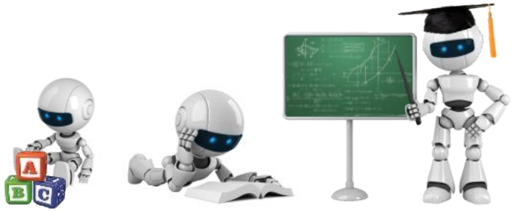
USC Viterbi
School of Engineering
Department of Computer Science

CSCI 544
Applied Natural Language Processing

Mohammad Rostami
USC Computer Science Department



Adapted from slides by Danqi Chen, Michael Collins, and Max Ma

Logistical Notes

- Project Group Formation Deadline: 09/12
  - Most students have joined a group (~220 students).
  - Contact groups with 3 or more members on Excel
  - Do NOT form more than 50 groups
  - Meet weekly, helpful for both HW and project
- Next Lecture
- Paper Selection Deadline: 09/19
  - Only one group!
  - Check and then enter your paper: https://docs.google.com/spreadsheets/d/1_vafG77ijmETCnuVZvKpT35k--5op5wn71GZXgAY7O0
- Project Proposal Deadline: 10/03
  - Think about extending your paper

HMM Assumptions

- Markov Assumption on **S**

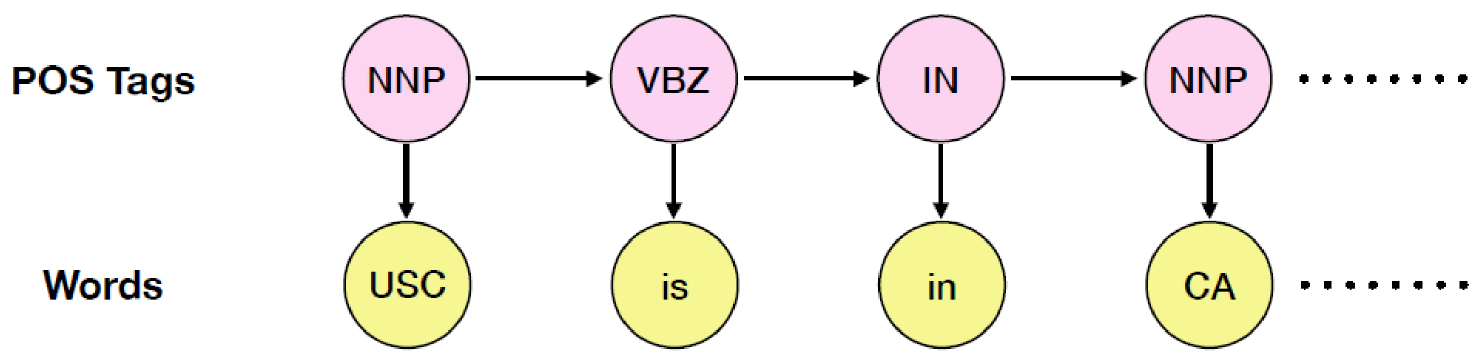
$$P(S_j = s_j \mid S_{j-1} = s_{j-1}, \ldots, S_1 = s_1) = P(S_j = s_j \mid S_{j-1} = s_{j-1})$$

Transition Probabilities

- Conditional independence of **X** and **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



