

Using MALLET for Conditional Random Fields

Matthew Michelson & Craig A. Knoblock
CSCI 548 – Lecture 3

The road to CRFs...

- In the beginning...Generative Models
(Probability of X and $Y \rightarrow P(X, Y)$?)
- *Markov assumption*: prob. in current state only depends on previous and current state
- Standard model: Hidden Markov Model (HMM)

Markov Process

Let's say we're independent of time, then we can define

- $a_{ij} = P(q_t=S_j|q_{t-1}=S_i)$ as a STATE TRANSITION from S_i to S_j

- $a_{ij} \geq 0$

- $$\sum_{j=1}^N a_{ij} = 1$$

This conserves all of the “Mass” of probability;
i.e. all outgoing probabilities sum to 1

Markov Process

- Two more terms to define:
 - $\pi_i = P(q_1=S_i)$ = probability that we start in state S_i
 - $b_j(k) = P(k|q_t = S_j)$ = probability of observation symbol k in State j .

So, lets say symbols = {A,B}, then we could have something like $b_1(A) = P(A|S_1)$

i.e. what is the probability that we output A in state 1?

Hidden Markov Model

- A Hidden Markov Model (HMM)
 - Set of states, Set of $a_{i,j}$, Set of π_i , Set of $b_j(k)$
- learn a set of sequence of observations, and their transition and emission probabilities. → Training
- When testing, input comes in, and fits model's internal observations with some probability, output best state transition sequence to produce the input observation → Decoding
- you can observe the sequence of emissions, but you do not know what state the model is in → “Hidden”
 - If 2 states output “yes”, all I see is “yes,” I have no idea what state or set of states produced this!

HMM - Example

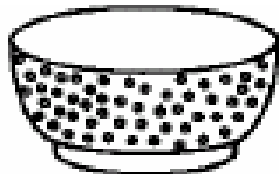
Urn and Ball Model – Each urn has large num. of M distinct colored balls. Randomly pick an urn, and pick out a colored ball, repeat.

S = set of states = set of urns

Transition Probs = choice of next urn

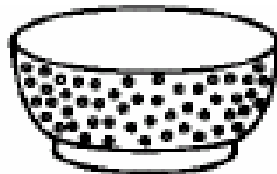
$b_i(\text{color})$ = prob. of getting that colored ball in urn _{i}

Urn and Ball Problem



URN 1

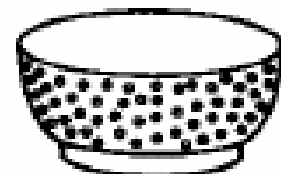
$P(\text{RED}) = b_1(1)$
 $P(\text{BLUE}) = b_1(2)$
 $P(\text{GREEN}) = b_1(3)$
 $P(\text{YELLOW}) = b_1(4)$
⋮
 $P(\text{ORANGE}) = b_1(M)$



URN 2

$P(\text{RED}) = b_2(1)$
 $P(\text{BLUE}) = b_2(2)$
 $P(\text{GREEN}) = b_2(3)$
 $P(\text{YELLOW}) = b_2(4)$
⋮
 $P(\text{ORANGE}) = b_2(M)$

...



URN N

$P(\text{RED}) = b_N(1)$
 $P(\text{BLUE}) = b_N(2)$
 $P(\text{GREEN}) = b_N(3)$
 $P(\text{YELLOW}) = b_N(4)$
⋮
 $P(\text{ORANGE}) = b_N(M)$

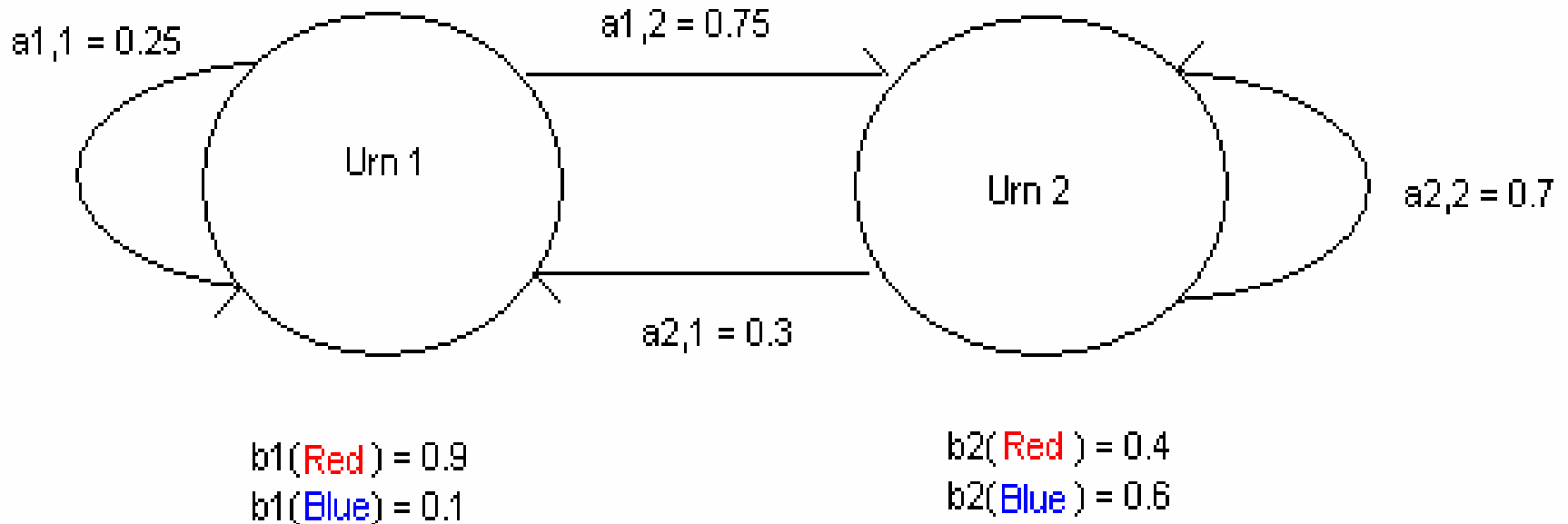
$O = \{\text{GREEN, GREEN, BLUE, RED, YELLOW, RED, \dots, BLUE}\}$

