Natural Language Processing

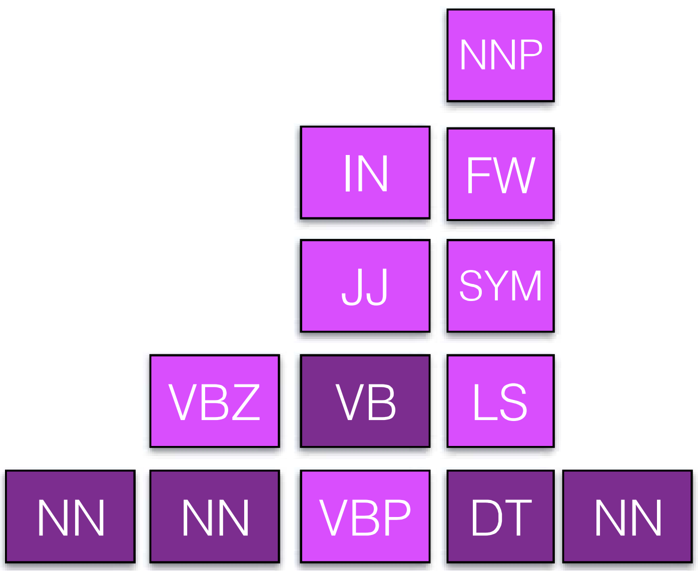
Info 159/259

Lecture 11: MEMM/CRF (Feb 25, 2020)

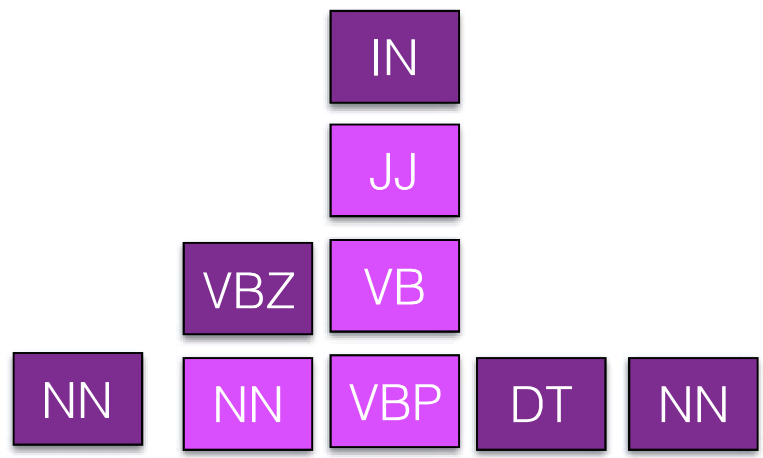
David Bamman, UC Berkeley

POS tagging

Labeling the tag that's correct for the context.



Fruit flies like a banana



Time flies like an arrow

(Just tags in evidence within the Penn Treebank — more are possible!)

Sequence labeling

$$x = \{x_1, \ldots, x_n\}$$

$$y = \{y_1, \ldots, y_n\}$$

- For a set of inputs x with n sequential time steps, one corresponding label y_i for each x_i

Generative vs. Discriminative models

- Generative models specify a joint distribution over the labels and the data. With this you could generate new data

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

- Discriminative models specify the conditional distribution of the label y given the data x. These models focus on how to discriminate between the classes

$$P(y \mid x)$$

Hidden Markov Model

Prior probability of label sequence

$$P(y) = P(y_1, \ldots, y_n)$$

$$P(y_1, \ldots, y_n) \approx \prod_{i=1}^{n+1} P(y_i \mid y_{i-1})$$

- We'll make a first-order Markov assumption and calculate the joint probability as the product the individual factors conditioned only on the previous tag.

Hidden Markov Model

$$\begin{aligned}
P(y_i, \ldots, y_n) = &\, P(y_1) \\
&\times P(y_2 \mid y_1) \\
&\times P(y_3 \mid y_1, y_2) \\
&\ldots \\
&\times P(y_n \mid y_1, \ldots, y_{n-1})
\end{aligned}$$

- Remember: a Markov assumption is an approximation to this exact decomposition (the chain rule of probability)

Hidden Markov Model

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

$$P(x_1, \ldots, x_n \mid y_1, \ldots, y_n) \approx \prod_{i=1}^{N} P(x_i \mid y_i)$$

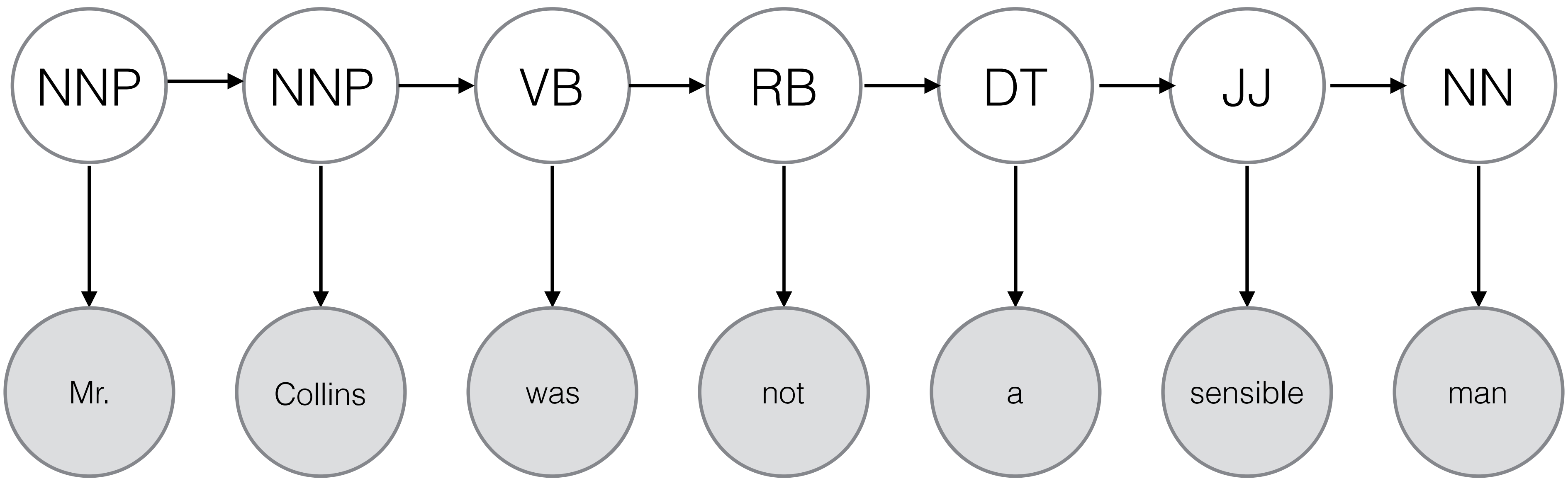
- Here again we'll make a strong assumption: the probability of the word we see at a given time step is only dependent on its label

HMM

$$P(x_1, \ldots, x_n, y_1, \ldots, y_n) \approx \prod_{i=1}^{n+1} P(y_i \mid y_{i-1}) \prod_{i=1}^{n} P(x_i \mid y_i)$$

HMM

$$P(VB \mid NNP)$$



$$P(was \mid VB)$$

Parameter estimation

$$P(y_t \mid y_{t-1}) \qquad \frac{c(y_1, y_2)}{c(y_1)}$$

MLE for both is just counting
(as in Naive Bayes)

$$P(x_t \mid y_t) \qquad \frac{c(x, y)}{c(y)}$$

Decoding

- Greedy: proceed left to right, committing to the best tag for each time step (given the sequence seen so far)

Fruit	flies	like	a	banana
NN	VB	IN	DT	NN

Decoding

DT	NN	VBD	IN	DT	NN	???
The	horse	raced	past	the	barn	fell
DT	NN	VBN	IN	DT	NN	VBD

Information later on in the sentence can influence the best tags earlier on.

