CS378: Natural Language Processing
Lecture 6: Maximum Entropy Markov Model

The University of Texas at Austin

Eunsol Choi

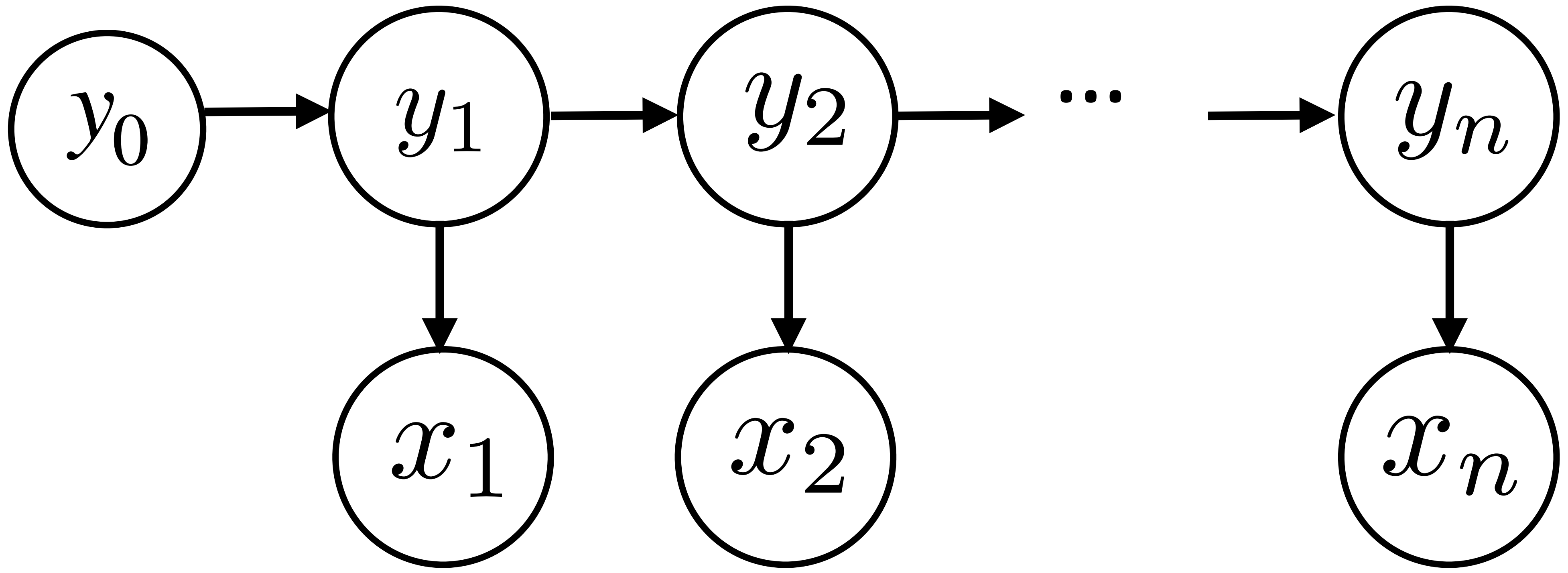
Parts of this lecture adapted from Greg Durrett, Yejin Choi, Yoav Artzi

Overview

- Sequence Modeling Problems in NLP
- Generative Model: Hidden Markov Models (HMM)
- Discriminative Model:
  Maximum Entropy Markov Models (MEMM)
  Conditional Random Fields
- Unsupervised Learning: Expectation Maximization

Recall: HMMs

- Observations (X) generated from hidden states (Y).



$$p(x_1 \dots x_n, y_1 \dots y_n) = q(STOP|y_n) \prod_{i=1}^{n} q(y_i|y_{i-1}) e(x_i|y_i)$$

- Training: maximum likelihood estimation

- Inference problem: $\text{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = \text{argmax}_{\mathbf{y}} \frac{P(\mathbf{y}, \mathbf{x})}{\cancel{P(\mathbf{x})}}$

Recall: The Viterbi Algorithm

- Dynamic program for computing (for all i)

the max score of a sequence of length i ending in tag y_i

$$\pi(i, y_i) = \max_{y_1 \ldots y_{i-1}} p(x_1 \ldots x_i, y_1 \ldots y_i)$$

- Iterative Computation:

$$\pi(0, y_0) = \begin{cases} 1 \text{ if } y_0 == START \\ 0 \text{ otherwise} \end{cases}$$

- For I = 1… n:
  - Store score

$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i | y_i) q(y_i | y_{i-1}) \pi(i-1, y_{i-1})$$

  - Store back-pointer

$$bp(i, y_i) = \arg\max_{y_{i-1}} e(x_i | y_i) q(y_i | y_{i-1}) \pi(i-1, y_{i-1})$$

The Viterbi Algorithm: Runtime

- Linear in sentence length (n)
- Polynomial in the number of possible tags (|K|)

$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i|y_i)q(y_i|y_{i-1})\pi(i-1, y_{i-1})$$

$$O(n|\mathcal{K}|) \text{ entries in } \pi(i, y_i)$$

$$O(|\mathcal{K}|) \text{ time to compute each } \pi(i, y_i)$$

- Total Runtime: $O(n|\mathcal{K}|^2)$

- Would there any scenarios where we would choose beam search?

Tagsets in Different Languages

Language	Source	# Tags
Arabic	PADT/CoNLL07 (Hajič et al., 2004)	21
Basque	Basque3LB/CoNLL07 (Aduriz et al., 2003)	64
Bulgarian	BTB/CoNLL06 (Simov et al., 2002)	54
Catalan	CESS-ECE/CoNLL07 (Martí et al., 2007)	54
Chinese	Penn ChineseTreebank 6.0 (Palmer et al., 2007)	34
Chinese	Sinica/CoNLL07 (Chen et al., 2003)	294
Czech	PDT/CoNLL07 (Böhmová et al., 2003)	63
Danish	DDT/CoNLL06 (Kromann et al., 2003)	25
Dutch	Alpino/CoNLL06 (Van der Beek et al., 2002)	12
English	PennTreebank (Marcus et al., 1993)	45
French	FrenchTreebank (Abeillé et al., 2003)	30
German	Tiger/CoNLL06 (Brants et al., 2002)	54
German	Negra (Skut et al., 1997)	54
Greek	GDT/CoNLL07 (Prokopidis et al., 2005)	38
Hungarian	Szeged/CoNLL07 (Csendes et al., 2005)	43
Italian	ISST/CoNLL07 (Montemagni et al., 2003)	28
Japanese	Verbmobil/CoNLL06 (Kawata and Bartels, 2000)	80
Japanese	Kyoto4.0 (Kurohashi and Nagao, 1997)	42
Korean	Sejong (http://www.sejong.or.kr)	187
Portuguese	Floresta Sintá(c)tica/CoNLL06 (Afonso et al., 2002)	22
Russian	SynTagRus-RNC (Boguslavsky et al., 2002)	11
Slovene	SDT/CoNLL06 (Džeroski et al., 2006)	29
Spanish	Ancora-Cast3LB/CoNLL06 (Civit and Martí, 2004)	47
Swedish	Talbanken05/CoNLL06 (Nivre et al., 2006)	41
Turkish	METU-Sabanci/CoNLL07 (Oflazer et al., 2003)	31