Urn and Ball Example

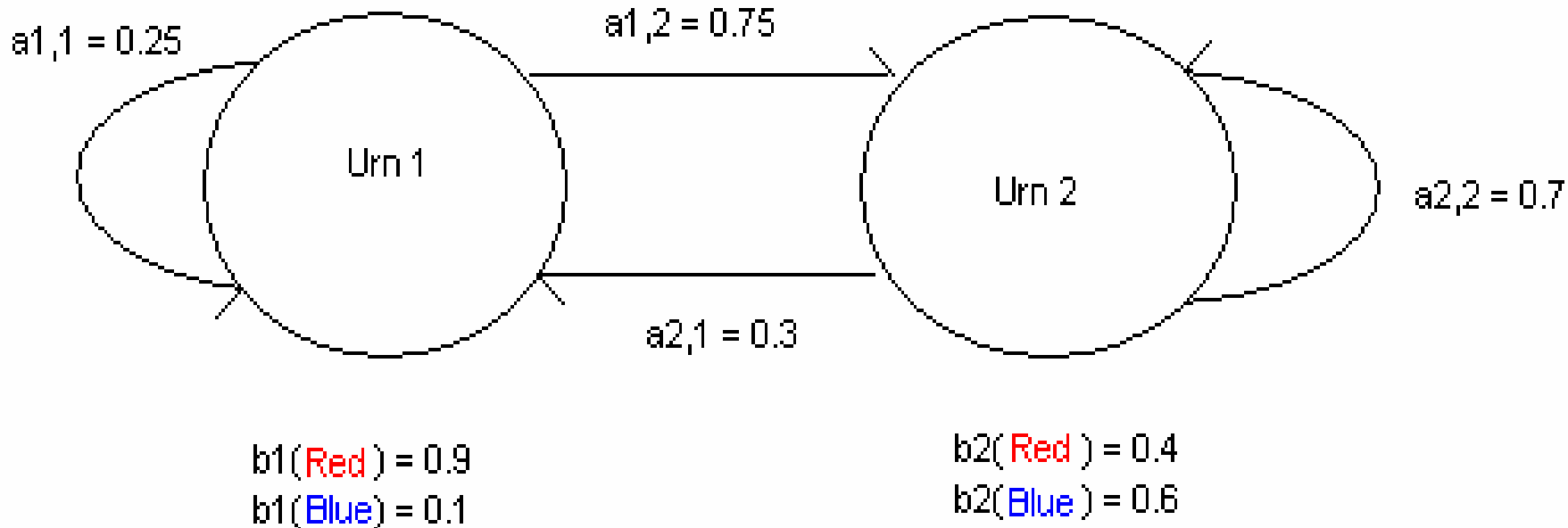
Let's say we have the following:

- 2 urns
- 2 colors (Red,Blue)
- $a_{1,1} = 0.25$ $a_{1,2} = 0.75$
- $a_{2,1} = 0.3$ $a_{2,2} = 0.7$
- $b_1(\text{Red}) = 0.9$, $b_1(\text{Blue}) = 0.1$
- $b_2(\text{Red}) = 0.4$, $b_2(\text{Blue}) = 0.6$

Urn and Ball Example



Let's say it's perfectly random to start with either urn, i.e. $\pi_1 = \pi_2 = 0.5$
What is the most probable state sequence that produces {Red,Red,Blue}?

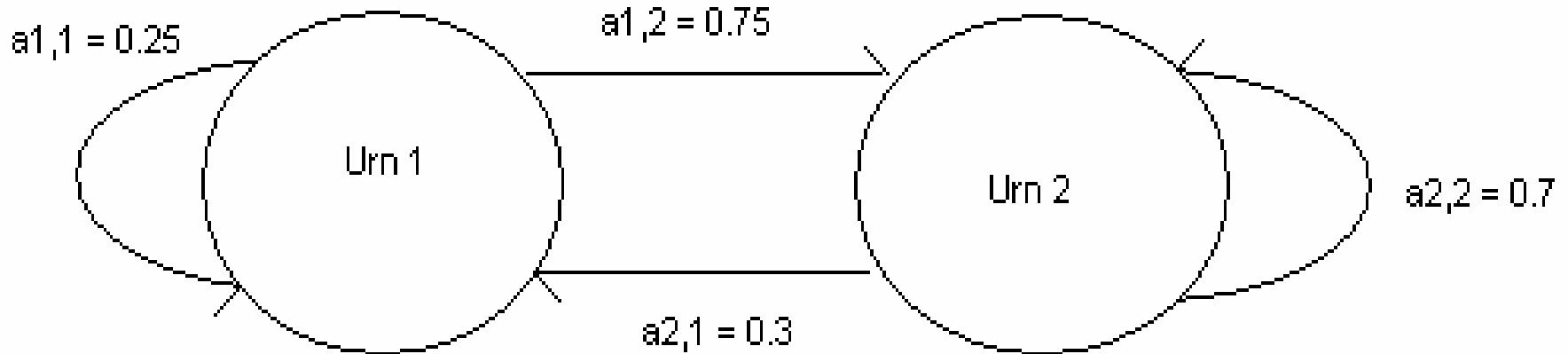


We will use the Viterbi algorithm to do this, recursively:

Define $\zeta(i) = \max P[q_1, q_2, \dots, q_t = i, O_1, O_2, \dots, O_n | \text{HMM model}]$

(Remember, q_t = current state, O are observations)

