Language Technologies Institute

Carnegie Mellon University

Algorithms for NLP

CS 11-711 · Fall 2020

Lecture 8: Viterbi, discriminative sequence labeling, NER

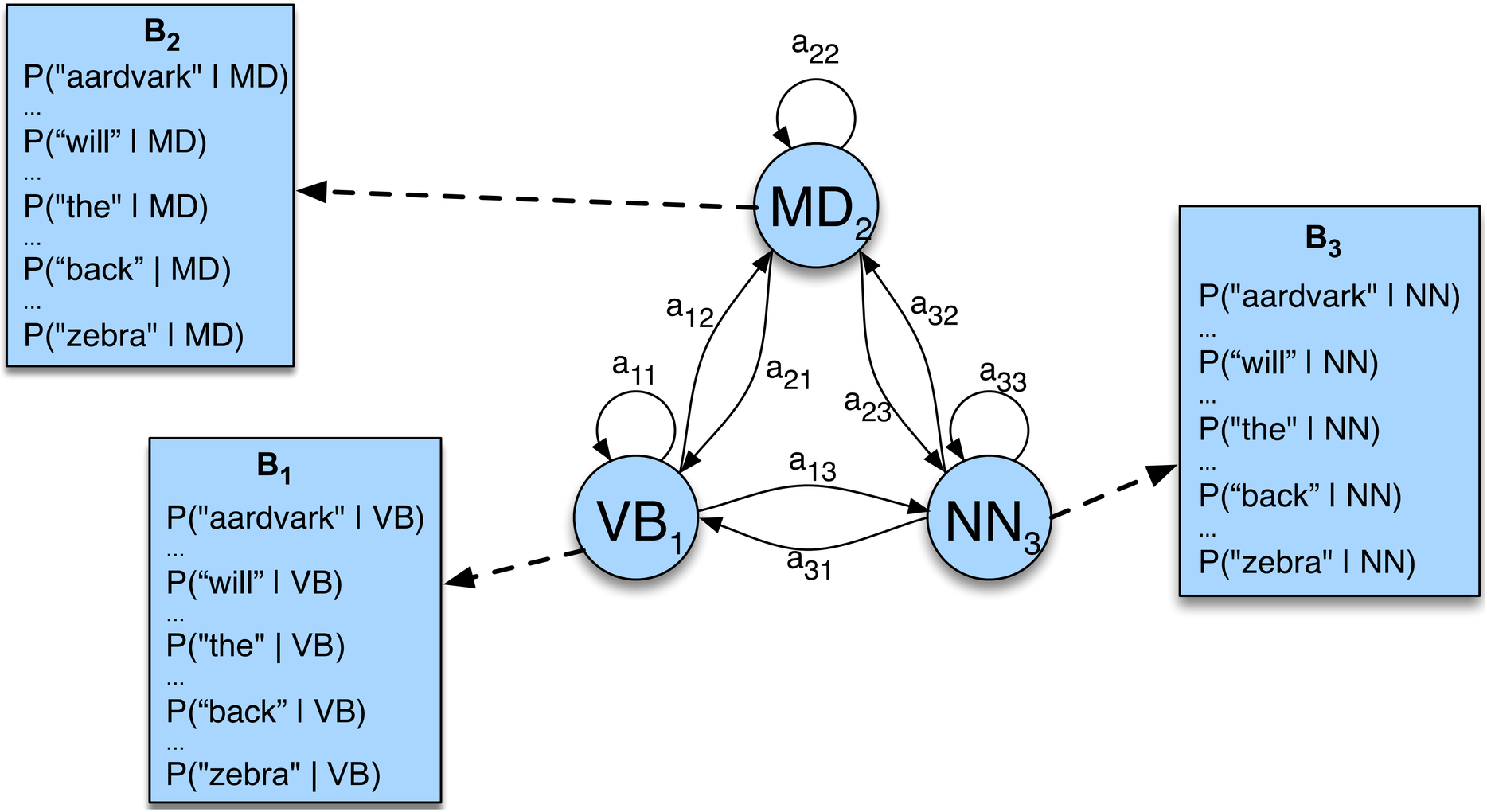
Emma Strubell

Announcements

- Project 1 is due tomorrow! You may submit up to 3 days late (out of a budget of 5 total for the semester).
- No recitation tomorrow (Friday). Do your homework.

Recap

Hidden Markov models (HMMs)



Recap

Hidden Markov models (HMMs)

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state q_i
$\pi = \pi_1, \pi_2, ..., \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

Recap

Hidden Markov models (HMMs)

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state q_i
$\pi = \pi_1, \pi_2, ..., \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

Recap

Hidden Markov models (HMMs)

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, \ldots, v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state q_i
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

Recap

Hidden Markov models (HMMs)

Forward

Viterbi

Forward-backward; Baum-Welch

Problem 1 (Likelihood):	Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O\|\lambda)$.
Problem 2 (Decoding):	Given an observation sequence O and an HMM $\lambda = (A, B)$, discover the best hidden state sequence Q.
Problem 3 (Learning):	Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

Recap

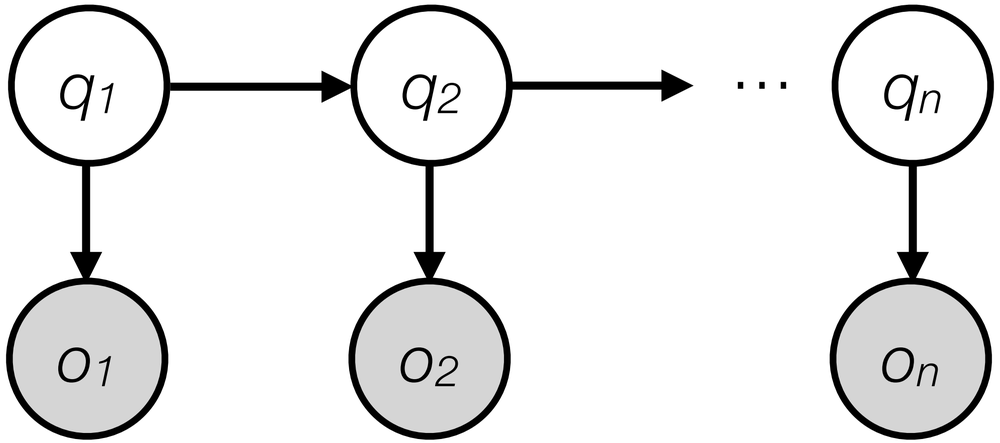
Hidden Markov models (HMMs)

Forward	**Problem 1 (Likelihood):**	Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O\|\lambda)$.
Viterbi	**Problem 2 (Decoding):**	Given an observation sequence O and an HMM $\lambda = (A, B)$, discover the best hidden state sequence Q.
Forward-backward; Baum-Welch	**Problem 3 (Learning):**	Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

HMM tagging as decoding

- **Decoding**: Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, \ldots, o_n$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_n$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n \mid w_1^n)$$



HMM tagging as decoding

- **Decoding**: Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, \ldots, o_n$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_n$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n \mid w_1^n)$$

$$= \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n \mid t_1^n) P(t_1^n)}{P(w_1^n)}$$

HMM tagging as decoding

- **Decoding**: Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, \ldots, o_n$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_n$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n \mid w_1^n)$$

$$= \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n \mid t_1^n) P(t_1^n)}{P(w_1^n)}$$

$$= \underset{t_1^n}{\operatorname{argmax}} P(w_1^n \mid t_1^n) P(t_1^n)$$

HMM tagging as decoding

- **Decoding**: Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, \ldots, o_n$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_n$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n \mid w_1^n)$$

$$= \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n \mid t_1^n) P(t_1^n)}{P(w_1^n)}$$

$$= \underset{t_1^n}{\operatorname{argmax}} P(w_1^n \mid t_1^n) P(t_1^n)$$

simplifying assumptions:

HMM tagging as decoding

- **Decoding**: Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, \ldots, o_n$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_n$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n \mid w_1^n)$$

$$= \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n \mid t_1^n) P(t_1^n)}{P(w_1^n)}$$

$$= \underset{t_1^n}{\operatorname{argmax}} P(w_1^n \mid t_1^n) P(t_1^n)$$

simplifying assumptions:

$$P(w_1^n \mid t_1^n) \approx \prod_{i=1}^{n} P(w_i \mid t_i)$$

HMM tagging as decoding

- **Decoding**: Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, \ldots, o_n$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_n$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n \mid w_1^n)$$

$$= \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n \mid t_1^n) P(t_1^n)}{P(w_1^n)}$$

$$= \underset{t_1^n}{\operatorname{argmax}} P(w_1^n \mid t_1^n) P(t_1^n)$$

simplifying assumptions:

$$P(w_1^n \mid t_1^n) \approx \prod_{i=1}^{n} P(w_i \mid t_i)$$

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i \mid t_{i-1})$$

HMM tagging as decoding

- **Decoding**: Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, \ldots, o_n$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_n$

$$\hat{t}_1^n = \underset{t_1^n}{\text{argmax}}\, P(t_1^n \mid w_1^n) \approx \underset{t_1^n}{\text{argmax}} \prod_{i=1}^{n} \underbrace{P(w_i \mid t_i)}_{\text{emission, } B} \underbrace{P(t_i \mid t_{i-1})}_{\text{transition, } A}$$

HMM tagging as decoding

- **Decoding**: Given as input an HMM $\lambda = (A, B)$ and sequence of observations $O = o_1, o_2, \ldots, o_n$, find the most probable sequence of states $Q = q_1, q_2, \ldots, q_n$

emission, B transition, A

$$\hat{t}_1^n = \underset{t_1^n}{\text{argmax}} P(t_1^n \mid w_1^n) \approx \underset{t_1^n}{\text{argmax}} \prod_{i=1}^{n} P(w_i \mid t_i) P(t_i \mid t_{i-1})$$

How many possible choices?