All paths

END							
DT							
NNP							
VB							
NN							
MD							
START							
	^	Janet	will	back	the	bill	$

Ideally, what we want is to calculate the joint probability of each path and pick the one with the highest probability. But for N time steps and K labels, number of possible paths = K^N

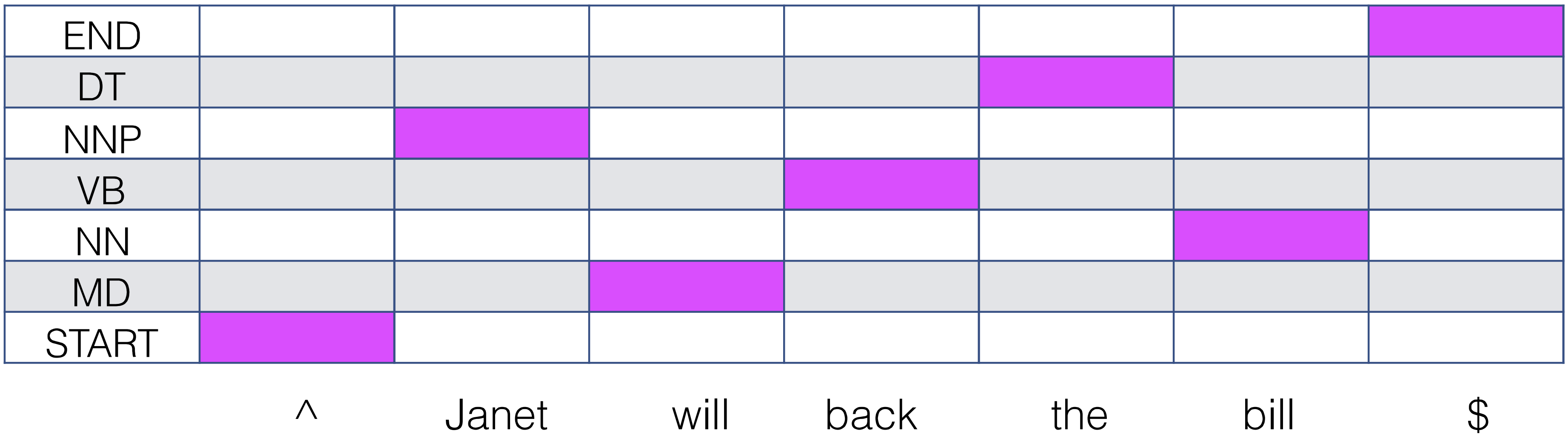
5 word sentence with 45 Penn Treebank tags

45^5 = 184,528,125 different paths

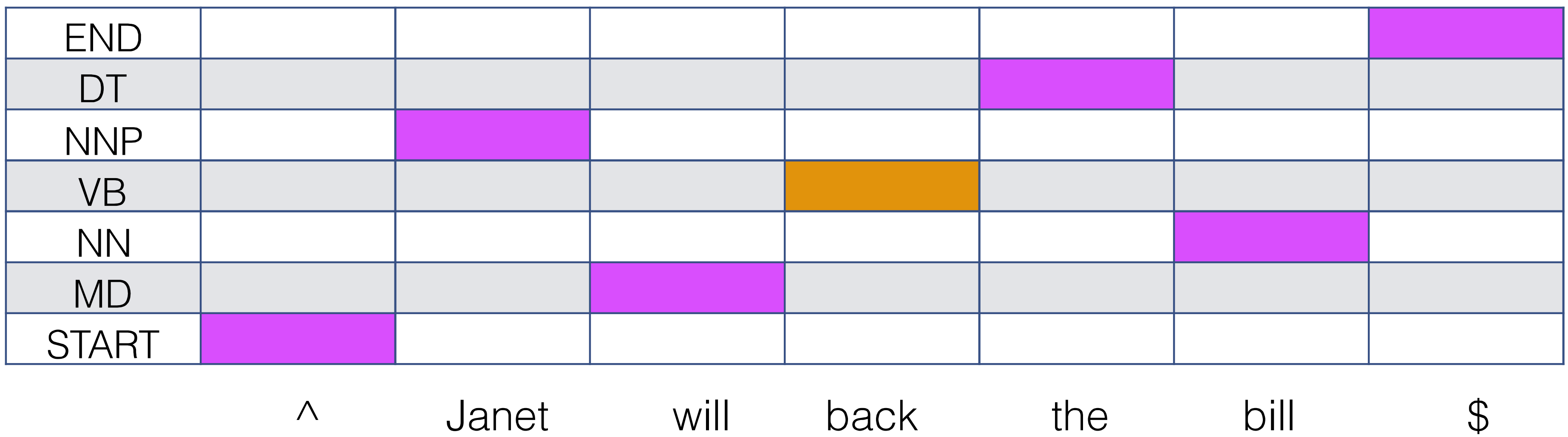
45^{20} = 1.16e33 different paths

Viterbi algorithm

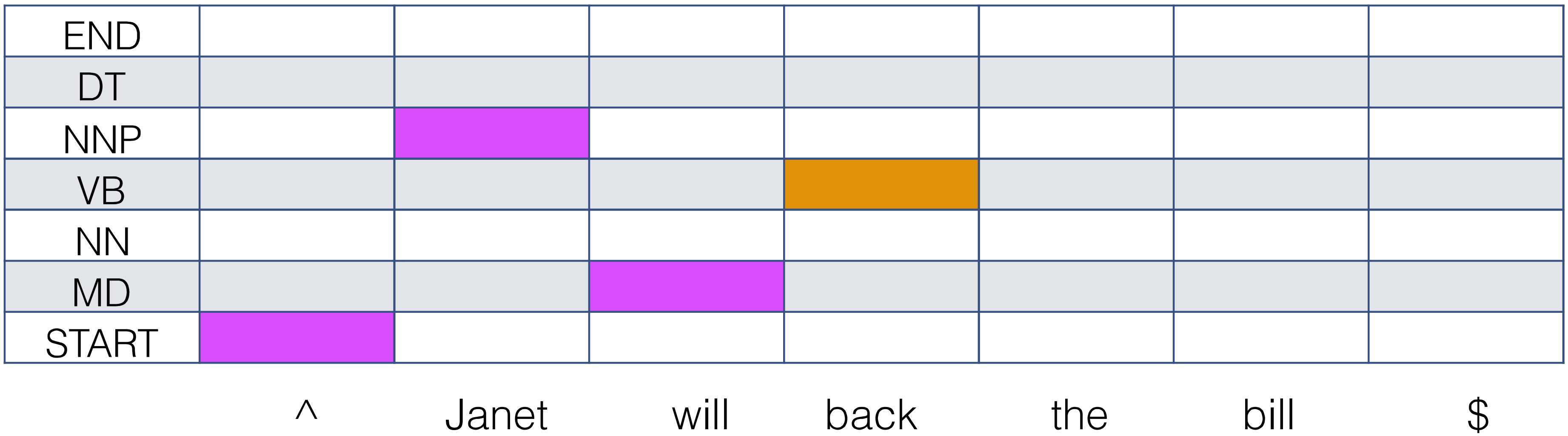
- Basic idea: if an optimal path through a sequence uses label L at time T, then it must have used an optimal path to get to label L at time T
- We can discard all non-optimal paths up to label L at time T



