CSCI 544: Applied Natural Language Processing

Sequence Labeling-I

Xuezhe Ma (Max)

USC University of Southern California

Logistical Notes

- **Project Topics from TAs**
  - https://docs.google.com/spreadsheets/d/1LtWwJXAFZcikdM7R_hW_EipqqPNQiFCDjKfwUdARTg4/edit?usp=sharing
- **PyTorch Lecture**
  - This Thursday (01/30/2025)

Course Organization

NLP Tasks

Models	Text Classification	Sequence Labeling	Syntactic Parsing	Seq2seq & MT	Language Modeling & Generation
	Naive Bayes Logistic Regression SVM …	HMMs MEMMs CRFs	Tree-structured Models/Algorithms	……	……
	Word Embedding & Neural Networks	RNN LSTM	Neural Structured Networks	Attention & Transformers	Generative Models & LLMs

Overview

- **The Sequence Labeling Problem**
  - General Structured Prediction Tasks
  - Part-of-speech Tagging: A case study
  - Generative Models vs. Discriminative Models
- **Hidden Markov Model (HMM)**
  - Basic definitions
  - Parameter estimation
  - The Viterbi algorithm
- **Log-Linear Models**
  - Maximum Entropy Markov Models (MEMMs)
  - Conditional Random Fields (CRFs)

Overview

- **The Sequence Labeling Problem**
  - General Structured Prediction Tasks
  - Part-of-speech Tagging: A case study
  - Generative Models vs. Discriminative Models
- **Hidden Markov Model (HMM)**
  - Basic definitions
  - Parameter estimation
  - The Viterbi algorithm
- **Log-Linear Models**
  - Maximum Entropy Markov Models (MEMMs)
  - Conditional Random Fields (CRFs)

Structured Prediction

What is Structured Prediction

- **Unstructured Prediction**
  - Output Y consists of a single component

X

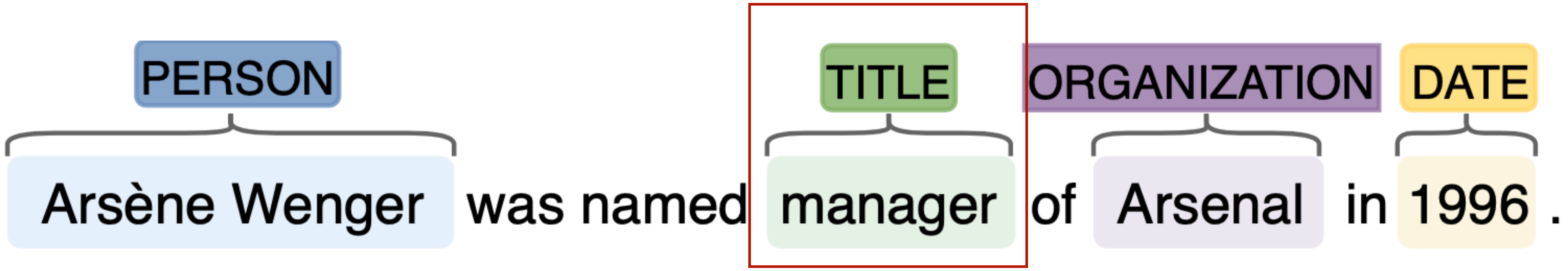
→ {Cat, Dog, Finch, Owl}

Y

Structured Prediction

- Y consists of **multiple components** $Y = \{y_1, y_2, \ldots, y_n\}$

Named Entity Recognition



$Y_i = <$ **manager** $\rightarrow$ **Title** $>$

Structured Prediction

- Y consists of multiple components $Y = \{y_1, y_2, \ldots, y_n\}$

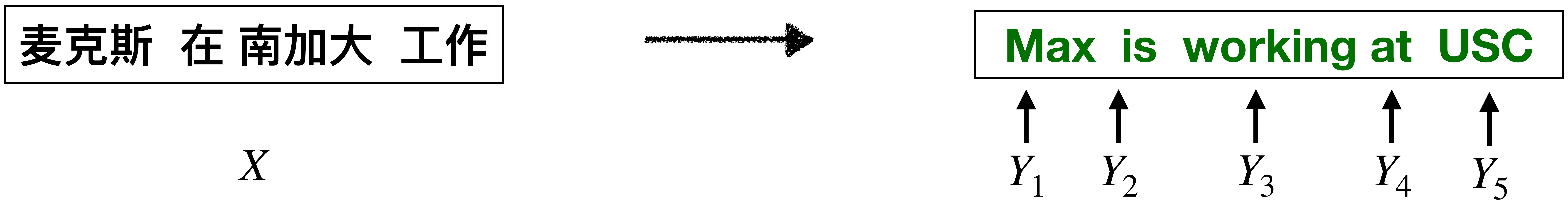
麦克斯 在 南加大 工作 $\longrightarrow$ Max is working at USC

X

Y_1 Y_2 Y_3 Y_4 Y_5

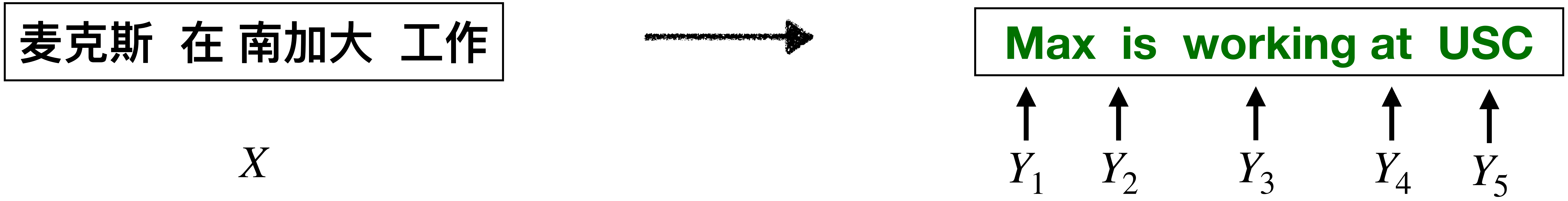
Structured Prediction

- Y consists of multiple components $Y = \{y_1, y_2, \ldots, y_n\}$
- (Strong) correlations between output components



Structured Prediction

- Y consists of multiple components $Y = \{y_1, y_2, \ldots, y_n\}$
- (Strong) correlations between output components
- Exponential output space
  - Decoding: $y^* = \mathrm{argmax}_{y \in \mathcal{Y}} p(y \mid x)$

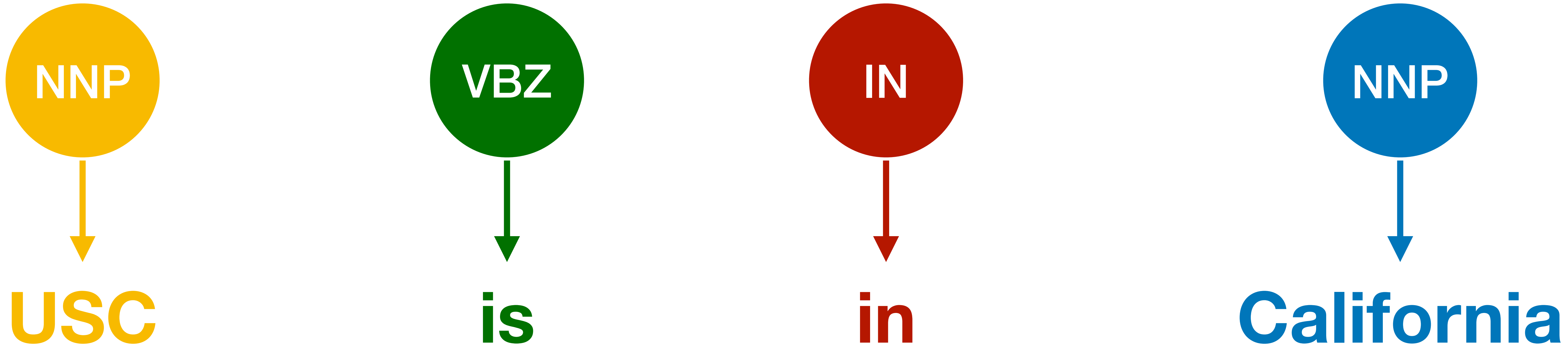


What is Sequence Labeling?

A type of structured prediction tasks

$$Y = < y_i, y_2, \ldots, y_n >$$

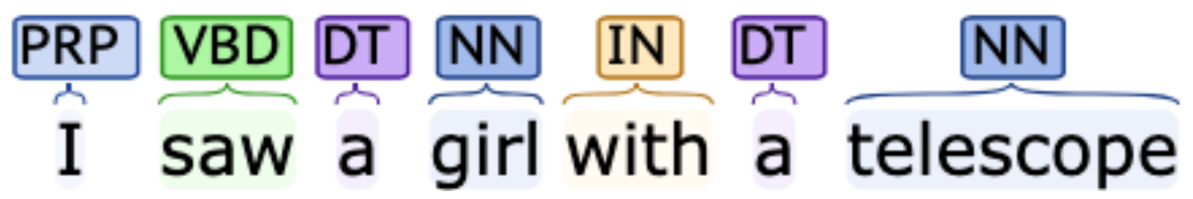
$$X = < x_i, x_2, \ldots, x_n >$$



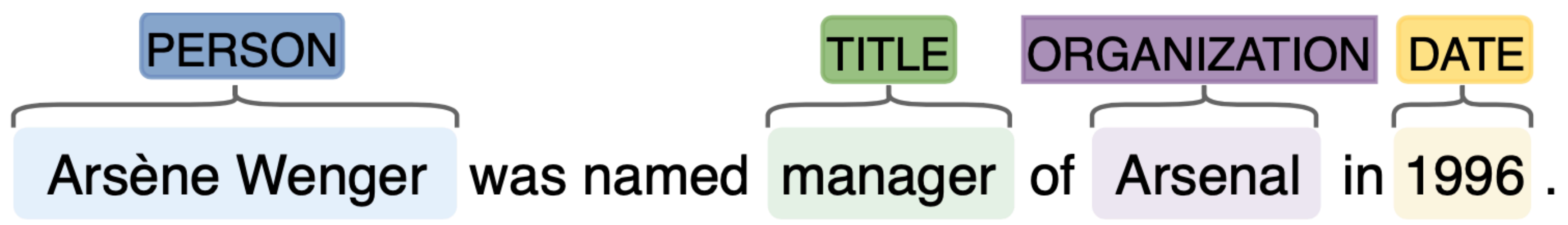
Assigning each token of X, e.g. x_i a corresponding label y_i

Why Sequence Labeling?