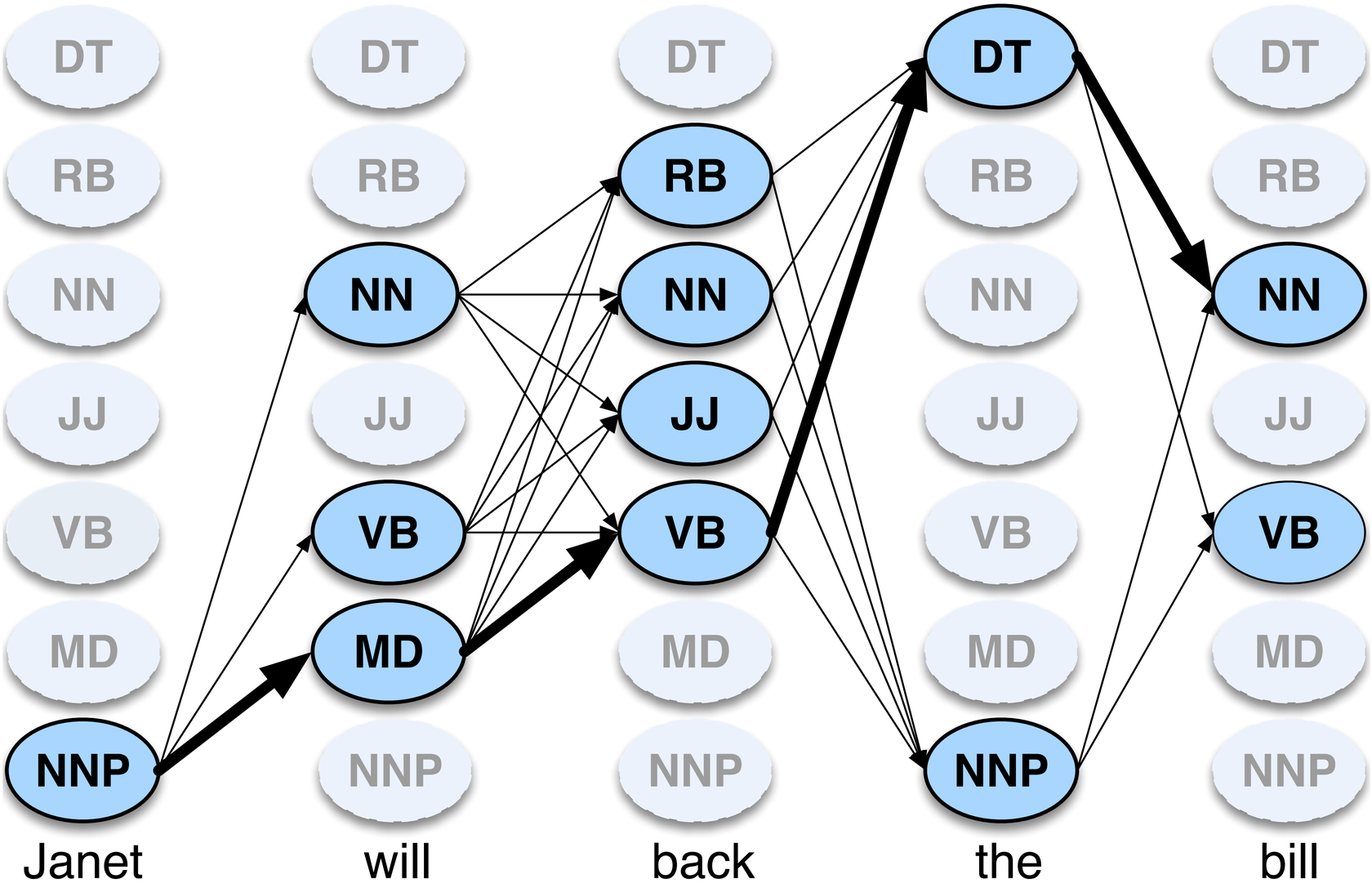
Part-of-speech tagging example

	I	suspect	the	present	forecast	is	pessimistic	.
noun	•	•	•	•	•	•		
adj.		•		•	•		•	
adv.				•				
verb		•		•	•	•		
num.	•							
det.			•					
punc.								•

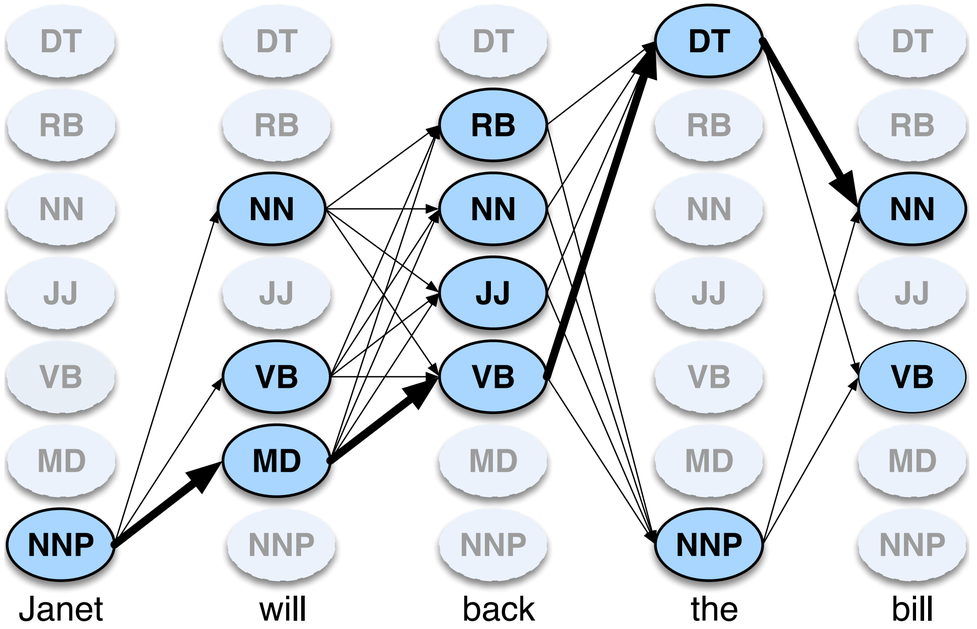
With this very simple tag set, $7^8 = 5.7$ million labelings.
(Even restricting to the possibilities above, 288 labelings.)