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

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

HMM Assumptions

- 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

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

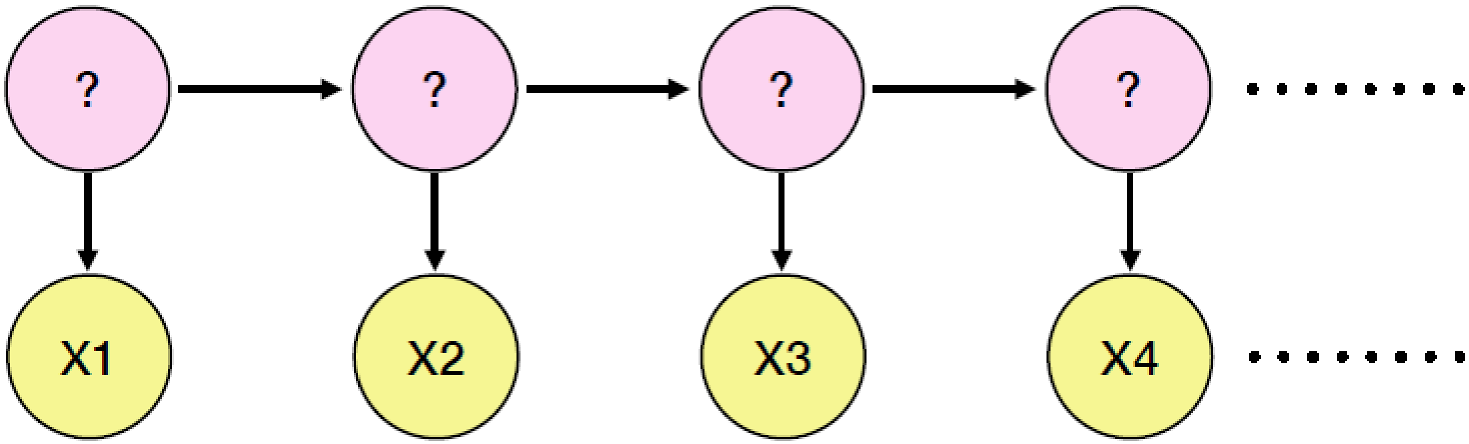
How to model $P(X_j = x_j \,|\, S_j = s_j)$

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

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

Decoding with HMM

- Given an input sequence $x_1, \ldots, x_m$ compute:



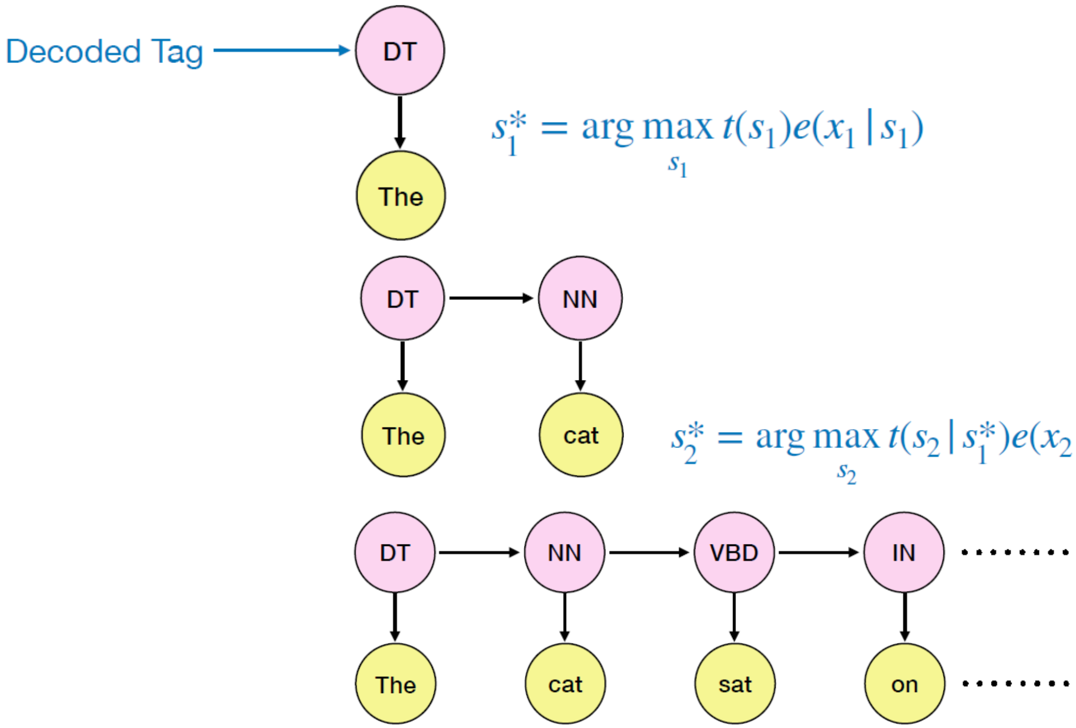
$$S^* = \arg\max_{s_1,\ldots,s_m} p(x_1,\ldots,x_m,s_1,\ldots,s_m) = t(s_1)\prod_{j=2}^{m} t(s_j \mid s_{j-1}) \prod_{j=1}^{m} e(x_j \mid s_j)$$

