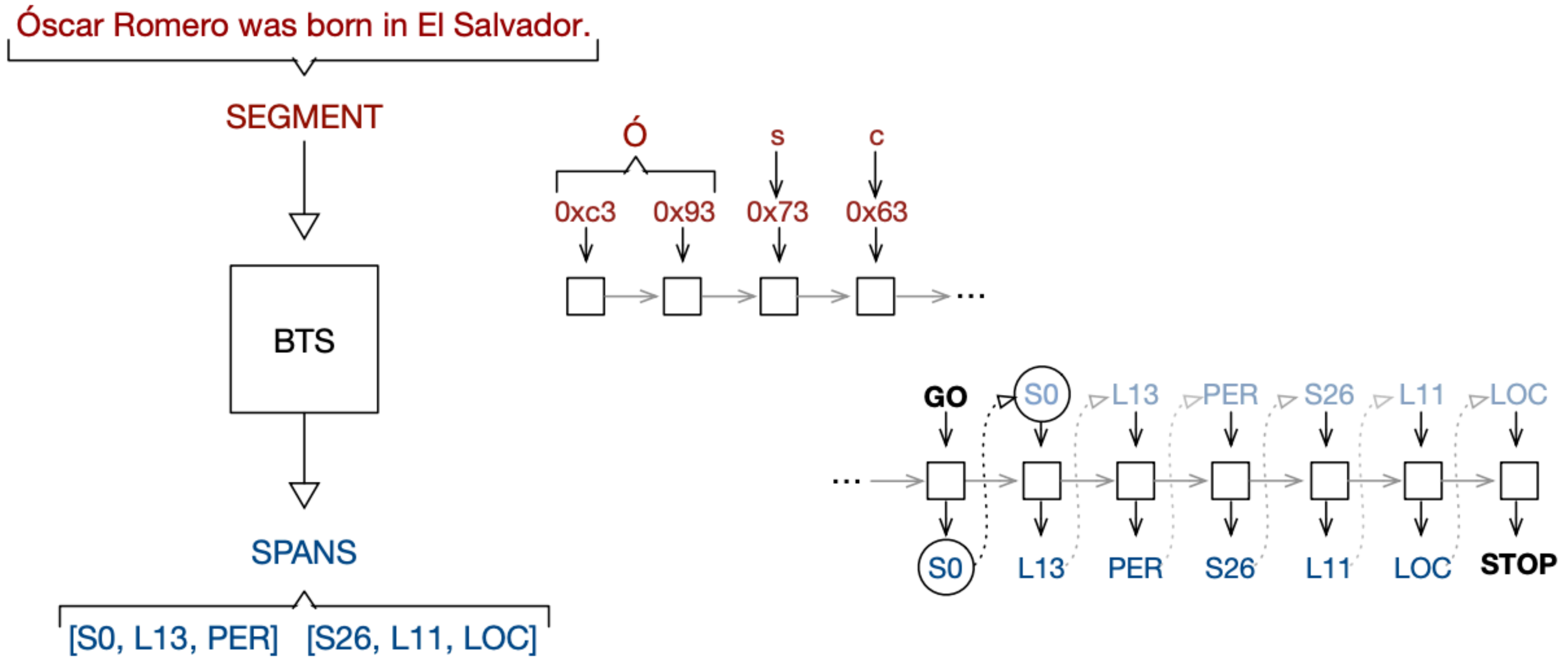


Figure 1: A diagram showing the way the Byte-to-Span (BTS) model converts an input text segment to a sequence of span annotations. The model reads the input segment one byte at a time (this can involve multibyte unicode characters), then a special Generate Output (GO) symbol, then produces the argmax output of a softmax over all possible start positions, lengths, and labels (as well as STOP, signifying no additional outputs). The prediction from the previous time step is fed as an input to the next time step.

