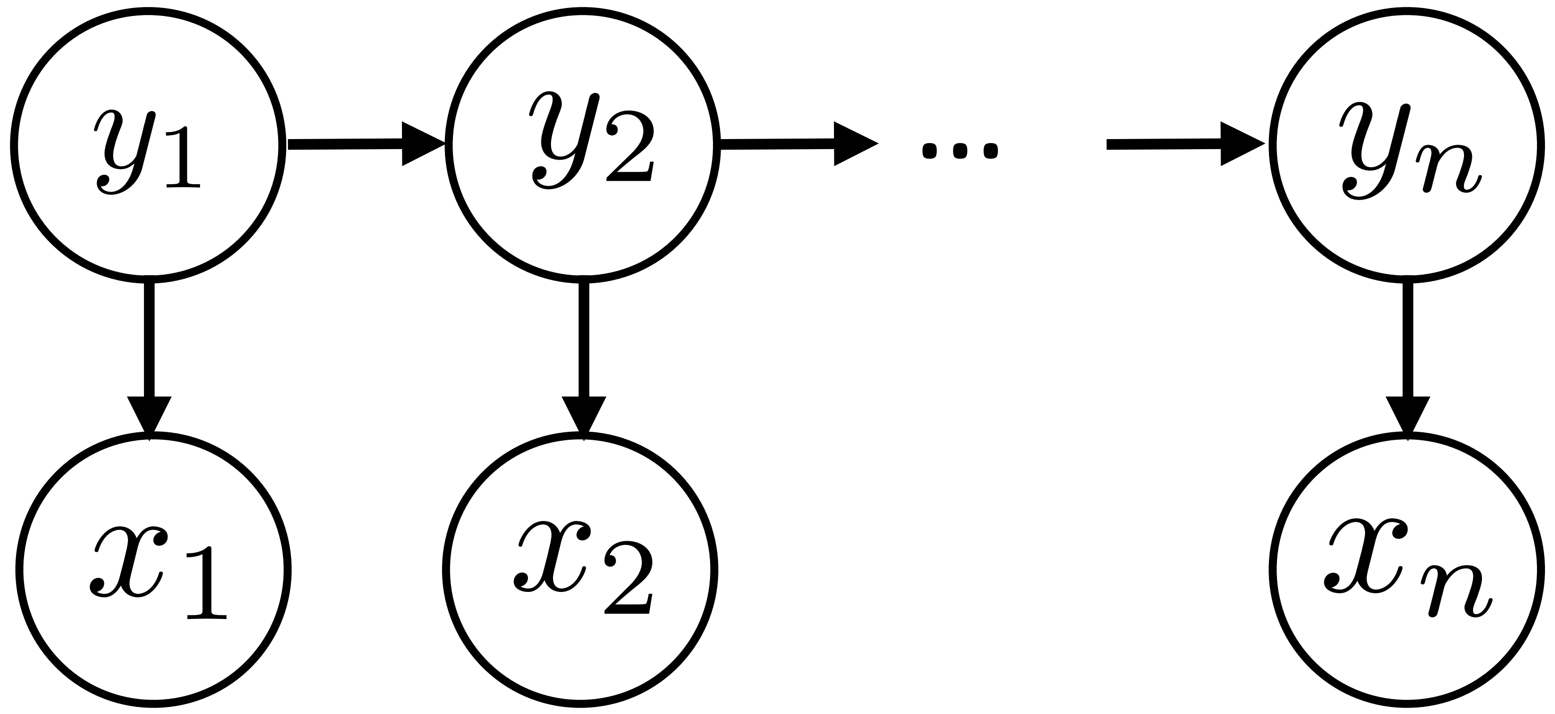
CS395T: Structured Models for NLP
Lecture 5: Sequence Models II

Greg Durrett

Some slides adapted from Dan Klein, UC Berkeley

Recall: 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: $\mathrm{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = \mathrm{argmax}_{\mathbf{y}} \frac{P(\mathbf{y}, \mathbf{x})}{\cancel{P(\mathbf{x})}}$
- Exponentially many possible ***y*** here!
- Viterbi: $\mathrm{score}_i(s) = \max_{y_{i-1}} P(s|y_{i-1}) P(x_i|s) \mathrm{score}_{i-1}(y_{i-1})$

This Lecture

- Generative vs. discriminative models
- CRFs for sequence modeling
- Named entity recognition (NER)
- Structured SVM
- (if time) Beam search

Named Entity Recognition

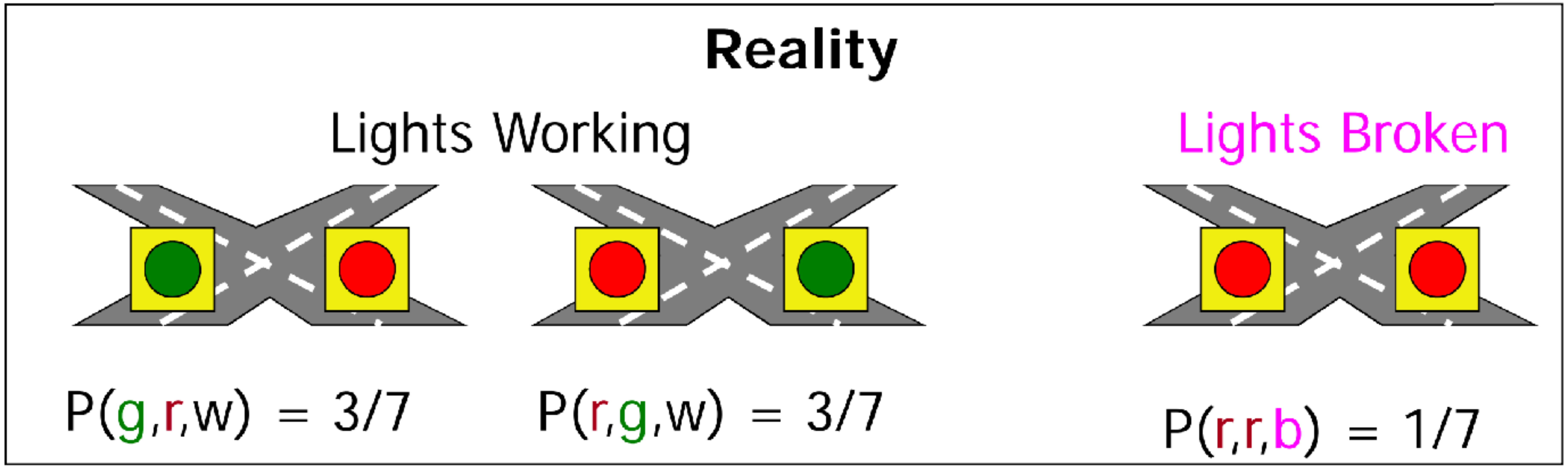
B-PER	I-PER	O	O	O	B-LOC	O	O	O	B-ORG	O	O
Barack	*Obama*	*will*	*travel*	*to*	*Hangzhou*	*today*	*for*	*the*	*G20*	*meeting*	*.*
PERSON					LOC				ORG		

- BIO tagset: begin, inside, outside

- POS tagging is a plausible generative model of language — NER with this vanilla tag set is not

- What's different about modeling P($\boldsymbol{y}$|$\boldsymbol{x}$) directly vs. P($\boldsymbol{x}$,$\boldsymbol{y}$) and computing the posterior later?