Let’s say this is the best sequence for the entire sentence

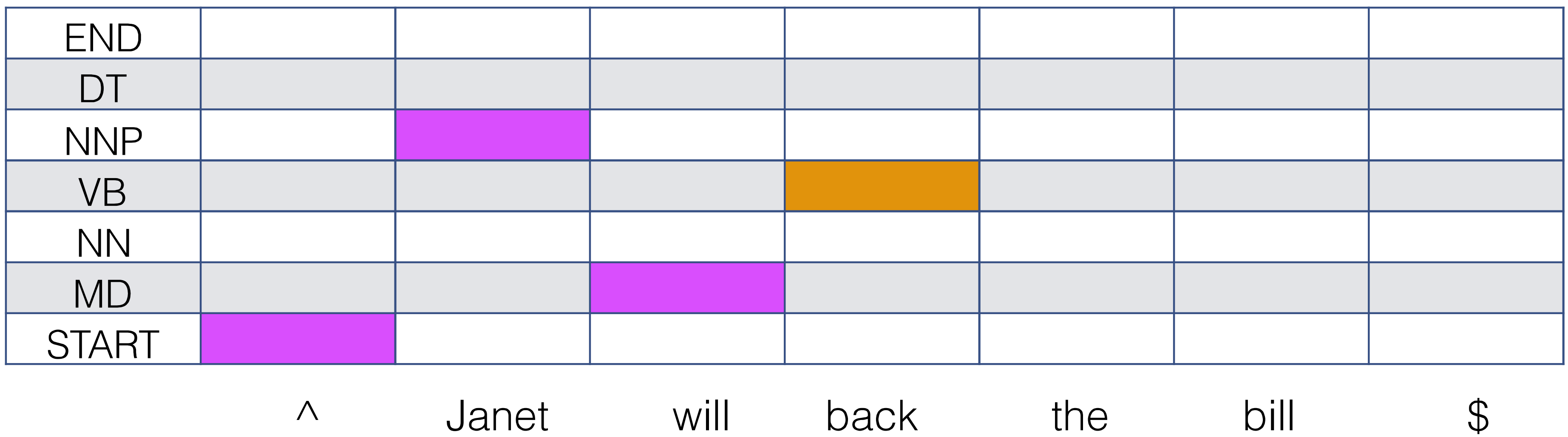


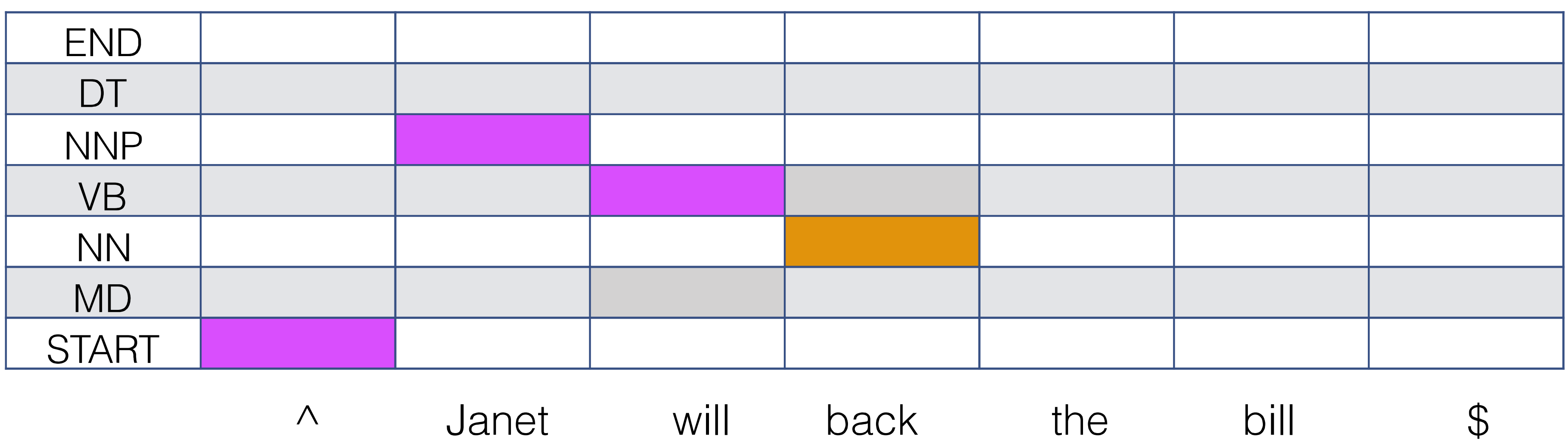
back = VB is in the optimal sequence



If this is the optimal path to *back* = VB in the entire sequence, then every other path to *back* = VB must be less likely (otherwise it would be the optimal path itself!)

Importantly, the best path to *back* = NN might look different.





END							
DT							
NNP							
VB							
NN							
MD							
START							
	^	Janet	will	back	the	bill	$

- At each time step t ending in label K, we find the max probability of any path that led to that state

END		
DT		$v_1(DT)$
NNP		$v_1(NNP)$
VB		$v_1(VB)$
NN		$v_1(NN)$
MD		$v_1(MD)$
START		
		Janet

What's the HMM probability of ending in Janet = NNP?

$$P(y_t \mid y_{t-1})P(x_t \mid y_t)$$

$$P(\text{NNP} \mid \text{START})P(Janet \mid \text{NNP})$$

END		
DT		v_1(DT)
NNP		v_1(NNP)
VB		v_1(VB)
NN		v_1(NN)
MD		v_1(MD)
START		
		Janet

Best path through time step 1 ending in tag y (trivially - best path for all is just START)

$$v_1(y) = \max_{u \in \mathcal{Y}} [P(y_t = y \mid y_{t-1} = u)P(x_t \mid y_t = y)]$$

END			
DT		$v_1(DT)$	$v_2(DT)$
NNP		$v_1(NNP)$	$v_2(NNP)$
VB		$v_1(VB)$	$v_2(VB)$
NN		$v_1(NN)$	$v_2(NN)$
MD		$v_1(MD)$	$v_2(MD)$
START			
		Janet	will

What's the max HMM probability of ending in will = MD?