How can we maximize this over all state sequences?

- Brute-force search: 45^14~10^23 -> +1000 years with THz speed!

Greedy Decoding

- Start from the first word and decode one state at a time



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

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

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

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

Viterbi Decoding

- A dynamic programming algorithm
- The basic data structure will be a table that stores the maximum probability for any state sequence ending in state ***s*** at position ***j***.

$\pi[1, s] = t(s)e(x_1|s)$, and for $j > 1$,

$$\pi[j, s] = \max_{s_1 \ldots s_{j-1}} \left[\underbrace{t(s_1)e(x_1|s_1) \left(\prod_{k=2}^{j-1} t(s_k|s_{k-1})e(x_k|s_k) \right)}_{\text{The value for position } j\text{-}1} \underbrace{t(s|s_{j-1})e(x_j|s)}_{\text{Word } j} \right]$$

- Ex: The man saw the rat: a table with the size 5*#(tag classes)

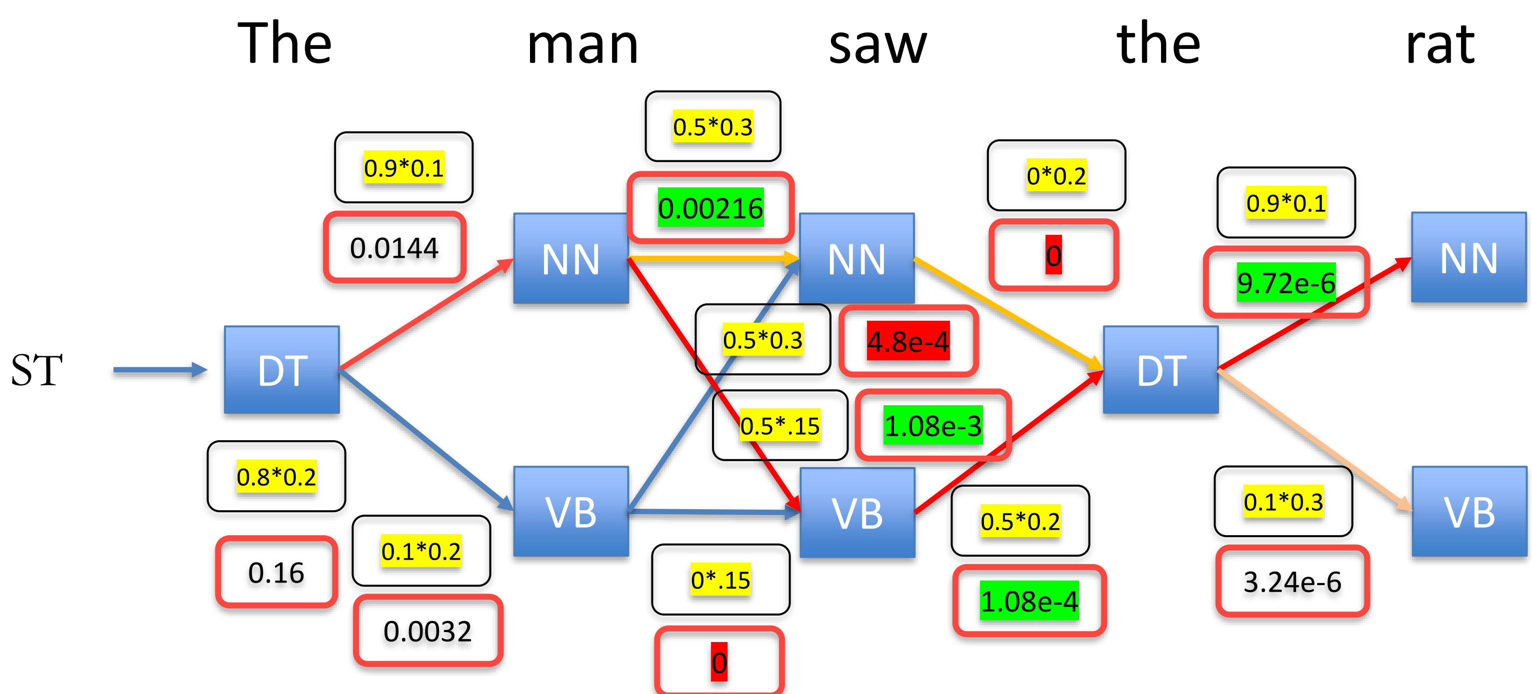