Generative vs. Discriminative Models



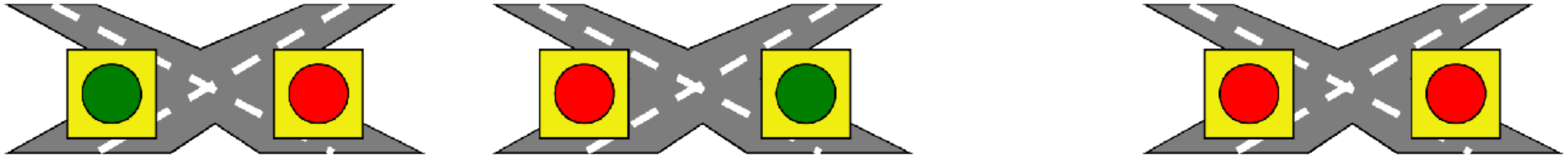
slide credit: Dan Klein

Generative vs. Discriminative Models

Reality

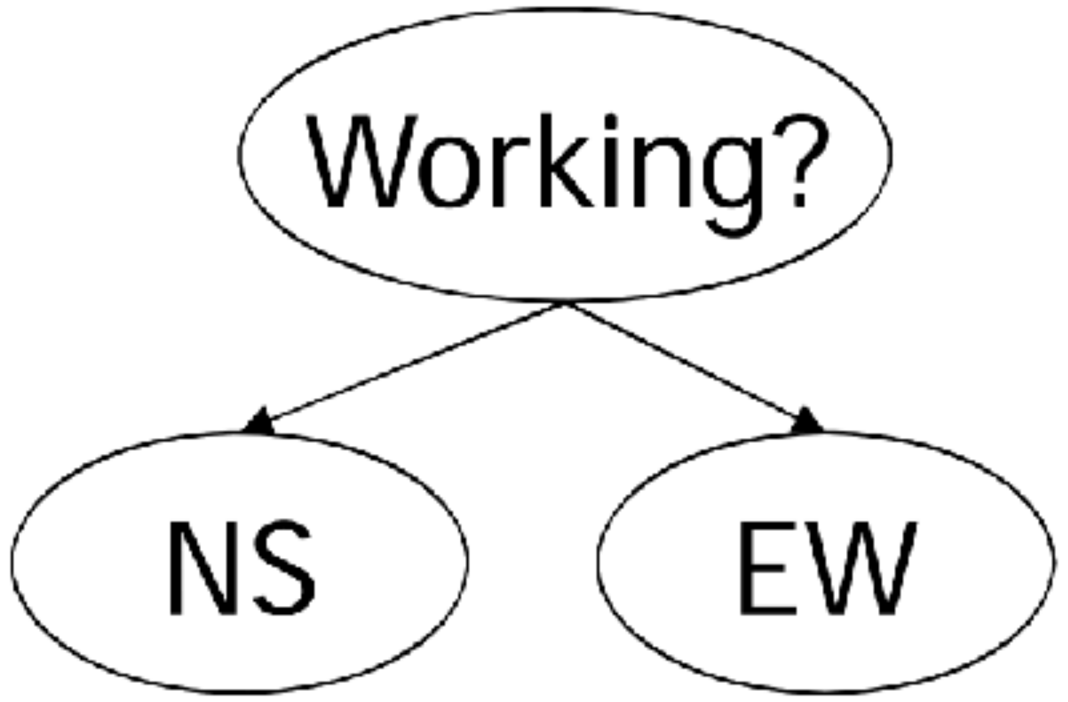
Lights Working

Lights Broken



$P(g,r,w) = 3/7$ $P(r,g,w) = 3/7$ $P(r,r,b) = 1/7$

NB Model



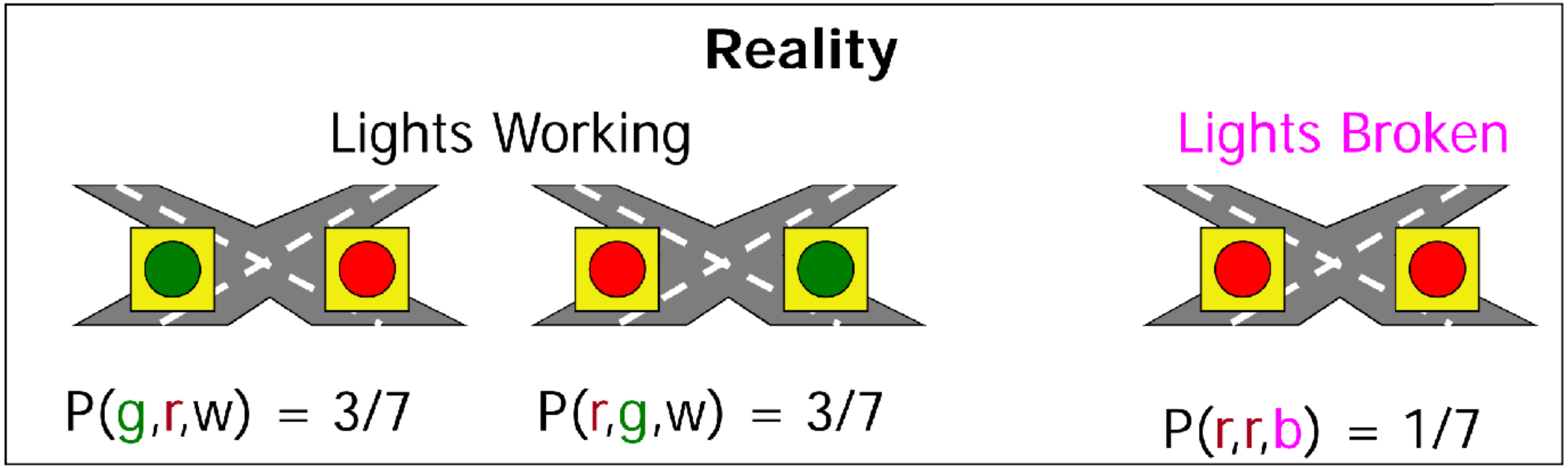
NB FACTORS:

- $P(w) = 6/7$
- $P(r|w) = 1/2$
- $P(g|w) = 1/2$

- $P(b) = 1/7$
- $P(r|b) = 1$
- $P(g|b) = 0$

slide credit: Dan Klein

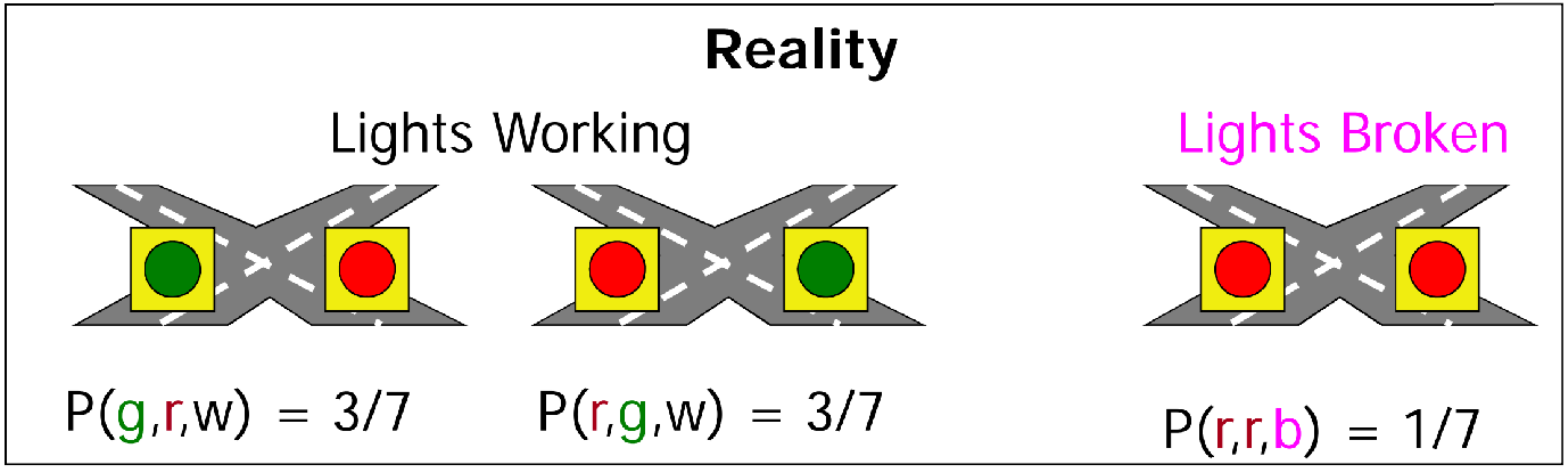
Generative vs. Discriminative Models



- What does the model say when both lights are red?
  - P(b,r,r) = (1/7)(1)(1) = 1/7 = 4/28
  - P(w,r,r) = (6/7)(1/2)(1/2) = 6/28 = 6/28
  - P(w|r,r) = 6/10!
- Lights are working — wrong!