First, what's the HMM probability of a single path ending in will = MD?

END			
DT		$v_1(DT)$	$v_2(DT)$
NNP		$v_1(NNP)$	$v_2(NNP)$
VB		$v_1(VB)$	$v_2(VB)$
NN		$v_1(NN)$	$v_2(NN)$
MD		$v_1(MD)$	$v_2(MD)$
START			
		Janet	will

$$P(y_1 \mid START)P(x_1 \mid y_1) \times P(y_2 = \mathrm{MD} \mid y_1)P(x_2 \mid y_2 = \mathrm{MD})$$

END			
DT		v_1(DT)	v_2(DT)
NNP		v_1(NNP)	v_2(NNP)
VB		v_1(VB)	v_2(VB)
NN		v_1(NN)	v_2(NN)
MD		v_1(MD)	v_2(MD)
START			
		Janet	will

Best path through time step 2 ending in tag MD

$$P(\mathrm{DT} \mid \mathrm{START}) \times P(Janet \mid \mathrm{DT}) \times P(y_t = \mathrm{MD} \mid P(y_{t-1} = \mathrm{DT}) \times P(will \mid y_t = \mathrm{MD})$$

$$P(\mathrm{NNP} \mid \mathrm{START}) \times P(Janet \mid \mathrm{NNP}) \times P(y_t = \mathrm{MD} \mid P(y_{t-1} = \mathrm{NNP}) \times P(will \mid y_t = \mathrm{MD})$$

$$P(\mathrm{VB} \mid \mathrm{START}) \times P(Janet \mid \mathrm{VB}) \times P(y_t = \mathrm{MD} \mid P(y_{t-1} = \mathrm{VB}) \times P(will \mid y_t = \mathrm{MD})$$

$$P(\mathrm{NN} \mid \mathrm{START}) \times P(Janet \mid \mathrm{NN}) \times P(y_t = \mathrm{MD} \mid P(y_{t-1} = \mathrm{NN}) \times P(will \mid y_t = \mathrm{MD})$$

$$P(\mathrm{MD} \mid \mathrm{START}) \times P(Janet \mid \mathrm{MD}) \times P(y_t = \mathrm{MD} \mid P(y_{t-1} = \mathrm{MD}) \times P(will \mid y_t = \mathrm{MD})$$

END			
DT		v_1(DT)	v_2(DT)
NNP		v_1(NNP)	v_2(NNP)
VB		v_1(VB)	v_2(VB)
NN		v_1(NN)	v_2(NN)
MD		v_1(MD)	v_2(MD)
START			
		Janet	will

Best path through time step 2 ending in tag MD

Let's say the best path ending will = MD includes Janet = NNP. By definition, every other path has lower probability.

END			
DT		$v_1(\text{DT})$	$v_2(\text{DT})$
NNP		$v_1(\text{NNP})$	$v_2(\text{NNP})$
VB		$v_1(\text{VB})$	$v_2(\text{VB})$
NN		$v_1(\text{NN})$	$v_2(\text{NN})$
MD		$v_1(\text{MD})$	$v_2(\text{MD})$
START			
		Janet	will

Best path through time step 2 ending in tag MD

$$P(\text{DT} \mid \text{START}) \times P(Janet \mid \text{DT}) \times P(y_t = \text{MD} \mid P(y_{t-1} = \text{DT}) \times P(will \mid y_t = \text{MD})$$

$$P(\text{NNP} \mid \text{START}) \times P(Janet \mid \text{NNP}) \times P(y_t = \text{MD} \mid P(y_{t-1} = \text{NNP}) \times P(will \mid y_t = \text{MD})$$

$$P(\text{VB} \mid \text{START}) \times P(Janet \mid \text{VB}) \times P(y_t = \text{MD} \mid P(y_{t-1} = \text{VB}) \times P(will \mid y_t = \text{MD})$$

$$P(\text{NN} \mid \text{START}) \times P(Janet \mid \text{NN}) \times P(y_t = \text{MD} \mid P(y_{t-1} = \text{NN}) \times P(will \mid y_t = \text{MD})$$

$$P(\text{MD} \mid \text{START}) \times P(Janet \mid \text{MD}) \times P(y_t = \text{MD} \mid P(y_{t-1} = \text{MD}) \times P(will \mid y_t = \text{MD})$$

$$v_1(y) = \max_{u \in \mathcal{Y}} \left[P(y_t = y \mid y_{t-1} = u) P(x_t \mid y_t = y) \right]$$

$v_1(u)$

$$\boxed{P(\text{DT} \mid \text{START}) \times P(Janet \mid \text{DT})} \times P(y_t = \text{MD} \mid P(y_{t-1} = \text{DT}) \times P(will \mid y_t = \text{MD})$$

$$\boxed{P(\text{NNP} \mid \text{START}) \times P(Janet \mid \text{NNP})} \times P(y_t = \text{MD} \mid P(y_{t-1} = \text{NNP}) \times P(will \mid y_t = \text{MD})$$

$$\boxed{P(\text{VB} \mid \text{START}) \times P(Janet \mid \text{VB})} \times P(y_t = \text{MD} \mid P(y_{t-1} = \text{VB}) \times P(will \mid y_t = \text{MD})$$

$$\boxed{P(\text{NN} \mid \text{START}) \times P(Janet \mid \text{NN})} \times P(y_t = \text{MD} \mid P(y_{t-1} = \text{NN}) \times P(will \mid y_t = \text{MD})$$

$$\boxed{P(\text{MD} \mid \text{START}) \times P(Janet \mid \text{MD})} \times P(y_t = \text{MD} \mid P(y_{t-1} = \text{MD}) \times P(will \mid y_t = \text{MD})$$

$$v_1(u) \times P(y_t = \text{MD} \mid y_{t-1} = \text{u}) \times P(will \mid y_t = \text{MD})$$

END			
DT		v_1(DT)	v_2(DT)
NNP		v_1(NNP)	v_2(NNP)
VB		v_1(VB)	v_2(VB)
NN		v_1(NN)	v_2(NN)
MD		v_1(MD)	v_2(MD)
START			
		Janet	will

$$v_t(y) = \max_{u \in \mathcal{Y}} \left[v_{t-1}(u) \times P(y_t = y \mid y_{t-1} = u) P(x_t \mid y_t = y) \right]$$

END				
DT		v_1(DT)	v_2(DT)	v_3(DT)
NNP		v_1(NNP)	v_2(NNP)	v_3(NNP)
VB		v_1(VB)	v_2(VB)	v_3(VB)
NN		v_1(NN)	v_2(NN)	v_3(NN)
MD		v_1(MD)	v_2(MD)	v_3(MD)
START				
		Janet	will	back

25 paths ending in back = VB

END				
DT		v_1(DT)	v_2(DT)	v_3(DT)
NNP		v_1(NNP)	v_2(NNP)	v_3(NNP)
VB		v_1(VB)	v_2(VB)	v_3(VB)
NN		v_1(NN)	v_2(NN)	v_3(NN)
MD		v_1(MD)	v_2(MD)	v_3(MD)
START				
		Janet	will	back

Let’s say the best path ending in back = VB includes will = MD.