- Universal POS tagset (~12 tags), cross-lingual model works as well as tuned CRF using external resources

Gillick et az. (2016)

Forward-Backward Algorithm

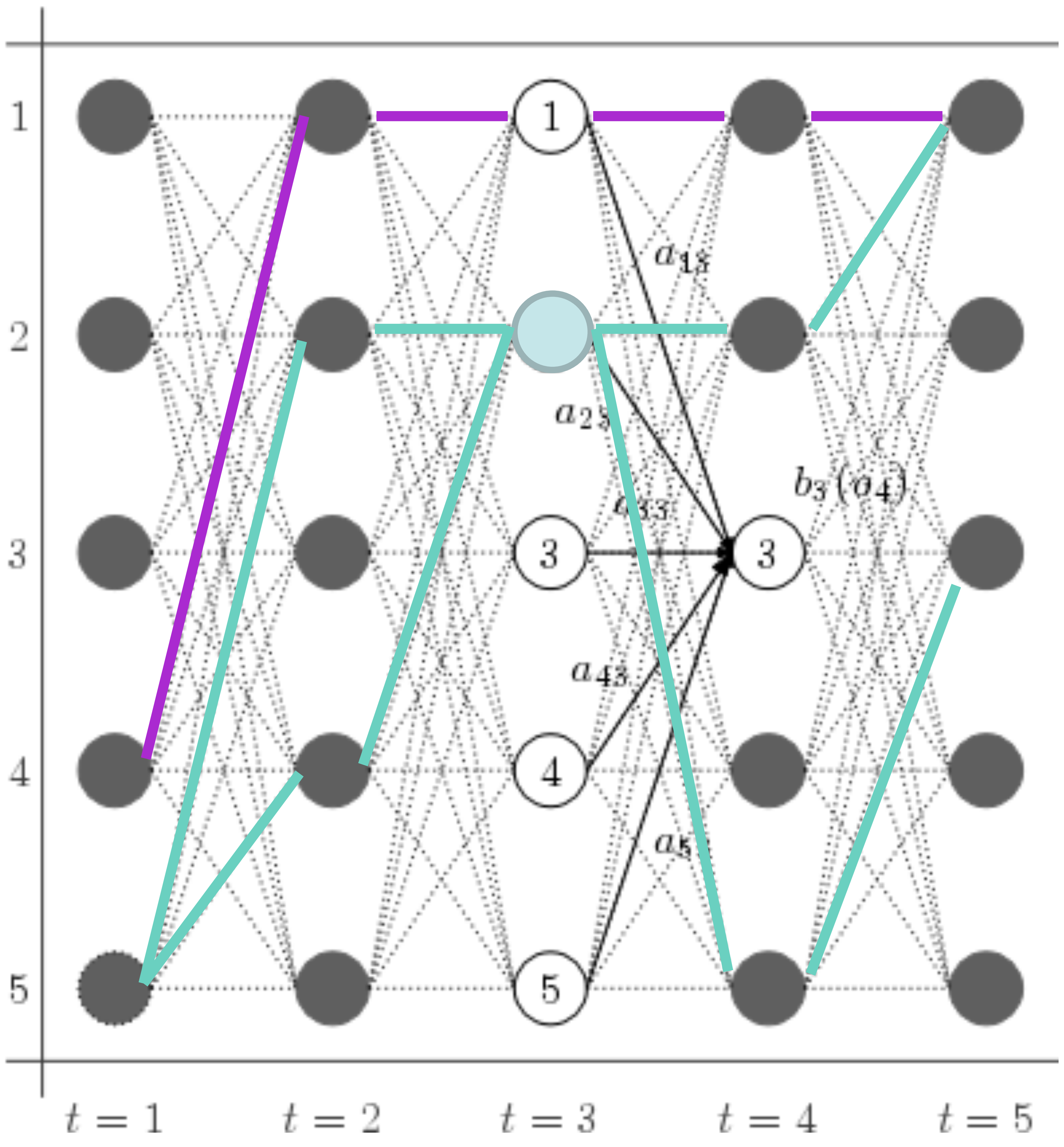
- What did Viterbi compute? $P(\mathbf{y}_{\max}|\mathbf{x}) = \max_{y_1,\ldots,y_n} P(\mathbf{y}|\mathbf{x})$