slide credit: Dan Klein

Generative vs. Discriminative Models

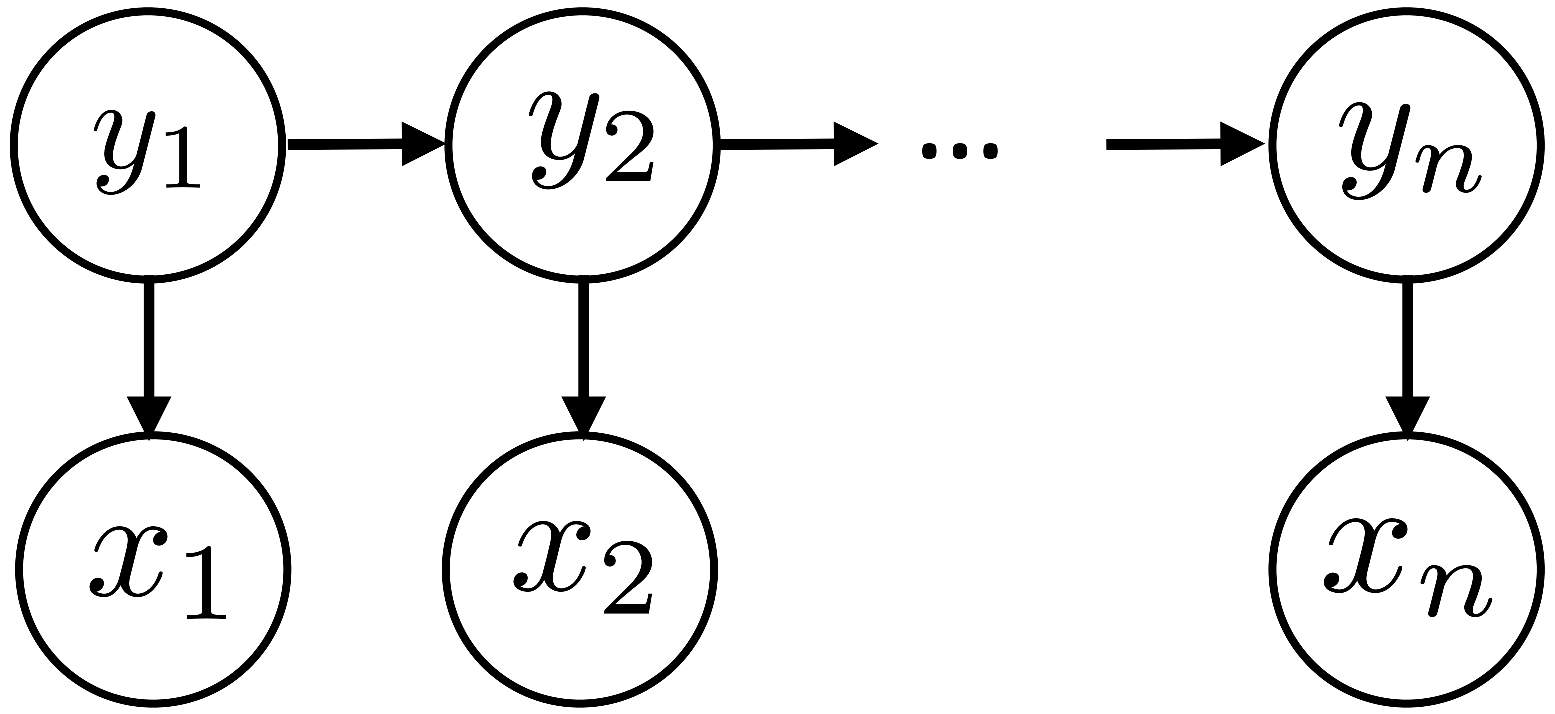


- What if P(*b*) were 1/2 instead of 1/7 (the NB estimate)?
  - P(b,r,r) = (1/2)(1)(1) = 1/2 = 4/8
  - P(w,r,r) = (1/2)(1/2)(1/2) = 1/8 = 1/8
  - P(w|r,r) = 1/5!
    - Lights are broken — correct! Data likelihood is lower but
- Data likelihood P(***x***,***y***) is lower but posterior P(***y***|***x***) is more accurate

slide credit: Dan Klein

Conditional Random Fields

- HMMs are expressible as Bayes nets (factor graphs)



- This reflects the following decomposition:

$$P(\mathbf{y}, \mathbf{x}) = P(y_1)P(x_1|y_1)P(y_2|y_1)P(x_2|y_2)\ldots$$

- Locally normalized model: each factor is a probability distribution that normalizes

Conditional Random Fields

- HMMs: $P(\mathbf{y}, \mathbf{x}) = P(y_1)P(x_1|y_1)P(y_2|y_1)P(x_2|y_2)\ldots$

- 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

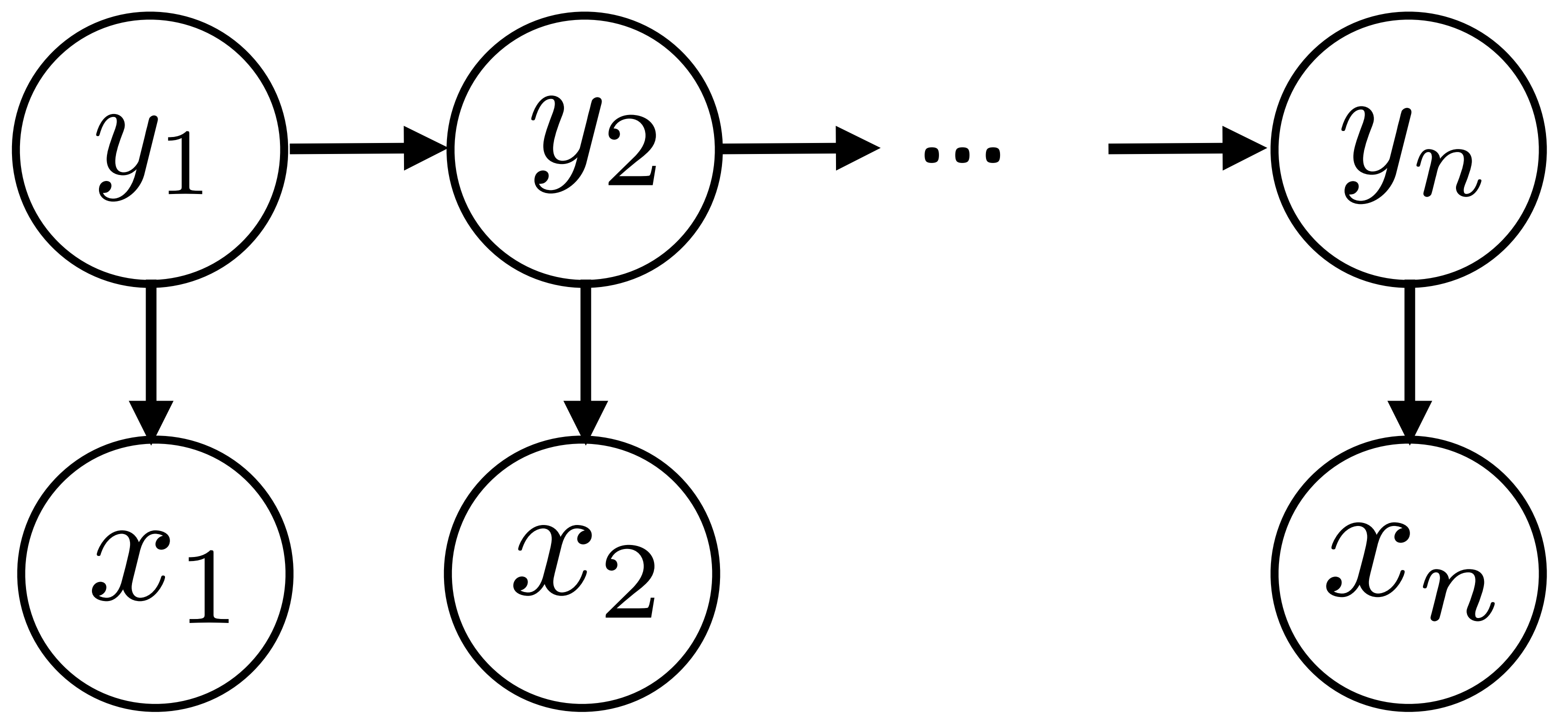
any real-valued scoring function of its arguments