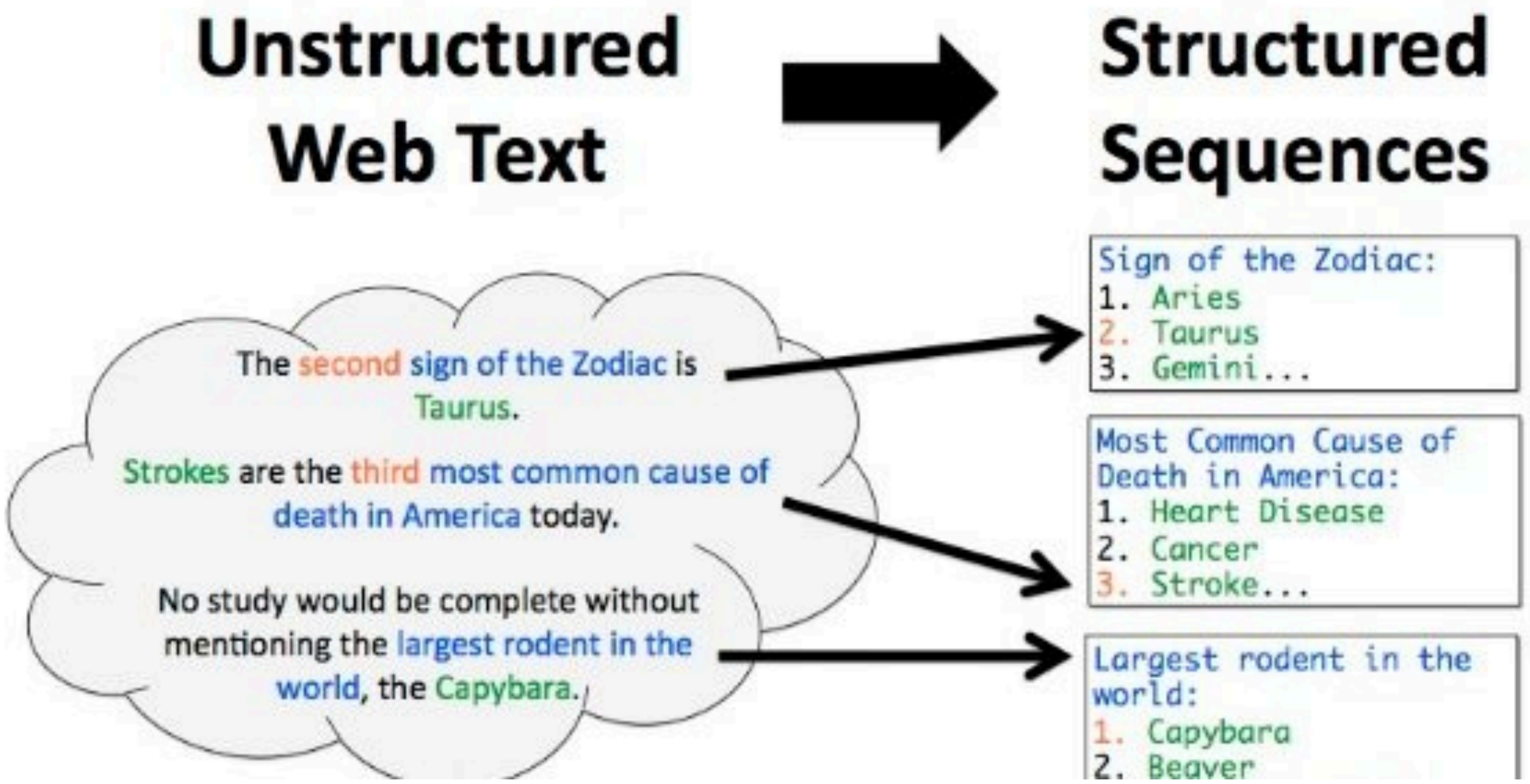
Part-of-Speech Tagging



Named Entity Recognition

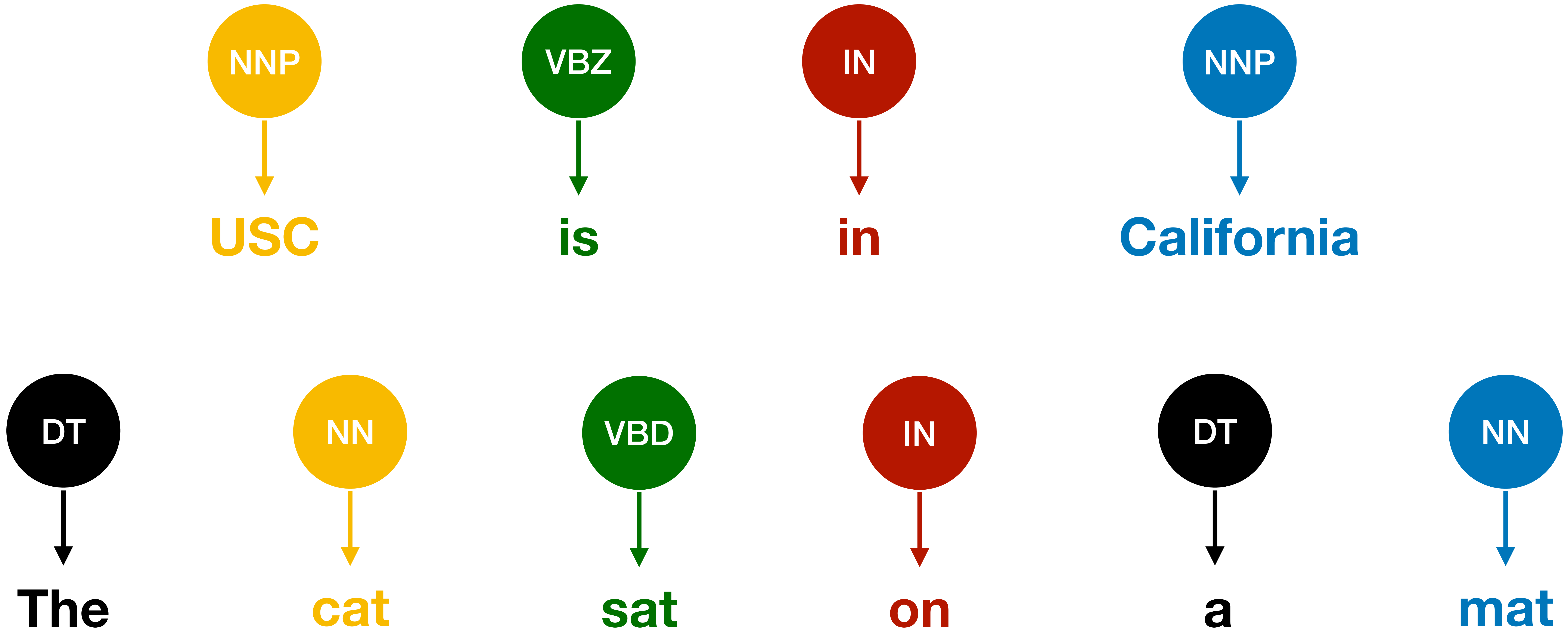


Information Extraction



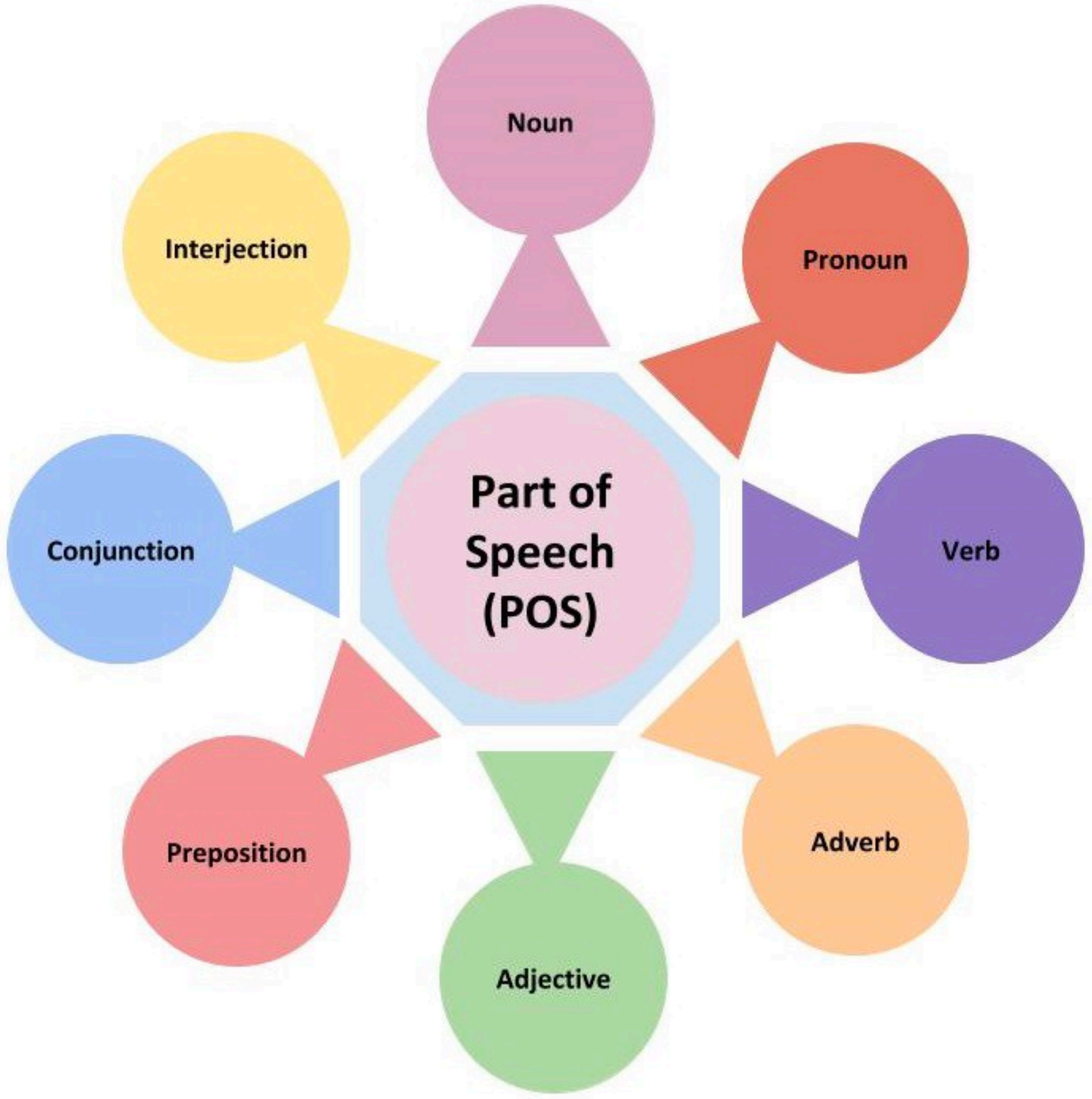
What are Part-of-Speech (POS) Tags

- Word classes or syntactic categories
- Reveal useful information about the syntactic role of a word (and its neighbors!)