Slide credit: Noah Smith

The Viterbi algorithm

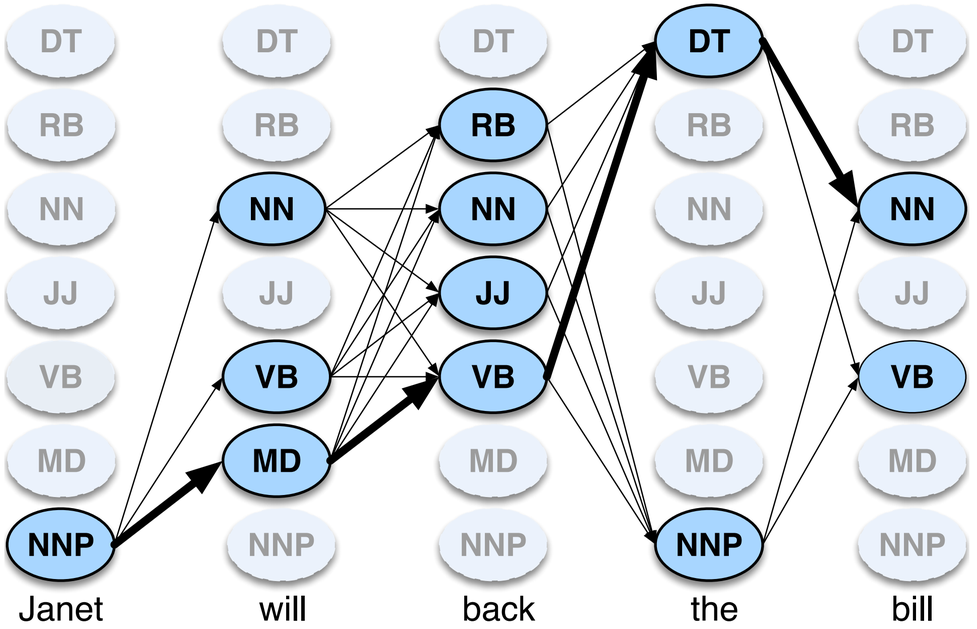


The Viterbi algorithm



$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$$

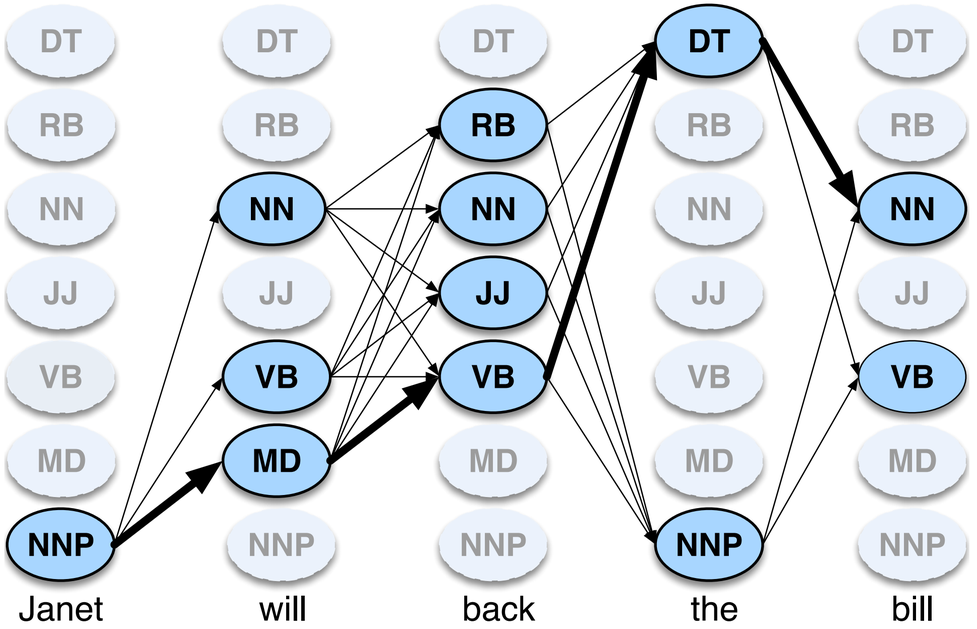
The Viterbi algorithm



$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

previous
Viterbi path
probability

The Viterbi algorithm

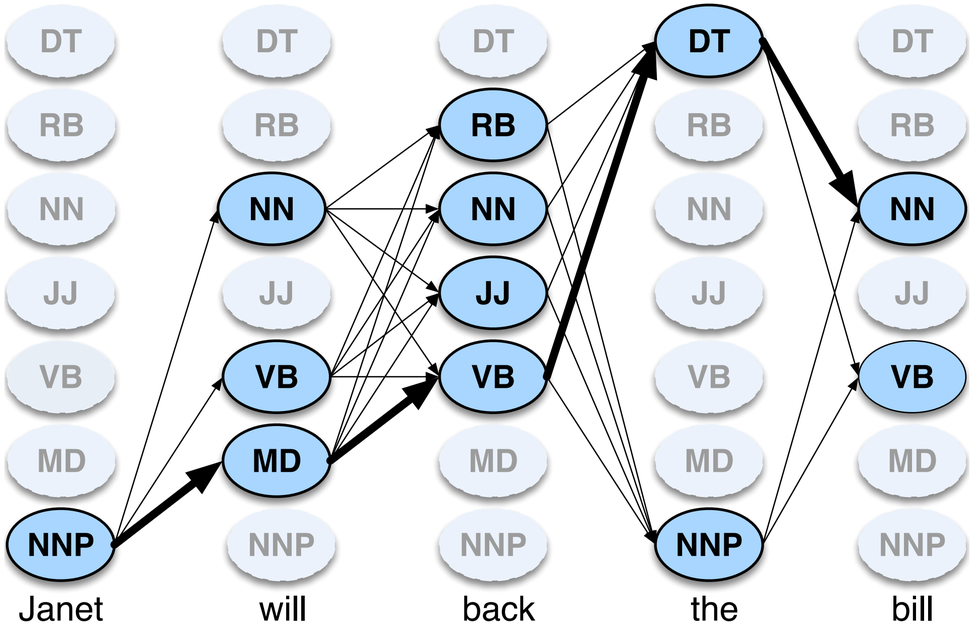


transition probability

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

previous Viterbi path probability

The Viterbi algorithm



transition probability

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

state observation likelihood

previous Viterbi path probability

The Viterbi algorithm

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

create a path probability matrix *viterbi[N,T]*
for each state s **from** 1 **to** N **do**
    $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
    backpointer[s,1] $\leftarrow 0$
for each time step t **from** 2 **to** T **do**
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$
bestpath $\leftarrow$ the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

The Viterbi algorithm

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

create a path probability matrix *viterbi[N,T]*
for each state *s* **from** 1 **to** *N* **do**
    $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
    backpointer[s,1] $\leftarrow 0$

initialization

for each time step *t* **from** 2 **to** *T* **do**
  for each state *s* **from** 1 **to** *N* **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$

$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$

$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$

bestpath $\leftarrow$ the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

The Viterbi algorithm

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

create a path probability matrix *viterbi[N,T]*
for each state s **from** 1 **to** N **do**
    $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
    backpointer[s,1] $\leftarrow 0$
for each time step t **from** 2 **to** T **do**
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$
bestpath $\leftarrow$ the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

initialization

recursion

The Viterbi algorithm

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

create a path probability matrix *viterbi[N,T]*
for each state s **from** 1 **to** N **do**
    $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
    backpointer[s,1] $\leftarrow 0$
for each time step t **from** 2 **to** T **do**
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$ ← $v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$

initialization

recursion

$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$

$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$

bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

The Viterbi algorithm

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

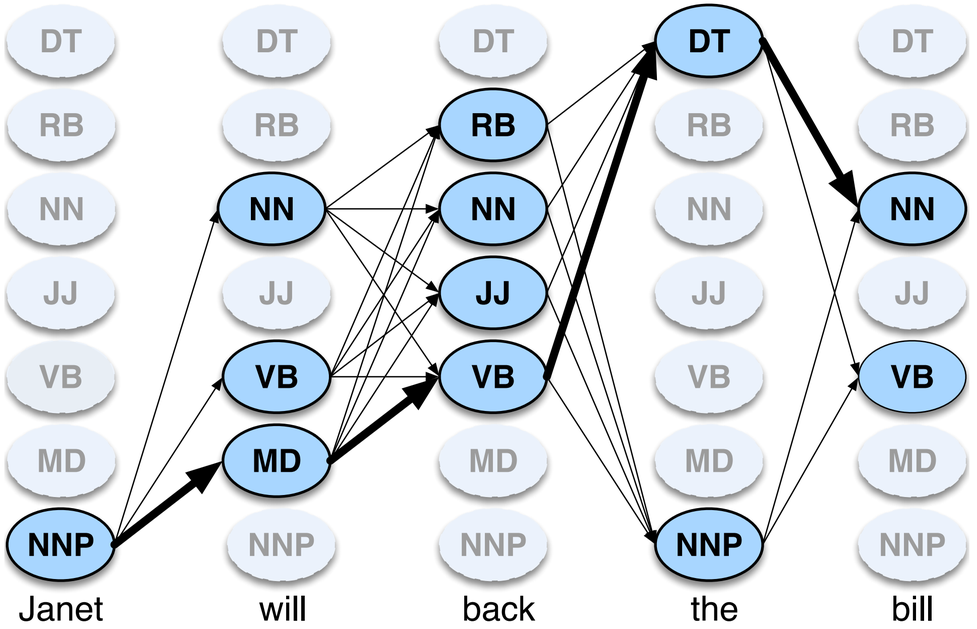
create a path probability matrix *viterbi[N,T]*
for each state s **from** 1 **to** N **do**
    $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
    backpointer[s,1] $\leftarrow 0$
for each time step t **from** 2 **to** T **do**
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$ ← $v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$
bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