END				
DT		v_1(DT)	v_2(DT)	v_3(DT)
NNP		v_1(NNP)	v_2(NNP)	v_3(NNP)
VB		v_1(VB)	v_2(VB)	v_3(VB)
NN		v_1(NN)	v_2(NN)	v_3(NN)
MD		v_1(MD)	v_2(MD)	v_3(MD)
START				
		Janet	will	back

If the best path ending in will = MD includes Janet=NNP, we can forget all paths with Janet != NNP for any path including will = MD because we know they are less likely.

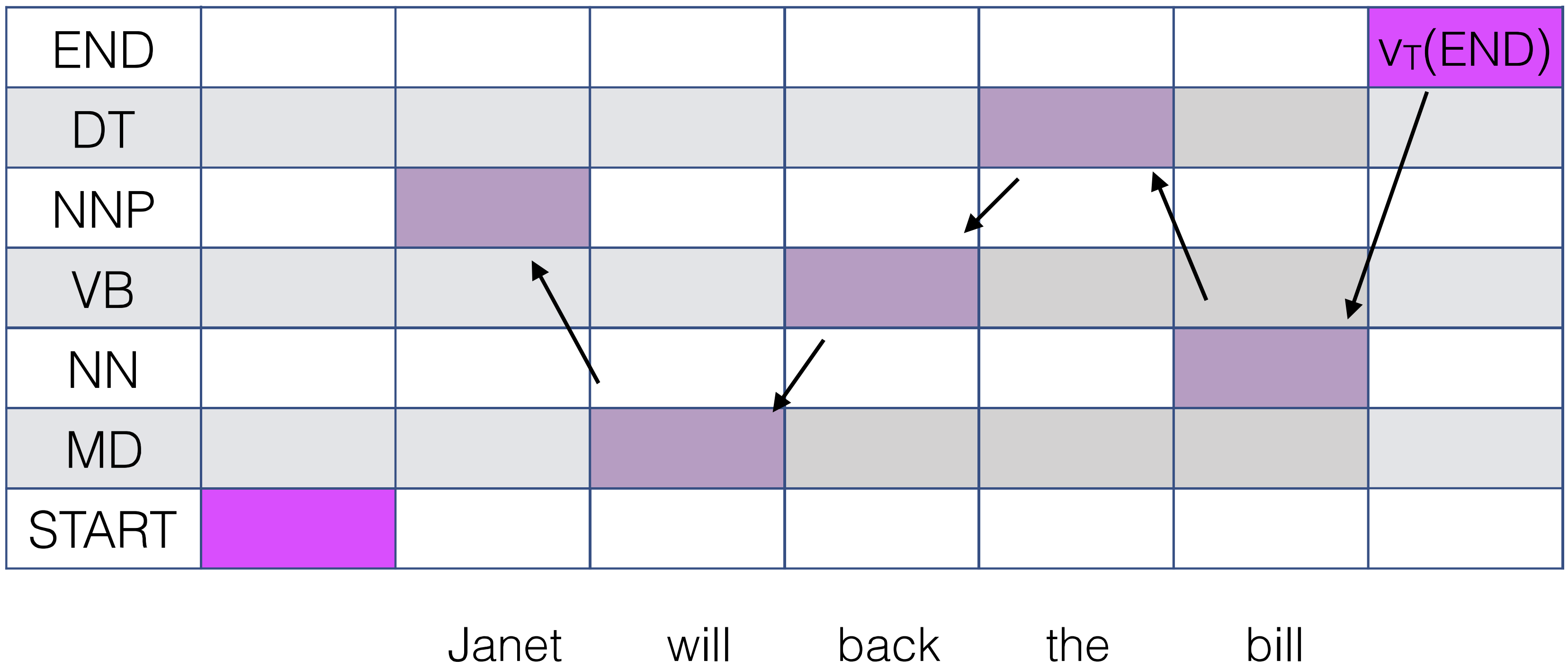
END					
DT		$v_1(DT)$	$v_2(DT)$	$v_3(DT)$	$v_4(DT)$
NNP		$v_1(NNP)$	$v_2(NNP)$	$v_3(NNP)$	$v_4(NNP)$
VB		$v_1(VB)$	$v_2(VB)$	$v_3(VB)$	$v_4(MD)$
NN		$v_1(NN)$	$v_2(NN)$	$v_3(NN)$	$v_4(NN)$
MD		$v_1(MD)$	$v_2(MD)$	$v_3(MD)$	$v_4(MD)$
START					
		Janet	will	back	the

125 possible paths ending in the = DT, but we only need to consider 5 (best path ending in back = DT, back = NNP, back = VB, back = NN, back = MD)

END						
DT		v_1(DT)	v_2(DT)	v_3(DT)	v_4(DT)	v_5(DT)
NNP		v_1(NNP)	v_2(NNP)	v_3(NNP)	v_4(NNP)	v_5(NNP)
VB		v_1(VB)	v_2(VB)	v_3(VB)	v_4(MD)	v_5(MD)
NN		v_1(NN)	v_2(NN)	v_3(NN)	v_4(NN)	v_5(NN)
MD		v_1(MD)	v_2(MD)	v_3(MD)	v_4(MD)	v_5(MD)
START						
		Janet	will	back	the	bill

END							v_T(END)
DT		v_1(DT)	v_2(DT)	v_3(DT)	v_4(DT)	v_5(DT)	
NNP		v_1(NNP)	v_2(NNP)	v_3(NNP)	v_4(NNP)	v_5(NNP)	
VB		v_1(VB)	v_2(VB)	v_3(VB)	v_4(MD)	v_5(MD)	
NN		v_1(NN)	v_2(NN)	v_3(NN)	v_4(NN)	v_5(NN)	
MD		v_1(MD)	v_2(MD)	v_3(MD)	v_4(MD)	v_5(MD)	
START							
		Janet	will	back	the	bill	

v_T(END) encodes the best path through the entire sequence



For each timestep t + label, keep track of the max element from t-1 to reconstruct best path

function VITERBI(*observations* of len *T*,*state-graph* of len *N*) **returns** *best-path*

create a path probability matrix *viterbi[N+2,T]*
for each state *s* **from** 1 **to** *N* **do** ; initialization step
$\quad viterbi[s,1] \leftarrow a_{0,s} * b_s(o_1)$
$\quad backpointer[s,1] \leftarrow 0$
for each time step *t* **from** 2 **to** *T* **do** ; recursion step
for each state *s* **from** 1 **to** *N* **do**
$\quad viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$\quad backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s}$
$viterbi[q_F,T] \leftarrow \max_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
$backpointer[q_F,T] \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
return the backtrace path by following backpointers to states back in time from $backpointer[q_F,T]$

Figure 10.8 Viterbi algorithm for finding optimal sequence of tags. Given an observation sequence and an HMM $\lambda = (A,B)$, the algorithm returns the state path through the HMM that assigns maximum likelihood to the observation sequence. Note that states 0 and q_F are non-emitting.