- In addition to finding the best path, we may want to compute marginal probabilities of paths $P(y_i = s|\mathbf{x})$

$$P(y_i = s|\mathbf{x}) = \sum_{y_1,\ldots,y_{i-1},y_{i+1},\ldots,y_n} P(\mathbf{y}|\mathbf{x})$$

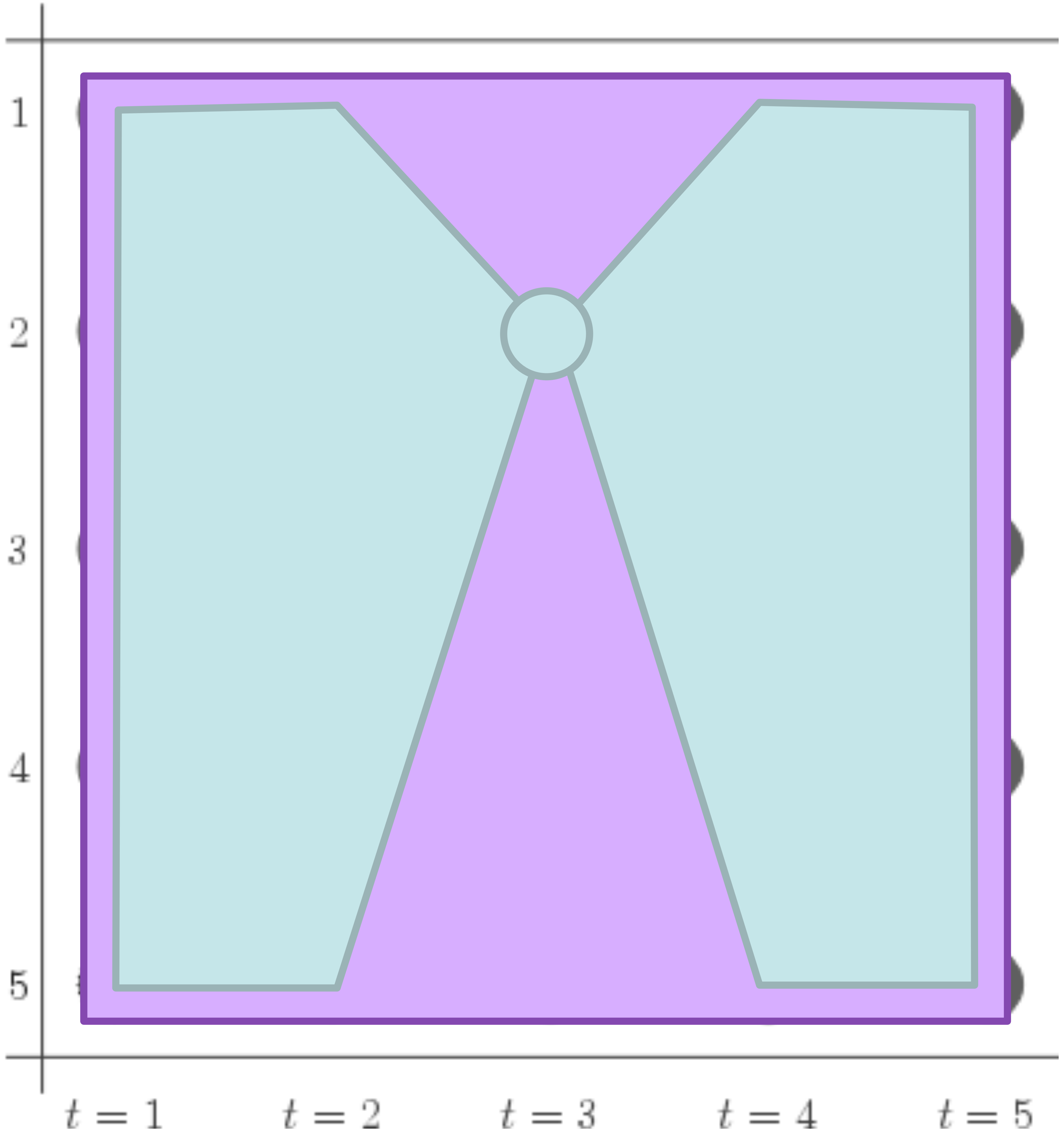
- Can compute marginals with dynamic programming as well using an algorithm called forward-backward