initialization
recursion
termination

The Viterbi algorithm

	NNP	MD	VB	JJ	NN	RB	DT
<*s*>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

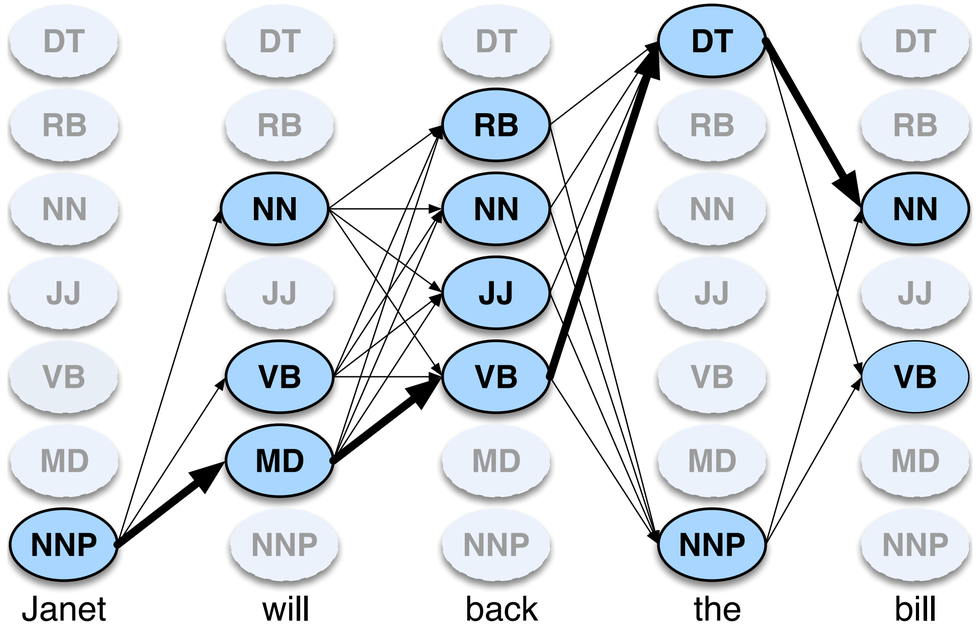


$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

The Viterbi algorithm

	NNP	MD	VB	JJ	NN	RB	DT
<*s*>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

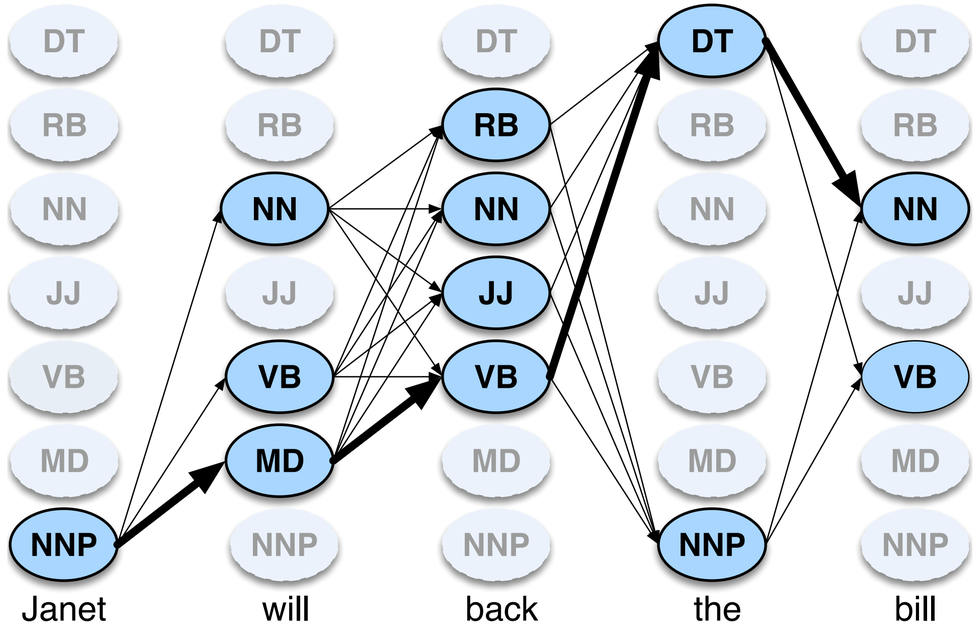


$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

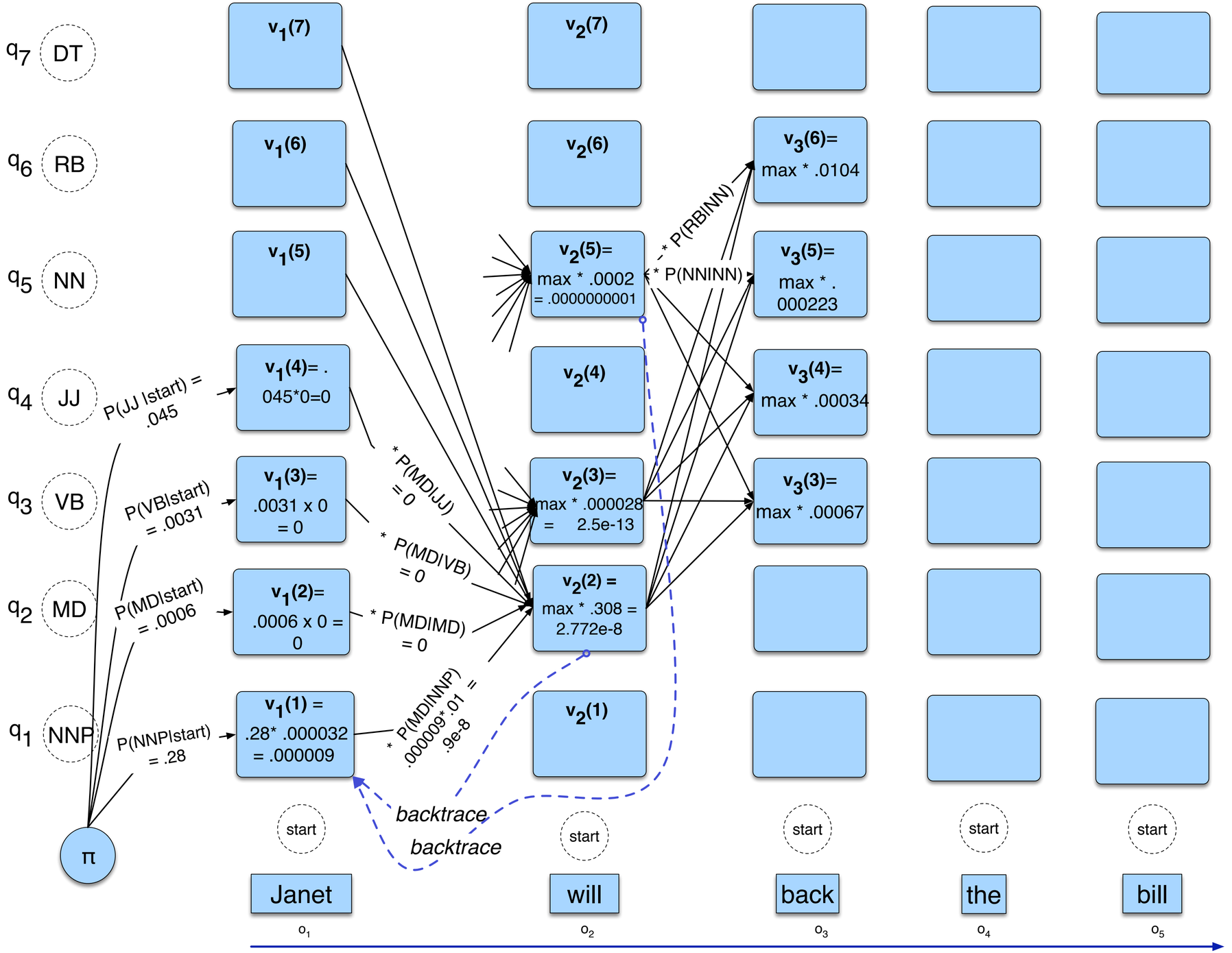
The Viterbi algorithm

	NNP	MD	VB	JJ	NN	RB	DT
<*s*>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0



$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$


q7 DT
q6 RB
q5 NN
q4 JJ
q3 VB
q2 MD
q1 NNP
π
v1(7)
v1(6)
v1(5)
v1(4)= . 045*0=0
v1(3)= .0031 x 0 = 0
v1(2)= .0006 x 0 = 0
v1(1) = .28* .000032 = .000009
P(JJ |start) = .045
P(VB|start) = .0031
P(MD|start) = .0006
P(NNP|start) = .28
* P(MD|JJ) = 0
* P(MD|VB) = 0
* P(MD|MD) = 0
* P(MD|NNP) .000009*.01 = .9e-8
v2(7)
v2(6)
v2(5)= max * .0002 = .0000000001
v2(4)
v2(3)= max * .000028 = 2.5e-13
v2(2) = max * .308 = 2.772e-8
v2(1)
* P(RB|NN)
* P(NN|NN)
v3(6)= max * .0104
v3(5)= max * . 000223
v3(4)= max * .00034
v3(3)= max * .00067
backtrace
backtrace
start
start
start
start
start
Janet
will
back
the
bill
o1
o2
o3
o4
o5

The Viterbi algorithm

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

create a path probability matrix *viterbi[N,T]*
for each state s **from** 1 **to** N **do**
    $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
    backpointer[s,1] $\leftarrow 0$
for each time step t **from** 2 **to** T **do**
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$
bestpath $\leftarrow$ the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