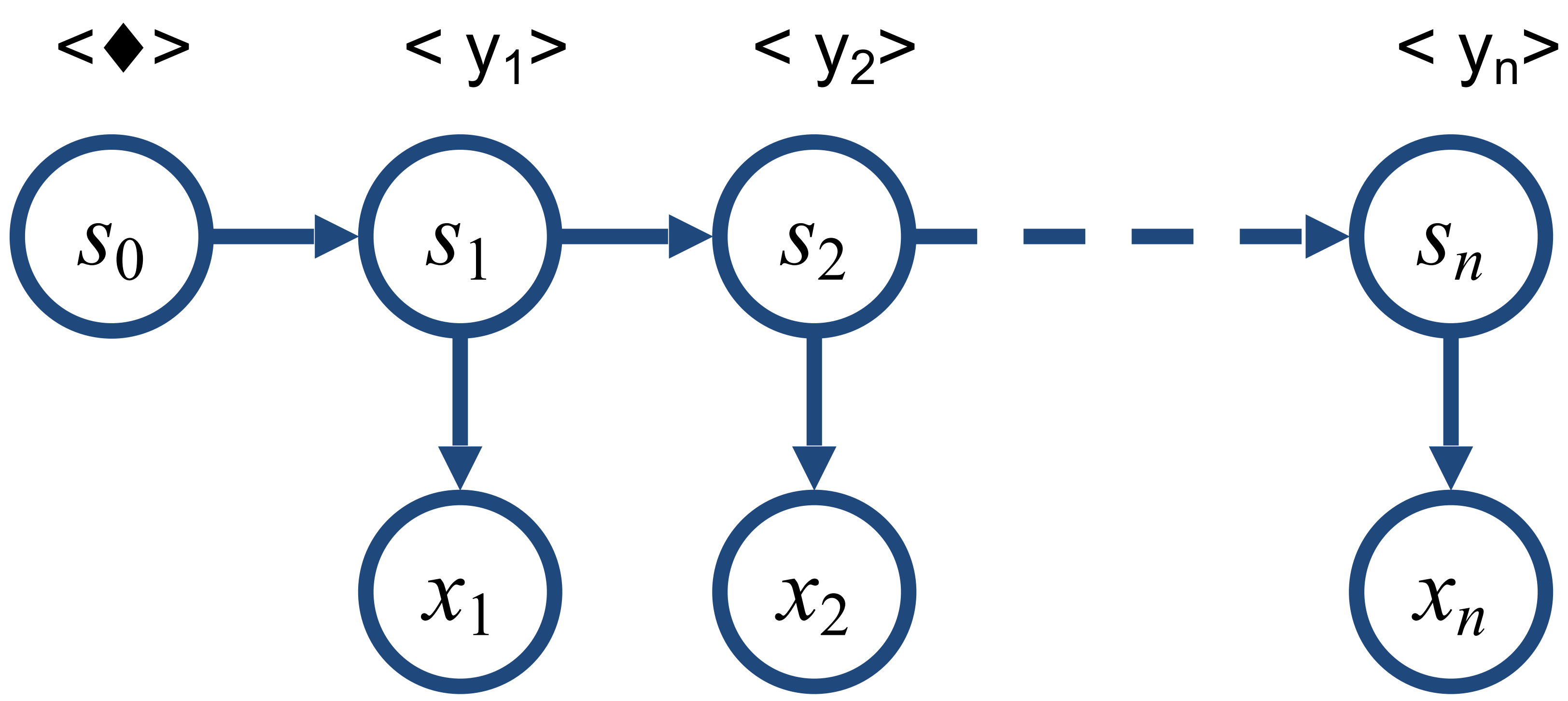
$294^2 = 86436$

$45^2 = 2045$

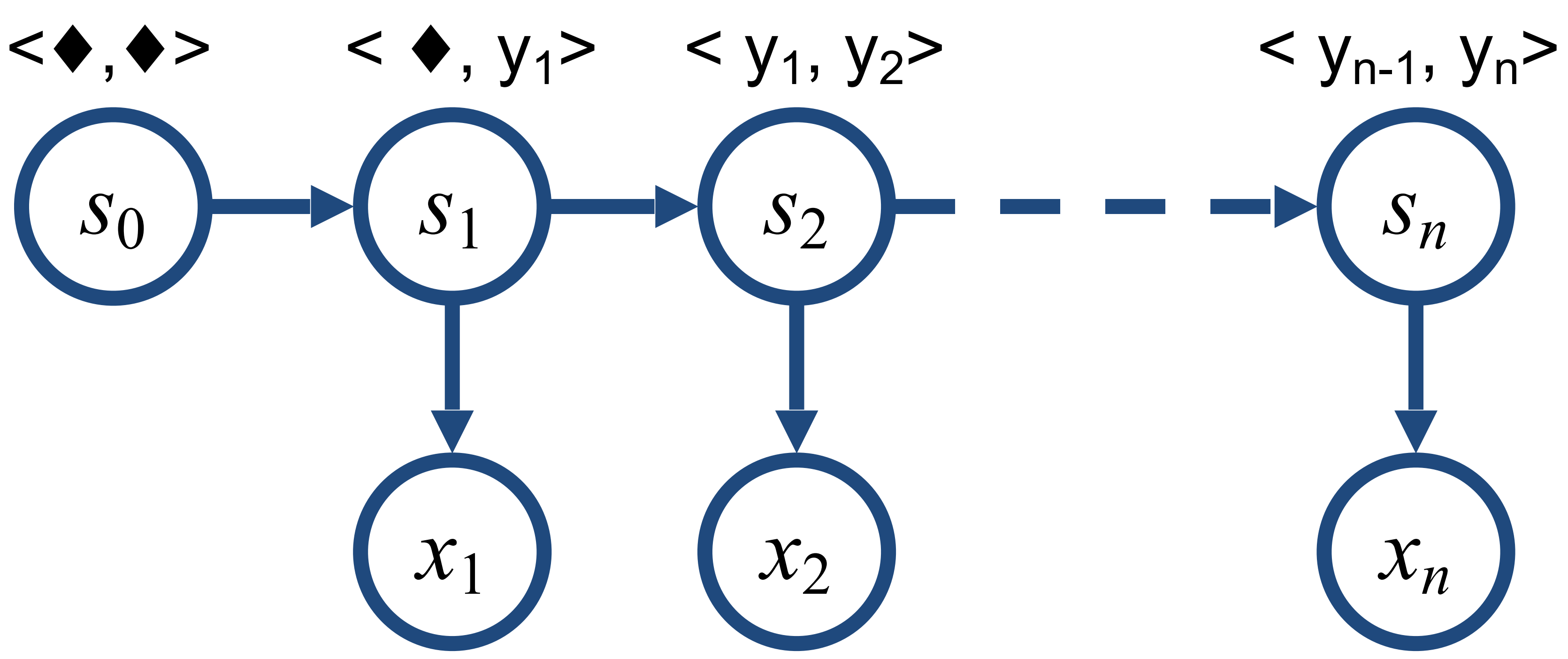
$11^2 = 121$

Trigram HMM Taggers

- Bigram model: y_1 = NNP, y_2 = NNP, …



- 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 (sparsity)
  - 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

Slide credit: Dan Klein

Can we do better?

- HMM is a generative model, estimation relies on counting!
  - Reminds you of something?

- Can we build a discriminative model, incorporating rich features?