Forward-Backward Algorithm

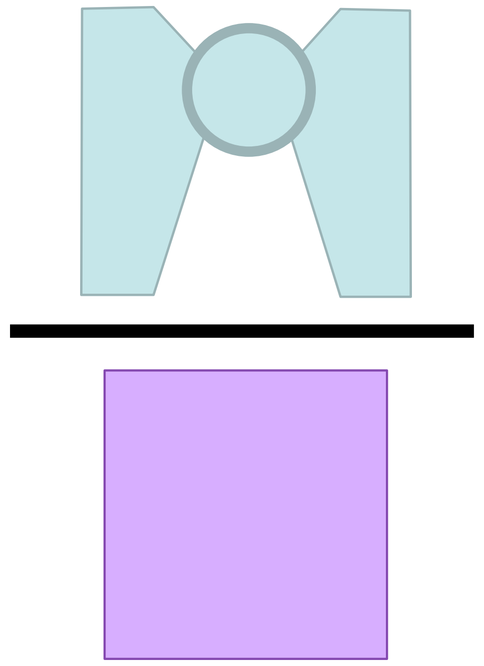


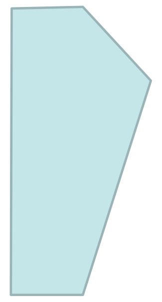
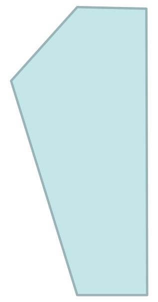
$$P(y_3 = 2|\mathbf{x}) = \frac{\text{sum of all paths through state 2 at time 3}}{\text{sum of all paths}}$$

Forward-Backward Algorithm



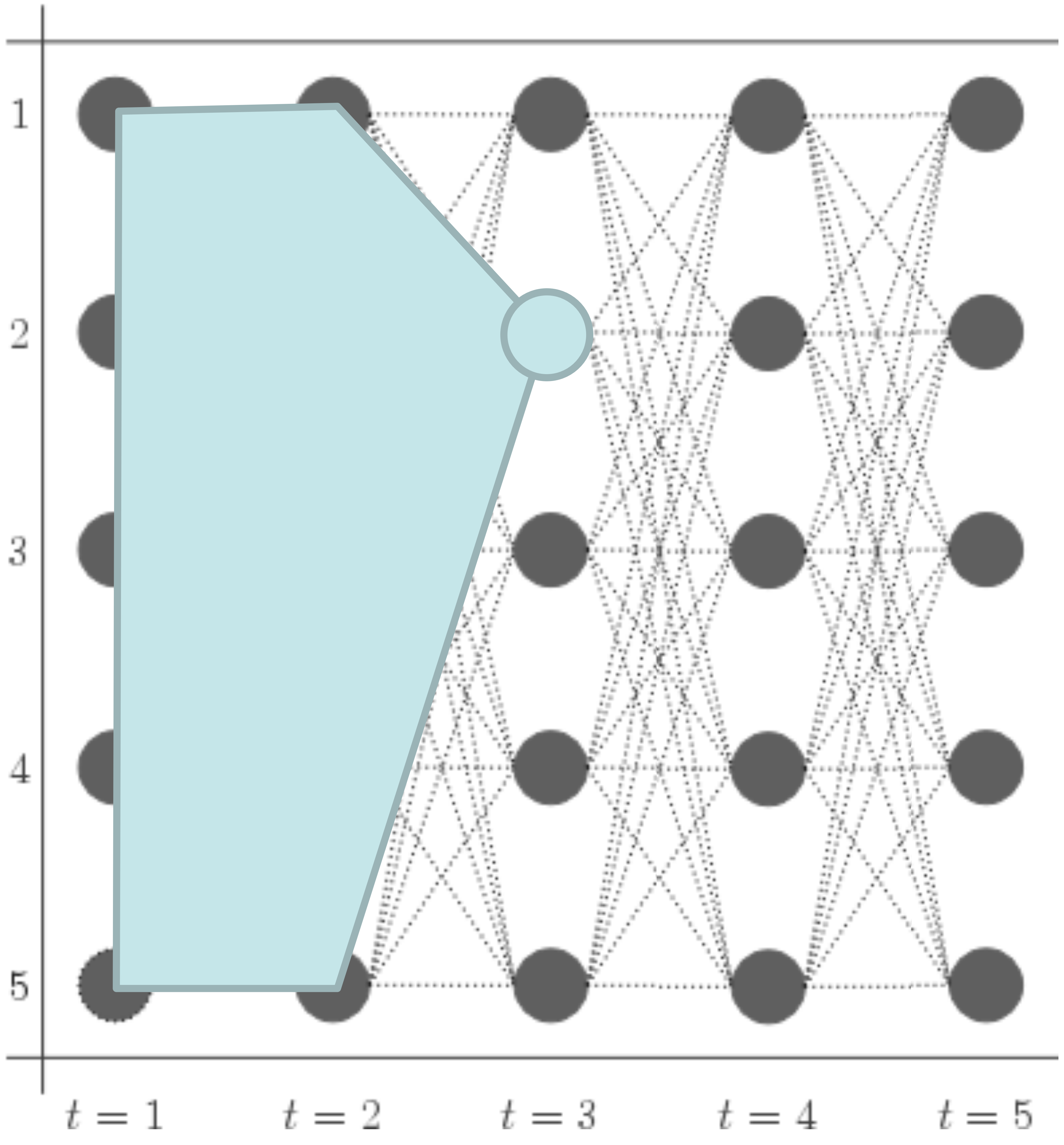
$$P(y_3 = 2|\mathbf{x}) = \frac{\text{sum of all paths through state 2 at time 3}}{\text{sum of all paths}}$$