Viterbi Decoding: Example

- Sentence: The man saw the rat -> 5^4 vs 5*4

	DT	NN	VB
S	0.8	0.2	0
DT	0	0.9	0.1
NN	0	0.5	0.5
VB	0.5	0.5	0

	The	man	saw	rat
DT	0.2	0	0	0
NN	0	0.1	0.3	0.1
VB	0	0.2	0.15	0.3

$$p(s_j|s_{j-1})p(x_j|s_j) = t(s_j|s_{j-1})e(x_j|s_j)$$



Viterbi Decoding

- Algorithm

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

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

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

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

  - We then have

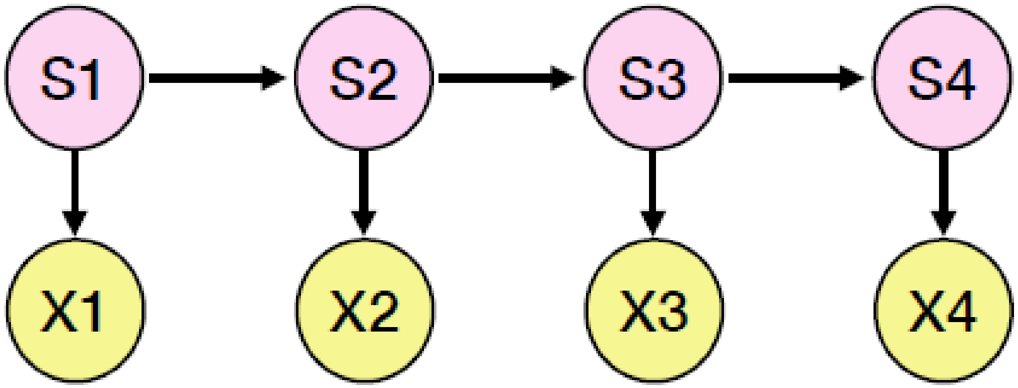
$$\max_{s_1 \ldots s_m} p(x_1 \ldots x_m, s_1 \ldots s_m; \underline{\theta}) = \max_s \pi[m, s]$$

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

HMM Limitations

- Transition probabilities are position independent
- Limited dependencies: Markov Assumption

- Is he informed? VS he informed



Generative vs Discriminative Models

- Can we have discriminative models for sequence labeling?