So, $\zeta_{t+1}(i) = [\max \zeta_t(i) * a_{i,j}] * b_j(O_{t+1})$



$$b_1(\text{Red}) = 0.9$$

$$b_1(\text{Blue}) = 0.1$$

$$b_2(\text{Red}) = 0.4$$

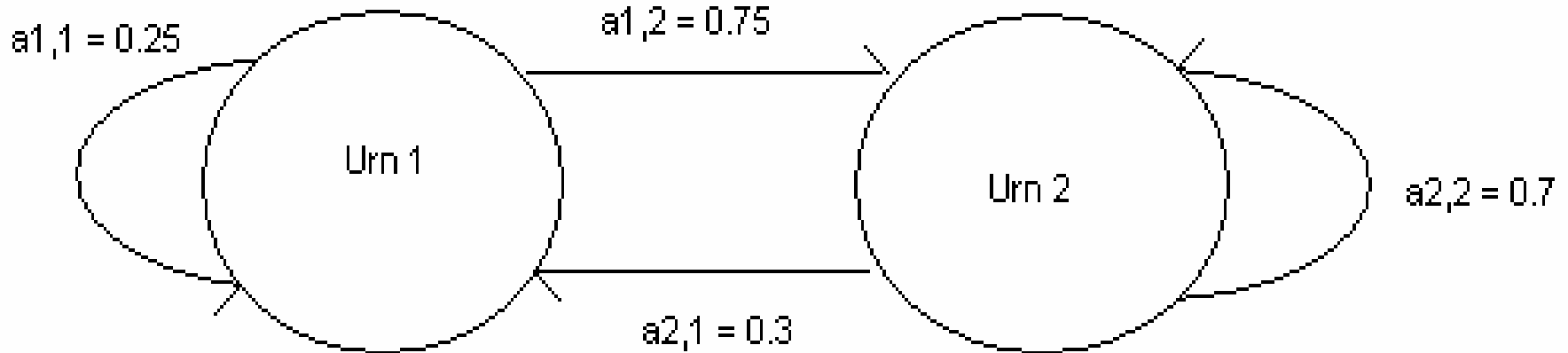
$$b_2(\text{Blue}) = 0.6$$

We need a first set of initialized values:

$$\zeta_1(i) = \pi_i * b_i(O_1 = \text{Red}) \quad i = \{1, 2\}$$

$$\zeta_1(1) = \pi_1 * b_1(O_1 = \text{Red}) = 0.5 * 0.9 = 0.45$$

$$\zeta_1(2) = \pi_2 * b_2(O_1 = \text{Red}) = 0.5 * 0.4 = 0.2$$



$$b_1(\text{Red}) = 0.9$$

$$b_1(\text{Blue}) = 0.1$$

$$b_2(\text{Red}) = 0.4$$

$$b_2(\text{Blue}) = 0.6$$

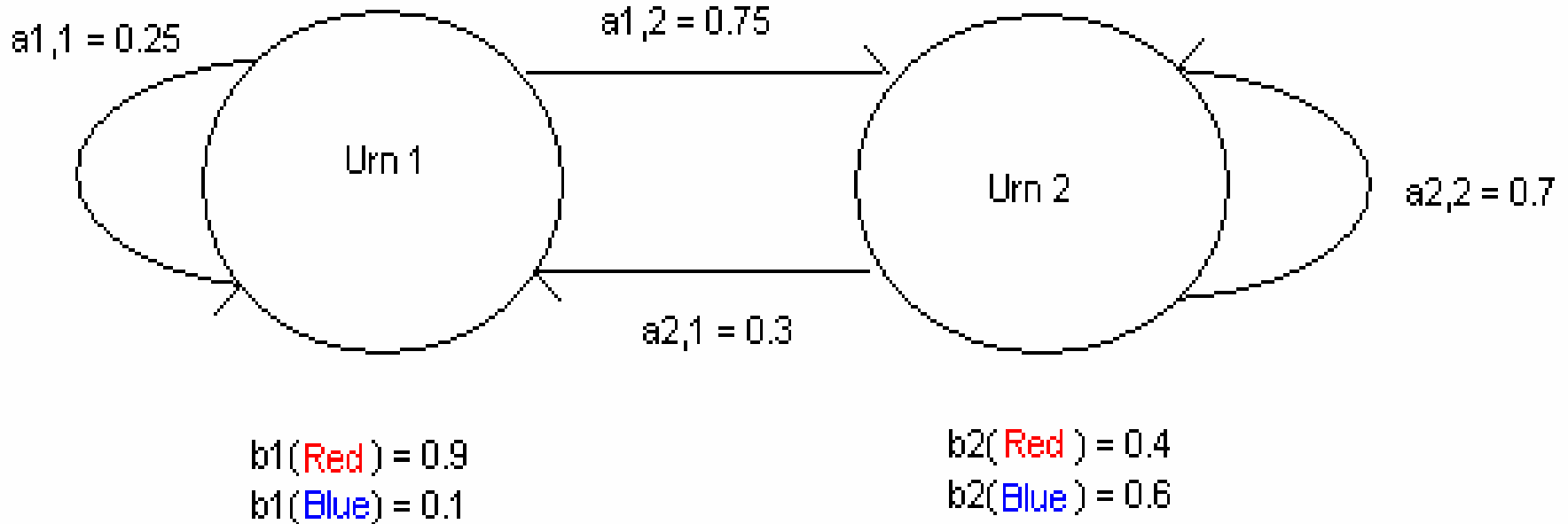
Now, recurse:

$$\zeta_2(1) = \max (\{ \zeta_1(1) * a_{1,1} , \zeta_1(2) * a_{2,1} \}) * b_1(O_2 = \text{Red})$$

$$= \max(\{ 0.45 * 0.25, 0.2 * 0.3 \}) * 0.9 = 0.10125$$

$$\zeta_2(2) = \max(\{ \zeta_1(1) * a_{1,2} , \zeta_1(2) * a_{2,2} \}) * b_2(O_2 = \text{Red})$$

$$= \max(\{ 0.45 * 0.75, 0.2 * 0.7 \}) * 0.4 = 0.135$$



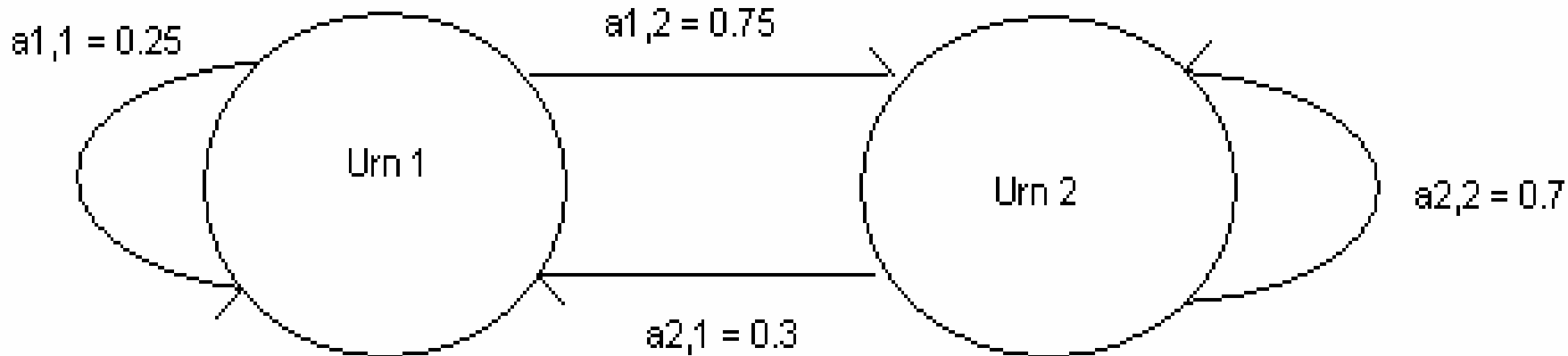
Now, recurse:

$$\zeta_3(1) = \max (\{ \zeta_2(1) * a_{1,1} , \zeta_2(2) * a_{2,1} \}) * b_1(O_3 = \text{Blue})$$

$$= \max(\{ 0.10125 * 0.25, 0.135 * 0.3 \}) * 0.1 = 0.00405$$

$$\zeta_3(2) = \max(\{ \zeta_2(1) * a_{1,2}, \zeta_2(2) * a_{2,2} \}) * b_2(O_3 = \text{Blue})$$

$$= \max(\{ 0.10125 * 0.75, 0.135 * 0.7 \}) * 0.6 = 0.0567$$



$$b1(\text{Red}) = 0.9$$

$$b1(\text{Blue}) = 0.1$$

$$b2(\text{Red}) = 0.4$$

$$b2(\text{Blue}) = 0.6$$

So, we see that at each step, maximally we have:

$$\zeta_3(2) = 0.0567, \zeta_2(2) = 0.135, \zeta_1(1) = 0.45$$

So, working backwards, know the state transitions went

Urn 2 \leftarrow Urn 2 \leftarrow Urn 1.

So, if we are given observation (Red,Red,Blue) we say that the most probable State transition set is

{Start in Urn 1/red, Go to Urn 2/red, Stay Urn 2/blue}



HMM Issues

1 – Independence Assumption

Current observation only depends on what state you are in right now.

Or, to say it differently, the current output has no dependence on previous outputs. For our urn example, we couldn't model the fact that if urn1 outputs a red ball, then urn2 should decrease its probability of doing so.

HMM Issues

2 – Multiple Features Issue

HMM generates a set of probabilities given an observation.

But what if you want to capture many features from an observation, and these features interact?

E.g. observation is “Doug.” This is a *noun*, *capital*, and *masculine*. Now, what if transition is into state = “MAN”?

Now, we know that state MAN probably depends on the observations noun and capital. But, what if we have state CITY too? Doesn't that depend on noun and cap?

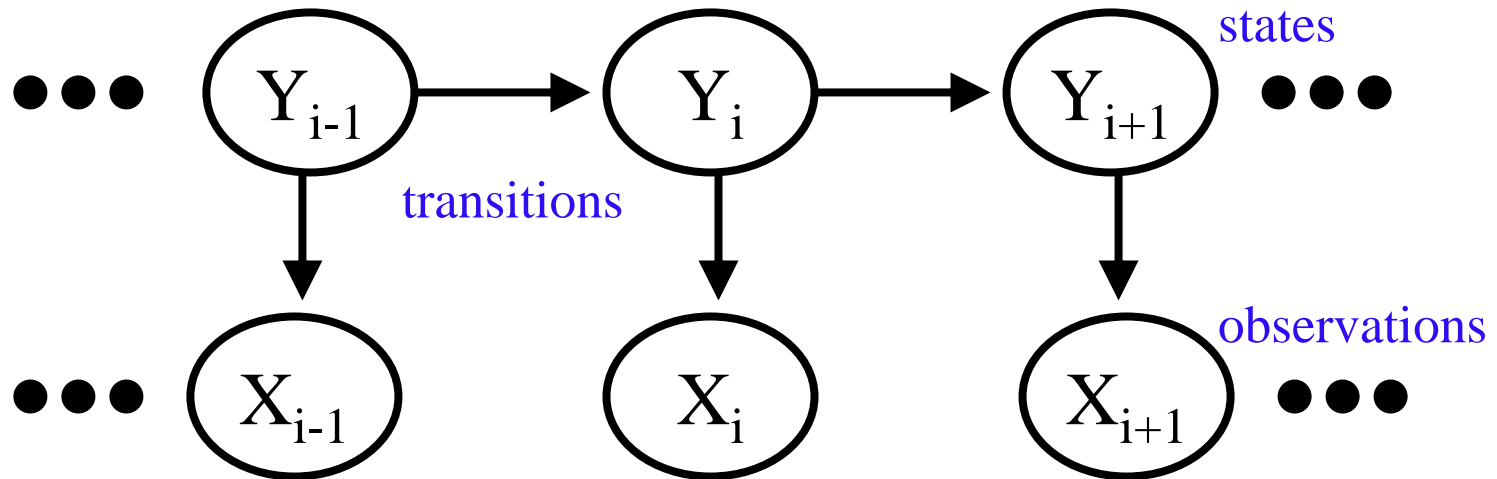
To transfer into MAN might require a masculine name. This observation strongly depends on the word having feature *masculine*.