= 

- Easiest and most flexible to do one pass to compute  and one to compute 

slide credit: Dan Klein

Forward-Backward Algorithm



- Initial:

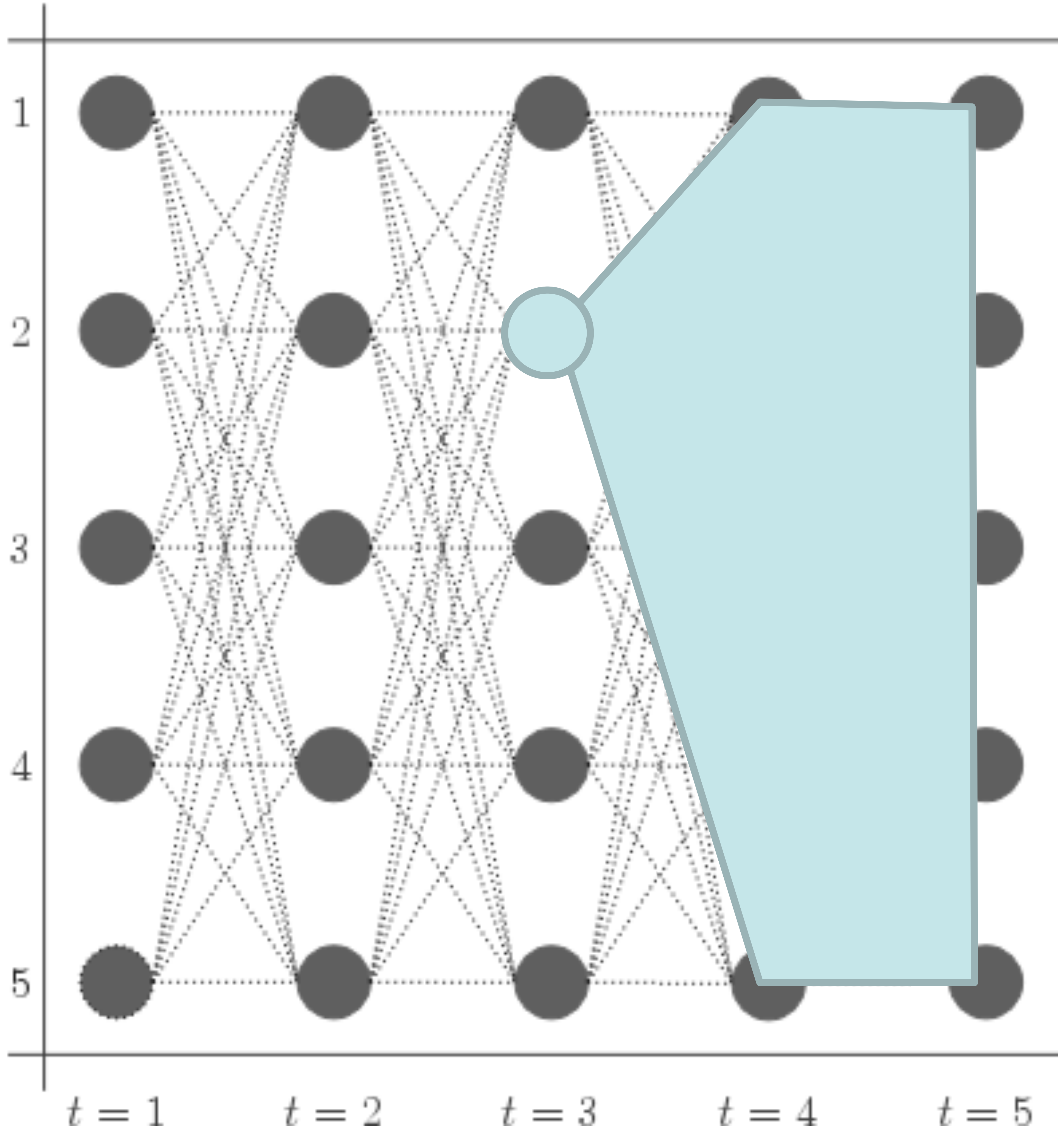
$$\alpha_1(s) = P(s)P(x_1|s)$$

- Recurrence:

$$\alpha_t(s_t) = \sum_{s_{t-1}} \alpha_{t-1}(s_{t-1})P(s_t|s_{t-1})P(x_t|s_t)$$

- Same as Viterbi but summing instead of maxing!
- These quantities get very small! Store everything as log probabilities

Forward-Backward Algorithm



- Initial:

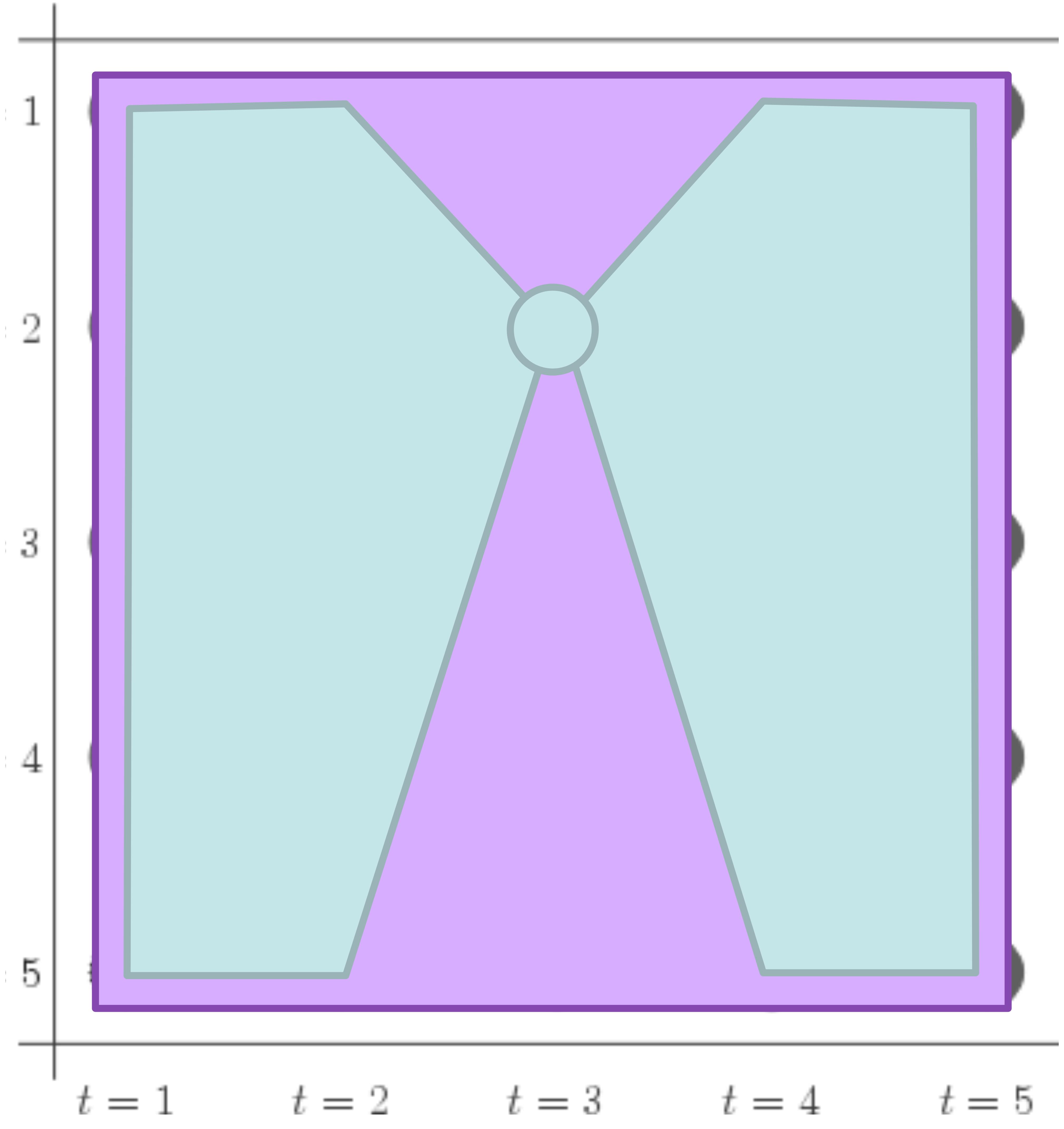
$$\beta_n(s) = 1$$

- Recurrence:

$$\beta_t(s_t) = \sum_{s_{t+1}} \beta_{t+1}(s_{t+1}) P(s_{t+1}|s_t) \boxed{P(x_{t+1}|s_{t+1})}$$

- Big differences: count emission for the *next* timestep (not current one)