END		
DT		v_1(DT)
NNP		v_1(NNP)
VB		v_1(VB)
NN		v_1(NN)
MD		v_1(MD)
START		
		Janet

Can Viterbi decoding help with independent preditions? (e.g., Naive Bayes or logreg)

$$v_1(y) = \max_{u \in \mathcal{Y}} \left[P(y_t = y \mid y_{t-1} = u) P(x_t \mid y_t = y) \right]$$

When making independent predictions:

$$P(y_t = y \mid y_{t-1} = u) = P(y_t = y)$$

Generative vs. Discriminative models

- Generative models specify a joint distribution over the labels and the data. With this you could generate new data

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

- Discriminative models specify the conditional distribution of the label y given the data x. These models focus on how to discriminate between the classes

$$P(y \mid x)$$

MEMM

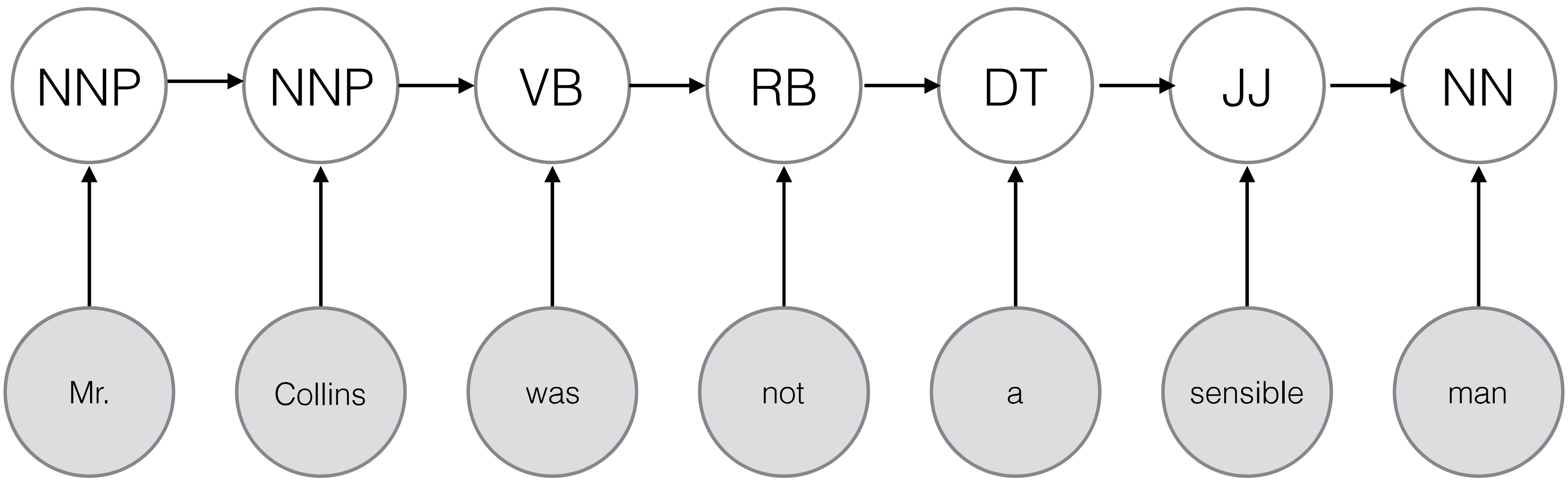
General maxent form

$$\arg\max_y P(y \mid x, \beta)$$

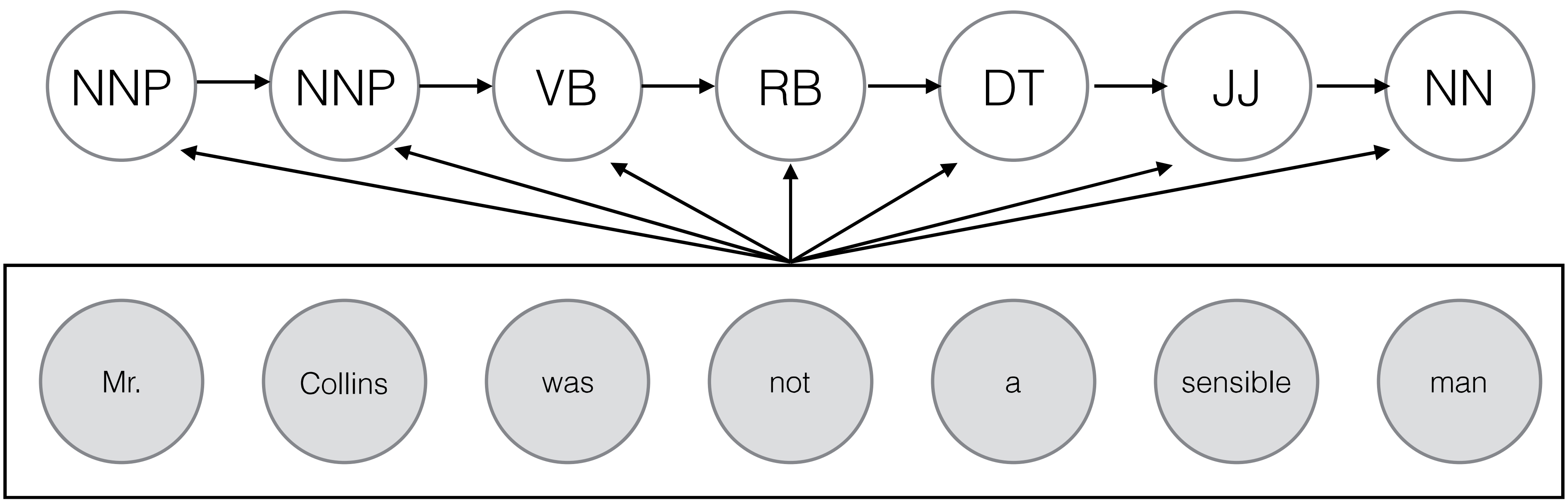
Maxent with first-order Markov assumption: Maximum Entropy Markov Model