Naive Bayes:

$$P(y)P(x \mid y)$$

HMM:

$$P(y_1, \ldots, y_n)P(x_1, \ldots, x_n \mid y_1, \ldots, y_n)$$

Logistic Regression:

$$P(y \mid x)$$

Maximum Entropy Markov Model :

$$P(y_1, \ldots, y_n \mid x_1, \ldots, x_n)$$

Named Entity Recognition (NER)

B-PER	I-PER	O	O	O	B-LOC	O	O	O	B-ORG	O	O
Barack	*Obama*	*will*	*travel*	*to*	*Hangzhou*	*today*	*for*	*the*	*G20*	*meeting*	*.*
PERSON					LOC				ORG		

- BIO tagset: begin, inside, outside
- Sequence of tags — can we use an HMM?
- Would it do well?
  - Lots of O's
  - Insufficient features/capacity with multinomials (especially for unks)

Emission Features for NER

LOC

Leicestershire is a nice place to visit…

PER

Leonardo DiCaprio won an award…

LOC

I took a vacation to Boston

ORG

Apple released a new version…

LOC PER

Texas governor Greg Abbott said

ORG

According to the New York Times…

Emission Features for NER

- Word features
  - Capitalization
  - Word shape
  - Prefixes/suffixes
  - Lexical indicators
- Context features
  - Words before/after
- Word clusters

Leicestershire

Boston

Apple released a new version…

According to the New York Times…

Maximum Entropy Markov Models (MEMM)

$$p(y_1 \dots y_n | x_1 \dots x_n) = \prod_{i=1}^{n} p(y_i | y_1 \dots y_{i-1}, x_1 \dots x_n)$$ Chain rule

$$= \prod_{i=1}^{n} p(y_i | y_{i-1}, x_1 \dots x_n)$$ Independence assumption

- Log linear model for sequence tagging problem
- Learning:
  - Train $p(y_i \,|\, y_{i-1}, x_1, \dots, x_n)$ as a discrete log-linear model
- Scoring:

$$p(y_i | y_{i-1}, x_1 \dots x_n) = \frac{e^{w \cdot \phi(x_1 \dots x_n, i, y_{i-1}, y_i)}}{\sum_{y'} e^{w \cdot \phi(x_1 \dots x_n, i, y_{i-1}, y')}}$$

Learning for MEMM

- Scoring:

$$p(y_i|y_{i-1}, x_1 \dots x_n) = \frac{e^{w \cdot \phi(x_1 \dots x_n, i, y_{i-1}, y_i)}}{\sum_{y'} e^{w \cdot \phi(x_1 \dots x_n, i, y_{i-1}, y')}}$$

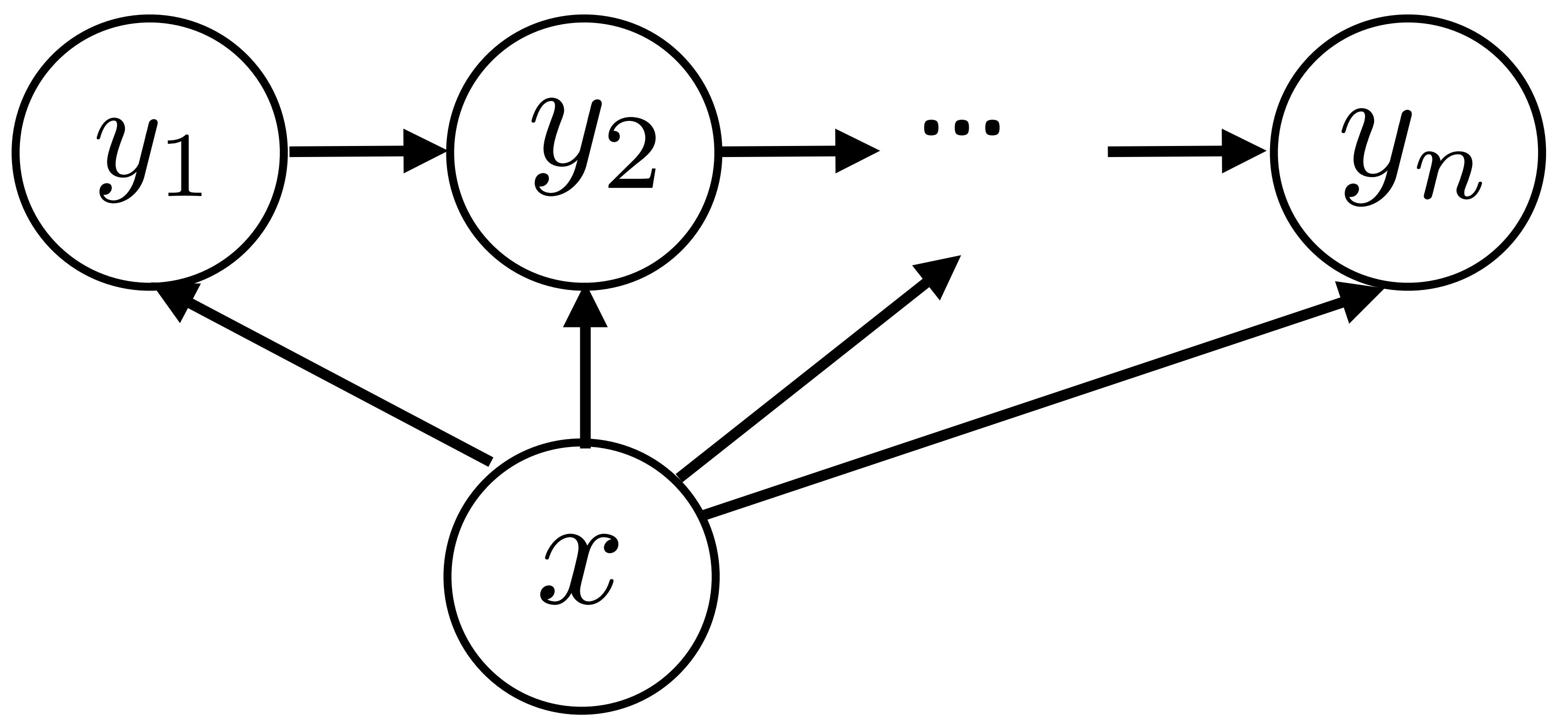
- Learning Objective: log likelihood of training data

$$L = \sum_{i=1}^{n} \log P(y_i \,|\, y_1, \dots, y_{i-1}, x_1, x_2 \dots x_n)$$

- Gradient ascent: same as logistic regression
  - Compute gradients with respect to weight w and update

Basic Features for NER

$$p(y_i|y_{i-1}, x_1 \dots x_n) = \frac{e^{w\cdot\phi(x_1\dots x_n, i, y_{i-1}, y_i)}}{\sum_{y'} e^{w\cdot\phi(x_1\dots x_n, i, y_{i-1}, y')}}$$



O B-LOC

Barack Obama will travel to Hangzhou today for the G20 meeting .