HMM Issues

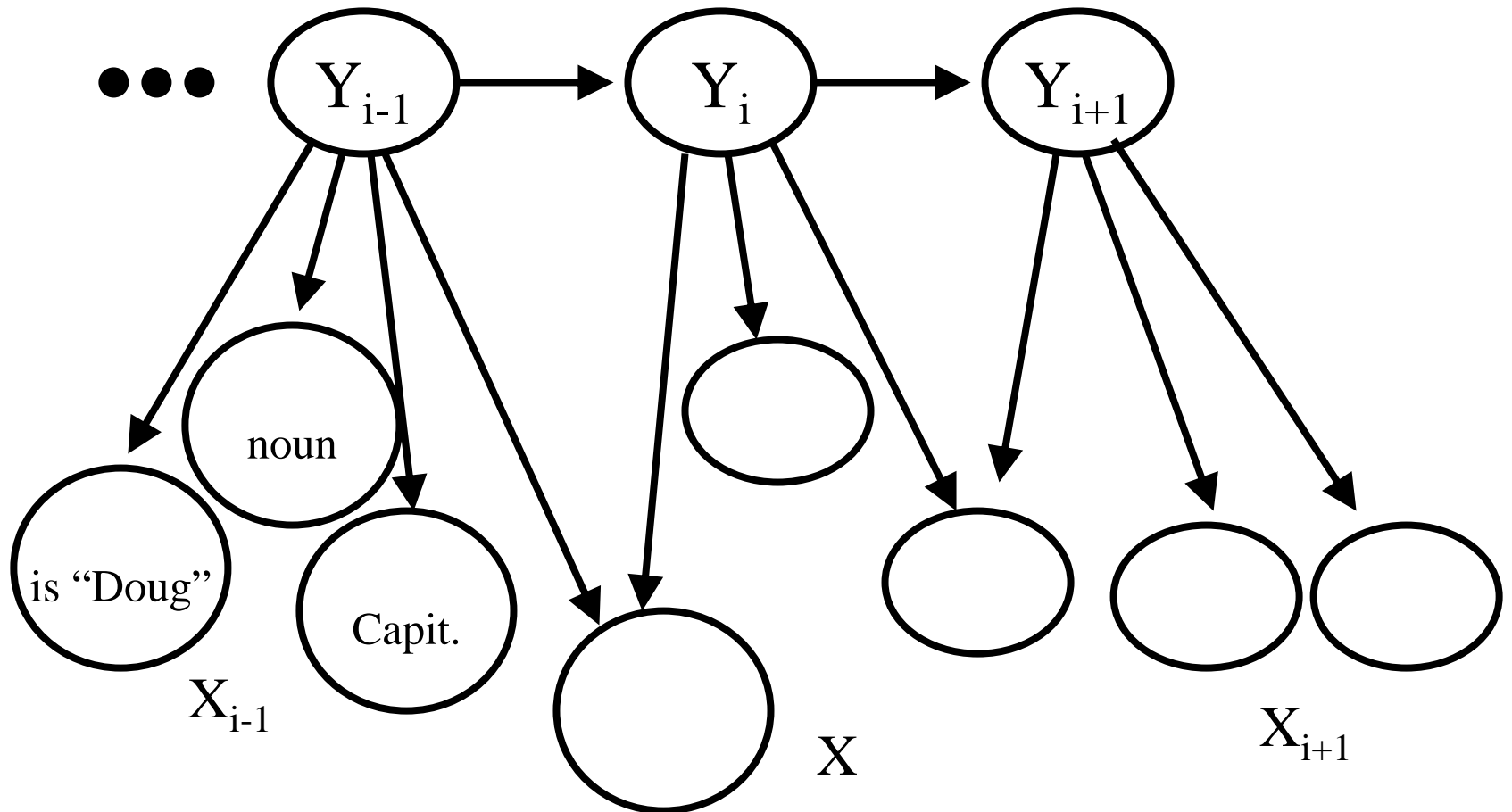
3 – an abundance of training data for one state has no effect on the others

Hidden Markov Model



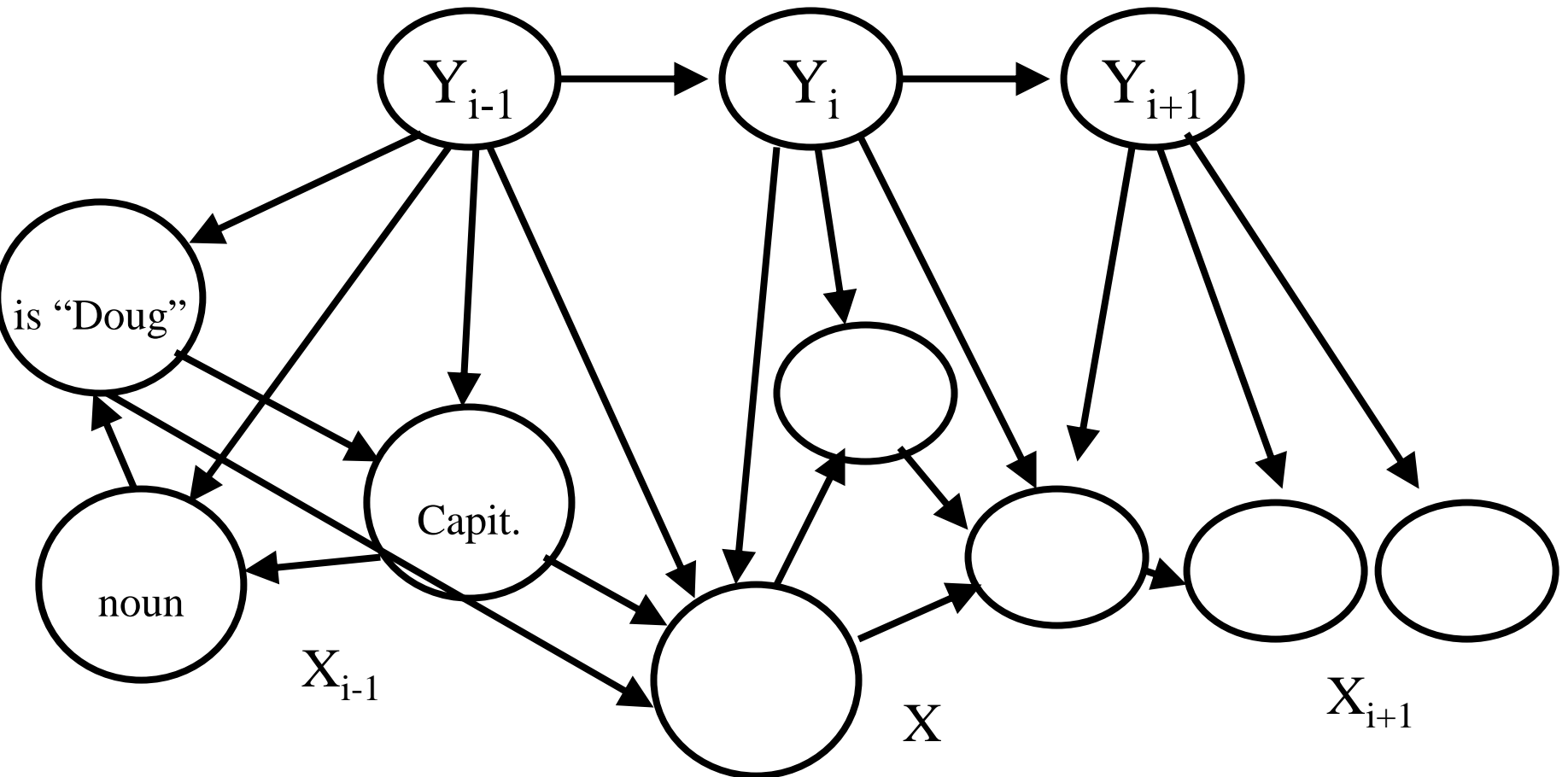
$$P(X, Y) = \prod_i P(X_i | Y_i) P(Y_i | Y_{i-1})$$

But how do we model this?