Part of Speech

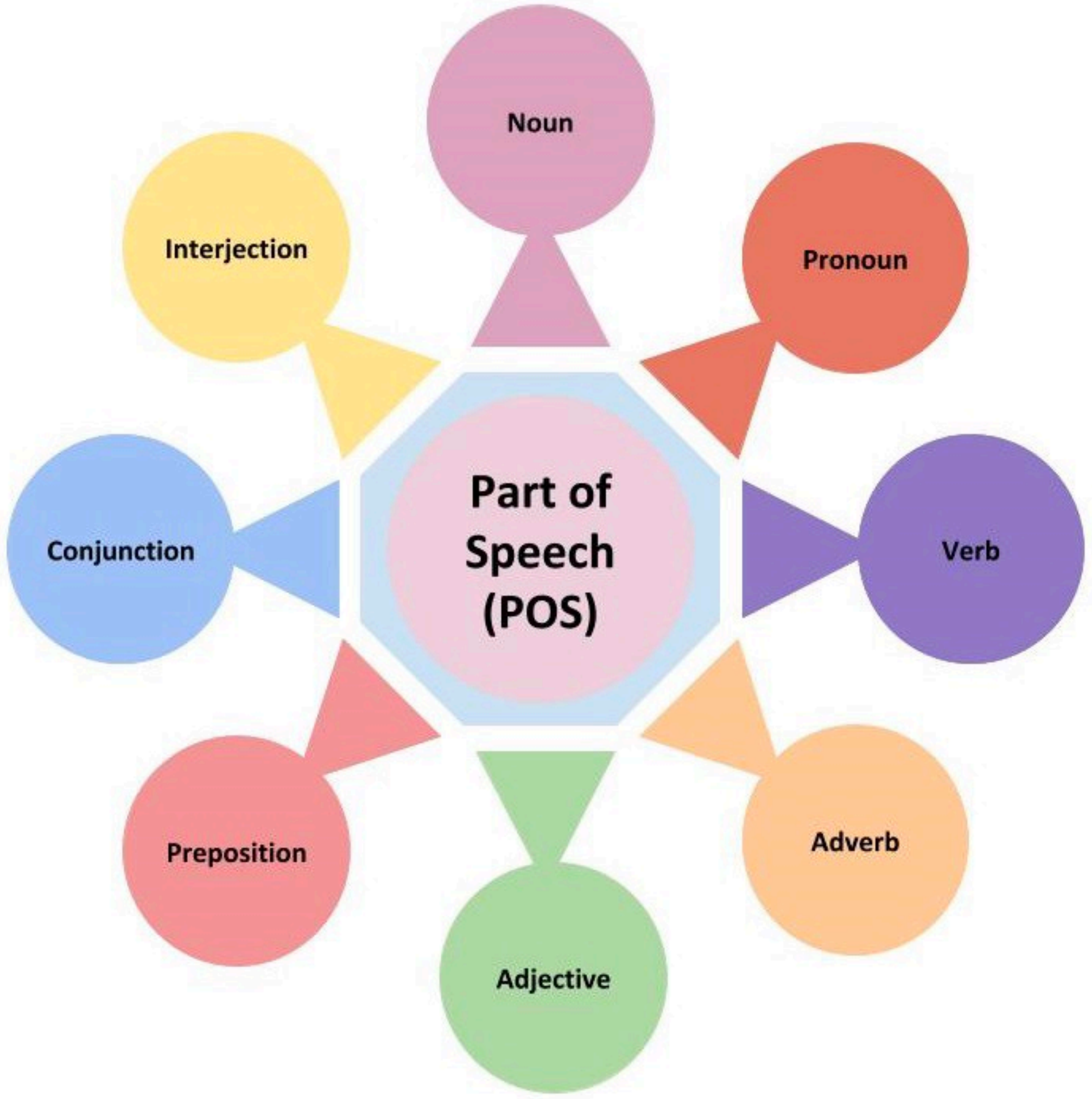
- **Different words have different syntactic functions**
- **Can be roughly divided into two classes**
  - Closed class: fixed membership, function words
    - e.g. prepositions (in, on, of), determiners (a, the)
  - Open class: New words get added frequently
    - e.g. nouns (Twitter, Facebook), verbs (google), adjectives and adverbs.



Slide credit: Danqi Chen COS 484/584

Part of Speech

- **How many part of speech tags do you think English has?**
  - A. < 10
  - B. 10 - 30
  - C. 30 - 50
  - D. > 50



Slide credit: Danqi Chen COS 484/584

Penn Tree Bank Tagset

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	*and, but, or*	PDT	predeterminer	*all, both*	VBP	verb non-3sg present	*eat*
CD	cardinal number	*one, two*	POS	possessive ending	*’s*	VBZ	verb 3sg pres	*eats*
DT	determiner	*a, the*	PRP	personal pronoun	*I, you, he*	WDT	wh-determ.	*which, that*
EX	existential ‘there’	*there*	PRP$	possess. pronoun	*your, one’s*	WP	wh-pronoun	*what, who*
FW	foreign word	*mea culpa*	RB	adverb	*quickly*	WP$	wh-possess.	*whose*
IN	preposition/ subordin-conj	*of, in, by*	RBR	comparative adverb	*faster*	WRB	wh-adverb	*how, where*
JJ	adjective	*yellow*	RBS	superlatv. adverb	*fastest*	$	dollar sign	$
JJR	comparative adj	*bigger*	RP	particle	*up, off*	#	pound sign	#
JJS	superlative adj	*wildest*	SYM	symbol	+,%, &	“	left quote	‘ or “
LS	list item marker	*1, 2, One*	TO	“to”	*to*	”	right quote	’ or ”
MD	modal	*can, should*	UH	interjection	*ah, oops*	(	left paren	[, (, {, <
NN	sing or mass noun	*llama*	VB	verb base form	*eat*	)	right paren	],), }, >
NNS	noun, plural	*llamas*	VBD	verb past tense	*ate*	,	comma	,
NNP	proper noun, sing.	*IBM*	VBG	verb gerund	*eating*	.	sent-end punc	. ! ?
NNPS	proper noun, plu.	*Carolinas*	VBN	verb past part.	*eaten*	:	sent-mid punc	: ; ... – -

45 tags!
(Marcus et al., 1993)

Slide credit: Danqi Chen COS 484/584

The Task of Part of Speech Tagging

- Tag each word with its part of speech
- Disambiguation task: each word might have different senses/functions
  - The/DT back/ADJ door/NN
  - On/IN my/PRP$ back/NN
  - Win/VB the/DT voters/NNS back/RP

Same word, different tags

		WSJ	Brown
Types:			
Unambiguous	(1 tag)	44,432 (**86%**)	45,799 (**85%**)
Ambiguous	(2+ tags)	7,025 (**14%**)	8,050 (**15%**)
Tokens:			
Unambiguous	(1 tag)	577,421 (**45%**)	384,349 (**33%**)
Ambiguous	(2+ tags)	711,780 (**55%**)	786,646 (**67%**)

Slide credit: Danqi Chen COS 484/584

A Simple Baseline

- **Many words might be easy to disambiguate**
- **Most Frequent Class: Assign each token (word) to the class it occurred most in the training data.** (e.g. student/NN)
  - Entirely discarding contextual information
- **How accurate do you think this baseline would be at tagging words?**
  - A. < 50%
  - B. 50% - 75%
  - C. 75% - 90%
  - D. > 90%

Accurately tags **92.34%** of word tokens on Wall Street Journal (WSJ)

Slide credit: Danqi Chen COS 484/584

POS Tagging Not Solved!

- **State of the art: $\sim$ 97%**
- **Sentence level accuracies**
  - Average length of English sentence $\sim$ 14 words
  - 0.92^{14} = 31% vs. 0.97^{14} = 65%
- **Highly relying on domain information**
  - Training data and testing data must be from the same domain
  - < 70% on data from social media

Slide credit: Danqi Chen COS 484/584

Some Observations

- **The function (or POS) of a word depends on its context**
  - The/DT back/ADJ door/NN
  - On/IN my/PRP$ back/NN
  - Win/VB the/DT voters/NNS back/RP
- **Certain POS combinations are extremely unlikely**
  - *<JJ, DT>* ("good the") or *<DT, IN>* ("the in")
- **Better to make predictions on entire sentences instead of individual words**

Sequence Labeling Models!

Slide credit: Danqi Chen COS 484/584

Generative Models vs Discriminative Models

Generative vs Discriminative: Revisit

- **Generative Models**
  - Modeling the joint distribution: $P(X, S)$
- **Discriminative Models**
  - Modeling $P(S \mid X)$ directly

	Generative	**Discriminative**
Classification	Naive Bayes: $P(s)P(x \mid s)$	Logistic Regression: $P(s \mid x)$
Sequence Labeling	HMM: $P(s_1, \ldots, s_n)P(x_1, \ldots, x_n \mid s_1, \ldots, s_n)$	MEMM/CRF: $P(s_1, \ldots, s_n \mid x_1, \ldots, x_n)$

Overview

- **The Sequence Labeling Problem**
  - General Structured Prediction Tasks
  - Part-of-speech Tagging: A case study
  - Generative Models vs. Discriminative Models
- **Hidden Markov Model (HMM)**
  - Basic definitions
  - Parameter estimation
  - The Viterbi algorithm
- **Log-Linear Models**
  - Maximum Entropy Markov Models (MEMMs)
  - Conditional Random Fields (CRFs)

Hidden Markov Models

Markov Sequences

- Consider a sequence of random variables $X_1, X_2, \ldots, X_m$ where m is the length of the sequence
- Each variable X_i can take any value in $\{1, 2, \ldots, k\}$
- How do we model the joint distribution

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_m = x_m)$$

?

Slide credit: Michael Collins COMS W4705

The Markov Assumption

$$
\begin{aligned}
& P(X_1 = x_1, X_2 = x_2, \ldots, X_m = x_m) \\
= \quad & P(X_1 = x_1) \prod_{j=2}^{m} P(X_j = x_j | X_1 = x_1, \ldots, X_{j-1} = x_{j-1}) \\
= \quad & P(X_1 = x_1) \prod_{j=2}^{m} P(X_j = x_j | X_{j-1} = x_{j-1})
\end{aligned}
$$

Markov assumption

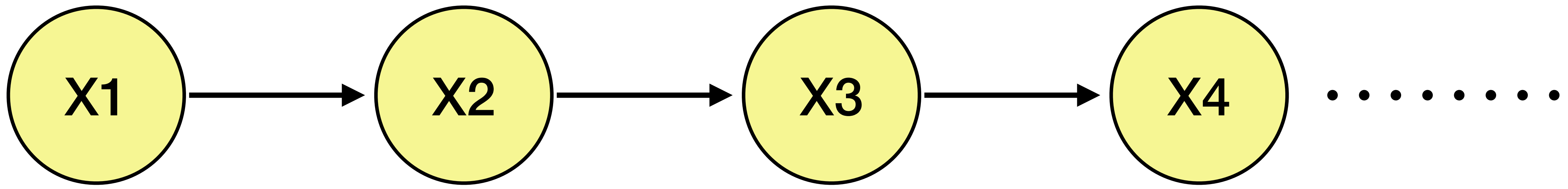
- The first equality is exact (by the chain rule).
- The second equality follows from *the Markov assumption*: for all $j = 2 \ldots m$,

$$P(X_j = x_j | X_1 = x_1, \ldots, X_{j-1} = x_{j-1}) = P(X_j = x_j | X_{j-1} = x_{j-1})$$

Slide credit: Michael Collins COMS W4705

Markov Sequences

- A Generative Model for Sequences



Pick x_1 **at random from the distribution** $P(X_1)$

Pick x_2 **at random from the distribution** $P(X_2 | X_1 = x_1)$

Pick x_t **at random from the distribution** $P(X_t | X_{t-1} = x_{t-1})$

Modeling Pairs of Sequences

- **In Sequence Labeling, we need to model pairs of sequences**

$S = S_i, S_2, \ldots, S_n$	NNP	VBZ	IN	NNP
$X = X_i, X_2, \ldots, X_n$	USC	is	in	California

Hidden Markov Models (HMMs) allow us to *jointly* reason over X and S

Hidden Markov Models

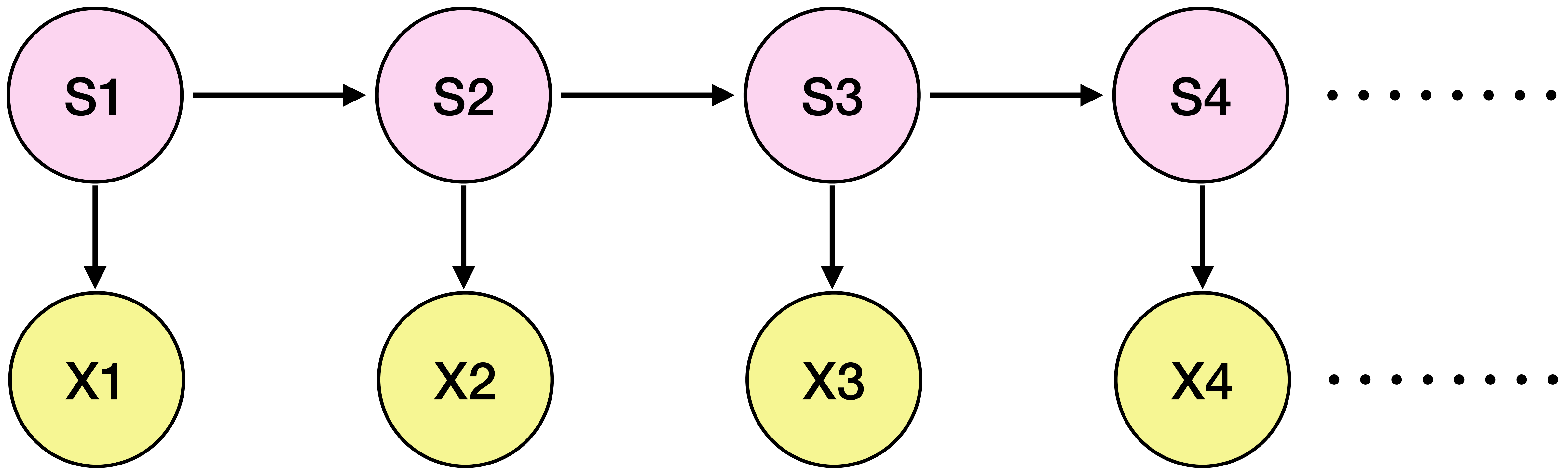
- We have two sequences of random variables: $X_1, X_2, \ldots, X_m$ and $S_1, S_2, \ldots, S_m$
- Intuitively, each X_i corresponds to an "observation" and each S_i corresponds to an underlying "state" that generated the observation. Assume that each S_i is in $\{1, 2, \ldots k\}$, and each X_i is in $\{1, 2, \ldots o\}$
- How do we model the joint distribution

$$P(X_1 = x_1, \ldots, X_m = x_m, S_1 = s_1, \ldots, S_m = s_m)$$

?