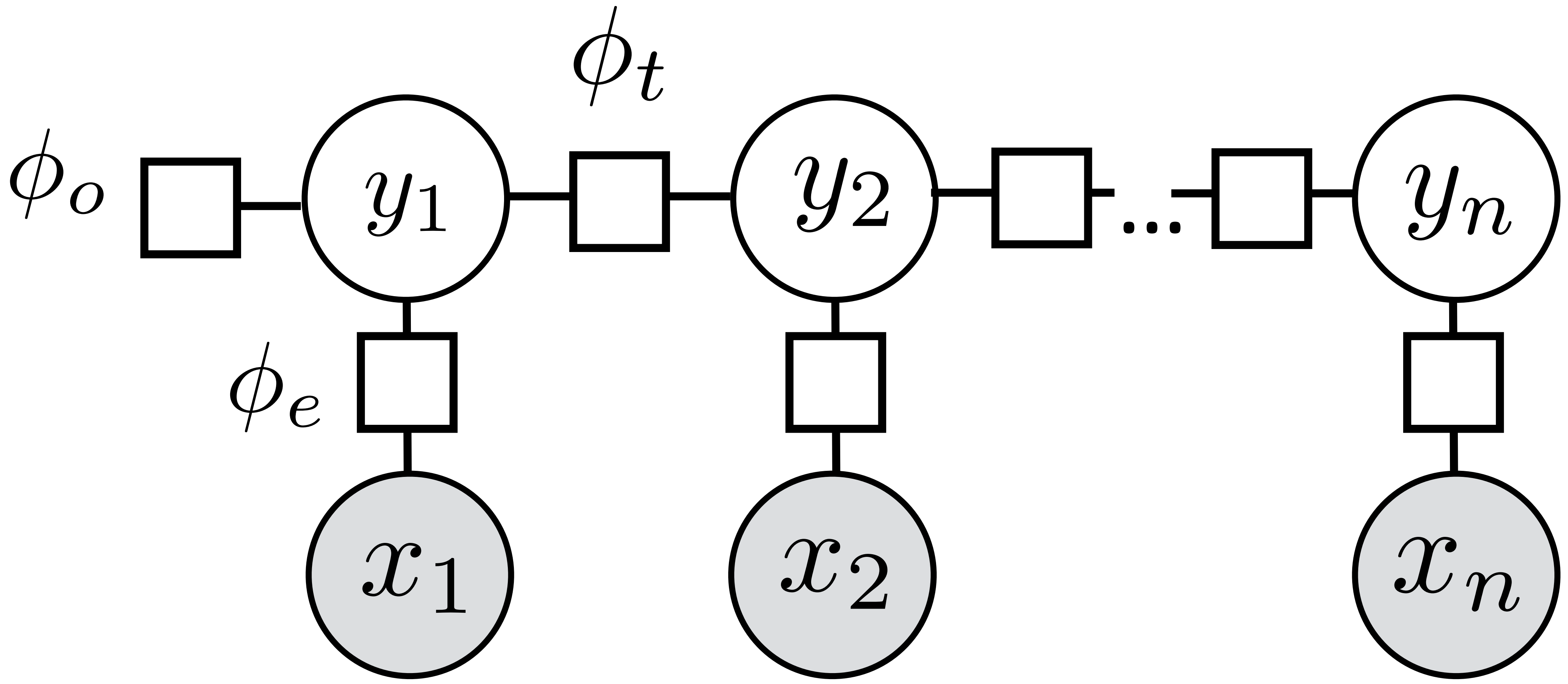
- Naive Bayes : logistic regression :: HMMs : CRFs
  local vs. global normalization <-> generative vs. discriminative

- How do we max over ***y***? Intractable in general — can we fix this?

Sequential CRFs

- HMMs: $P(\mathbf{y}, \mathbf{x}) = P(y_1)P(x_1|y_1)P(y_2|y_1)P(x_2|y_2)\ldots$



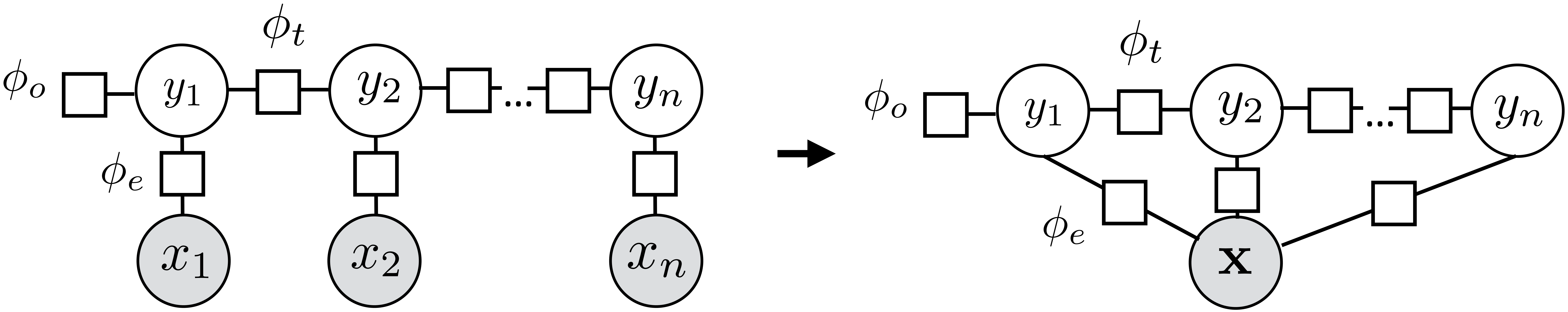


- CRFs:

$$P(\mathbf{y}|\mathbf{x}) \propto \prod_k \exp(\phi_k(\mathbf{x}, \mathbf{y}))$$

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

Sequential CRFs



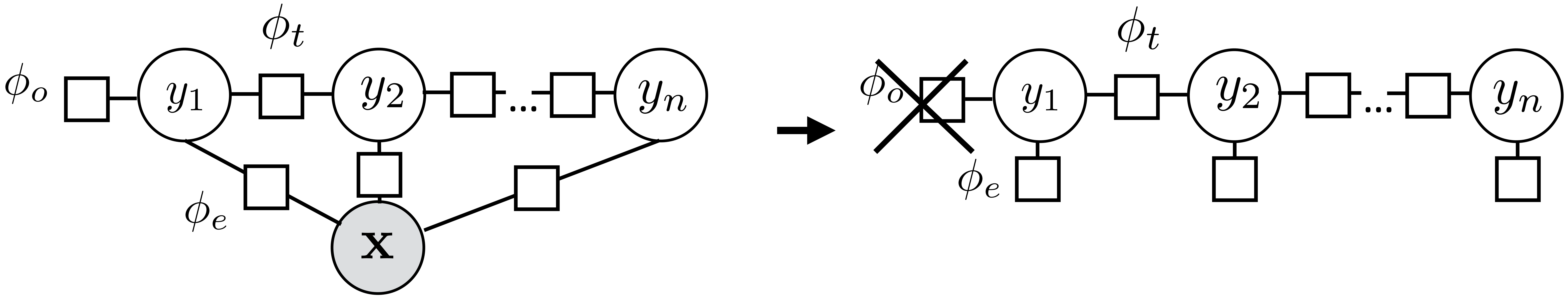
$$P(\mathbf{y}|\mathbf{x}) \propto \exp(\phi_o(y_1)) \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \cancel{\prod_{i=1}^{n} \exp(\phi_e(x_i, y_i))}$$

$$\prod_{i=1}^{n} \exp(\phi_e(y_i, i, \mathbf{x}))$$

token index — lets us look at current word

- We condition on **x**, so every variable can depend on all of **x**
- **x** can't depend arbitrarily on **y** in a generative model — would make inference hard