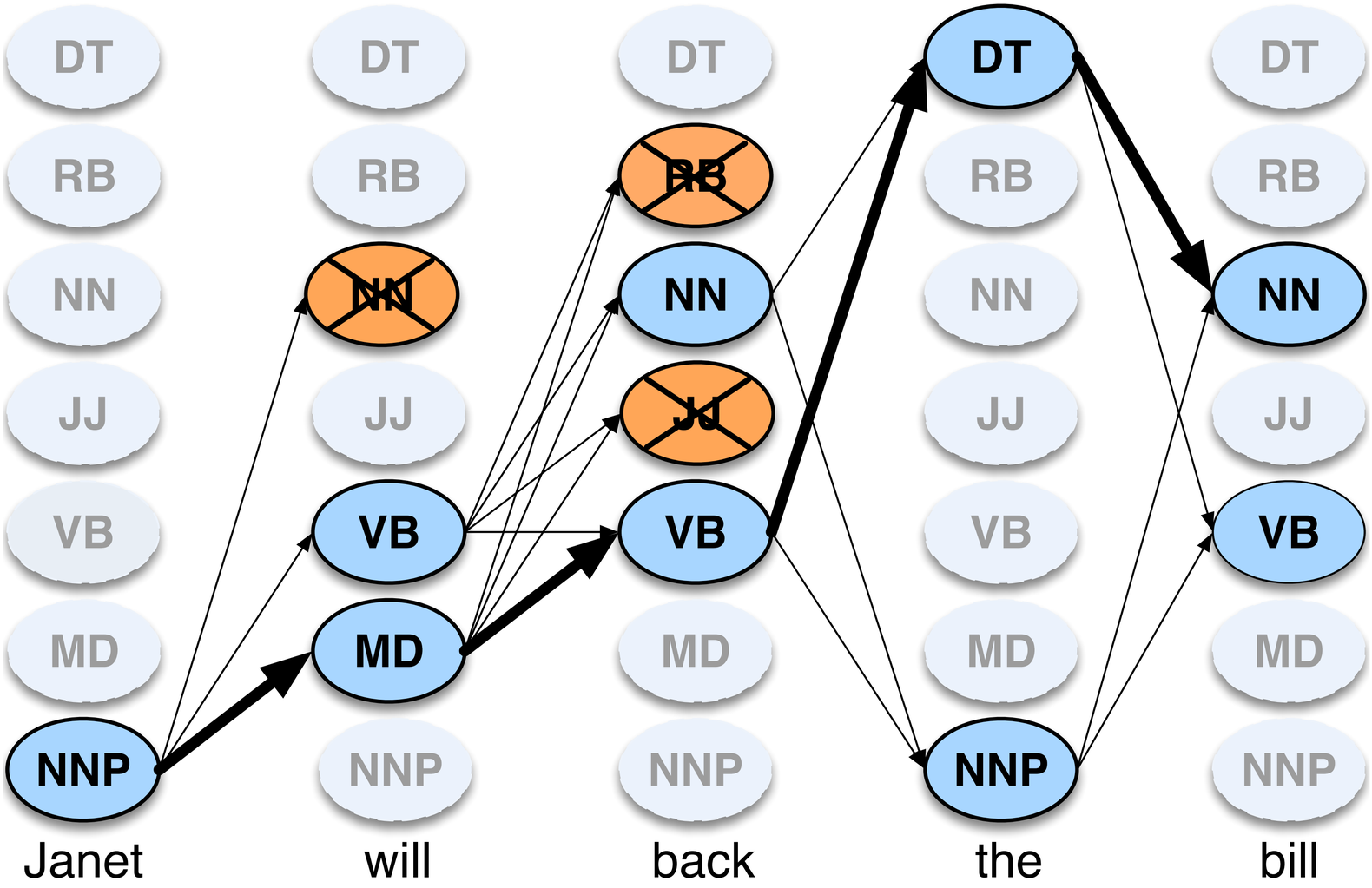
The Viterbi algorithm

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

create a path probability matrix *viterbi[N,T]*
for each state s **from** 1 **to** N **do**
    $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
    backpointer[s,1] $\leftarrow 0$
for each time step t **from** 2 **to** T **do**
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$
bestpath $\leftarrow$ the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

Computational complexity in N and T?

Beam search


DT
RB
NN
JJ
VB
MD
NNP
DT
RB
NN
JJ
VB
MD
NNP
DT
RB
NN
JJ
VB
MD
NNP
DT
RB
NN
JJ
VB
MD
NNP
DT
RB
NN
JJ
VB
MD
NNP
Janet
will
back
the
bill

Hidden Markov models

Forward	**Problem 1 (Likelihood):**	Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O\|\lambda)$.
Viterbi	**Problem 2 (Decoding):**	Given an observation sequence O and an HMM $\lambda = (A, B)$, discover the best hidden state sequence Q.
Forward-backward; Baum-Welch	**Problem 3 (Learning):**	Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

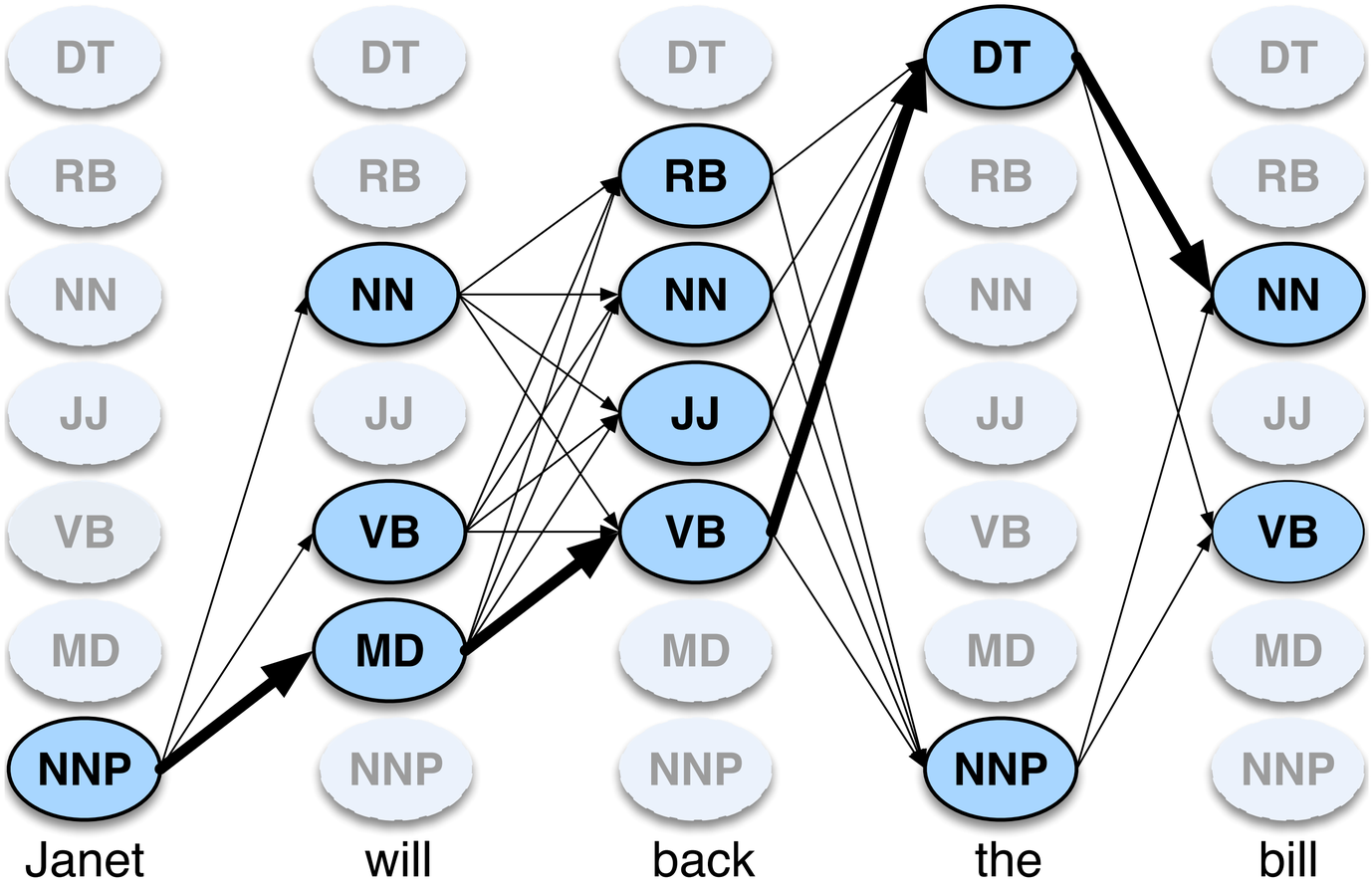
Hidden Markov models

Forward	**Problem 1 (Likelihood):**	Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O\|\lambda)$.
Viterbi	**Problem 2 (Decoding):**	Given an observation sequence O and an HMM $\lambda = (A, B)$, discover the best hidden state sequence Q.
Forward-backward; Baum-Welch	**Problem 3 (Learning):**	Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

The forward algorithm

- Just sum instead of max!

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t)$$



Problems with HMMs

Problems with HMMs

- HMMs have a lot in common with Naive Bayes classifiers, n-gram LMs:

Problems with HMMs

- HMMs have a lot in common with Naive Bayes classifiers, n-gram LMs:
  - Need smoothing to improve generalization

Problems with HMMs

- HMMs have a lot in common with Naive Bayes classifiers, n-gram LMs:
  - Need smoothing to improve generalization
  - How to handle unknown words?

Problems with HMMs

- HMMs have a lot in common with Naive Bayes classifiers, n-gram LMs:
  - Need smoothing to improve generalization
  - How to handle unknown words?
  - Would like to easily add arbitrary features that might help model probabilities.

Problems with HMMs

- HMMs have a lot in common with Naive Bayes classifiers, n-gram LMs:
  - Need smoothing to improve generalization
  - How to handle unknown words?
  - Would like to easily add arbitrary features that might help model probabilities.
- Previously we solved some of these problems by training a **discriminative model** like logistic regression to *predict* rather than *count*.

Problems with HMMs

- HMMs have a lot in common with Naive Bayes classifiers, n-gram LMs:
  - Need smoothing to improve generalization
  - How to handle unknown words?
  - Would like to easily add arbitrary features that might help model probabilities.
- Previously we solved some of these problems by training a **discriminative model** like logistic regression to *predict* rather than *count*.
- How to use logistic regression for sequence labeling?