Slide credit: Michael Collins COMS W4705

The HMM Assumptions



1. **Markov Assumption on S**

$$P(S_j = s_j \mid S_{j-1} = s_{j-1}, \ldots, S_1 = s_1) = P(S_j = s_j \mid S_{j-1} = s_{j-1})$$

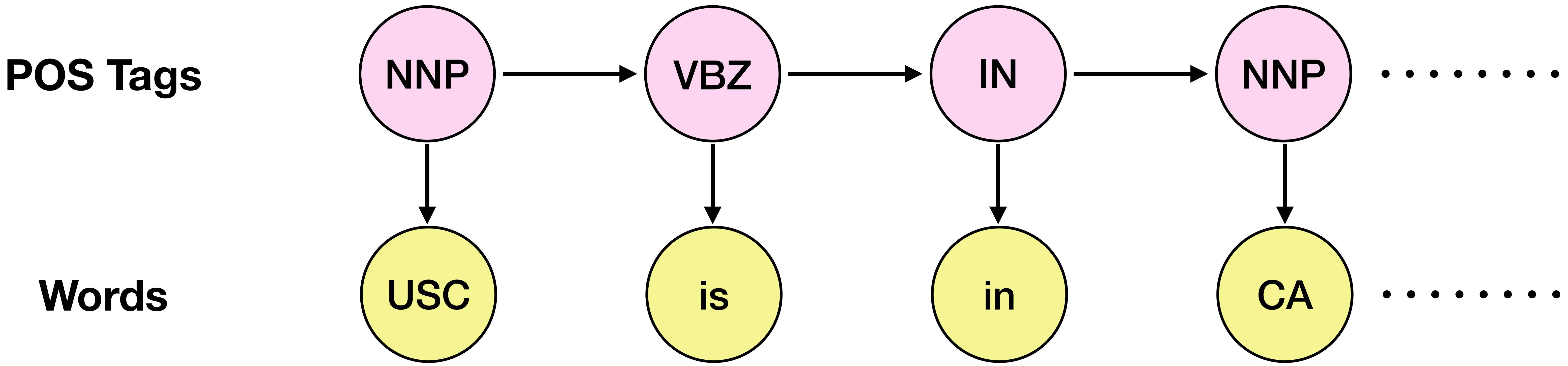
Transition Probabilities

2. **Conditional Independence on X given S**

$$P(X_1 = x_j, \ldots, X_m = x_m \mid S_1 = s_1, \ldots, S_m = s_m) = \prod_{j=1}^{m} P(X_j = x_j \mid S_j = s_j)$$

Emission Probabilities

The HMM Assumptions



1. **Markov Assumption on S**

$$P(S_3 = \text{IN} \mid S_2 = \text{VBZ}, S_1 = \text{NNP}) = P(S_3 = \text{IN} \mid S_2 = \text{VBZ})$$

2. **Conditional Independence on X given S**

$$P(\text{USC is in CA} \mid \text{NNP VBZ IN NNP}) = P(\text{USC} \mid \text{NNP})P(\text{is} \mid \text{VBZ})P(\text{in} \mid \text{IN})P(\text{CA} \mid \text{NNP})$$

Which assumption do you think is stronger?

Joint Distribution of Sequence Pairs in HMMs

$$P(X_1 = x_j, \ldots, X_m = x_m, S_1 = s_1, \ldots, S_m = s_m)$$

$$= \boxed{P(X_1 = x_j, \ldots, X_m = x_m \,|\, S_1 = s_1, \ldots, S_m = s_m)}$$

Output Independence

$$\boxed{\times P(S_1 = s_1, \ldots, S_m = s_m)}$$

Markov Assumption

$$= \prod_{j=1}^{m} P(X_j = x_j \,|\, S_j = s_j)$$

How to model $P(X_j = x_j \,|\, S_j = s_j)$ and $P(S_j = s_j \,|\, S_{j-1} = s_{j-1})$?

$$\times P(S_1 = s_1) \prod_{j=1}^{m} P(S_j = s_j \,|\, S_{j-1} = s_{j-1})$$

Homogeneous HMMs

- In a *homogeneous* HMM, we make an additional assumption:

$$P(S_j = s_j \,|\, S_{j-1} = s_{j-1}) = t(s_j \,|\, s_{j-1})$$

$$P(X_j = x_j \,|\, S_j = s_j) = e(x_j \,|\, s_j)$$

- Idea behind this assumption: the transition and emission probabilities do NOT depend on the position in the Markov chain (do not depend on the index j)

The Model Form for Homogeneous HMMs

- The model takes the following form:

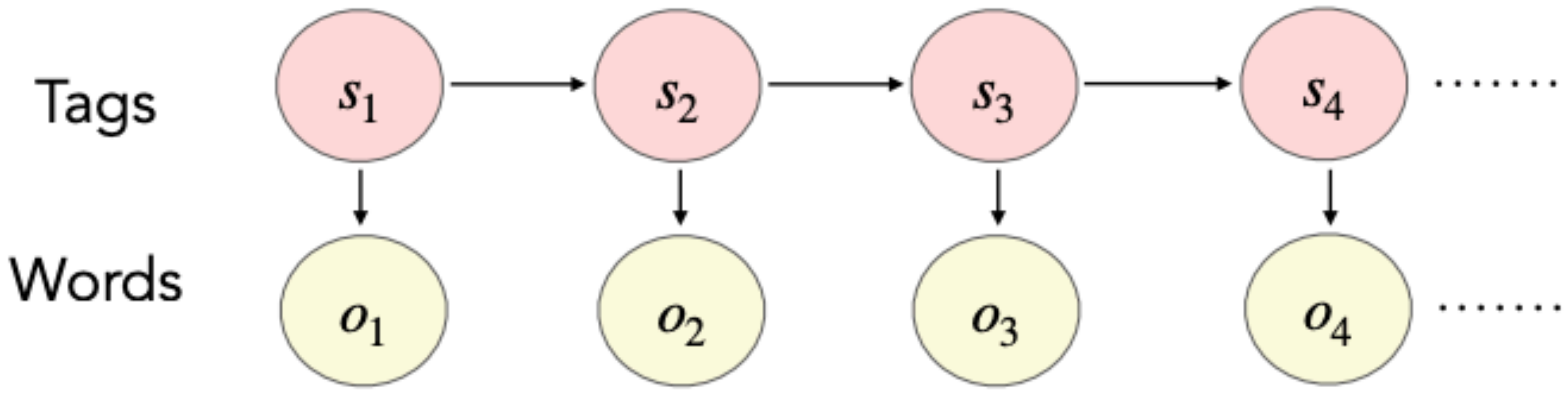
$$p(x_1 \dots x_m, s_1 \dots s_m; \underline{\theta}) = t(s_1) \prod_{j=2}^{m} t(s_j | s_{j-1}) \prod_{j=1}^{m} e(x_j | s_j)$$

- Parameters in the model:

  1. Initial state parameters $t(s)$ for $s \in \{1, 2, \dots, k\}$
  2. Transition parameters $t(s'|s)$ for $s, s' \in \{1, 2, \dots, k\}$
  3. Emission parameters $e(x|s)$ for $s \in \{1, 2, \dots, k\}$ and $x \in \{1, 2, \dots, o\}$

Slide credit: Michael Collins COMS W4705

Example: Sequence Probability



Dummy start state

s_{t+1}

s_t	DT	NN
∅	0.8	0.2
DT	0.2	0.8
NN	0.3	0.7

o_t

	the	cat
DT	0.9	0.1
NN	0.5	0.5

What is the joint probability P(the cat, DT NN)?

A) $(0.8 * 0.8) * (0.9 * 0.5)$
B) $(0.2 * 0.8) * (0.9 * 0.5)$
C) $(0.3 * 0.7) * (0.5 * 0.5)$

Slide credit: Danqi Chen COS 484/584

Learning a Hidden Markov Model

Parameter Estimation

- Assuming we have fully observed data $\{X_i, S_i\}_{i=1}^{N}$, e.g. WSJ

Training set:

1 Pierre/NNP Vinken/NNP ,/, 61/CD years/NNS old/JJ ,/
join/VB the/DT board/NN as/IN a/DT nonexecutive/JJ di
Nov./NNP 29/CD ./.
2 Mr./NNP Vinken/NNP is/VBZ chairman/NN of/IN Elsev
N.V./NNP ,/, the/DT Dutch/NNP publishing/VBG group/
3 Rudolph/NNP Agnew/NNP ,/, 55/CD years/NNS old/JJ
chairman/NN of/IN Consolidated/NNP Gold/NNP Fields/N
,/, was/VBD named/VBN a/DT nonexecutive/JJ director/
this/DT British/JJ industrial/JJ conglomerate/NN ./.
. . .
38,219 It/PRP is/VBZ also/RB pulling/VBG 20/CD peopl
of/IN Puerto/NNP Rico/NNP ,/, who/WP were/VBD help
Huricane/NNP Hugo/NNP victims/NNS ,/, and/CC sendin
them/PRP to/TO San/NNP Francisco/NNP instead/RB ./