Sequential CRFs



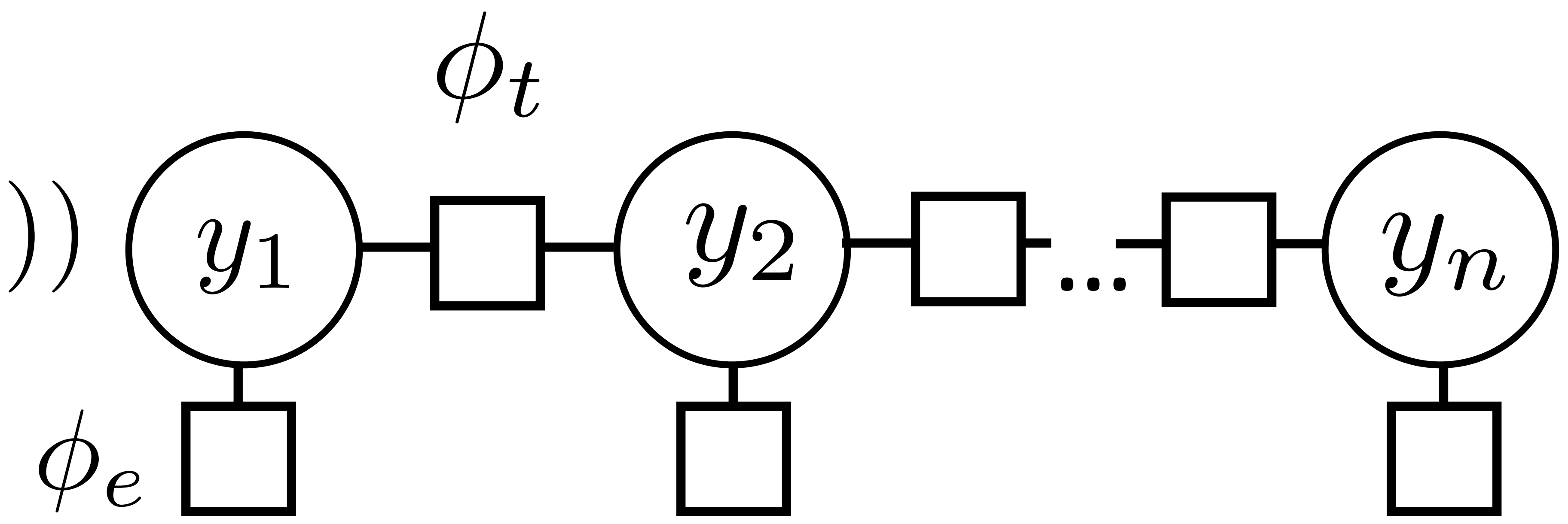
- ...in fact, we typically don't show **x** at all
- Don't include initial distribution, can bake into other factors

Sequential CRFs:

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

Computing (arg)maxes

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



- $\text{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x})$: can use Viterbi exactly as in HMM case

$$\max_{y_1,\ldots,y_n} e^{\phi_t(y_{n-1},y_n)} e^{\phi_e(y_n,n,\mathbf{x})} \ldots e^{\phi_e(y_2,2,\mathbf{x})} e^{\phi_t(y_1,y_2)} e^{\phi_e(y_1,1,\mathbf{x})}$$

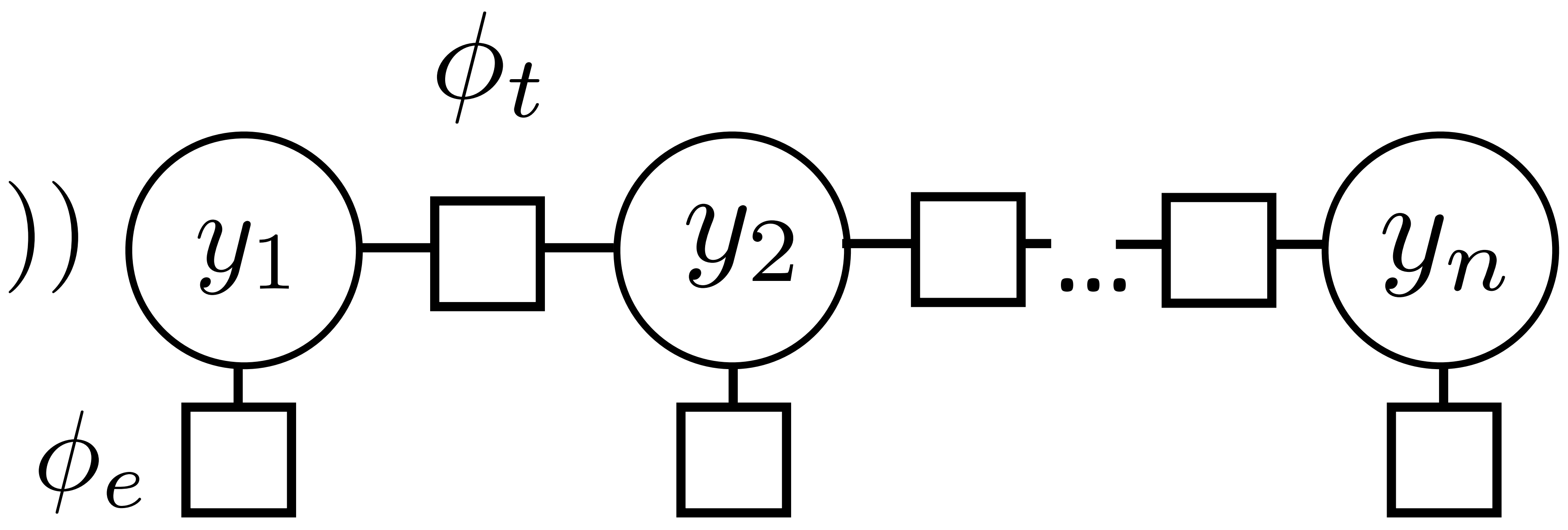
$$= \max_{y_2,\ldots,y_n} e^{\phi_t(y_{n-1},y_n)} e^{\phi_e(y_n,n,\mathbf{x})} \ldots e^{\phi_e(y_2,2,\mathbf{x})} \boxed{\max_{y_1}} e^{\phi_t(y_1,y_2)} \underbrace{e^{\phi_e(y_1,1,\mathbf{x})}}$$

$$= \max_{y_3,\ldots,y_n} e^{\phi_t(y_{n-1},y_n)} e^{\phi_e(y_n,n,\mathbf{x})} \ldots \max_{y_2} e^{\phi_t(y_2,y_3)} \underbrace{e^{\phi_e(y_2,2,\mathbf{x})} \max_{y_1} e^{\phi_t(y_1,y_2)} \text{score}_1(y_1)}$$

- $\exp(\phi_t(y_{i-1}, y_i))$ and $\exp(\phi_e(y_i, i, \mathbf{x}))$ play the role of the Ps now, same dynamic program

Computing Marginals

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



- Normalizing constant $Z = \sum_{\mathbf{y}}\prod_{i=2}^{n}\exp(\phi_t(y_{i-1}, y_i))\prod_{i=1}^{n}\exp(\phi_e(y_i, i, \mathbf{x}))$
- Analogous to P(**x**) for HMMs
- For both HMMs and CRFs:

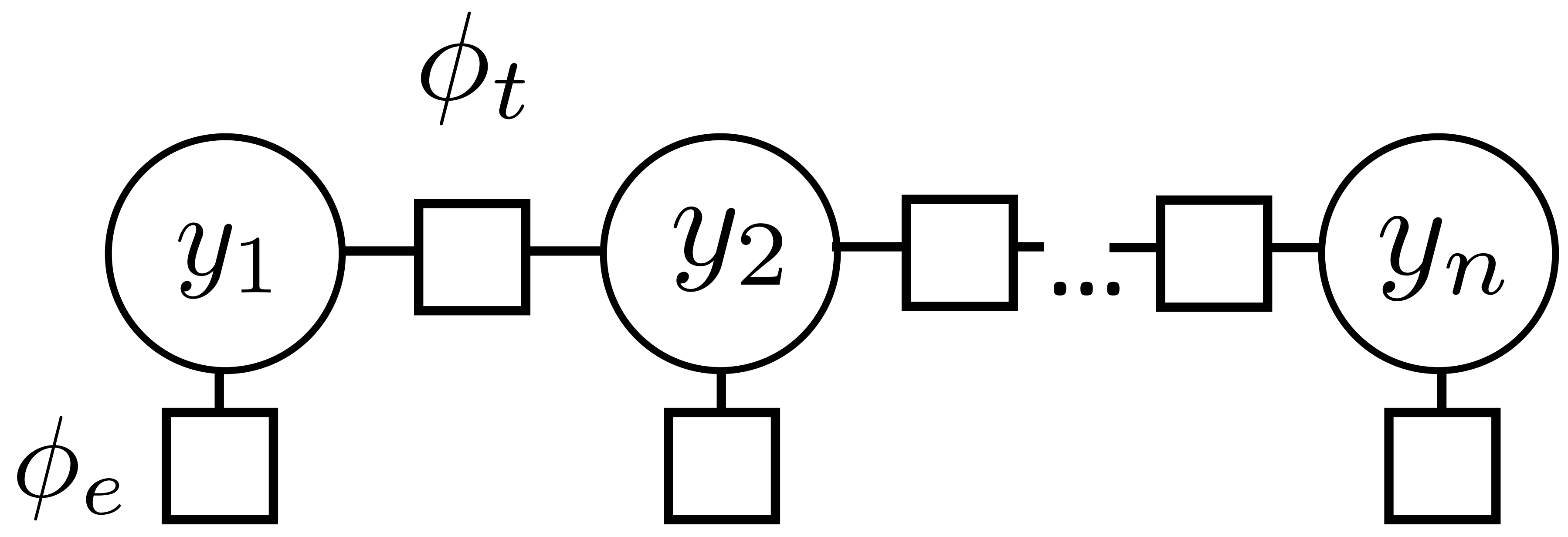
$$P(y_i = s|\mathbf{x}) = \frac{\text{forward}_i(s)\text{backward}_i(s)}{\sum_{s'}\text{forward}_i(s')\text{backward}_i(s')}$$

$P(y_i = s, \mathbf{x})$ for HMMs; sums out other *y*s

Z for CRFs, P(**x**) for HMMs

Inference in General CRFs

- Can do inference in any tree-structured CRF

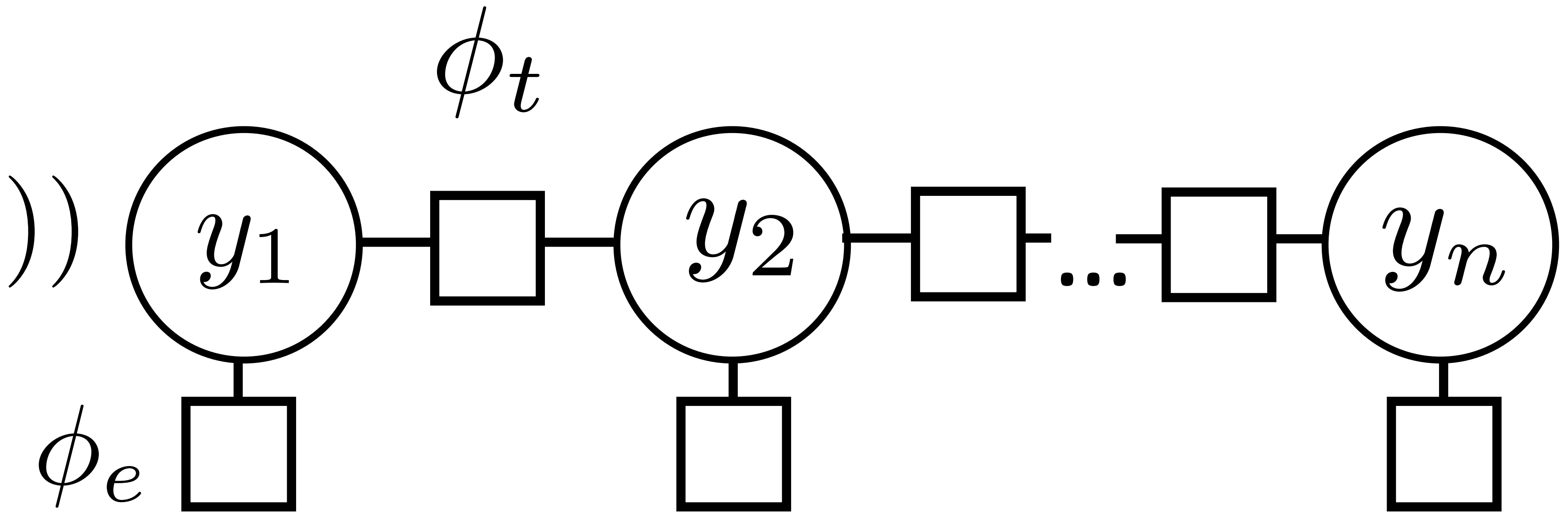


- Sum-product algorithm: generalization of forward-backward to arbitrary tree-structured graphs

- We'll come back to this in a few lectures when we deal with other kinds of graphs

Feature Functions

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



- Phi can have sophisticated features! Generally look like linear models

$$\phi_e(y_i, i, \mathbf{x}) = w^\top f_e(y_i, i, \mathbf{x}) \qquad \phi_t(y_{i-1}, y_i) = w^\top f_t(y_{i-1}, y_i)$$

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

- Log-linear model — structurally like logistic regression!

Training CRFs

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

- Assume ϕ_t and ϕ_e are both linear feature functions $w^\top f(\text{args})$

$$\mathcal{L}(\mathbf{y}^*, \mathbf{x}) = \log P(\mathbf{y}^*|\mathbf{x}) = \sum_{i=2}^{n} w^\top f_t(y_{i-1}^*, y_i^*) + \sum_{i=1}^{n} w^\top f_e(x_i, y_i^*) - \log Z$$

- Gradient is gold features minus expected features under model, like in LR

$$\frac{\partial}{\partial w_j}\mathcal{L}(\mathbf{y}^*, \mathbf{x}) = \sum_{i=2}^{n} f_{t,j}(y_{i-1}^*, y_i^*) + \sum_{i=1}^{n} f_{e,j}(x_i, y_i^*)$$
$$-\mathbb{E}_{\mathbf{y}}\left[\sum_{i=2}^{n} f_{t,j}(y_{i-1}, y_i) + \sum_{i=1}^{n} f_{e,j}(x_i, y_i)\right]$$

Training CRFs

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

- How to compute expectations?
- Forward-backward helps you compute $P(y_i = s|\mathbf{x})$
- Take weighted sum over all features at all tags and positions
- Transition features: need to compute $P(y_i = s_1, y_{i+1} = s_2|\mathbf{x})$ using forward-backward as well
- …but you can build a pretty good system without transition features

Implementation Tips

- Often many features but only a few are active on a single sentence even across many different labels
- Maintain the gradient as a sparse vector for efficiency
  - `Counter` in `utils.py` is a way to do this

Basic Features for NER

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

B-LOC O

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

Transitions: $f_t(y_{i-1}, y_i) = \mathrm{Ind}[y_{i-1} \ \& \ y_i]$

Emissions: $f_e(y_6, 6, \mathbf{x}) =$ Ind[B-LOC & Current word = *Hangzhou*]

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

Features for NER

$$\phi_e(y_i, i, \mathbf{x})$$

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...

Features for NER

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

Leicestershire

Boston

Apple released a new version…

According to the New York Times…

Nonlocal Features

The news agency Tanjug reported on the outcome of the meeting.

ORG?

PER?

The delegation met the president at the airport, Tanjug said.