Transitions: $\mathrm{Ind}[y_{i-1}\ \&\ y_i]$ = Ind[O — B-LOC]

Emissions: Ind[B-LOC & Current word = *Hangzhou*]

Ind[B-LOC & Prev word = *to*]

Decoding for MEMM

- Given your model, finding the highest scoring **y**
- Not very different from decoding for HMMs
- Viterbi for HMMs

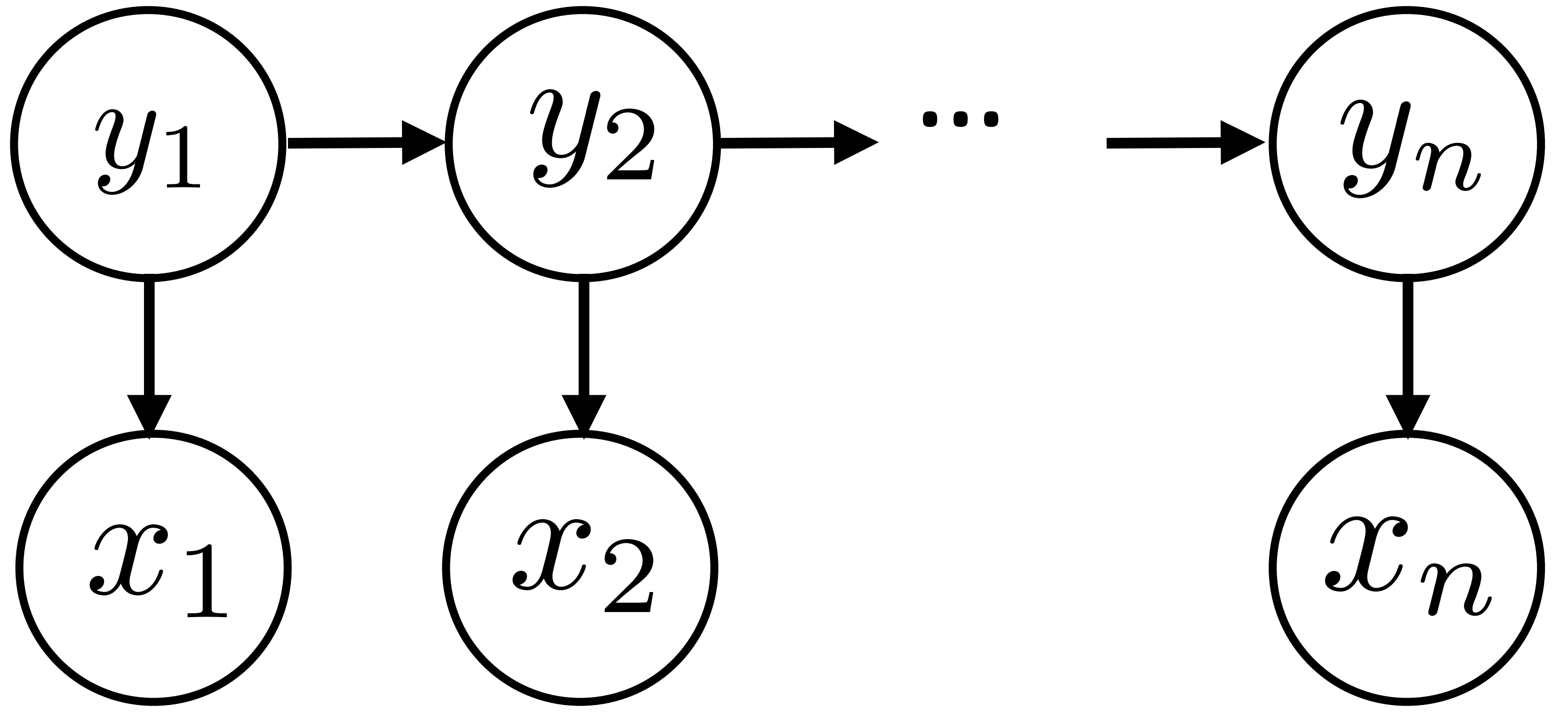
$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i|y_i)q(y_i|y_{i-1})\pi(i-1, y_{i-1})$$

- Viterbi for MEMM

$$\pi(i, y_i) = \max_{y_{i-1}} p(y_i|y_{i-1}, x_1 \ldots x_n)\pi(i-1, y_{i-1})$$

HMM vs. MEMM

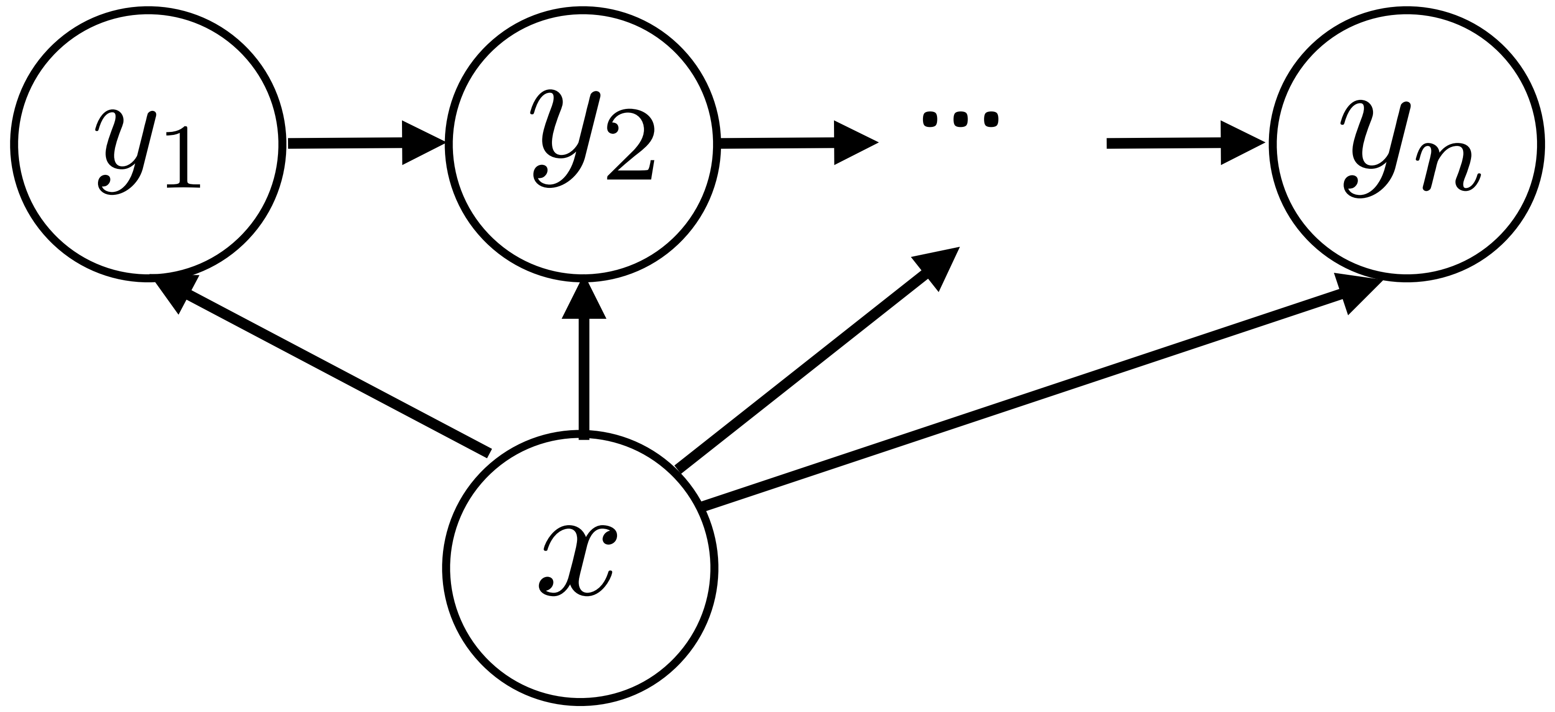
- HMM models joint distribution:



$$p(x_1 \dots x_n, y_1 \dots y_n) = q(STOP|y_n) \prod_{i=1}^{n} q(y_i|y_{i-1}) e(x_i|y_i)$$