Maximum Likelihood Estimate:

$$\max_{t(\cdot|\cdot), e(\cdot|\cdot)} \prod_{i=1}^{N} P(X_i, S_i)$$

$$t(s' \mid s) = \frac{\text{count}(s \rightarrow s')}{\text{count}(s)}$$

$$e(x \mid s) = \frac{\text{count}(s \rightarrow x)}{\text{count}(s)}$$

Slide credit: Danqi Chen COS 484/584

Learning Example

1. the/DT cat/NN sat/VBD on/IN the/DT mat/NN

2. Princeton/NNP is/VBZ in/IN New/NNP Jersey/NNP

3. the/DT old/NN man/VB the/DT boats/NNS

$$t(\textbf{NN} \mid \textbf{DT}) = \frac{3}{4}$$

$$e(\textbf{cat} \mid \textbf{NN}) = \frac{1}{3}$$

Maximum Likehood Estimate:

$$\max_{t(\cdot|\cdot),e(\cdot|\cdot)} \prod_{i=1}^{N} P(X_i, S_i)$$

$$t(s' \mid s) = \frac{\text{count}(s \to s')}{\text{count}(s)}$$

$$e(x \mid s) = \frac{\text{count}(s \to x)}{\text{count}(s)}$$

Slide credit: Danqi Chen COS 484/584

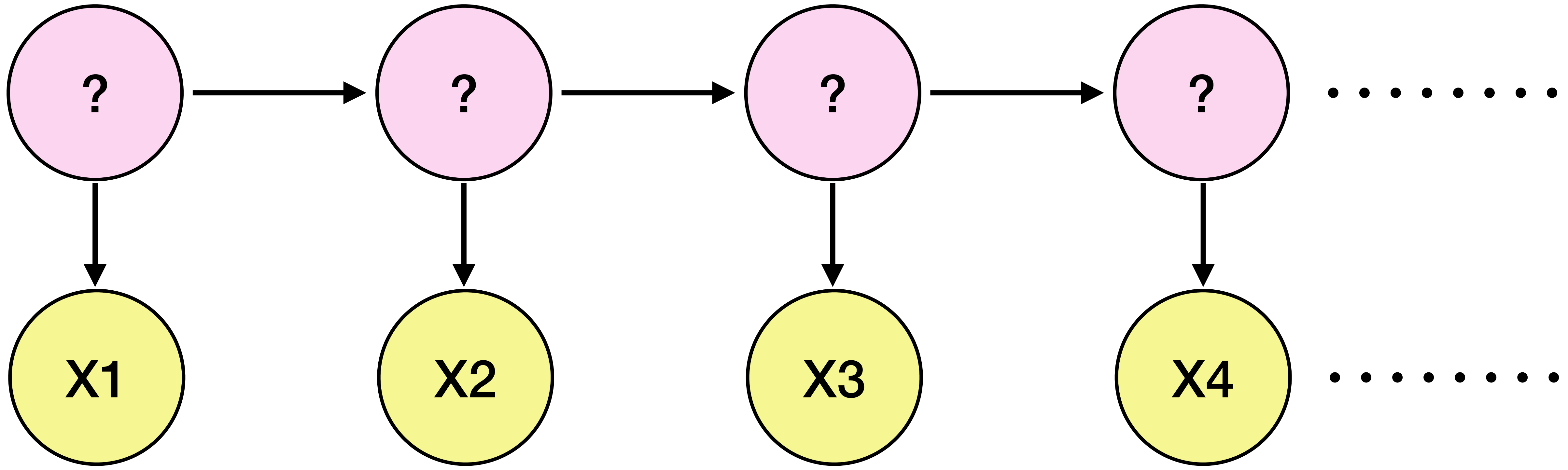
Decoding with HMMs

Decoding with HMMs

- Goal: for a given input sequence $x_1, \ldots, x_m$, find

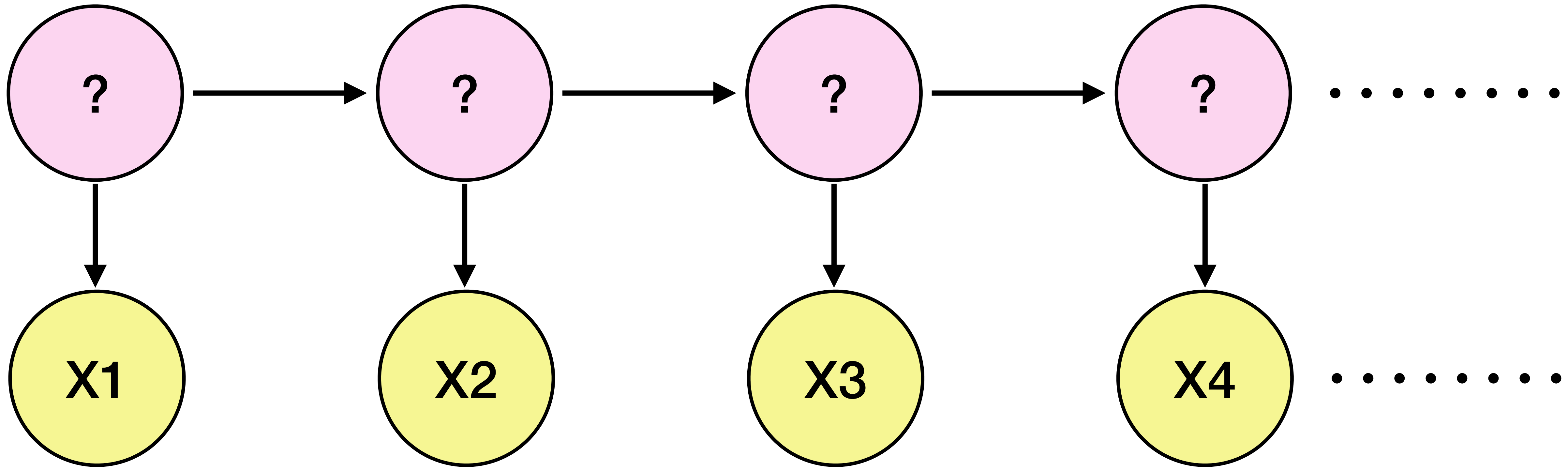
$$\arg \max_{s_1,\ldots,s_m} p(x_1 \ldots x_m, s_1 \ldots s_m; \underline{\theta})$$

- This is the most likely state sequence $s_1 \ldots s_m$ for the given input sequence $x_1 \ldots x_m$



Slide credit: Michael Collins COMS W4705

Decoding with HMMs



$$S^* = \arg\max_{s_1,\ldots,s_m} p(x_1, \ldots, x_m, s_1, \ldots, s_m) = t(s_1)\prod_{j=2}^{m} t(s_j \mid s_{j-1}) \prod_{j=1}^{m} e(x_j \mid s_j)$$

How can we maximize this over all state sequences?

Greedy Decoding

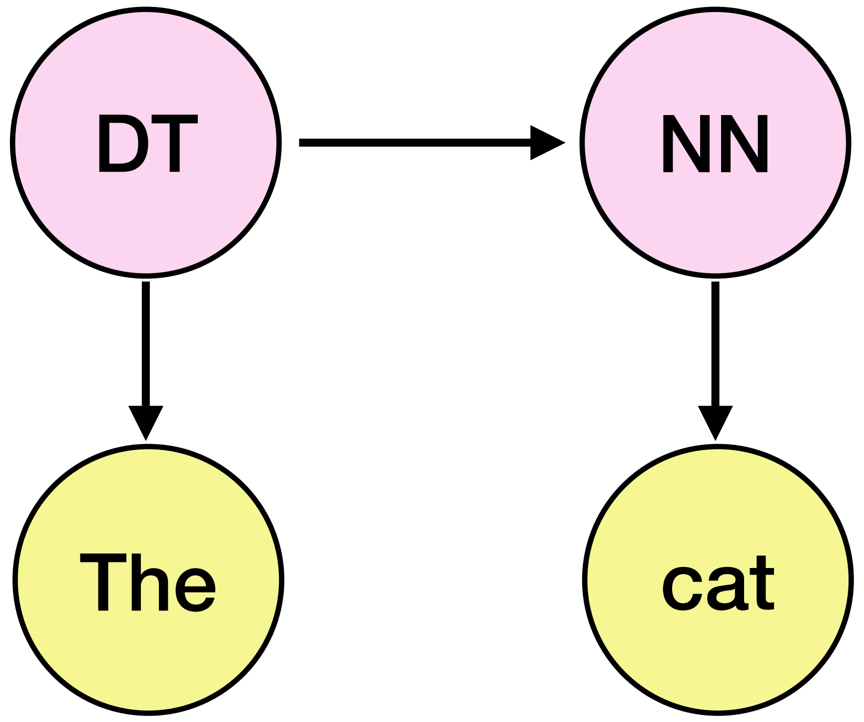
Decode/reveal one state at a time

$$s_1^* = \arg\max_{s_1} t(s_1)e(x_1 \mid s_1)$$

$$S^* = \arg\max_{s_1,\ldots,s_m} p(x_1, \ldots, x_m, s_1, \ldots, s_m) = t(s_1)\prod_{j=2}^{m} t(s_j \mid s_{j-1}) \prod_{j=1}^{m} e(x_j \mid s_j)$$

Slide credit: Danqi Chen COS 484/584

Greedy Decoding



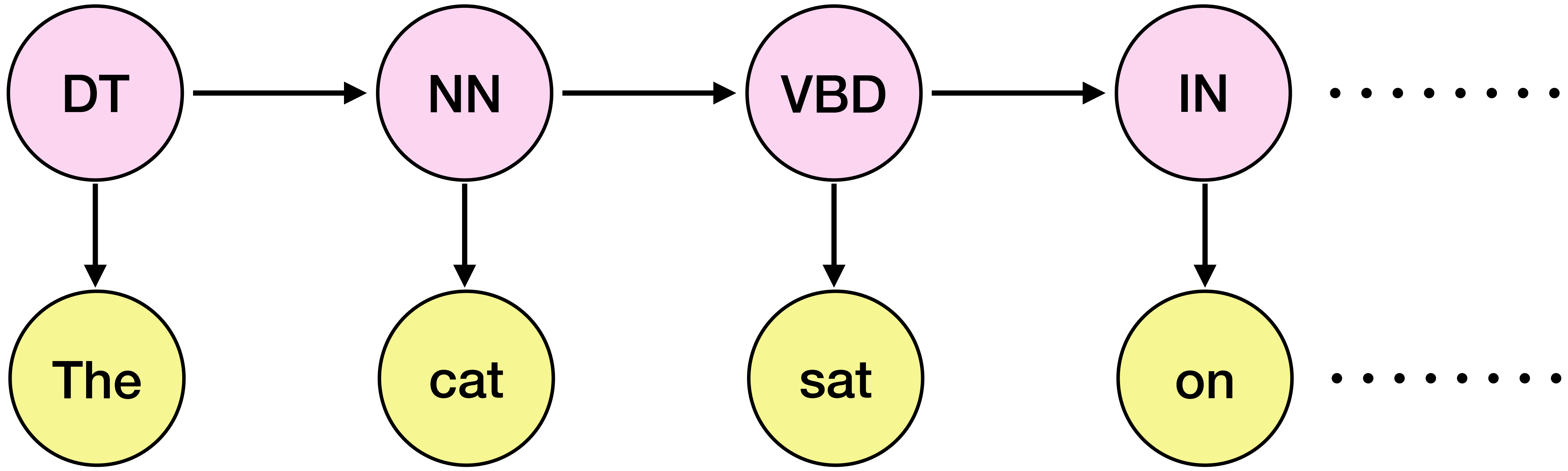
Decode/reveal one state at a time

$$s_2^* = \arg\max_{s_2} t(s_2 \mid s_1^*) e(x_2 \mid s_2)$$

$$S^* = \arg\max_{s_1,\ldots,s_m} p(x_1, \ldots, x_m, s_1, \ldots, s_m) = t(s_1) \prod_{j=2}^{m} t(s_j \mid s_{j-1}) \prod_{j=1}^{m} e(x_j \mid s_j)$$

Slide credit: Danqi Chen COS 484/584

Greedy Decoding



$$s_j^* = \arg\max_{s_j} t(s_j \mid s_{j-1}^*) e(x_j \mid s_j), \quad \forall j$$

- Local decisions
- Not guaranteed to produce the overall optimal sequence

Slide credit: Danqi Chen COS 484/584

Viterbi Decoding

- The *Viterbi algorithm* is a dynamic programming algorithm. Basic data structure:

$$\pi[j, s]$$

will be a table entry that stores the maximum probability for any state sequence ending in state s at position j. More formally: $\pi[1, s] = t(s)e(x_1|s)$, and for $j > 1$,

$$\pi[j, s] = \max_{s_1 \ldots s_{j-1}} \left[t(s_1)e(x_1|s_1) \left(\prod_{k=2}^{j-1} t(s_k|s_{k-1})e(x_k|s_k) \right) \boxed{t(s|s_{j-1})} e(x_j|s) \right]$$

Slide credit: Michael Collins COMS W4705

Viterbi Decoding

- Initialization: for $s = 1 \dots k$

$$\pi[1, s] = t(s)e(x_1|s)$$

- For $j = 2 \dots m$, $s = 1 \dots k$:

$$\pi[j, s] = \max_{s' \in \{1 \dots k\}} [\pi[j-1, s'] \times t(s|s') \times e(x_j|s)]$$