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

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

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

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

POS Features for Log-Linear Models

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

- We can use the current word, POS, and the surrounding words and their POS

Number of Features

$$f_1 = \begin{cases} 1, & \text{if } x_i = \text{is}, \ s_i = \text{VBZ} \\ 0, & \text{otherwise} \end{cases}$$

Vocabulary size $\longrightarrow VK$

POS tag size $\uparrow$

$$f_2 = \begin{cases} 1, & \text{if } x_{i-1} = \text{USC}, \ x_i = \text{is}, \ s_{i-1} = \text{NNP}, \text{ and } s_i = \text{VBZ} \\ 0, & \text{otherwise} \end{cases}$$

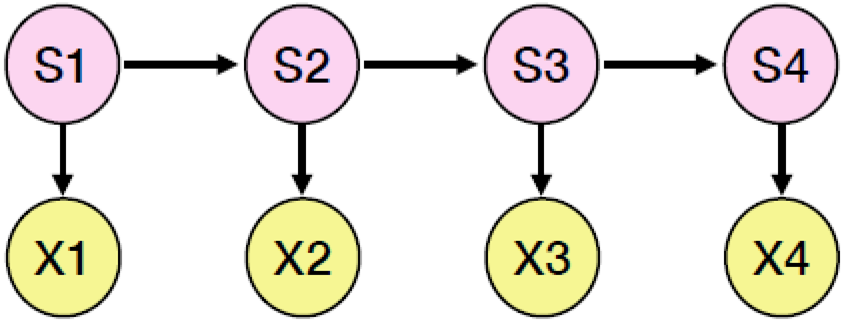
V^2K^2

$$f_3 = \begin{cases} 1, & \text{if } x_{i-1} = \text{USC}, \ x_i = \text{is}, \ x_{i+1} = \text{in}, \ s_{i-1} = \text{NNP}, \text{ and } s_i = \text{VBZ} \\ 0, & \text{otherwise} \end{cases}$$

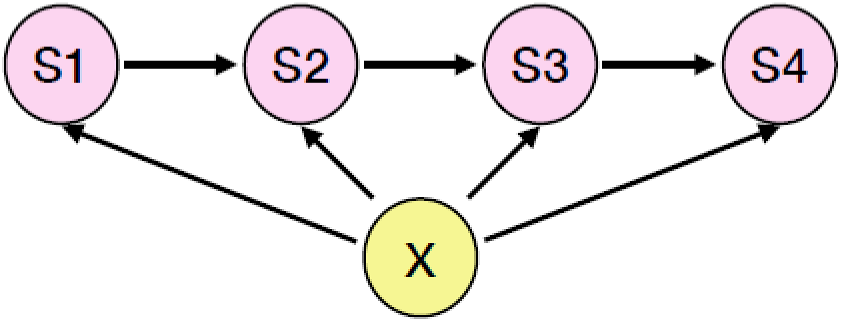
V^3K^2

Maximum-Entropy Markov Models (MEMMs)

- We assume that the hidden states are generated from the observations
- States are hidden and by observing data, we want to assign a probability for any possible state
- We still use Markov assumption on states
- Context is considered to determine POS



HMM



MEMM

Maximum-Entropy Markov Models (MEMMs)

- Markov assumption on ***S***

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

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

- We model each term using a log-linear model

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

MEMM: Training

- We use a training dataset and solve for optimal ***v***

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

- Potential solution: gradient-based techniques

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

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

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

MEMM: Decoding

- Viterbi Algorithm

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

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

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

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

MEMM: Decoding

- Viterbi Algorithm

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

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

    where s_0 is a special "initial" state.