- MEMM models conditional distribution:

$$p(y_1 \dots y_n|x_1 \dots x_n) = \prod_{i=1}^{n} p(y_i|y_{i-1}, x_1 \dots x_n)$$



POS Tagging Performances

- 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 : 93.7% accuracy / 82.6% on unks
- MEMM [Ratnaparkhi 1996]: 96.8% accuracy / 86.9% on unks

Slide credit: Dan Klein

Problems with MEMM / HMM

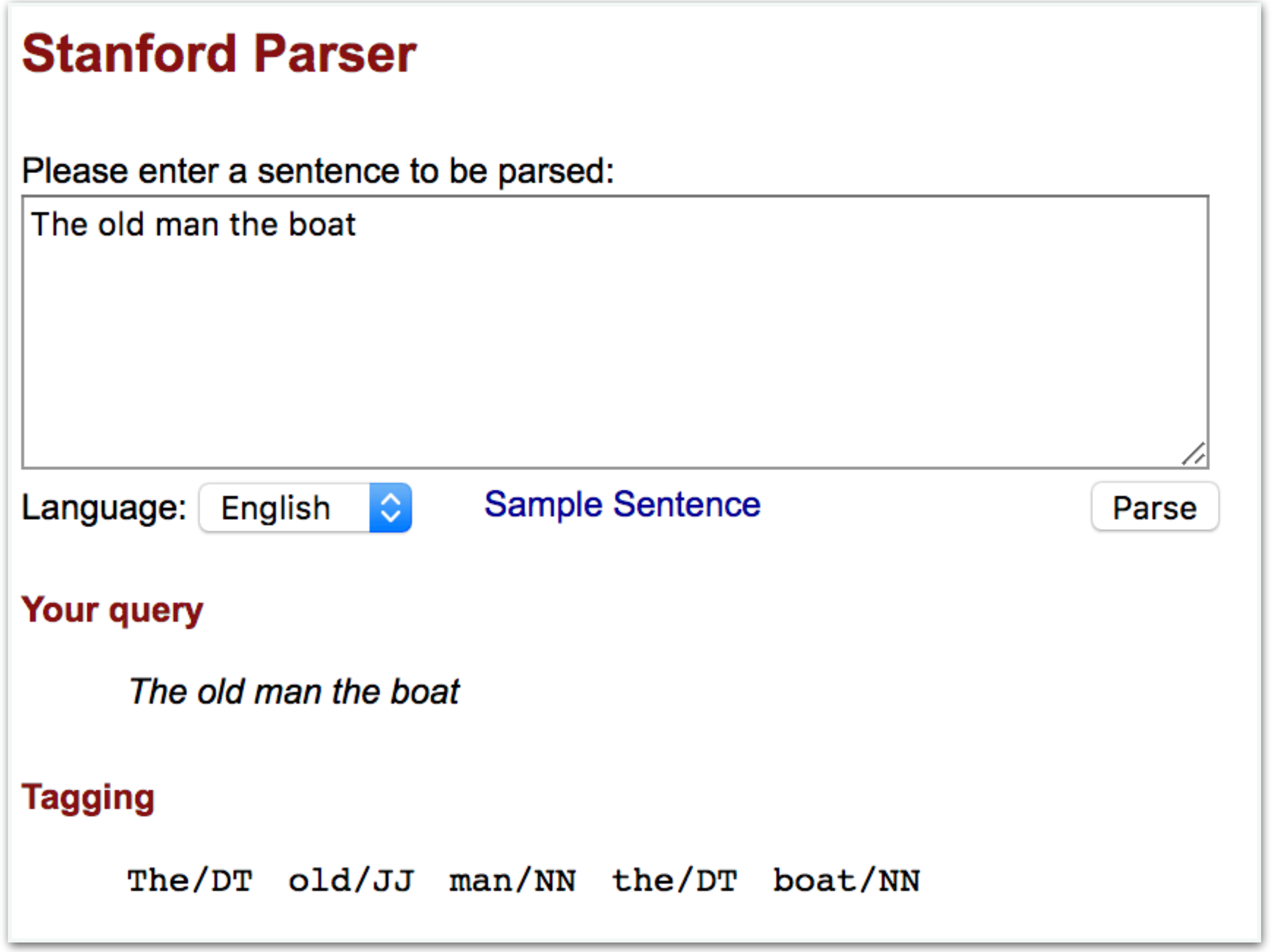
- Left-to-right assumption

The/? old/? man/?

The/? old/? man/? the/? boat/?

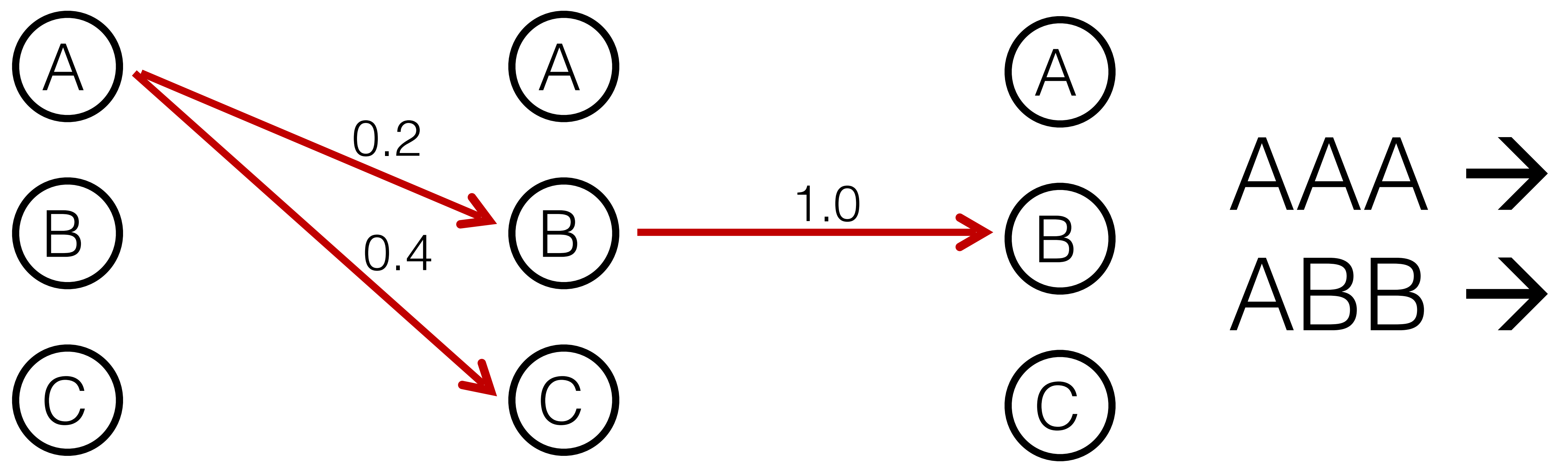
$$P(\text{The} \mid DT)P(JJ \mid DT)\ P(\text{old} \mid JJ)\ P(NN \mid JJ)\ P(\text{man} \mid NN)\ P(DT \mid NN)$$

$$P(\text{The} \mid DT)P(NN \mid DT)\ P(\text{old} \mid NN)\ P(VB \mid NN)\ P(\text{man} \mid VB)\ P(DT \mid VB)$$



Locally Normalized Model

- Probabilities are products of locally normalized probabilities
- Label bias:
  - States with fewer transitions are likely to be preferred.



from \ to	A	B	C
A	0.4	0.2	0.4
B	0.0	1.0	0.0
C	0.6	0.2	0.2

B -> B transitions are likely to take over even if rarely observed!