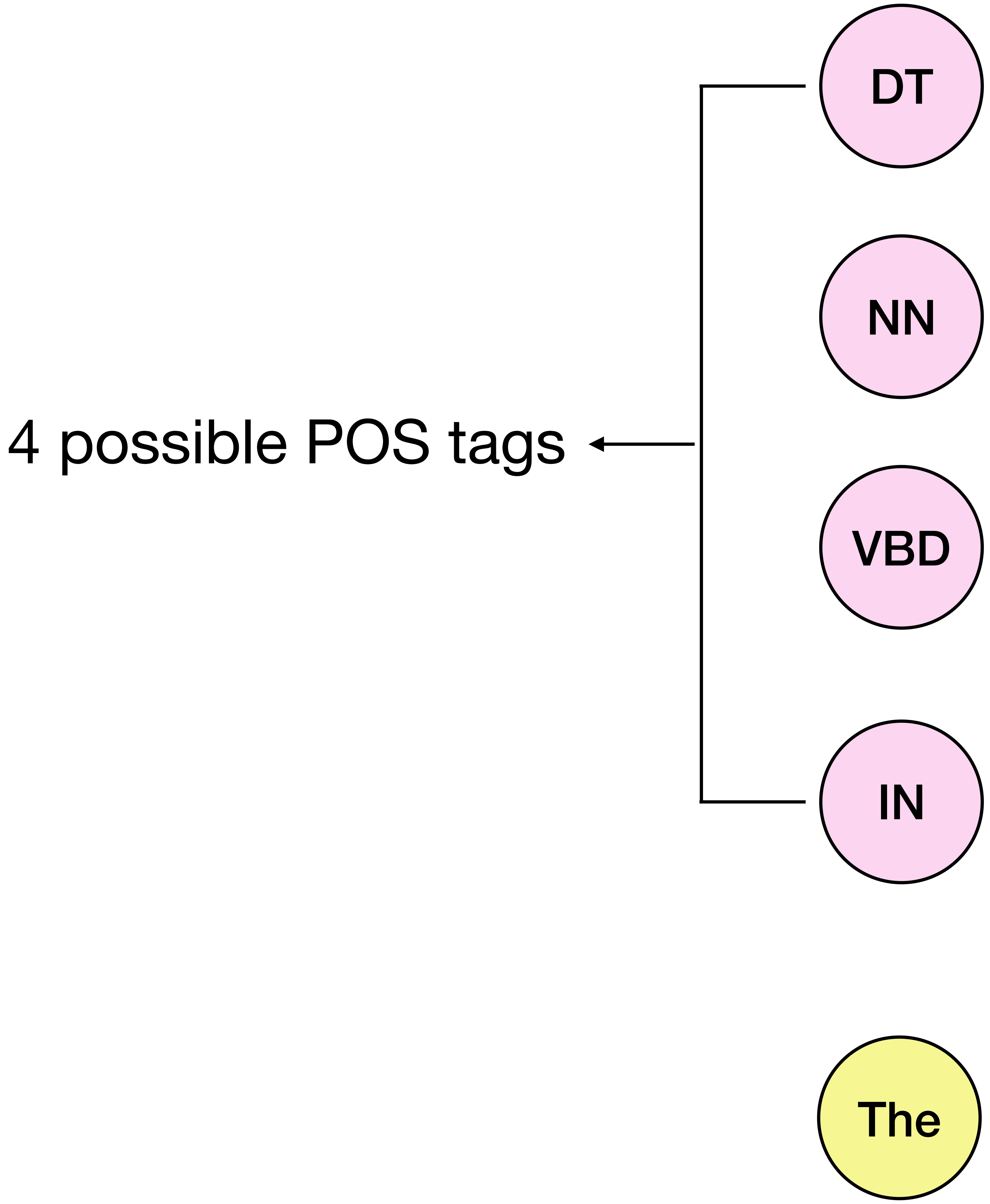
- We then have

$$\max_{s_1 \dots s_m} p(x_1 \dots x_m, s_1 \dots s_m; \underline{\theta}) = \max_s \pi[m, s]$$

- The algorithm runs in $O(mk^2)$ time

Slide credit: Michael Collins COMS W4705

Viterbi Decoding



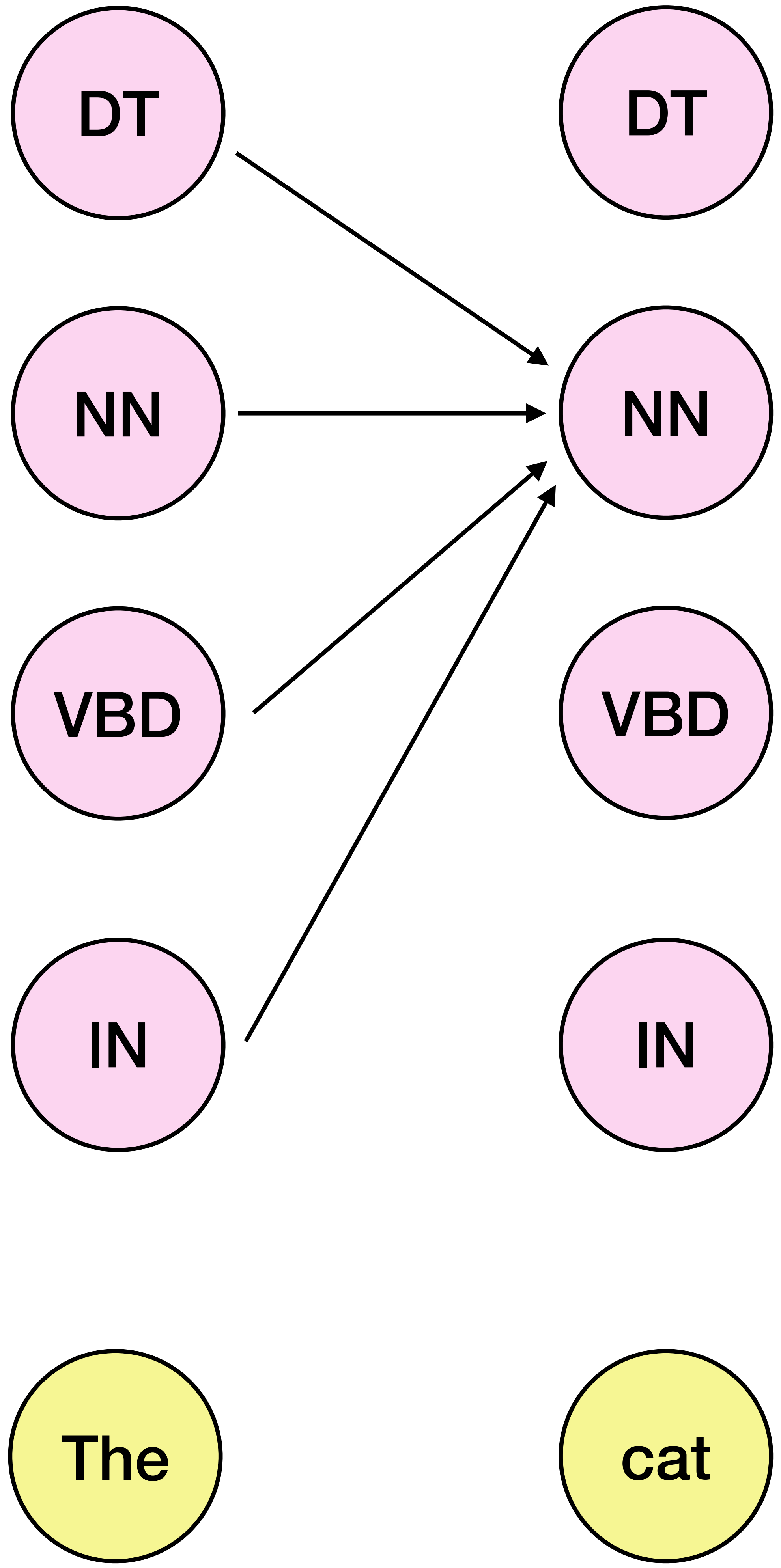
$$\pi[1,\textbf{DT}] = t(\textbf{DT})e(\textbf{The}\,|\,\textbf{DT})$$

$$\pi[1,\textbf{NN}] = t(\textbf{NN})e(\textbf{The}\,|\,\textbf{NN})$$

$$\pi[1,\textbf{VBD}] = t(\textbf{VBD})e(\textbf{The}\,|\,\textbf{VBD})$$

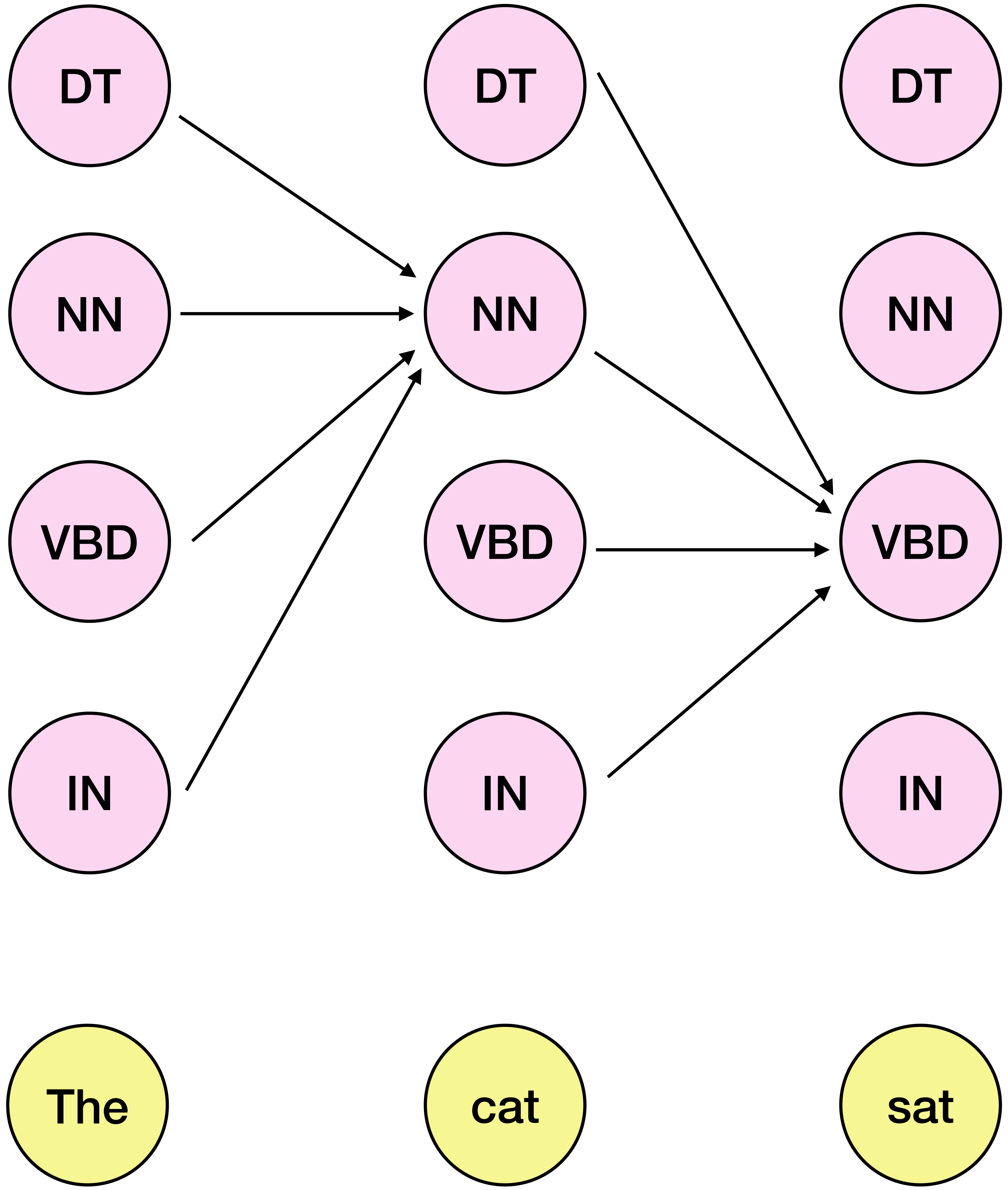
$$\pi[1,\textbf{IN}] = t(\textbf{IN})e(\textbf{The}\,|\,\textbf{IN})$$