$$\arg\max_y \prod_{i=1}^{n} P(y_i \mid y_{i-1}, x)$$

MEMM

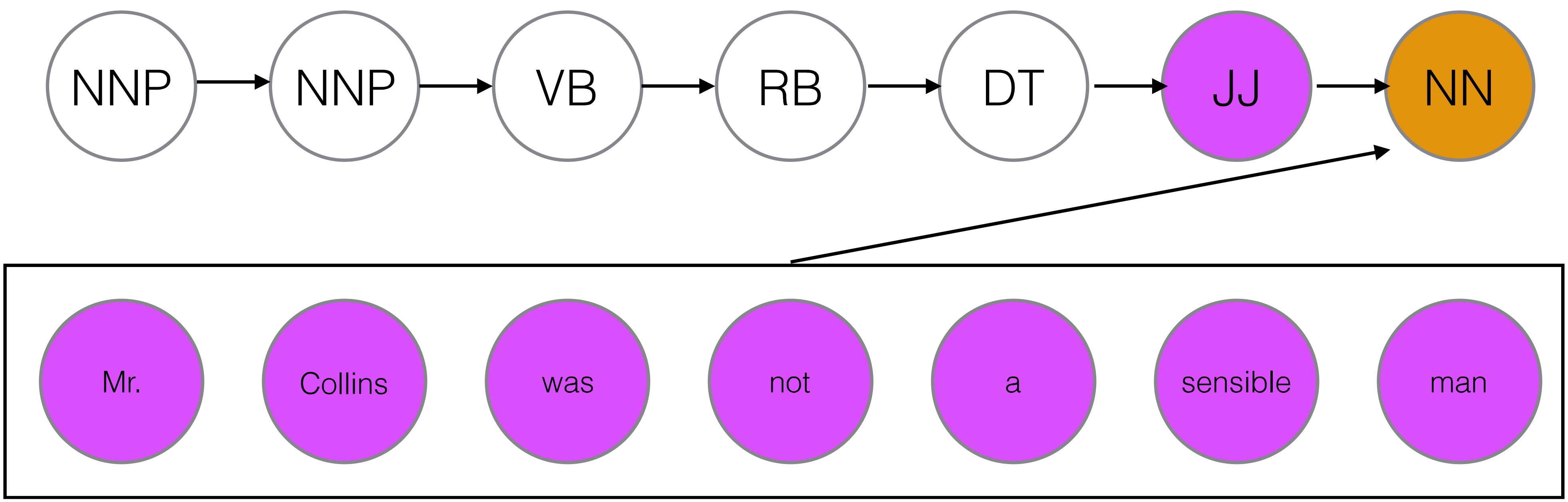


MEMM



MEMMs condition on the *entire* input

MEMM



Features

$$f(y_i, y_{i-1}; x_1, \ldots, x_n)$$

Features are scoped over the previous predicted tag and the entire observed input

feature	example
x_i = man	1
y_{i-1} = JJ	1
i=n (last word of sentence)	1
x_i ends in -ly	0

Training

$$\prod_{i=1}^{n} P(y_i \mid y_{i-1}, x, \beta)$$

For all training data, we want probability of the true label y_i conditioned on the previous true label y_{i-1} to be high.

This is simply multiclass logistic regression

Decoding

- With logistic regression, our prediction is simply the argmax y:

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

- With an MEMM, we know the true y_{i-1} during training but we never of course know it at test time

$$P(y_i \mid y_{i-1}, x, \beta)$$

Greedy decoding

- A i=1, predict the argmax given START:

$$P(y_1 \mid START, x, \beta)$$

- For each subsequent time step, condition on the y just predicted during the step before

$$P(y_i \mid y_{i-1}, x, \beta)$$

Viterbi decoding

Viterbi for HMM: max joint probability

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

$$v_t(y) = \max_{u \in \mathcal{Y}} \left[v_{t-1}(u) \times P(y_t = y \mid y_{t-1} = u) P(x_t \mid y_t = y) \right]$$

Viterbi for MEMM: max conditional probability

$$P(y \mid x)$$

$$v_t(y) = \max_{u \in \mathcal{Y}} [v_{t-1}(u) \times P(y_t = y \mid y_{t-1} = u, x, \beta)]$$

MEMM Training

$$\prod_{i=1}^{n} P(y_i \mid y_{i-1}, x, \beta)$$

For all training data, we want probability of the true label y_i conditioned on the previous true label y_{i-1} to be high.

This is simply multiclass logistic regression

MEMM Training

$$\prod_{i=1}^{n} P(y_i \mid y_{i-1}, x, \beta)$$

Locally normalized — at each time step, each conditional distribution sums to 1

Label bias

$$\prod_{i=1}^{n} P(y_i \mid y_{i-1}, x, \beta)$$

- For a given conditioning context, the probability of a tag (e.g., VBZ) only competes against other tags with that same context (e.g., NN)

Bottou 2001; Lafferty et al. 2001

Label bias

NN	TO	VB

will to fight

	NN	MD
x_i=will	10	40
y_{i-1}=START	-1	7
BIAS	7	-2

Modals show up much more frequently at the start of the sentence than nouns do (e.g., questions)

Toutanova et al. 2003

Label bias

NN TO VB

will to fight

But we know that MD + TO is very rare

- *can to eat
- *would to eat
- *could to eat
- *may to eat

Toutanova et al. 2003

Label bias

NN	TO	VB

will to fight

	TO
x_i=to	10000000
y_{i-1}=NN	0
y_{i-1}=MD	0

to is relatively deterministic (almost always TO) so it doesn't matter what tag precedes it.

Toutanova et al. 2003

Label bias

NN TO VB

will to fight

$$\prod_{i=1}^{n} P(y_i \mid y_{i-1}, x, \beta)$$

Because of this local normalization, P(TO | context) will always be 1 if x="to"

Toutanova et al. 2003

Label bias

NN TO VB

will to fight

That means our prediction for *to* can't help us disambiguate *will*. We lose the information that MD + TO sequences rarely happen.

Toutanova et al. 2003

Label bias

Viterbi decoding doesn't help in this case

$$v_t(y) = \max_{u \in \mathcal{Y}} [v_{t-1}(u) \times P(y_t = y \mid y_{t-1} = u, x, \beta)]$$

$$P(y_t = \text{TO} \mid y_{t-1} = \text{MD}, x, \beta) = 1$$

$$P(y_t = \text{TO} \mid y_{t-1} = \text{NN}, x, \beta) = 1$$

Conditional random fields

- We can solve this problem using global normalization (over the entire sequences) rather than locally normalized factors.

MEMM

$$P(y \mid x, \beta) = \prod_{i=1}^{n} P(y_i \mid y_{i-1}, x, \beta)$$

CRF

$$P(y \mid x, \beta) = \frac{\exp(\Phi(x, y)^\top \beta)}{\sum_{y' \in \mathcal{Y}} \exp(\Phi(x, y')^\top \beta)}$$

Conditional random fields

$$P(y \mid x, \beta) = \frac{\exp(\Phi(x,y)^\top \beta)}{\sum_{y' \in \mathcal{Y}} \exp(\Phi(x,y')^\top \beta)}$$

Feature vector scoped over the entire input and label sequence

$$\Phi(x,y) = \sum_{i=1}^{n} \phi(x, i, y_i, y_{i-1})$$

ϕ is the same feature vector we used for local predictions using MEMMs

Features

$$\phi(x, i, y_i, y_{i-1})$$

Features are scoped over the previous predicted tag and the entire observed input

feature	example
x_i = man	1
y_{i-1} = JJ	1
i=n (last word of sentence)	1
x_i ends in -ly	0

In an MEMM, we estimate $P(y_t \mid y_{t-1}, x, \beta)$ from each $\phi(x, t, y_t, y_{t-1})$ independently

	will $\phi(x, 1, y_1, y_0)$	to $\phi(x, 2, y_2, y_1)$	fight $\phi(x, 3, y_3, y_2)$
x_i=will ^ y_i = NN	1	0	0
y_{i-1}=START ^ y_i = NN	1	0	0
x_i=will ^ y_i = MD	0	0	0
y_{i-1}=START ^ y_i = MD	0	0	0
…			
x_i=to ^ y_i = TO	0	1	0
y_{i-1}=NN ^ y_i = TO	0	1	0
y_{i-1}=MD ^ y_i = TO	0	0	0
…			
x_i=fight ^ y_i = VB	0	0	1
y_{i-1}=TO ^ y_i = VB	0	0	1