Maximum entropy Markov models (MEMMs)

- Simply predict the label for each word using logistic regression, using the label assigned to the previous word as a feature (along with any other useful features)!

$$\hat{T} = \operatorname*{argmax}_{T} \prod_{i} P(t_i \mid w_{i-j}^{i+j}, t_{i-k}^{i-1}))) \quad P(t_i \mid w_i, t_{i-1}) = \frac{\exp(\theta \cdot f(t_i, w_{i-j}^{i+j}, t_{i-k}^{i-1}))}{\sum_{t' \in \mathcal{Y}} \exp(\theta \cdot f(t', w_{i-j}^{i+j}, t_{i-k}^{i-1}))}$$

Maximum entropy Markov models (MEMMs)

- Simply predict the label for each word using logistic regression, using the label assigned to the previous word as a feature (along with any other useful features)!

$$\hat{T} = \operatorname*{argmax}_{T} \prod_{i} P(t_i \mid w_{i-j}^{i+j}, t_{i-k}^{i-1}))) \quad P(t_i \mid w_i, t_{i-1}) = \frac{\exp(\theta \cdot f(t_i, w_{i-j}^{i+j}, t_{i-k}^{i-1}))}{\sum_{t' \in \mathcal{Y}} \exp(\theta \cdot f(t', w_{i-j}^{i+j}, t_{i-k}^{i-1}))}$$

<s> Janet will back the bill

Maximum entropy Markov models (MEMMs)

- Simply predict the label for each word using logistic regression, using the label assigned to the previous word as a feature (along with any other useful features)!

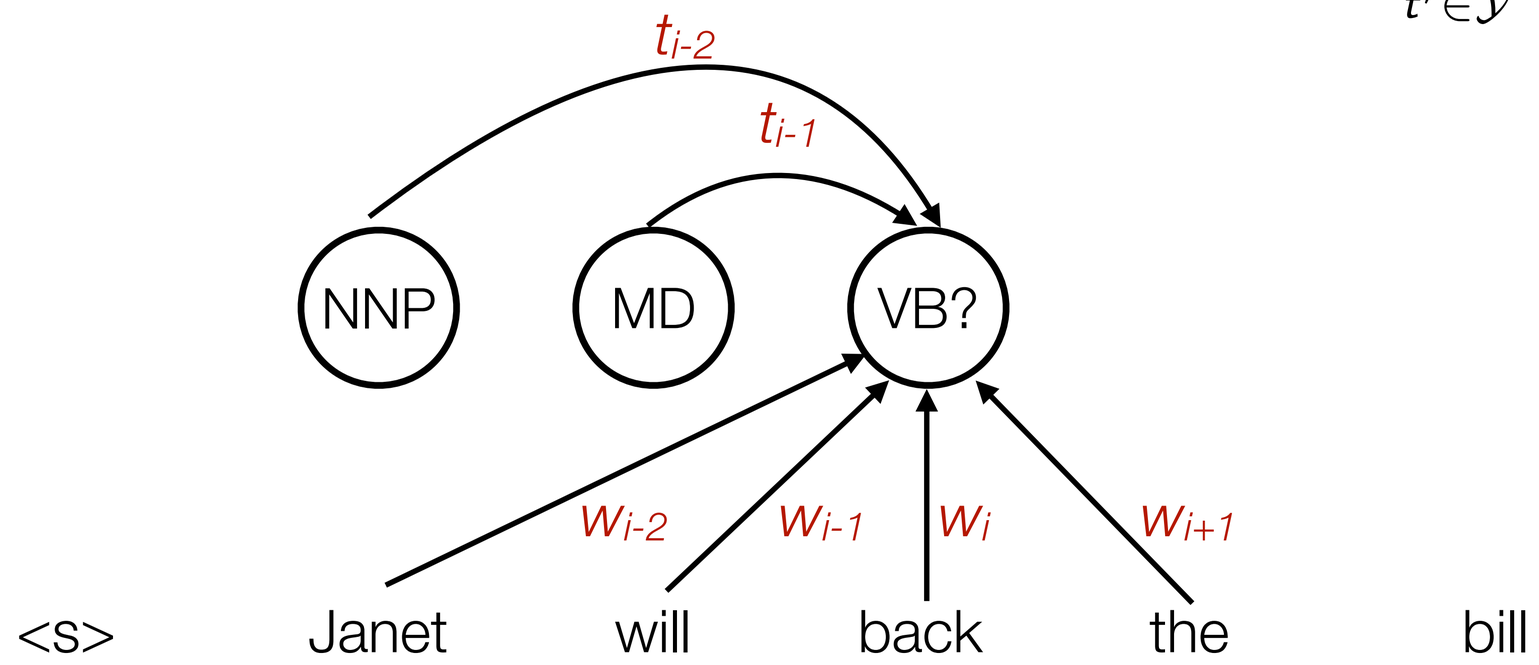
$$\hat{T} = \operatorname*{argmax}_{T} \prod_{i} P(t_i \mid w_{i-j}^{i+j}, t_{i-k}^{i-1}))) \quad P(t_i \mid w_i, t_{i-1}) = \frac{\exp(\theta \cdot f(t_i, w_{i-j}^{i+j}, t_{i-k}^{i-1}))}{\sum_{t' \in \mathcal{Y}} \exp(\theta \cdot f(t', w_{i-j}^{i+j}, t_{i-k}^{i-1}))}$$

<s> Janet will back the bill

Maximum entropy Markov models (MEMMs)

- Simply predict the label for each word using logistic regression, using the label assigned to the previous word as a feature (along with any other useful features)!

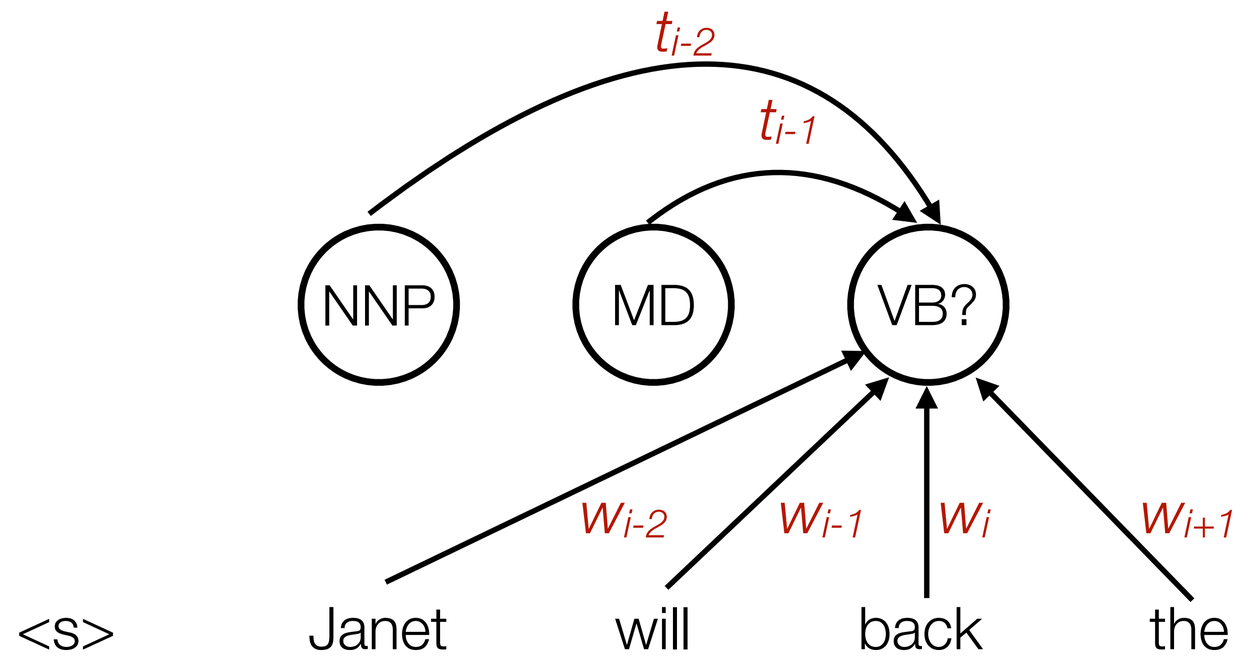
$$\hat{T} = \operatorname*{argmax}_T \prod_i P(t_i \mid w_{i-j}^{i+j}, t_{i-k}^{i-1}))) \quad P(t_i \mid w_i, t_{i-1}) = \frac{\exp(\theta \cdot f(t_i, w_{i-j}^{i+j}, t_{i-k}^{i-1}))}{\sum_{t' \in \mathcal{Y}} \exp(\theta \cdot f(t', w_{i-j}^{i+j}, t_{i-k}^{i-1}))}$$



Maximum entropy Markov models (MEMMs)

- Simply predict the label for each word using logistic regression, using the label assigned to the previous word as a feature (along with any other useful features)!