- Various ways to capture this information — we'll talk about this in a few lectures

Finkel and Manning (2008), Ratinov and Roth (2009)

Semi-Markov Models

$$\underbrace{\textit{Barack Obama}}_{\text{PER}}\ \underbrace{\textit{will travel to}}_{\text{O}}\ \underbrace{\textit{Hangzhou}}_{\text{LOC}}\ \underbrace{\textit{today for the}}_{\text{O}}\ \underbrace{\textit{G20}}_{\text{ORG}}\ \underbrace{\textit{meeting .}}_{\text{O}}$$

- Chunk-level prediction rather than token-level BIO
- ***y*** is a set of touching spans of the sentence
- Viterbi looks like looping over all spans that could lead to a given point
- Pros: features can look at whole span at once
- Cons: there's an extra factor of *n* during inference

Sarawagi and Cohen (2004)

Evaluating 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		

- Prediction of all Os still gets 66% accuracy on this example!
- What we really want to know: how many named entity *chunk* predictions did we get right?
  - Precision: of the ones we predicted, how many are right?
  - Recall: of the gold named entities, how many did we find?
  - F-measure: harmonic mean of these two
  - Partial credit? Typically no but more complex metrics exist

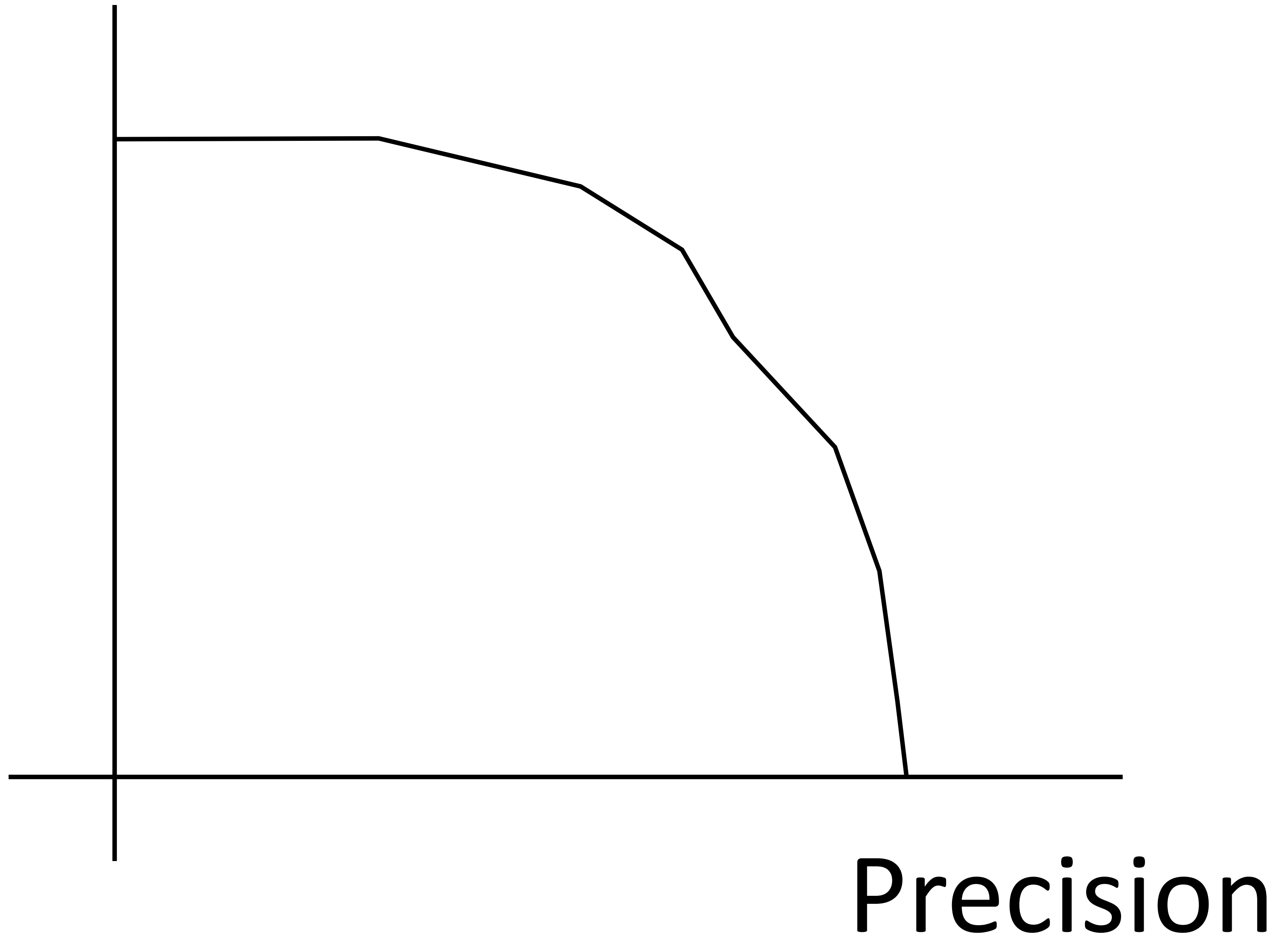
Evaluating NER

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

- More correct: ROC curve
- Measure the area under the curve as a way of evaluating the system holistically



How well do NER systems do?

	System	Resources Used	F_1
+	LBJ-NER	Wikipedia, Nonlocal Features, Word-class Model	90.80
-	(Suzuki and Isozaki, 2008)	Semi-supervised on 1G-word unlabeled data	89.92
-	(Ando and Zhang, 2005)	Semi-supervised on 27M-word unlabeled data	89.31
-	(Kazama and Torisawa, 2007a)	Wikipedia	88.02
-	(Krishnan and Manning, 2006)	Non-local Features	87.24
-	(Kazama and Torisawa, 2007b)	Non-local Features	87.17
+	(Finkel et al., 2005)	Non-local Features	86.86

Ratinov and Roth (2009)

Model	$\mathbf{F_1}$
Collobert et al. (2011)*	89.59
Lin and Wu (2009)	83.78
Lin and Wu (2009)*	90.90
Huang et al. (2015)*	90.10
Passos et al. (2014)	90.05
Passos et al. (2014)*	90.90
Luo et al. (2015)* + gaz	89.9
Luo et al. (2015)* + gaz + linking	**91.2**
Chiu and Nichols (2015)	90.69
Chiu and Nichols (2015)*	90.77
LSTM-CRF (no char)	90.20
LSTM-CRF	**90.94**
S-LSTM (no char)	87.96
S-LSTM	90.33

Lample et al. (2016)

Structured SVM

- CRF: $\log P(\mathbf{y}|\mathbf{x}) \propto \sum_{i=2}^{n} w^\top f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} w^\top f_e(x_i, y_i)$
- We can formulate an SVM using the same features

$$w^\top f(\mathbf{x}, \mathbf{y}) = \sum_{i=2}^{n} w^\top f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} w^\top f_e(x_i, y_i)$$

Minimize $\lambda \|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

s.t. $\forall j \quad \xi_j \geq 0$

$\forall j \forall \mathbf{y} \in \mathcal{Y} \quad w^\top f(\mathbf{x}_j, \mathbf{y}_j^*) \geq w^\top f(\mathbf{x}_j, \mathbf{y}) + \ell(\mathbf{y}, \mathbf{y}_j^*) - \xi_j$

Structured SVM

$$w^\top f(\mathbf{x}, \mathbf{y}) = \sum_{i=2}^{n} w^\top f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} w^\top f_e(x_i, y_i)$$

Minimize $\lambda \|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

s.t. $\forall j \quad \xi_j \geq 0$

$\forall j \forall \mathbf{y} \in \mathcal{Y} \quad w^\top f(\mathbf{x}_j, \mathbf{y}_j^*) \geq w^\top f(\mathbf{x}_j, \mathbf{y}) + \ell(\mathbf{y}, \mathbf{y}_j^*) - \xi_j$