Viterbi Decoding



$$\pi[2,\textbf{NN}] = \max_{s_1} \boxed{\pi[1,\ s_1]}\ t(\textbf{NN} \mid s_1)e(\textbf{cat} \mid \textbf{NN})$$

Viterbi Decoding



$$\pi[2,\textbf{NN}] = \max_{s_1} \pi[1,\ s_1]\ \ t(\textbf{NN} \mid s_1) e(\textbf{cat} \mid \textbf{NN})$$

$$\pi[3,\textbf{VBD}] = \max_{s_2} \boxed{\pi[2,\ s_2]}\ \ t(\textbf{VBD} \mid s_2) e(\textbf{sat} \mid \textbf{VBD})$$

Pros and Cons

- **HMMs are simple to train**
  - Just need to compile counts from the training corpus
- **Performs relatively well**
  - > 96% on POS tagging (92.3% of most frequent class)
  - > 90% on Named Entity Recognition
- **Main difficulty is modeling** $e(word \mid tag)$
  - Words are very complex
  - Unknown words

Overview

- **The Sequence Labeling Problem**
  - General Structured Prediction Tasks
  - Part-of-speech Tagging: A case study
  - Generative Models vs. Discriminative Models
- **Hidden Markov Model (HMM)**
  - Basic definitions
  - Parameter estimation
  - The Viterbi algorithm
- **Log-Linear Models**
  - Maximum Entropy Markov Models (MEMMs)
  - Conditional Random Fields (CRFs)
  - Parameter estimation

Log-Linear Models

Log-Linear Models

- **The General Problem**

  - We have some **input domain** $\mathcal{X}$
  - Have a finite **label set** $\mathcal{Y}$
  - Aim is to provide a **conditional probability** $p(y \mid x)$ for any x, y where $x \in \mathcal{X}$, $y \in \mathcal{Y}$

Slide credit: Michael Collins COMS W4705

Log-Linear Models

- We have some input domain $\mathcal{X}$, and a finite label set $\mathcal{Y}$. Aim is to provide a conditional probability $p(y \mid x)$ for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.
- A feature is a function $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$
  (Often binary features or indicator functions
  $f_k : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$).
- Say we have m features f_k for $k = 1 \ldots m$
  $\Rightarrow$ A feature vector $f(x, y) \in \mathbb{R}^m$ for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.
- We also have a **parameter vector** $v \in \mathbb{R}^m$
- We define

$$p(y \mid x; v) = \frac{e^{v \cdot f(x,y)}}{\sum_{y' \in \mathcal{Y}} e^{v \cdot f(x,y')}}$$

Slide credit: Michael Collins COMS W4705

Why the name?

$$p(y \mid x; v) = \frac{e^{v \cdot f(x,y)}}{\sum_{y' \in \mathcal{Y}} e^{v \cdot f(x,y')}}$$

$$\log p(y \mid x; v) = \underbrace{v \cdot f(x, y)}_{\text{Linear term}} - \underbrace{\log \sum_{y' \in \mathcal{Y}} e^{v \cdot f(x,y')}}_{\text{Normalization term}}$$

Features in Log-linear Models

$$S = S_i, S_2, \ldots, S_n$$

$$X = X_i, X_2, \ldots, X_n$$

NNP	VBZ	IN	NNP
USC	is	in	California

Theoretically, we can use any features in X and S: $f(X, S)$

- The current word: <is>
- The surrounding words: <USC, is>), …
- The current POS tag: <VBZ>
- The surrounding tags: <NNP, VBZ>, …
- The current word and tag: <is, VBZ>

How to design these features into numerical vectors?

Feature Vocabulary

$S = S_i, S_2, \ldots, S_n$

$X = X_i, X_2, \ldots, X_n$

NNP	VBZ	IN	NNP
USC	is	in	California

Theoretically, we can use any features in X and S: $f(X, S)$

- The current word: <is>
- The surrounding words: <USC, is>), …
- The current POS tag: <VBZ>
- The surrounding tags: <NNP, VBZ>, …
- The current word and tag: <is, VBZ>

Vocab = [is, in, USC, California, …]

Vocab += [<USC, is>, <is, in>, …]

Vocab += [NNP, VBZ, IN, …]

Vocab += [<USC,NNP>, <is,VBZ>, …]

Feature Sparsity Problem

$$S = S_i, S_2, \ldots, S_n$$

$$X = X_i, X_2, \ldots, X_n$$

NNP	VBZ	IN	NNP
USC	**is**	**in**	**California**

Theoretically, we can use any features in X and S: $f(X, S)$

	Number of Features
• The current word: <is>	V
• The surrounding words: <USC, is>), …	V^2
• The current POS tag: <VBZ>	
• The surrounding tags: <NNP, VBZ>, …	
• The current word and tag: <is, VBZ>	VK

Features vs. Independence

- We need independence assumptions to compute the nominator
- Stronger assumptions lead to less flexible features

$$p(y \mid x; v) = \frac{e^{v \cdot f(x,y)}}{\boxed{\sum_{y' \in \mathcal{Y}} e^{v \cdot f(x,y')}}}$$

Overview

- The Sequence Labeling Problem
  - General Structured Prediction Tasks
  - Part-of-speech Tagging: A case study
  - Generative Models vs. Discriminative Models
- Hidden Markov Model (HMM)
  - Basic definitions
  - Parameter estimation
  - The Viterbi algorithm
- **Log-Linear Models**
  - Maximum Entropy Markov Models (MEMMs)
  - Conditional Random Fields (CRFs)
  - Parameter estimation

Independence Assumptions in MEMMs

- **Goal: modeling the distribution**

$$p(s_1, \ldots, s_m \mid x_1, \ldots, x_m)$$

- **Markov Assumption on S**

$$p(s_1, \ldots, s_m \mid x_1, \ldots, x_m) = \prod_{j=1}^{m} p(s_j \mid s_1, \ldots, s_{j-1}, x_1, \ldots, x_m)$$ chain rule (no assumptions)

$$= \prod_{j=1}^{m} p(s_j \mid s_{j-1}, x_1, \ldots, x_m)$$ Markov assumption

Using Log-Linear Models

- **We then model each term using a log-linear model:**

$$p(s_j \mid s_{j-1}, x_1, \ldots, x_m) = \frac{\exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s_j))}{\sum_{s_j' \in \mathbb{S}} \exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s_j'))}$$

- **Here $f(x_1, \ldots, x_m, j, s, s')$ is the feature vector:**
  - $x_1, \ldots, x_m$ is the sequence of words to be tagged
  - j is the position to be tagged (any value from $1, \ldots, m$)
  - s is the previous state
  - s' is the current state

Using Log-Linear Models

- **We then model each term using a log-linear model:**

$$p(s_j \,|\, s_{j-1}, x_1, \ldots, x_m) = \frac{\exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s_j))}{\boxed{\sum_{s_j' \in \mathbb{S}} \exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s_j'))}}$$

Trackable

- **Here** $\boxed{f(x_1, \ldots, x_m, j, s, s')}$ **is the feature vector:**
  - $x_1, \ldots, x_m$ is the sequence of words to be tagged
  - j is the position to be tagged (any value from $1, \ldots, m$)
  - s is the previous state
  - s' is the current state

The whole sequence of X
Only two successive tags

Features in MEMMs

$$S = S_1, S_2, \ldots, S_n$$

$$X = X_1, X_2, \ldots, X_n$$

NNP	VBZ	IN	NNP
USC	is	in	California

What are the most important features $f(x_1, \ldots, x_m, j, s, s')$?

- The current word: <is>
- The surrounding words: <USC, is>), …
- The current POS tag: <VBZ>
- The previous tag: <NNP, VBZ>, …
- The current word and tag: <is, VBZ>
- Prefix or suffix features: <ing>
- …

Decoding with MEMMs: Viterbi Algorithm

- Goal: for a given input sequence $x_1, \ldots, x_m$, find

$$\arg \max_{s_1, \ldots, s_m} p(s_1 \ldots s_m | x_1 \ldots x_m)$$

- We can use the *Viterbi* algorithm again (see last lecture on HMMs). Basic data structure:

$$\pi[j, s]$$

will be a table entry that stores the maximum probability for any state sequence ending in state s at position j. More formally:

$$\pi[j, s] = \max_{s_1 \ldots s_{j-1}} \left(p(s | s_{j-1}, x_1 \ldots x_m) \prod_{k=1}^{j-1} p(s_k | s_{k-1}, x_1 \ldots x_m) \right)$$

Slide credit: Michael Collins COMS W4705

Decoding with MEMMs: Viterbi Algorithm

- Initialization: for $s \in \mathcal{S}$

$$\pi[1, s] = p(s|s_0, x_1 \dots x_m)$$

where s_0 is a special "initial" state.

- For $j = 2 \dots m$, $s = 1 \dots k$:

$$\pi[j, s] = \max_{s' \in \mathcal{S}} [\pi[j-1, s'] \times p(s|s', x_1 \dots x_m)]$$

- We then have

$$\max_{s_1 \dots s_m} p(s_1 \dots s_m | x_1 \dots x_m) = \max_s \pi[m, s]$$

Slide credit: Michael Collins COMS W4705

Model Performance

	POS Tagging	NER
HMM	96.4%	75.3
MEMM	96.9%	85.9

HMMs vs. MEMMs

- **In MEMMs, each state transition has probability**

$$p(s_j \mid s_{j-1}, x_1, \ldots, x_m) = \frac{\exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s_j))}{\sum_{s' \in \mathbb{S}} \exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s_j'))}$$

- **In HMMs, each state transition has probability**

$$p(s_j \mid s_{j-1}) p(x_j \mid s_j)$$

- **Feature vectors f allows much richer representations in MEMMs:**
  - Sensitivity to *any* word in the input sequence $x_1, \ldots, x_m$, not just x_j
  - Sensitivity to spelling features (prefixes, suffixes etc.) of current or surrounding words
- **Parameter estimation in MEMMs is more expensive than in HMMs (but is still not prohibitive for most tasks)**

Can we relax the Markov assumption in MEMMs but keep the same features?