$$\hat{T} = \underset{T}{\text{argmax}} \prod_i P(t_i \mid w_{i-j}^{i+j}, t_{i-k}^{i-1}))) \quad P(t_i \mid w_i, t_{i-1}) = \frac{\exp(\theta \cdot f(t_i, w_{i-j}^{i+j}, t_{i-k}^{i-1}))}{\sum_{t' \in \mathcal{Y}} \exp(\theta \cdot f(t', w_{i-j}^{i+j}, t_{i-k}^{i-1}))}$$



Lexical	$f_{i\pm\{0,1,2,3\}}$, $(m_{i-2,i-1})$, $(m_{i-1,i})$, $(m_{i-1,i+1})$, $(m_{i,i+1})$, $(m_{i+1,i+2})$, $(m_{i-2,i-1,i})$, $(m_{i-1,i,i+1})$, $(m_{i,i+1,i+2})$, $(m_{i-2,i-1,i+1})$, $(m_{i-1,i+1,i+2})$
POS	$p_{i-\{3,2,1\}}$, $a_{i+\{0,1,2,3\}}$, $(p_{i-2,i-1})$, $(a_{i+1,i+2})$, (p_{i-1}, a_{i+1}), (p_{i-2}, p_{i-1}, a_i), $(p_{i-2}, p_{i-1}, a_{i+1})$, (p_{i-1}, a_i, a_{i+1}), $(p_{i-1}, a_{i+1}, a_{i+2})$
Affix	$c_{:1}$, $c_{:2}$, $c_{:3}$, $c_{n:}$, $c_{n-1:}$, $c_{n-2:}$, $c_{n-3:}$
Binary	initial uppercase, all uppercase/lowercase, contains 1/2+ capital(s) not at the beginning, contains a (period/number/hyphen)

Table from: Choi and Palmer. Fast and Robust Part-of-Speech Tagging using Dynamic Model Selection. ACL 2012.

Part-of-speech tagging features

w_i contains a particular prefix (from all prefixes of length ≤ 4)
w_i contains a particular suffix (from all suffixes of length ≤ 4)
w_i contains a number
w_i contains an upper-case letter
w_i contains a hyphen
w_i is all upper case
w_i's word shape
w_i's short word shape
w_i is upper case and has a digit and a dash (like *CFC-12*)
w_i is upper case and followed within 3 words by Co., Inc., etc.

Part-of-speech tagging features

w_i contains a particular prefix (from all prefixes of length ≤ 4)
w_i contains a particular suffix (from all suffixes of length ≤ 4)
w_i contains a number
w_i contains an upper-case letter
w_i contains a hyphen
w_i is all upper case
w_i's word shape
w_i's short word shape
w_i is upper case and has a digit and a dash (like *CFC-12*)
w_i is upper case and followed within 3 words by Co., Inc., etc.

Features for NER

- Example extracted features for the token *L'occitane*

prefix(w_i) = `L`	suffix(w_i) = `tane`
prefix(w_i) = `L'`	suffix(w_i) = `ane`
prefix(w_i) = `L'O`	suffix(w_i) = `ne`
prefix(w_i) = `L'Oc`	suffix(w_i) = `e`
word-shape(w_i) = `X'Xxxxxxxx`	short-word-shape(w_i) = `X'Xx`

Features for NER

- Example extracted features for the token *L'occitane*

prefix(w_i) = L	suffix(w_i) = tane
prefix(w_i) = L'	suffix(w_i) = ane
prefix(w_i) = L'O	suffix(w_i) = ne
prefix(w_i) = L'Oc	suffix(w_i) = e
word-shape(w_i) = X'Xxxxxxxx	short-word-shape(w_i) = X'Xx

Label bias problem

Label bias problem

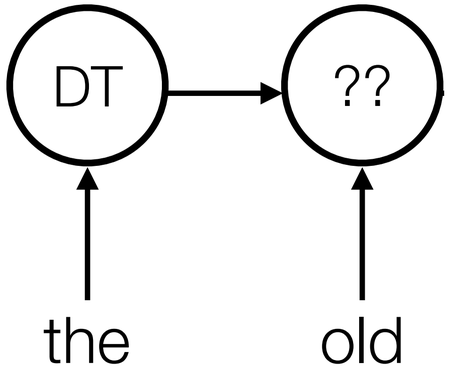
- **Greediness**: Transitions leaving a given state compete only against each other, rather than against all other transitions in the model.

Label bias problem

- **Greediness**: Transitions leaving a given state compete only against each other, rather than against all other transitions in the model.
  - Leads to locally optimal decisions that are not globally optimal.

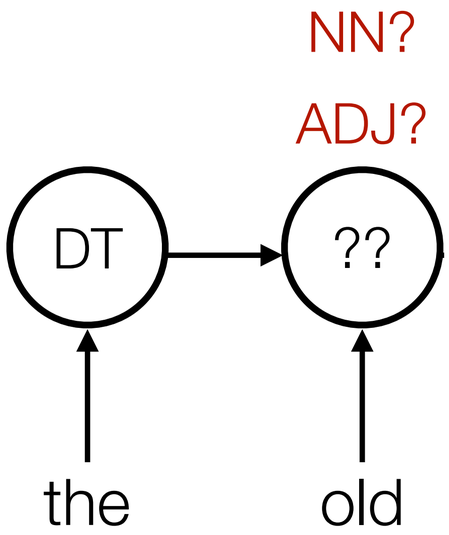
Label bias problem

- **Greediness**: Transitions leaving a given state compete only against each other, rather than against all other transitions in the model.
  - Leads to locally optimal decisions that are not globally optimal.



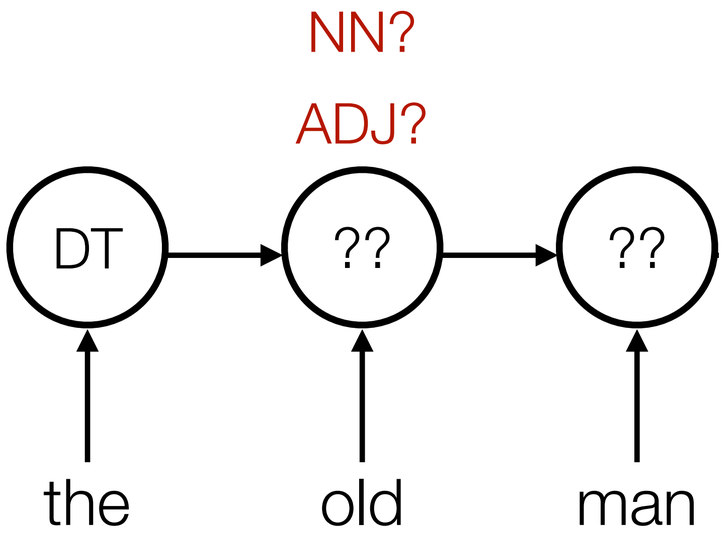
Label bias problem

- **Greediness**: Transitions leaving a given state compete only against each other, rather than against all other transitions in the model.
  - Leads to locally optimal decisions that are not globally optimal.



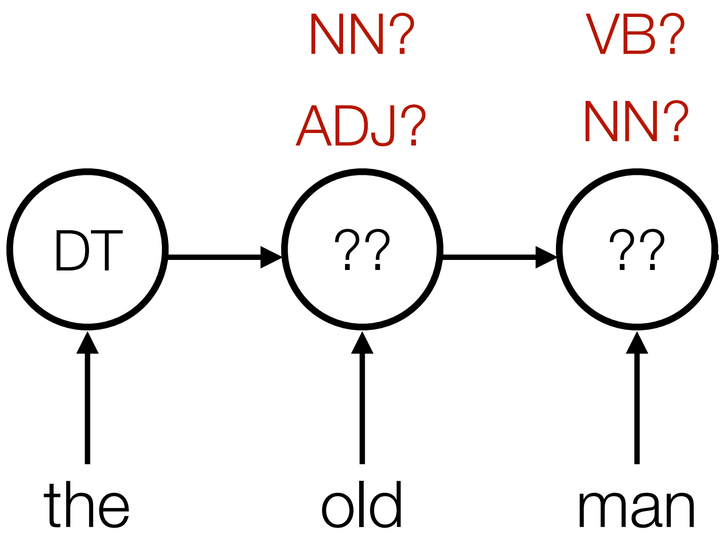
Label bias problem

- **Greediness**: Transitions leaving a given state compete only against each other, rather than against all other transitions in the model.
  - Leads to locally optimal decisions that are not globally optimal.