Forward-Backward Algorithm



$$\alpha_1(s) = P(s)P(x_1|s)$$

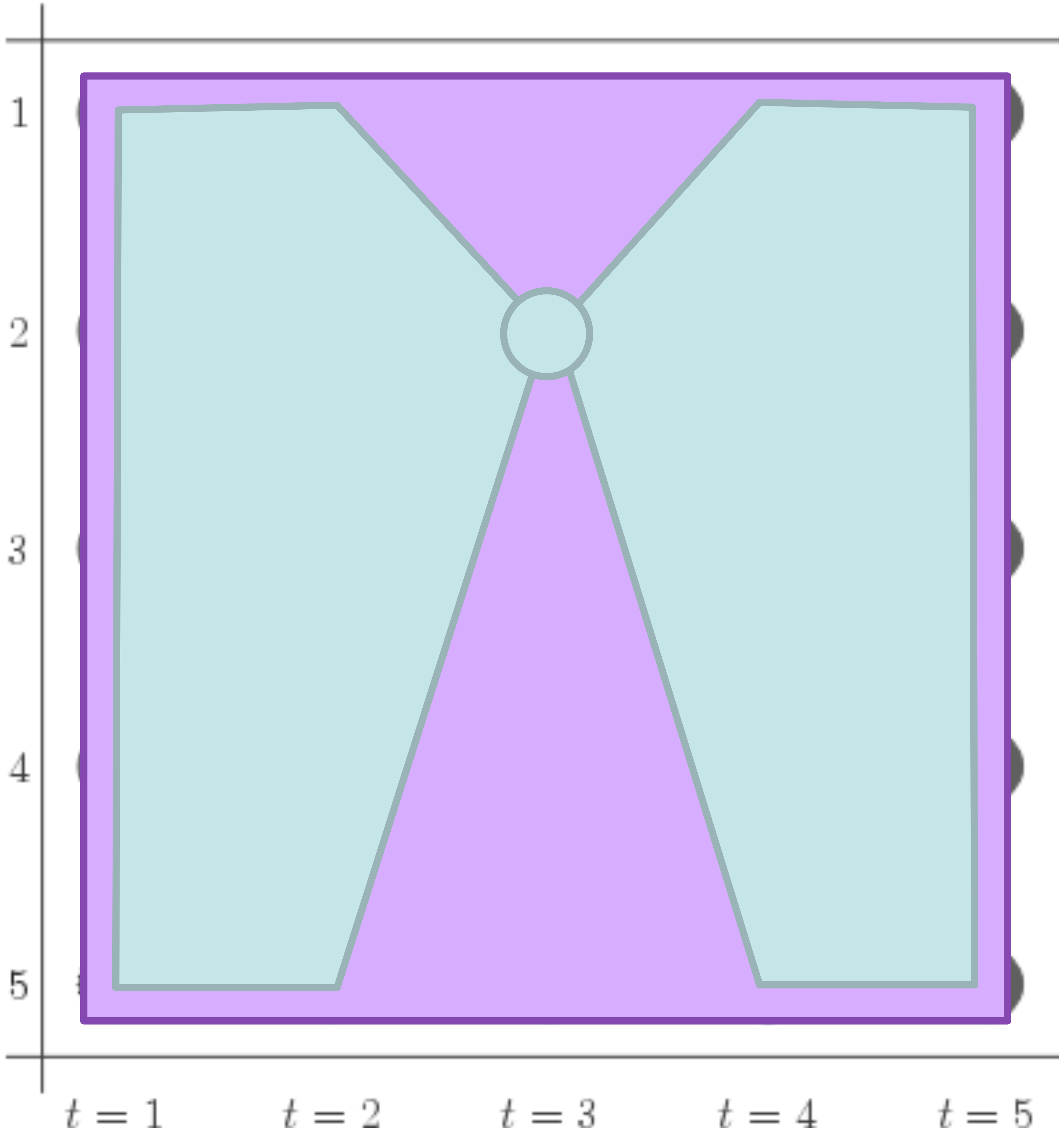
$$\alpha_t(s_t) = \sum_{s_{t-1}} \alpha_{t-1}(s_{t-1})P(s_t|s_{t-1})P(x_t|s_t)$$

$$\beta_n(s) = 1$$

$$\beta_t(s_t) = \sum_{s_{t+1}} \beta_{t+1}(s_{t+1})P(s_{t+1}|s_t)\boxed{P(x_{t+1}|s_{t+1})}$$

- Big differences: count emission for the *next* timestep (not current one)

Forward-Backward Algorithm



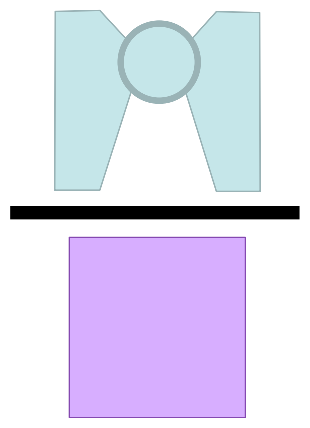
$$\alpha_1(s) = P(s)P(x_1|s)$$

$$\alpha_t(s_t) = \sum_{s_{t-1}} \alpha_{t-1}(s_{t-1})P(s_t|s_{t-1})P(x_t|s_t)$$

$$\beta_n(s) = 1$$

$$\beta_t(s_t) = \sum_{s_{t+1}} \beta_{t+1}(s_{t+1})P(s_{t+1}|s_t)\boxed{P(x_{t+1}|s_{t+1})}$$

$$P(s_3 = 2|\mathbf{x}) = \frac{\alpha_3(2)\beta_3(2)}{\sum_i \alpha_3(i)\beta_3(i)} =$$



- What is the denominator here? $P(\mathbf{x})$

Next Up

- More sequential models
  - CRFs: feature-based discriminative models
    - sequential as HMM + ability to use rich features as in LR
  - Named entity recognition