Overview

- **The Sequence Labeling Problem**
  - General Structured Prediction Tasks
  - Part-of-speech Tagging: A case study
  - Generative Models vs. Discriminative Models
- **Hidden Markov Model (HMM)**
  - Basic definitions
  - Parameter estimation
  - The Viterbi algorithm
- **Log-Linear Models**
  - Maximum Entropy Markov Models (MEMMs)
  - Conditional Random Fields (CRFs)
  - Parameter estimation

Conditional Random Fields (CRFs)

- **Goal: modeling the distribution**

$$p(s_1, \ldots, s_m \mid x_1, \ldots, x_m)$$

- **In MEMMs we had**

$$p(s_1, \ldots, s_m \mid x_1, \ldots, x_m) = \prod_{j=1}^{m} p(s_j \mid s_1, \ldots, s_{j-1}, x_1, \ldots, x_m)$$

chain rule (no assumptions)

$$= \prod_{j=1}^{m} p(s_j \mid s_{j-1}, x_1, \ldots, x_m)$$

Markov assumption

- **Using log-linear model**

$$p(s_j \mid s_{j-1}, x_1, \ldots, x_m) = \frac{\exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s_j))}{\sum_{s' \in \mathbb{S}} \exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s'_j))}$$

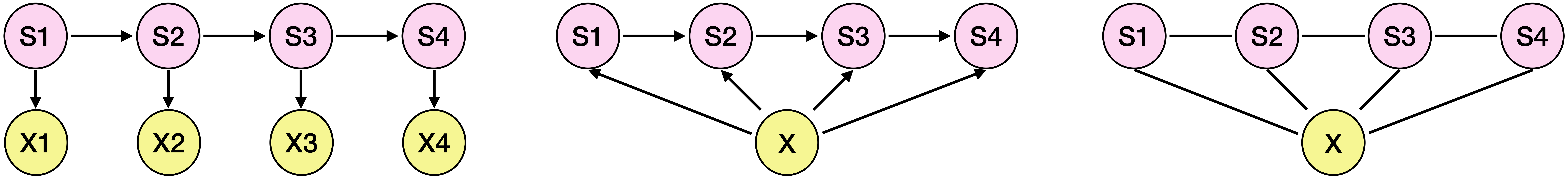
Can we build a *giant* log-linear model?

$$p(s_1, \ldots, s_m \mid x_1, \ldots, x_m)$$

Conditional Random Fields (CRFs)

- **Globally Normalized Model**

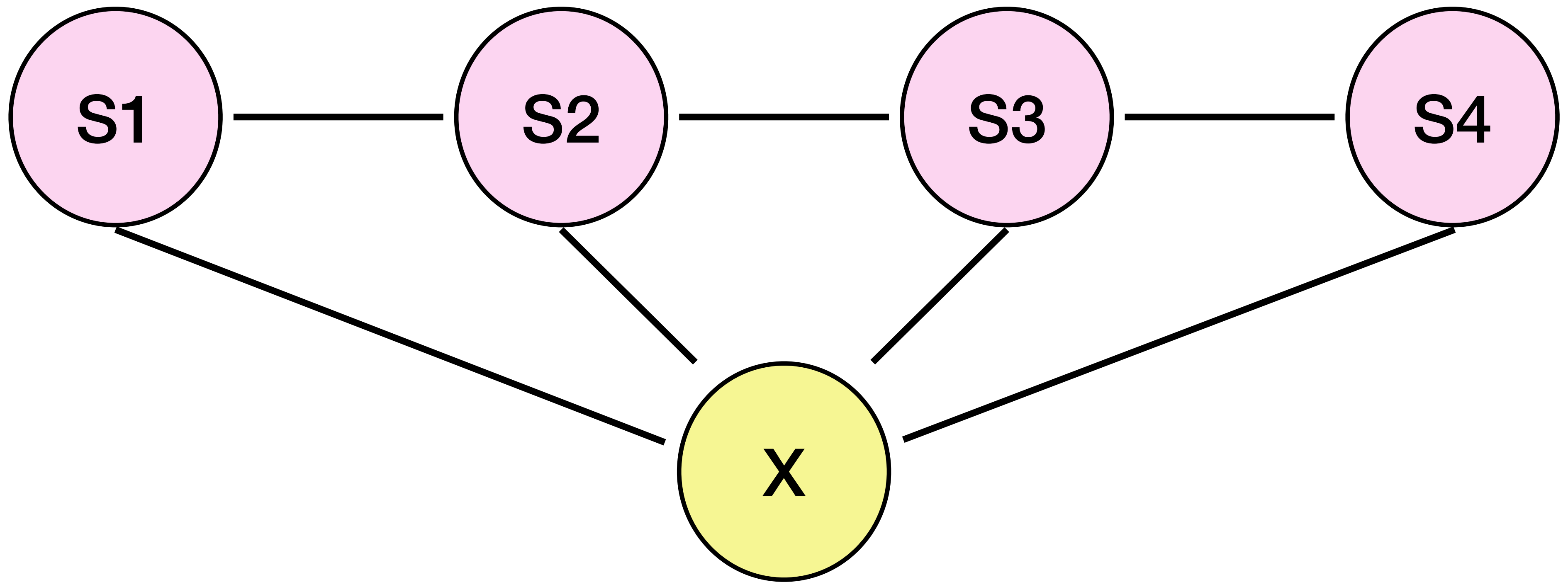
$$p(s_1, \ldots, s_m \,|\, x_1, \ldots, x_m) = \frac{\prod_{j-1}^{m} \exp(v \cdot \boxed{f(x_1, \ldots, x_m, i, s_{j-1}, s_j)})}{\boxed{\sum_{s'_1, \ldots, s'_m \in \mathbb{S}} \prod_{j=1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s'_{j-1}, s'_j))}}$$



HMM **MEMM** **CRF**

Independence Assumptions in CRFs

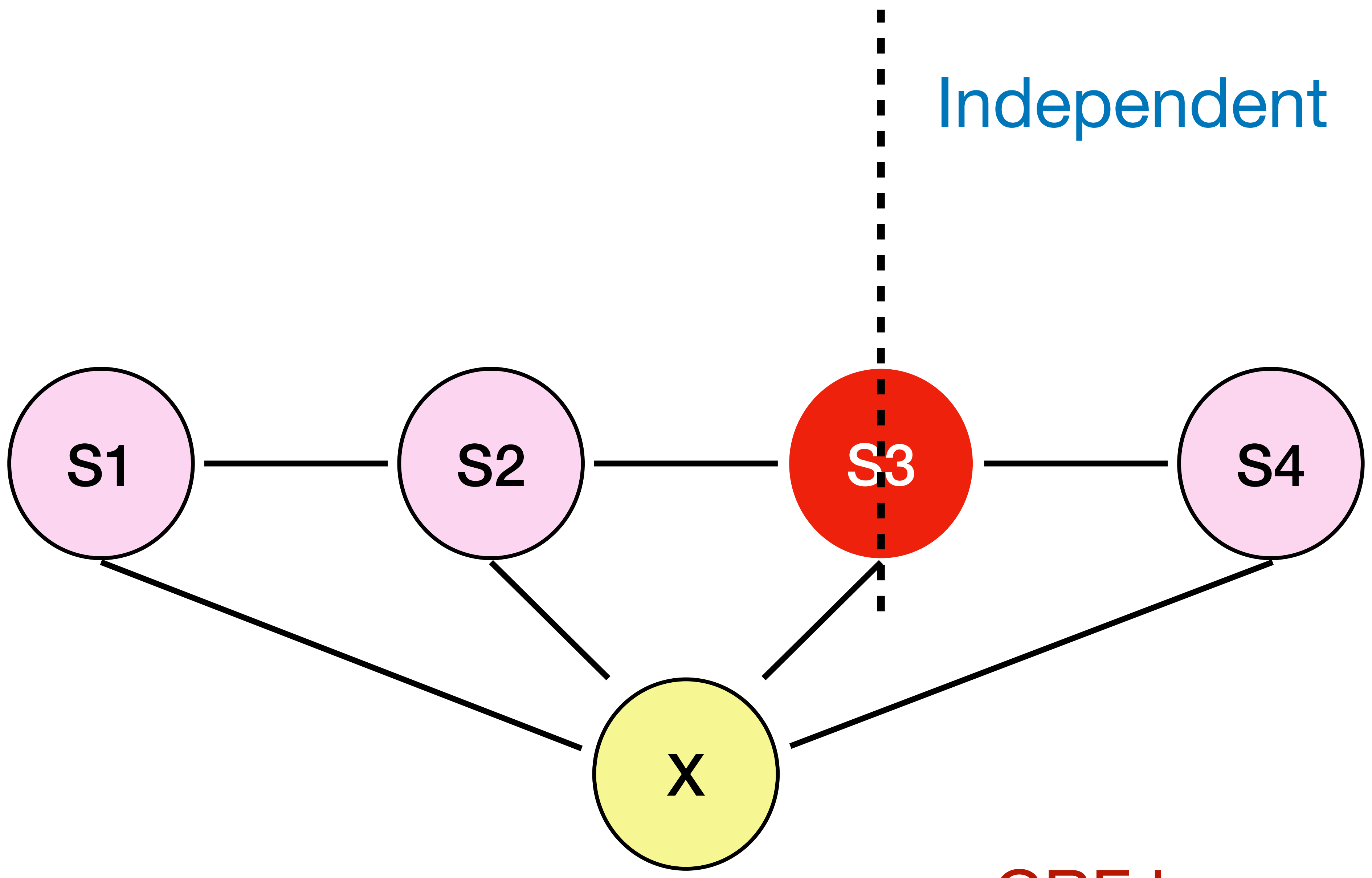
$$p(s_1, \ldots, s_m \,|\, x_1, \ldots, x_m) = \frac{\prod_{j-1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s_j))}{\sum_{s'_1, \ldots, s'_m \in \mathbb{S}} \prod_{j=1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s'_{j-1}, s'_j))}$$



CRF

Independence Assumptions in CRFs

$$p(s_1, \ldots, s_m \mid x_1, \ldots, x_m) = \frac{\prod_{j-1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s_j))}{\sum_{s'_1,\ldots,s'_m \in \mathbb{S}} \prod_{j=1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s'_{j-1}, s'_j))}$$



CRF

CRF has weaker independence assumption than MEMMs!

Decoding with CRFs

- **Viterbi Algorithm still applicable!**

$$p(s_1, \ldots, s_m \mid x_1, \ldots, x_m) = \frac{\prod_{j-1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s_j))}{\boxed{\sum_{s'_1, \ldots, s'_m \in \mathbb{S}} \prod_{j=1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s'_{j-1}, s'_j))}}$$

Makes no effects on decoding!

Computation of the Global Nominator

- **How to compute the global nominator?**
  - Dynamic programming similar to the Viterbi algorithm
  - Replacing the maximum operation in decoding with sum operation

$$p(s_1, \ldots, s_m | x_1, \ldots, x_m) = \frac{\prod_{j-1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s_j))}{\boxed{\sum_{s'_1, \ldots, s'_m \in \mathbb{S}} \prod_{j=1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s'_{j-1}, s'_j))}}$$

Partition Function

$$\pi[j, s] = \boxed{\sum_{s_1, \ldots, s_{j-1}}} \left[\prod_{k=1}^{j-1} \exp(v \cdot f(x_1, \ldots, x_m, k, s_{k-1}, s_k)) \right] \exp(v \cdot f(x_1, \ldots, x_m, k, s_{j-1}, s)))$$

Overview

- **The Sequence Labeling Problem**
  - General Structured Prediction Tasks
  - Part-of-speech Tagging: A case study
  - Generative Models vs. Discriminative Models
- **Hidden Markov Model (HMM)**
  - Basic definitions
  - Parameter estimation
  - The Viterbi algorithm
- **Log-Linear Models**
  - Maximum Entropy Markov Models (MEMMs)
  - Conditional Random Fields (CRFs)
  - Parameter estimation

Maximum Likelihood Estimation

- **Need to maximize:**

$$\max_{v} L(v) = \sum_{i=1}^{N} \log P(S_i \mid X_i; v)$$

$$= \sum_{i=1}^{N} v \cdot f(X_i, S_i) - \sum_{i=1}^{N} \log \sum_{s' \in \mathbb{S}} e^{v \cdot f(X_i, S')}$$

Do we need to manually derive the gradients?

Back-propagation!

- **Calculating gradients:**

$$\frac{\partial L(v)}{\partial v_k} = \underbrace{\sum_{i=1}^{N} f_k(X_i, S_i)}_{\text{Empirical counts}} - \underbrace{\sum_{i=1}^{N} \sum_{S' \in \mathbb{S}} f_k(X_i, S') p(S' \mid X_i; v)}_{\text{Expected counts}}$$

See Collin's notes for derivations

Model Performance

	POS Tagging	NER
HMM	96.4%	75.3
MEMM	96.9%	85.9
CRF	97.3%	88.7

Reading Materials

- **Notes from Michael Collins:**
  - Sequence Labeling and HMMs
  - EM Algorithm
  - Log-linear Models
  - MEMMs and CRFs