Can we build perceptrons - which wouldn't normalize?

- Perceptron:
  - Iteratively processes the data, reacting to training errors

- The (online structured) perceptron algorithm:
  - Start with zero weights
  - Visit training instances one by one
    - Make predictions: $\mathbf{y}^* = \text{argmax}_{\mathbf{y} \in Y} w \cdot \phi(\mathbf{x}, \mathbf{y})$
    - If correct ($\mathbf{y}^* == \mathbf{y_i}$), do nothing
    - If incorrect, adjust weights: $w = w + \phi(\mathbf{x_i}, \mathbf{y_i}) - \phi(\mathbf{x_i}, \mathbf{y}^*)$

How to find argmax efficiently??

Linear Perceptron Decoding

$$Y^* = \arg\max_Y w \cdot \phi(X, Y)$$

- Local Features

$$\phi(X, Y) = \sum_{j=1}^{n} \phi(X, j, y_{j-1}, y_j)$$

- Define $\pi(i, y_i)$ to be the max score of a sequence of length i ending in tag y_i

$$\pi(i, y_i) = \max_{y_{i-1}} w \cdot \phi(X, i, y_{i-1}, y_i) + \pi(i-1, y_{i-1})$$

- HMM Decoding:

$$\pi(i, y_i) = \max_{y_{i-1}} e(x_i|y_i) q(y_i|y_{i-1}) \pi(i-1, y_{i-1})$$

- Viterbi for MEMM

$$\pi(i, y_i) = \max_{y_{i-1}} p(y_i|y_{i-1}, x_1 \ldots x_n) \pi(i-1, y_{i-1})$$

POS Tagging Performances

- 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 : 93.7% accuracy / 82.6% on unks
- MEMM [Ratnaparkhi 1996]: 96.8% accuracy / 86.9% on unks
- Perceptron: 97.1% accuracy

Slide credit: Dan Klein

Let’s bring back probabilistic model again..

Conditional Random Fields

- CRFs: discriminative models with the following **globally-normalized** form:

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z}\prod_k \exp(\phi_k(\mathbf{x},\mathbf{y}))$$

normalizer

any real-valued scoring function of its arguments

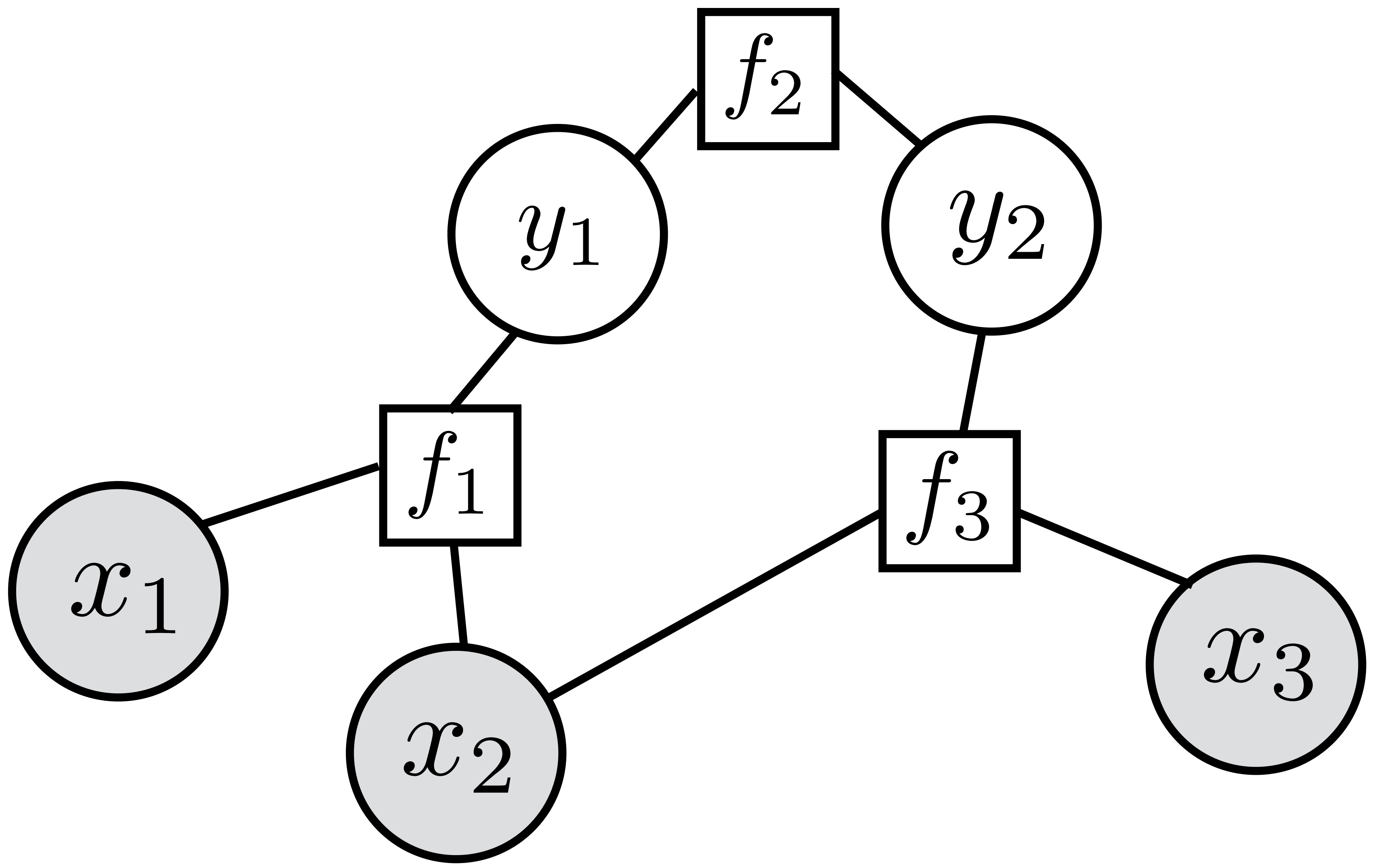
- We look at linear feature-based potentials $\phi_k(\mathbf{x},\mathbf{y}) = w^\top f_k(\mathbf{x},\mathbf{y})$

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z}\exp\left(\sum_{k=1}^{n} w^\top f_k(\mathbf{x},\mathbf{y})\right)$$