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

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

  - We then have

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

MEMM vs HMM

- Performance: 96.9% vs 96.4% on WSJ
- Transition probability modeling

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

- MEMM feature vector encode contextual information
- Training procedure for parameter estimation in MEMMs is more expensive than in HMMs

Conditional Random Fields (CRFs)

- Motivation

- HMM Assumption:

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

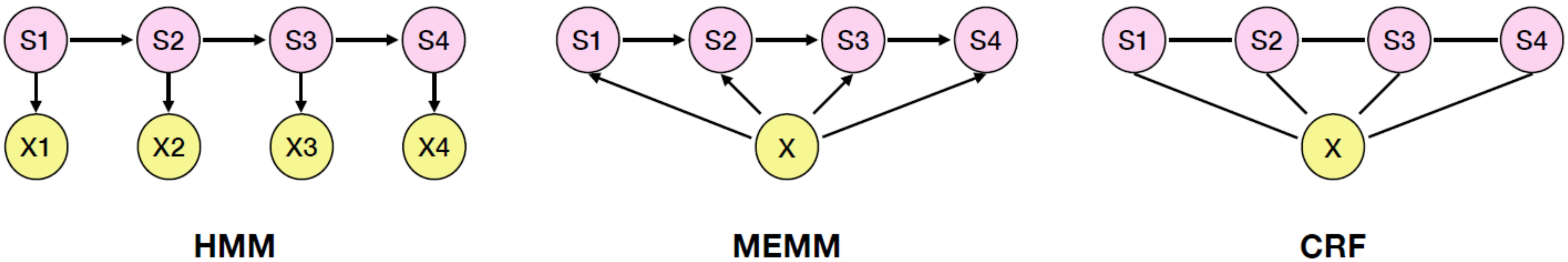
$$= \prod_{j=1}^{m} p(s_j \mid s_{j-1}, x_1, \ldots, x_m)$$ Markov assumption

- Can we build a log-linear model to directly model the conditional distribution?

Conditional Random Fields (CRFs)

- Formulation

- Can we have more complex models?



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

CRFs: Training and Decoding

- Training is similar to MEMM
- Decoding: Viterbi Algorithm

- We replace the maximum operation with summation operation

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

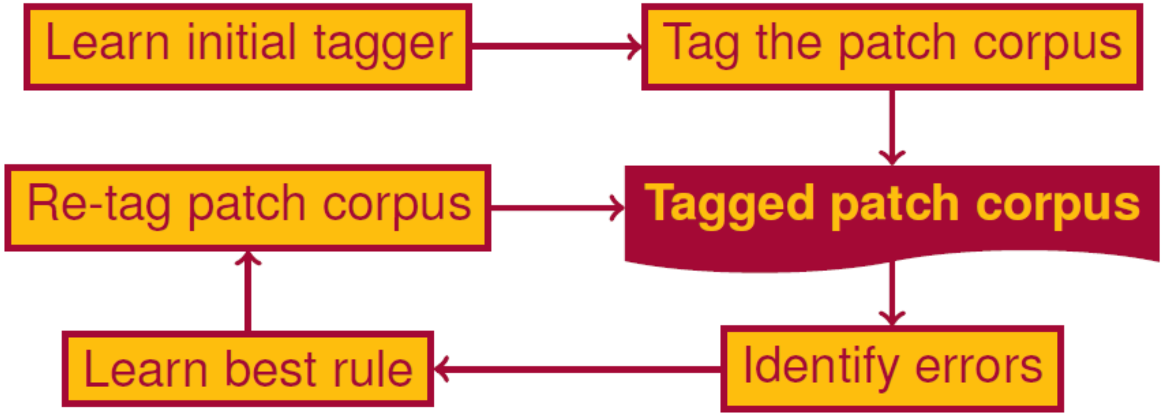
$$\pi[j, s] = \boxed{\sum_{s_1, \ldots, s_{j-1}}} \left[\prod_{k=1}^{j-1} \exp(v \cdot f(x_1, \ldots, x_m, k, s_{k-1}, s_k)) \right] \exp(v \cdot f(x_1, \ldots, x_m, k, s_{j-1}, s)))$$

Performance Comparison

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

Brill Tagger

- Brill (1992): A Simple Rule-Based Part of Speech Tagger
- Explainable Model



- Procedure for finding best rule
- Find most common error (e.g., noun tagged as verb)
- Find best rule to correct that error

- Rules must be applied in the order they are learned