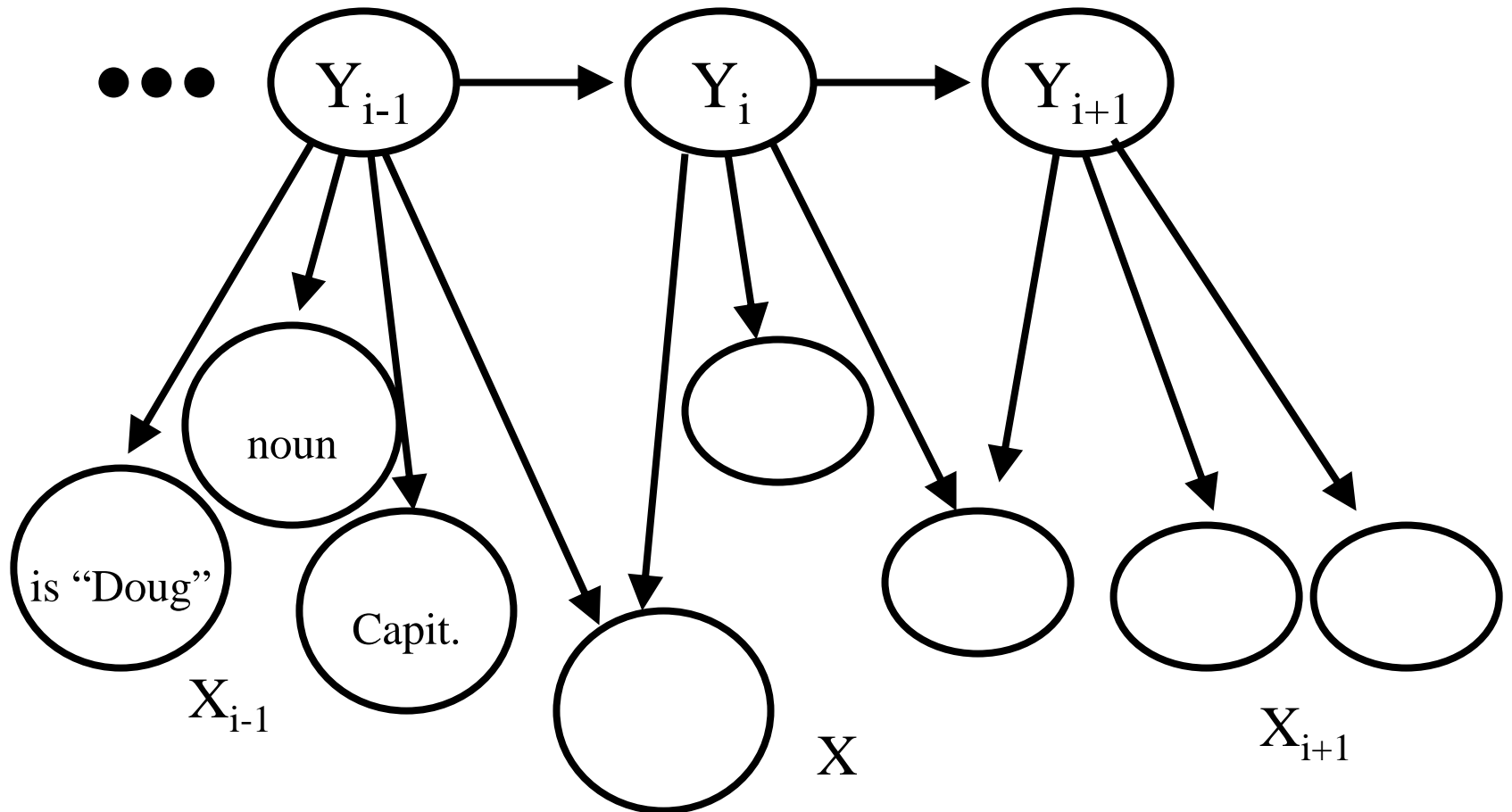
DEPENDENT FEATURES!!

Choice #1: Model all dependencies



Grows infeasible. Need LOTS of training data...

Choice #2: Ignore dependencies



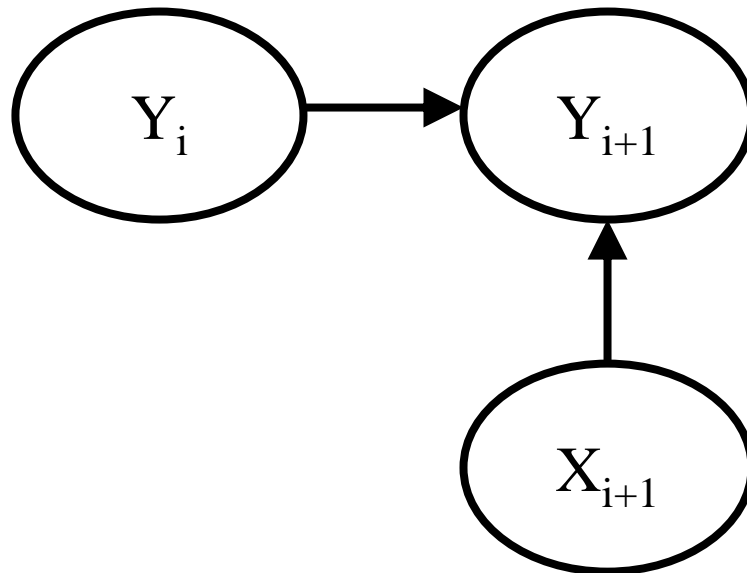
Not really a solution...