In a CRF, we use features from the entire sequence (by summing the individual features at each time step)

	will $\phi(x, 1, y_1, y_0)$	to $\phi(x, 2, y_2, y_1)$	fight $\phi(x, 3, y_3, y_2)$	$\mathbf{\Phi}$(x, NN TO VB)
x_i=will ^ y_i = NN	1	0	0	1
y_{i-1}=START ^ y_i = NN	1	0	0	1
x_i=will ^ y_i = MD	0	0	0	0
y_{i-1}=START ^ y_i = MD	0	0	0	0
…				
x_i=to ^ y_i = TO	0	1	0	1
y_{i-1}=NN ^ y_i = TO	0	1	0	1
y_{i-1}=MD ^ y_i = TO	0	0	0	0
…				
x_i=fight ^ y_i = VB	0	0	1	1
y_{i-1}=TO ^ y_i = VB	0	0	1	1

In a CRF, we use features from the entire sequence (by summing the individual features at each time step)

	$\mathbf{\Phi}$(x, NN TO VB)
x_i=will ^ y_i = NN	1
y_{i-1}=START ^ y_i = NN	1
x_i=will ^ y_i = MD	0
y_{i-1}=START ^ y_i = MD	0
…	
x_i=to ^ y_i = TO	1
y_{i-1}=NN ^ y_i = TO	1
y_{i-1}=MD ^ y_i = TO	0
…	
x_i=fight ^ y_i = VB	1
y_{i-1}=TO ^ y_i = VB	1

The sequence contains a position *at the beginning* whose word = will and whose tag is NN

The sequence contains a position *somewhere* whose word = to, whose tag is TO, and follows an NN

The sequence contains a position *somewhere* whose word = fight, whose tag is VB, and follows a TO

This lets us isolate the global sequence features that separate good sequences (in our training data) from bad sequences (not in our training data)

	Φ(x, NN TO VB) GOOD	**Φ**(x, MD TO VB) BAD
x_i=will ^ y_i = NN	1	0
y_{i-1}=START ^ y_i = NN	1	0
x_i=will ^ y_i = MD	0	1
y_{i-1}=START ^ y_i = MD	0	1
…		
y_{i-1}=NN ^ y_i = TO	1	0
y_{i-1}=MD ^ y_i = TO	0	1
x_i=to ^ y_i = TO	1	1
x_i=fight ^ y_i = VB	1	1
y_{i-1}=TO ^ y_i = VB	1	1

these are the different (and so are potentially predictive of a good label sequence)

these are the same (and so are not)

Conditional random fields

$$P(y \mid x, \beta) = \frac{\exp(\Phi(x,y)^\top \beta)}{\sum_{y' \in \mathcal{Y}} \exp(\Phi(x,y')^\top \beta)}$$

- In MEMMs, we normalize over the set of 45 POS tags
- CRFs are globally normalized, but the normalization complexity is huge — every possible sequence of labels of length n.

Forward algorithm (CRF)

$$P(y \mid x, \beta) = \frac{\exp(\Phi(x,y)^\top \beta)}{\sum_{y' \in \mathcal{Y}} \exp(\Phi(x,y')^\top \beta)}$$

- Calculating the denominator naively would involve a summation over K^N terms
- But we can do this efficiently in NK^2 time using the forward algorithm

For details, see: Collins, "The Forward-Backward Algorithm"

Forward algorithm (CRF)

END							
DT							
NNP							
VB							
NN							
MD							
START							
	^	Janet	will	back	the	bill	$

$$\alpha(1, y) = \exp\left(\phi(x, i, y, \text{START})^\top \beta\right)$$

$$\alpha(i, y) = \sum_{y' \in \mathcal{S}} \alpha(i-1, y') \times \exp\left(\phi(x, i, y, y')^\top \beta\right)$$

Forward algorithm (CRF)

END							
DT							
NNP							
VB							
NN							
MD							
START							
	^	Janet	will	back	the	bill	$

$$Z = \sum_{y' \in \mathcal{Y}} \exp\left(\Phi(x, y')^\top \beta\right) = \sum_{s \in \mathcal{S}} \alpha(n, s)$$

Conditional random fields

With a CRF, we have exactly the same parameters as we do with an equivalent MEMM; but we learn the best values of those parameters that leads to the best probability of the sequence overall (in our training data)

MEMM

	TO
x_i=to ^ y_i = TO	10000000
y_{i-1}=NN ^ y_i = TO	0
y_{i-1}=MD ^ y_i = TO	0

CRF

	TO
x_i=to ^ y_i = TO	7.8
y_{i-1}=NN ^ y_i = TO	1.4
y_{i-1}=MD ^ y_i = TO	-5.8

Parameter estimation

- Just like logistic regression/MEMM, we can find the optimal values for the parameters using stochastic gradient descent for a given sequence x and true labels y.

$$\frac{\partial L}{\partial \beta_k} = \Phi_k(x, y) - \sum_{y' \in \mathcal{Y}} P(y' \mid x, \beta)\Phi_k(x, y')$$

Features for the true label y

Expected feature counts under the probability for a sequence assigned by current β

Parameter estimation

- Just like logistic regression/MEMM, we can find the optimal values for the parameters using stochastic gradient descent for a given sequence x and true labels y.

$$\frac{\partial L}{\partial \beta_k} = \Phi_k(x, y) - \sum_{y' \in \mathcal{Y}} P(y' \mid x, \beta)\Phi_k(x, y')$$

If current model assigns probability of 1 to true sequence and 0 to all other K^N sequences, then gradient = 0

Parameter estimation

$$\sum_{y' \in \mathcal{Y}} P(y' \mid x, \beta) \Phi_k(x, y')$$

$$= \sum_{i=1}^{n} \sum_{a \in \mathcal{S}, b \in \mathcal{S}} \phi_k(x, i, a, b) \sum_{y' \in \mathcal{Y} : y'_{i-1} = a, y'_i = b} P(y' \mid x, \beta)$$

This entire sum can be found in NK^2 time using the forward-backward algorithm

For details, see: Collins, "The Forward-Backward Algorithm"

Viterbi Decoding

$$v_1(y) = \exp\left(\phi(x, 1, START, y)^\top \beta\right)$$

$$v_t(y) = \max_{u \in \mathcal{S}} [v_{t-1}(u) \times \exp\left(\phi(x, t, y, u)^\top \beta\right)]$$

(equivalently)

$$v_1(y) = \phi(x, 1, START, y)^\top \beta$$

$$v_t(y) = \max_{u \in \mathcal{S}} [v_{t-1}(u) + \phi(x, t, y, u)^\top \beta]$$

In practice

- CRF training is slow! NK^2 complexity for each sequence at each gradient step.
- Accuracy is typically better than MEMMs