- Looks like our single weight vector multiclass logistic regression model

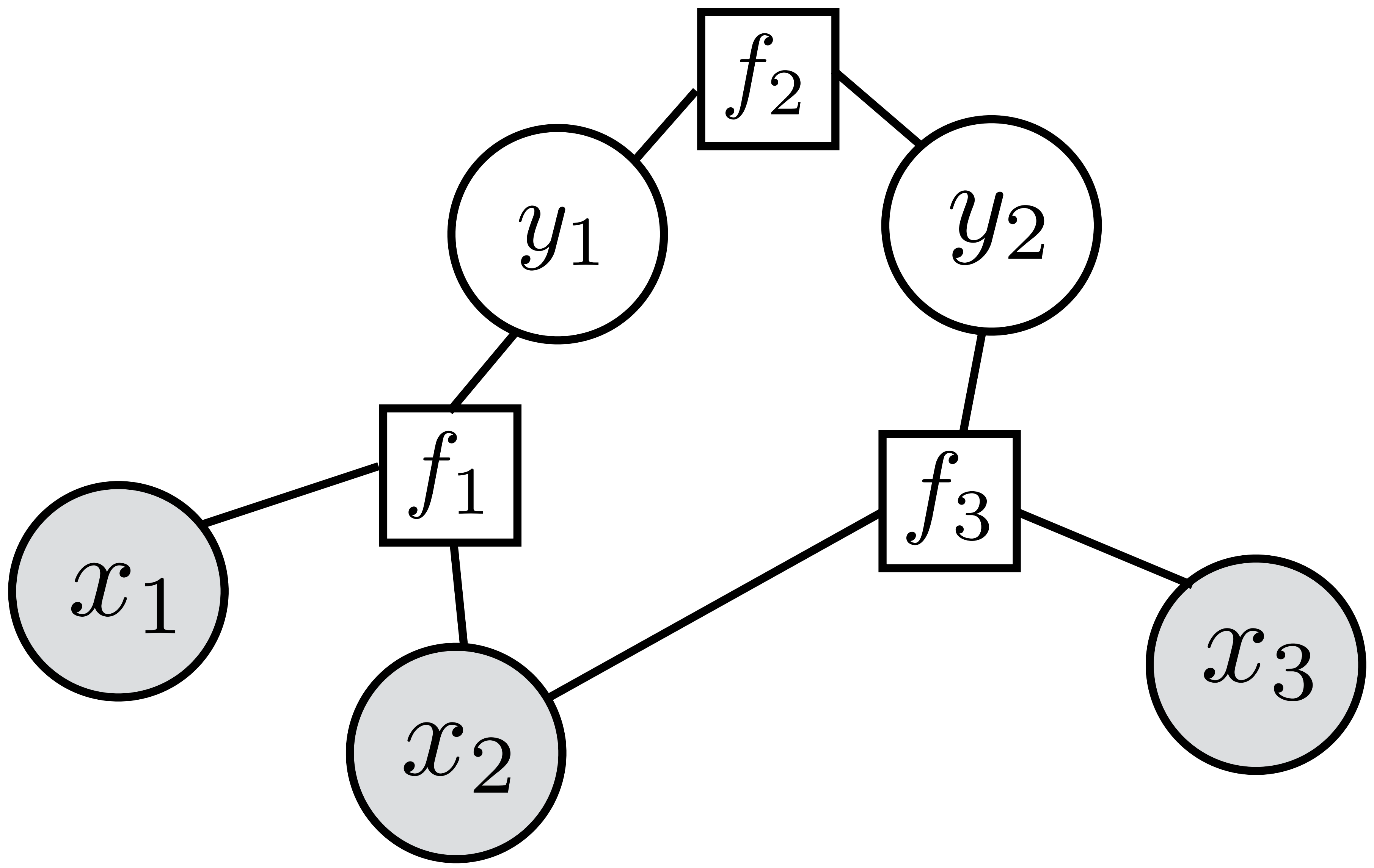
Conditional Random Fields

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \exp\left(\sum_{k=1}^{n} w^\top f_k(\mathbf{x}, \mathbf{y}) \right)$$



Conditional Random Fields

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \exp\left(\sum_{k=1}^{n} w^\top f_k(\mathbf{x}, \mathbf{y})\right)$$



- Normalizing constant

$$Z = \sum_{\mathbf{y}'} \exp\left(\sum_{k=1}^{n} w^\top f_k(\mathbf{x}, \mathbf{y}')\right)$$

- Inference: $\mathbf{y}_{\text{best}} = \text{argmax}_{\mathbf{y}'} \exp\left(\sum_{k=1}^{n} w^\top f_k(\mathbf{x}, \mathbf{y}')\right)$

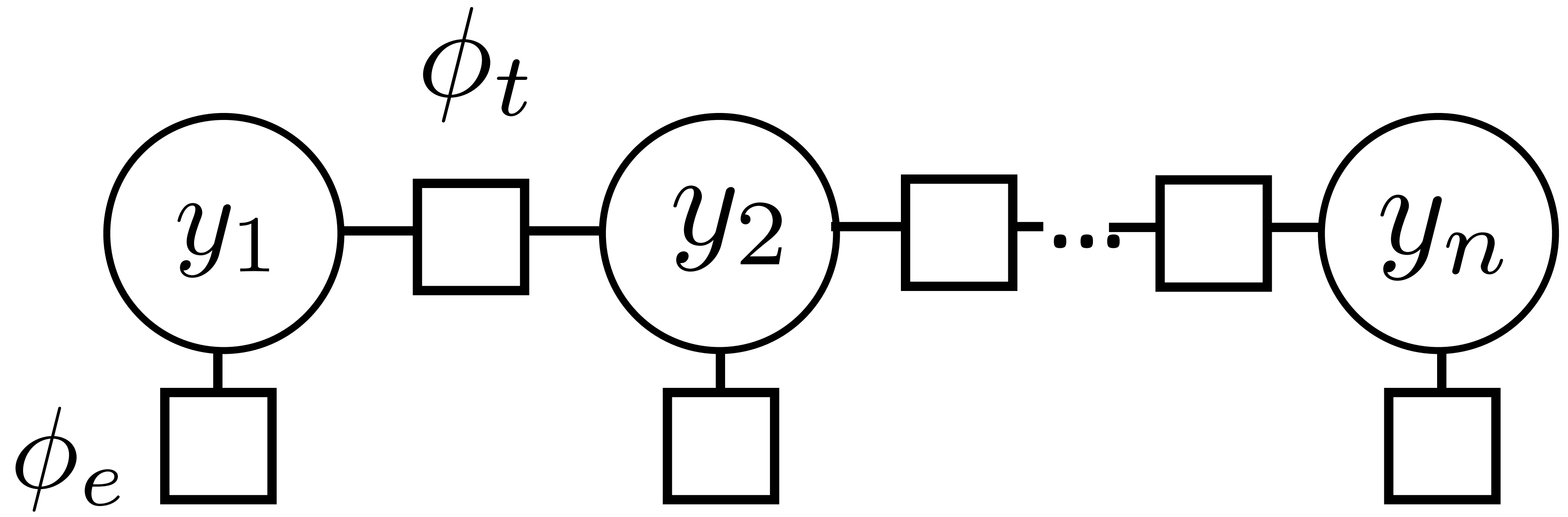
- Need to constrain the form of our CRFs to make it tractable:
  - Local features

Sequential CRFs

Sequential CRF: (one form)