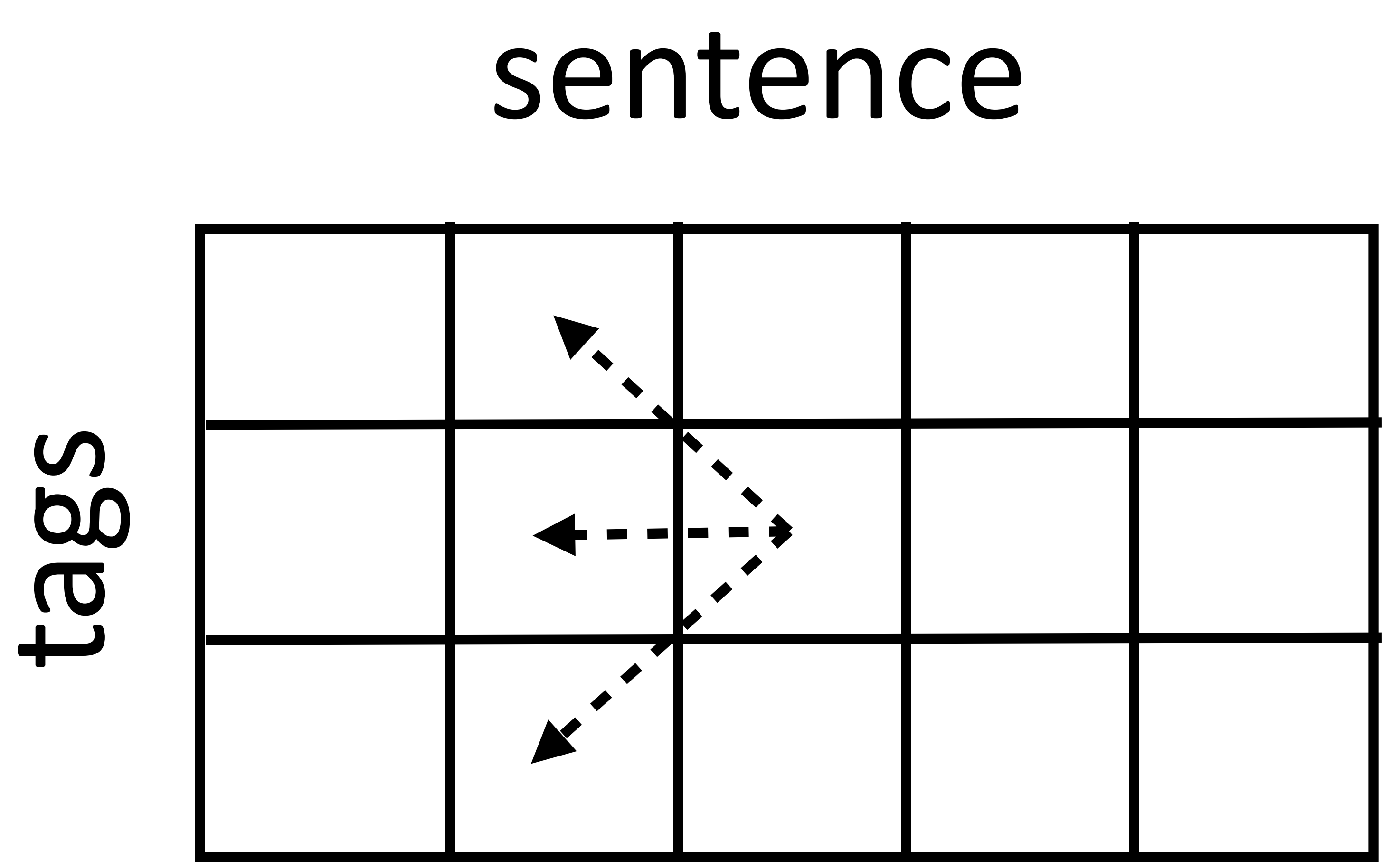
- Exponentially large state space! Use Viterbi for loss-augmented decode
- Same as normal Viterbi but boost wrong labels' scores by 1 (if using Hamming loss)
- Only need Viterbi, not forward-backward…hmm…

Viterbi Time Complexity

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

- n word sentence, s tags to consider — what is the time complexity?



- $O(ns^2)$ — s is ~40 for POS, n is ~20

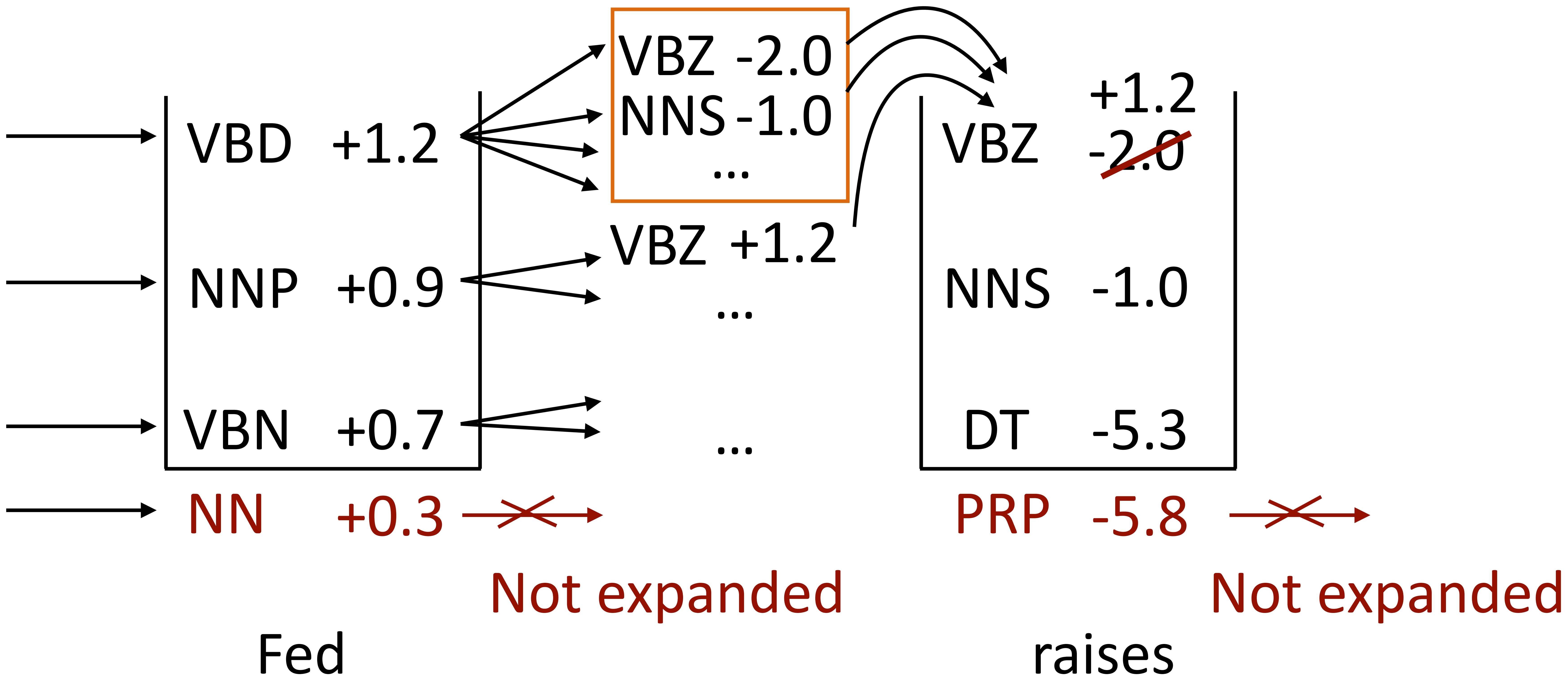
Viterbi Time Complexity

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

- Many tags are totally implausible
- Can any of these be:
  - Determiners?
  - Prepositions?
  - Adjectives?
- Features quickly eliminate many outcomes from consideration — don't need to consider these going forward

Beam Search

- Maintain a beam of *k* plausible states at the current timestep
- Expand all states, only keep *k* top hypotheses at new state



- O(nks) time complexity with beam size of *k*

How good is beam search?

- Big enough beam size: always exact! Usually works well even with smaller beams
- What's the case when k=1?
- How about when there's no transition model?
  - Depends on the strength of nonlocal interactions — we'll come back to this later!

Implementation Tips for CRFs

- Caching is your friend! Cache feature vectors especially

- Try to reduce redundant computation, e.g. if you compute both the gradient and the objective value, don’t rerun the dynamic program

- Exploit sparsity in feature vectors where possible. The weight vector needs to be stored explicitly, but all features and gradients are typically faster to handle sparsely

- Think about your data structures: if things are too slow

Debugging Tips for CRFs

- Hard to know whether inference, learning, or the model is broken!
- Compute the objective — is optimization working?
  - **Inference**: check gradient computation (most likely place for bug)
    - Are expectations being computed correctly? Do probabilities normalize / expectations look reasonable?
  - **Learning**: are you applying the gradient correctly?
- If objective is going down but model performance is bad:
  - **Inference**: check performance if you decode the training set
  - **Model**: if dev set performance is bad: work on features more!

Next Time

- Unsupervised sequence modeling
- Writing tips as you prepare your report