Conditional Model

- We prefer a model that is trained to maximize a *conditional* probability rather than *joint* probability: **$P(s|o)$ instead of $P(s,o)$**
 - Allow arbitrary, non-independent features on the observation sequence X
 - Examine features, but don't generate them. (There is not a directed transition from a state to an output)
 - Don't have to explicitly model their dependencies.
 - Conditionally trained means, “Given a set of observations (input) what is the most likely set of labels (states,nodes in the graph) that the model has been trained to traverse given this input”

Maximum Entropy Markov Models (MEMMs)

- Exponential model
- Given training set X with label sequence Y :
 - Train a model θ that maximizes $P(Y|X, \theta)$
 - For a new data sequence \mathbf{x} , the predicted label \mathbf{y} maximizes $P(\mathbf{y}|\mathbf{x}, \theta)$



MEMMs (cont'd)

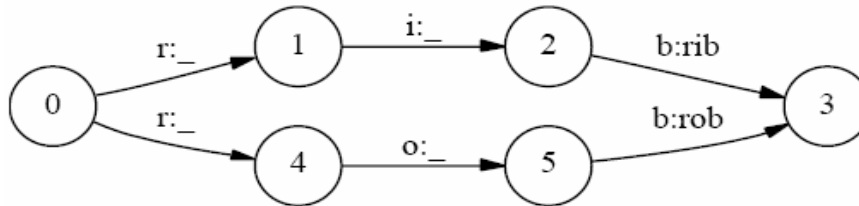
- MEMMs have all the advantages of Conditional Models

$$P(y' | y, x) = \frac{1}{Z(y, x)} \exp \left(\sum_k \underbrace{\lambda_k}_{\text{weight}} \underbrace{f_k(x, y, y')}_{\text{feature}} \right)$$

- Per-state normalization: all the mass that arrives at a state must be distributed among the possible successor states (“conservation of score mass”)
- Subject to Label Bias Problem

Label Bias Problem

- Consider this MEMM:



- $P(1 \text{ and } 2 \mid ro) = P(2 \mid 1 \text{ and } ro)P(1 \mid ro) = P(2 \mid 1 \text{ and } o)P(1 \mid r)$
 $P(1 \text{ and } 2 \mid ri) = P(2 \mid 1 \text{ and } ri)P(1 \mid ri) = P(2 \mid 1 \text{ and } i)P(1 \mid r)$
- Since $P(2 \mid 1 \text{ and } x) = 1$ for all x , $P(1 \text{ and } 2 \mid ro) = P(1 \text{ and } 2 \mid ri)$
In the training data, label value 2 is the only label value observed after label value 1
Therefore $P(2 \mid 1) = 1$, so $P(2 \mid 1 \text{ and } x) = 1$ for all x
- However, we expect $P(1 \text{ and } 2 \mid ri)$ to be greater than $P(1 \text{ and } 2 \mid ro)$.
- Per-state normalization does not allow the required expectation

Another view of Label Bias

However, since sequential classifiers are trained to make the best **local** decision, unlike generative models they cannot trade off decisions at **different positions** against each other. In other words, sequential classifiers are **myopic** about the impact of their current decision on later decisions (Bottou, 1991; Lafferty et al., 2001).

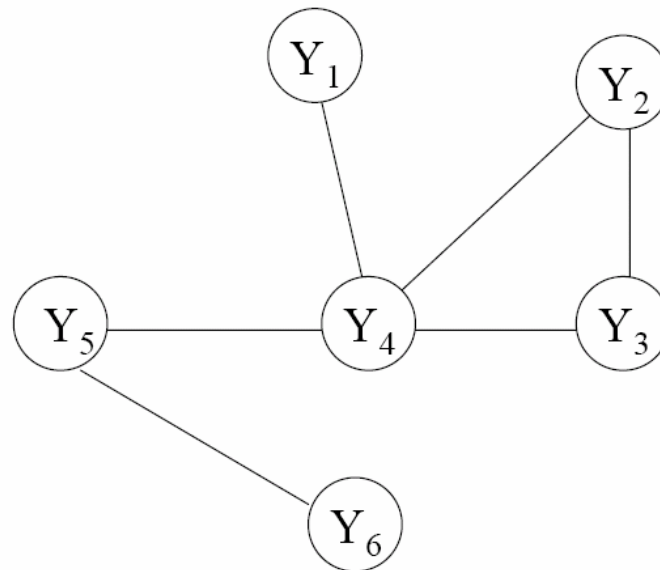
Conditional Random Fields (CRFs)

- CRFs have all the advantages of MEMMs without label bias problem
 - MEMM uses per-state exponential model for the conditional probabilities of next states given the current state
 - CRF has a single exponential model for the joint probability of the entire sequence of labels given the observation sequence
- Undirected acyclic graph
- Allow some transitions “vote” more strongly than others depending on the corresponding observations

Random Field – what it looks like

Let $G = (Y, E)$ be a graph where each vertex Y_v is a random variable
Suppose $P(Y_v | \text{all other } Y) = P(Y_v | \text{neighbors}(Y_v))$ then Y is a
random field

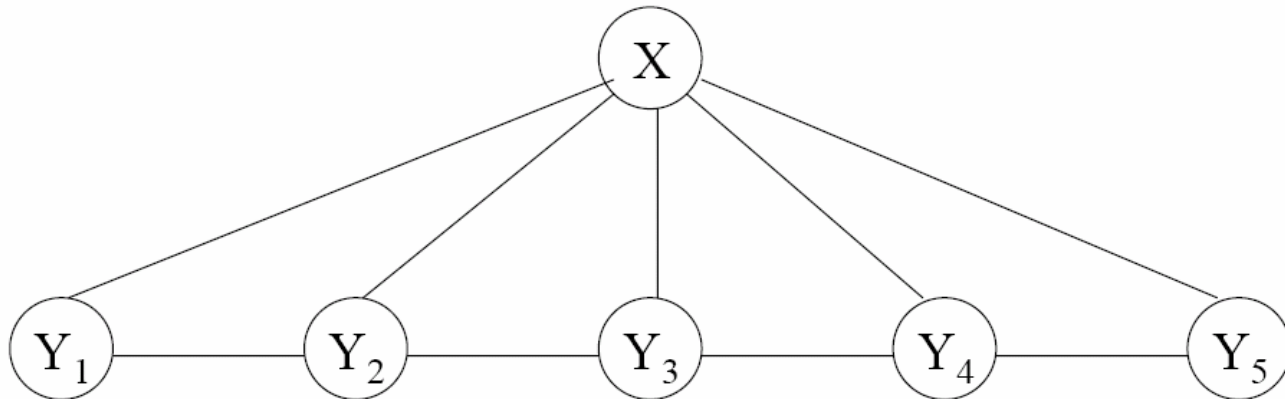
Example:



- $P(Y_5 | \text{all other } Y) = P(Y_5 | Y_4, Y_6)$

CRF – what it looks like

Suppose $P(Y_v | X, \text{all other } Y) = P(Y_v | X, \text{neighbors}(Y_v))$
then X with Y is a **conditional** random field



- $P(Y_3 | X, \text{all other } Y) = P(Y_3 | X, Y_2, Y_4)$
- Think of X as observations and Y as labels

CRF – the guts

vector \mathbf{f} of *local feature functions* $\mathbf{f} = \langle f^1, \dots, f^K \rangle$

map (\mathbf{x}, \mathbf{v}) and an index i to a measurement $f^k(i, \mathbf{x}, \mathbf{y}) \in \mathbb{R}$.

$$\mathbf{F}(\mathbf{x}, \mathbf{y}) = \sum_i^{|\mathbf{x}|} \mathbf{f}(i, \mathbf{x}, \mathbf{y})$$

For the case of NER $f^{13}(i, \mathbf{x}, \mathbf{y}) = [x_i \text{ is capitalized}] \cdot [y_i = I]$.

$$F^{13}(\mathbf{x}, \mathbf{y})$$

number of capitalized words x_i paired with the label I .

CRF...defined

We make feature functions to define features –
Not generated by model (X's of HMM)

$$\Pr(y|x, W) = \frac{1}{Z(x)} e^{W \cdot F(x,y)} \quad (1)$$

where W is a weight vector over the components of F , and $Z(x) = \sum_{y'} e^{W \cdot F(x,y')}$.

CRF

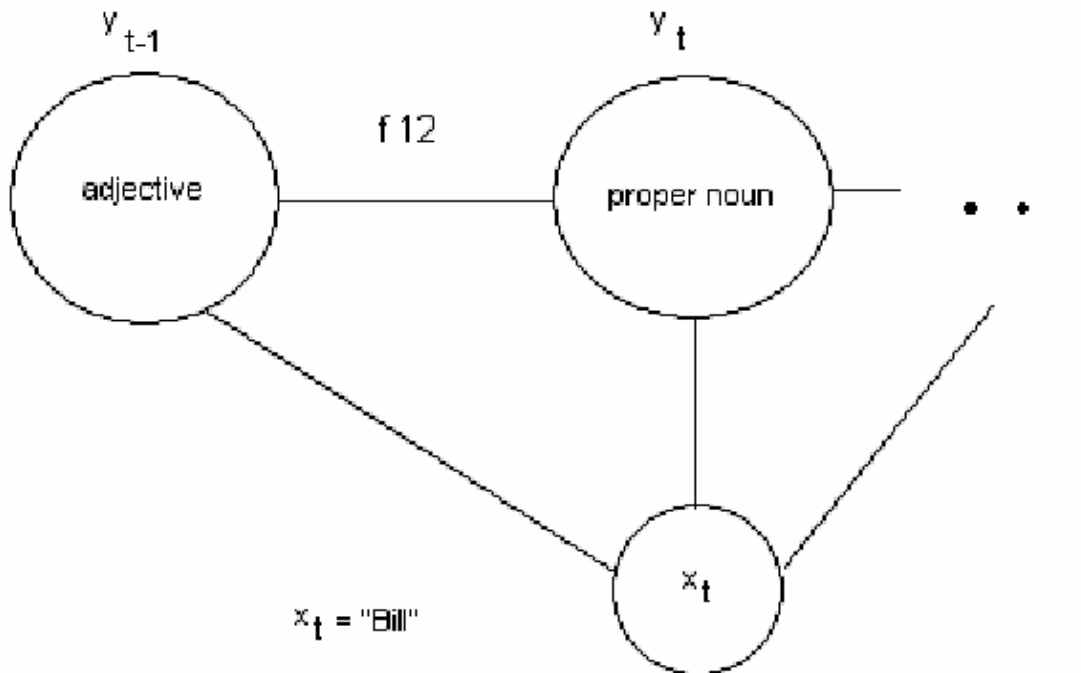
- Now we have $\text{Pr}(\text{label}|\text{obs.},\text{model})$
 - Find most probable label sequence (y's), given an observation sequence (x's)
 - No more independence assumption
 - conditionally trained for a whole label sequence given an input sequence (so long range and multi-feature reflected by this)

Example of a feature funct.

(y's are labels, x's are input obs)

$$f_{\mathbf{k}} (y_{t-1}, y_t, x, t)$$

$$f_{12} (\text{adjective, proper noun, } x, t) = \begin{cases} 1 & \text{if } x_t \text{ begins with Capital Letter} \\ 0 & \text{otherwise} \end{cases}$$





MALLET

- Machine learning toolkit specifically for language tasks
- Developed at U. Mass. by Andrew McCallum and his group
- For our purposes, we will use the SimpleTagger class which implements Conditional Random Fields



Getting MALLET to work...

1. Install Cygwin (HW Instructions)
2. Install Ant (HW Inst.)
3. Install MALLET (HW Inst.)
4. Train/Test/Label with SimpleTagger

SimpleTagger

□ Training

- Each line is of the form:

<feature1> <feature2> ... <featureN> <label>

Let's start with an example of a sentence:

Los Angeles is a great city!

We want to find all nouns, like the example in:

http://mallet.cs.umass.edu/index.php/SimpleTagger_example

Training CRFs

Let's say we have some tools that can identify features:

Colors → List of colors

Regex → Apostrophe finder

Regex → Capitalized

Stop-Words → Common tokens: a, the, etc.. (not etc. the word..)



Training CRFs

GOAL: Find NOUNS

Note: In SimpleTagger, the default “ignore” label is O (Used in HW)

LABELED INPUTS:

The SW CAP not-noun

red COLOR not-noun

bear’s APOS noun



Train SimpleTagger

- `java -cp "class;lib/mallet-deps.jar"`
`edu.umass.cs.mallet.base.fst.SimpleTagger --`
`train true --model-file SAVEDMODEL`
`TrainingData.txt`

Labeling with SimpleTagger

- Once you have a trained model, can re-use it to label new data!
- `java -cp "class;lib/mallet-deps.jar" edu.umass.cs.mallet.base.fst.SimpleTagger --include-input true --model-file SAVEDMODEL NotLabeledText.txt > LabeledOutput.txt`



CRFs and MALLETT

- Have fun!