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(y_i, i, \mathbf{x}))$$

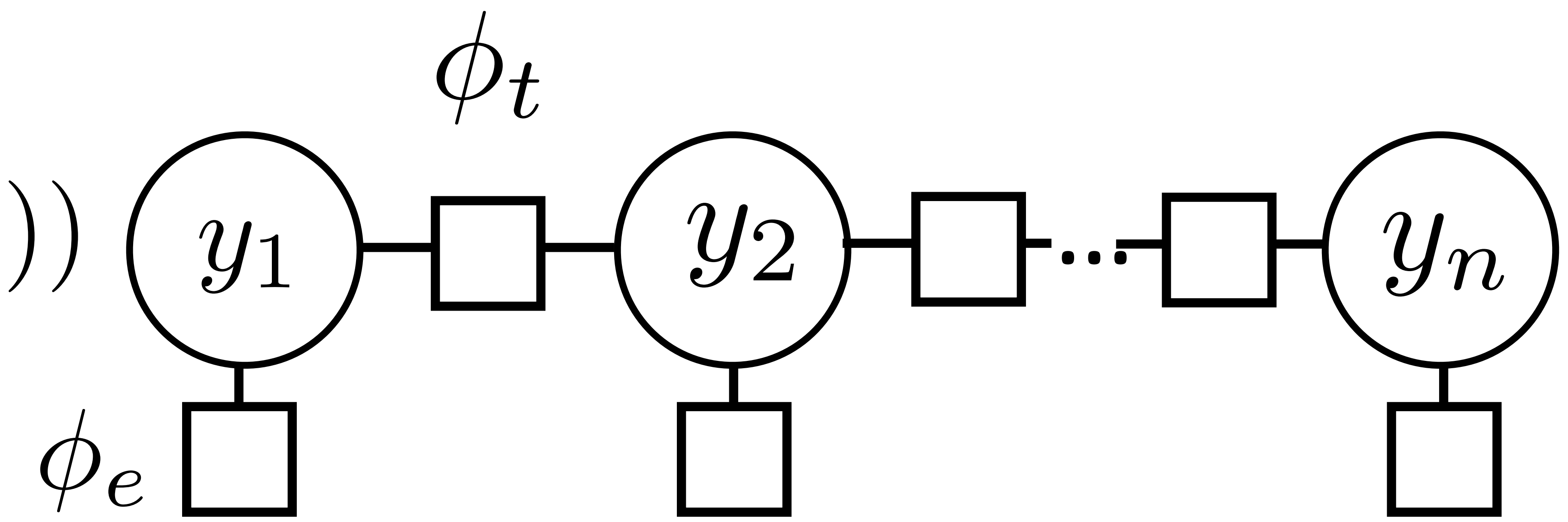
- Notation: omit **x** from the factor graph entirely (implicit), but every feature function connects to it



- Two types of factors: *transitions* ϕ_t (look at previous y, but not x) and *emissions* ϕ_e (look at y and all of **x**)

Feature Functions

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(y_i, i, \mathbf{x}))$$



- You can define features as you like (can be NN with 1B+ parameters).

$$P(\mathbf{y}|\mathbf{x}) \propto \exp w^\top \left[\sum_{i=2}^{n} f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} f_e(y_i, i, \mathbf{x}) \right]$$

CRFs Outline

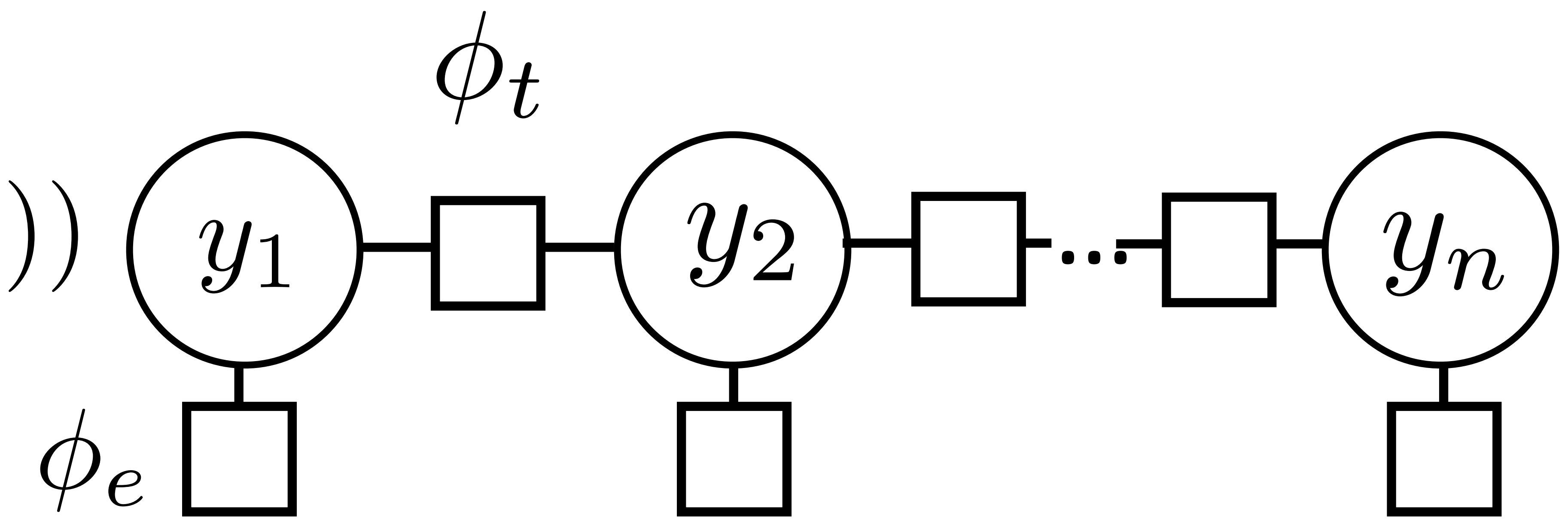
- Model: $$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z}\prod_{i=2}^{n}\exp(\phi_t(y_{i-1}, y_i))\prod_{i=1}^{n}\exp(\phi_e(y_i, i, \mathbf{x}))$$
  $$P(\mathbf{y}|\mathbf{x}) \propto \exp w^\top\left[\sum_{i=2}^{n} f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} f_e(y_i, i, \mathbf{x})\right]$$

- Inference

- Learning - a bit more complex! revisit next week

Decoding

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z}\prod_{i=2}^{n}\exp(\phi_t(y_{i-1}, y_i))\prod_{i=1}^{n}\exp(\phi_e(y_i, i, \mathbf{x}))$$



- How to compute? $\text{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x})$

- Similar to Viterbi during linear perceptron!

$$\pi(i, y_i) = \max_{y_{i-1}} w \cdot \phi(X, i, y_{i-1}, y_i) + \pi(i-1, y_{i-1})$$

- CRF decoding:

$$\pi(i, y_i) = \max_{y_{i-1}} \phi(x, i, y_{i-i}, y_i) + \pi(i-1, y_{i-1})$$

$$\pi(i, y_i) = max_{y_{i-1}} \phi_t(y_{i-1}, y_i) + \phi_e(y_i, i, x) + \pi(i-1, y_{i-1})$$

POS Tagging Performances

- 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 : 93.7% accuracy / 82.6% on unks
- MEMM [Ratnaparkhi 1996]: 96.8% accuracy / 86.9% on unks
- Perceptron: 97.1% accuracy
- CRF: 97.3% accuracy
- State-of-the-art (neural model w/CRF): 97.5% / 89%+

Slide credit: Dan Klein

Errors

	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

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)

Remaining Errors

- Lexicon gap (word not seen with that tag in training) 4.5%
- Unknown word: 4.5%
- Could get right: 16%
- 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?”

Remaining Questions

- How to handle noisy text?
- How to keep high accuracy when we go to new domains?
- Can we incorporate domain specific lexicon? (List of protein names?)
- How can we combine unlabeled data efficiently to labeled data?

Summary

- Sequence Modeling Problems in NLP
- Generative Model: Hidden Markov Models (HMM)
- Discriminative Model:
  Maximum Entropy Markov Models (MEMM)
  Conditional Random Fields