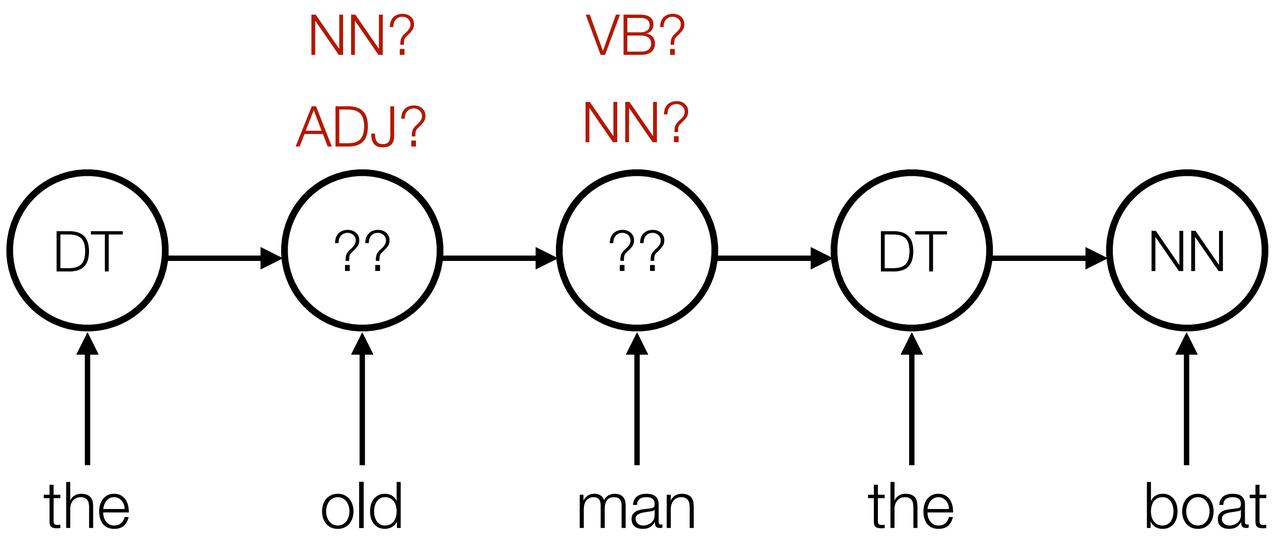
Label bias problem

- **Greediness**: Transitions leaving a given state compete only against each other, rather than against all other transitions in the model.
  - Leads to locally optimal decisions that are not globally optimal.



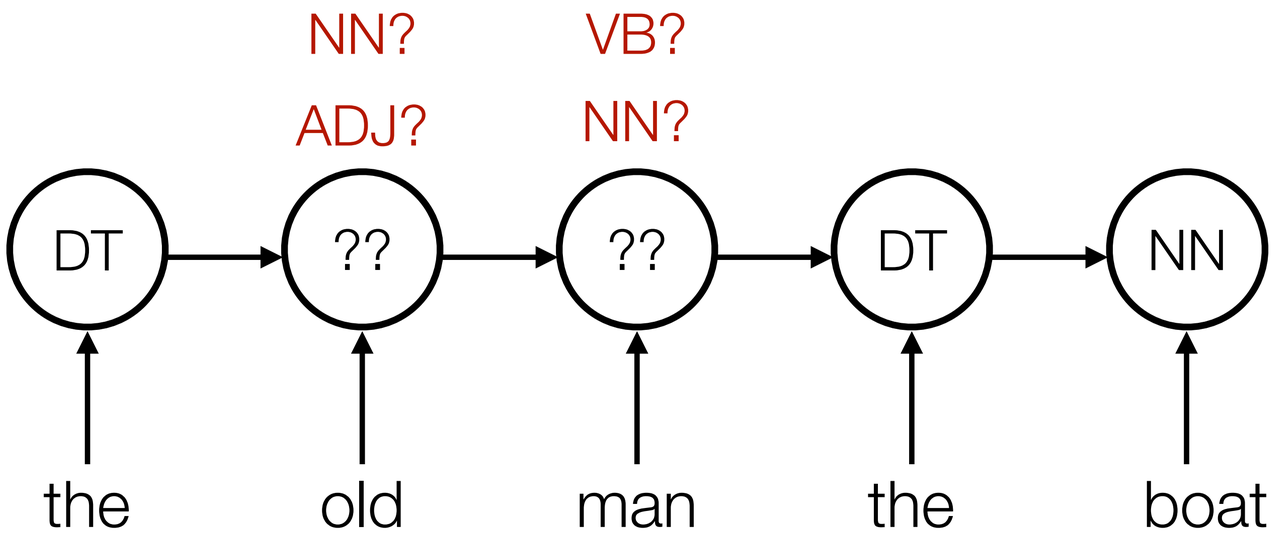
Label bias problem

- **Greediness**: Transitions leaving a given state compete only against each other, rather than against all other transitions in the model.
  - Leads to locally optimal decisions that are not globally optimal.



Label bias problem

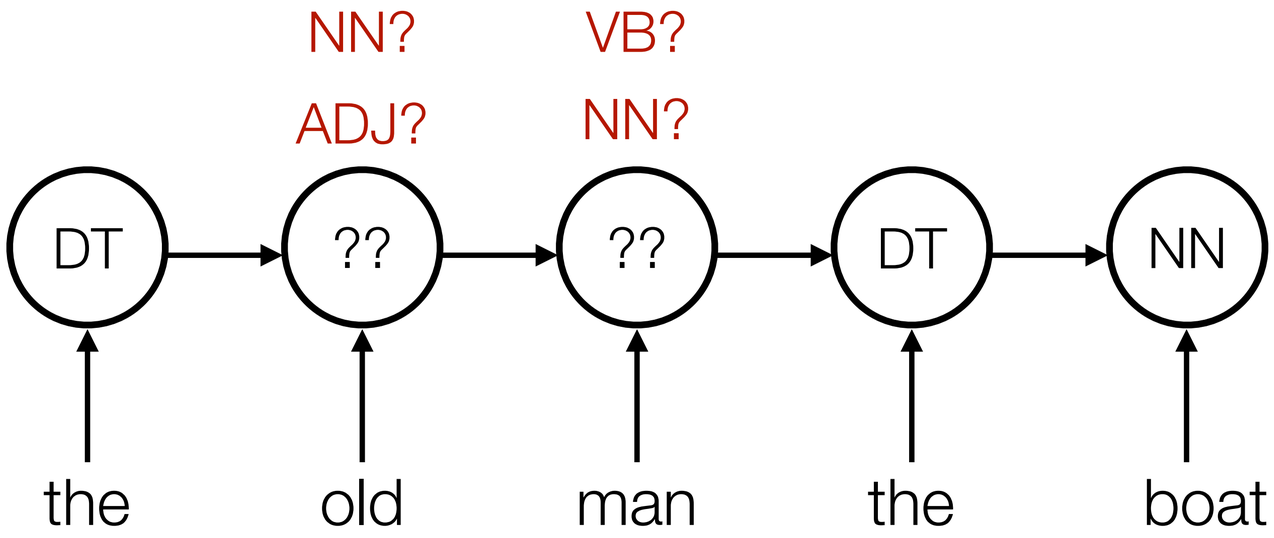
- **Greediness**: Transitions leaving a given state compete only against each other, rather than against all other transitions in the model.
  - Leads to locally optimal decisions that are not globally optimal.



Are HMMs subject to the label bias problem?

Label bias problem

- **Greediness**: Transitions leaving a given state compete only against each other, rather than against all other transitions in the model.
  - Leads to locally optimal decisions that are not globally optimal.
- All **locally normalized** models (vs. **globally normalized**) suffer from this problem.



Are HMMs subject to the label bias problem?

Conditional random fields (CRFs)

Conditional random fields (CRFs)

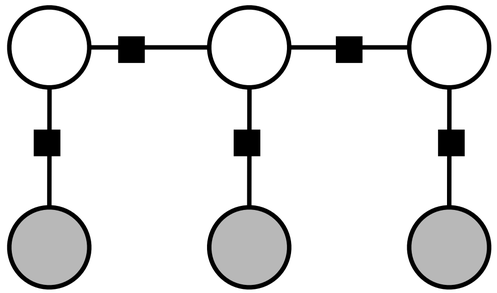
- **CRFs**: Globally-normalized discriminative sequence labeling models!

Conditional random fields (CRFs)

- **CRFs**: Globally-normalized discriminative sequence labeling models!
  - A family of graphical models where the probability of a configuration of variables is proportional to a product of scores across pairs (or cliques) of variables (suitable for **structured prediction**, not just sequence labeling).

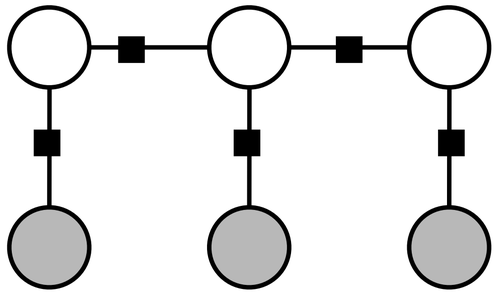
Conditional random fields (CRFs)

- **CRFs**: Globally-normalized discriminative sequence labeling models!
  - A family of graphical models where the probability of a configuration of variables is proportional to a product of scores across pairs (or cliques) of variables (suitable for **structured prediction**, not just sequence labeling).
  - In NLP, usually we mean a **linear-chain CRF** where pairs of variables are labels for adjacent tokens.



Conditional random fields (CRFs)

- **CRFs**: Globally-normalized discriminative sequence labeling models!
  - A family of graphical models where the probability of a configuration of variables is proportional to a product of scores across pairs (or cliques) of variables (suitable for **structured prediction**, not just sequence labeling).
  - In NLP, usually we mean a **linear-chain CRF** where pairs of variables are labels for adjacent tokens.



  - Semi-Markov CRFs [Sarawagi and Cohen 2004] are also useful for sequence labeling in NLP: feature functions, normalization over possible segments.

Conditional random fields (CRFs)

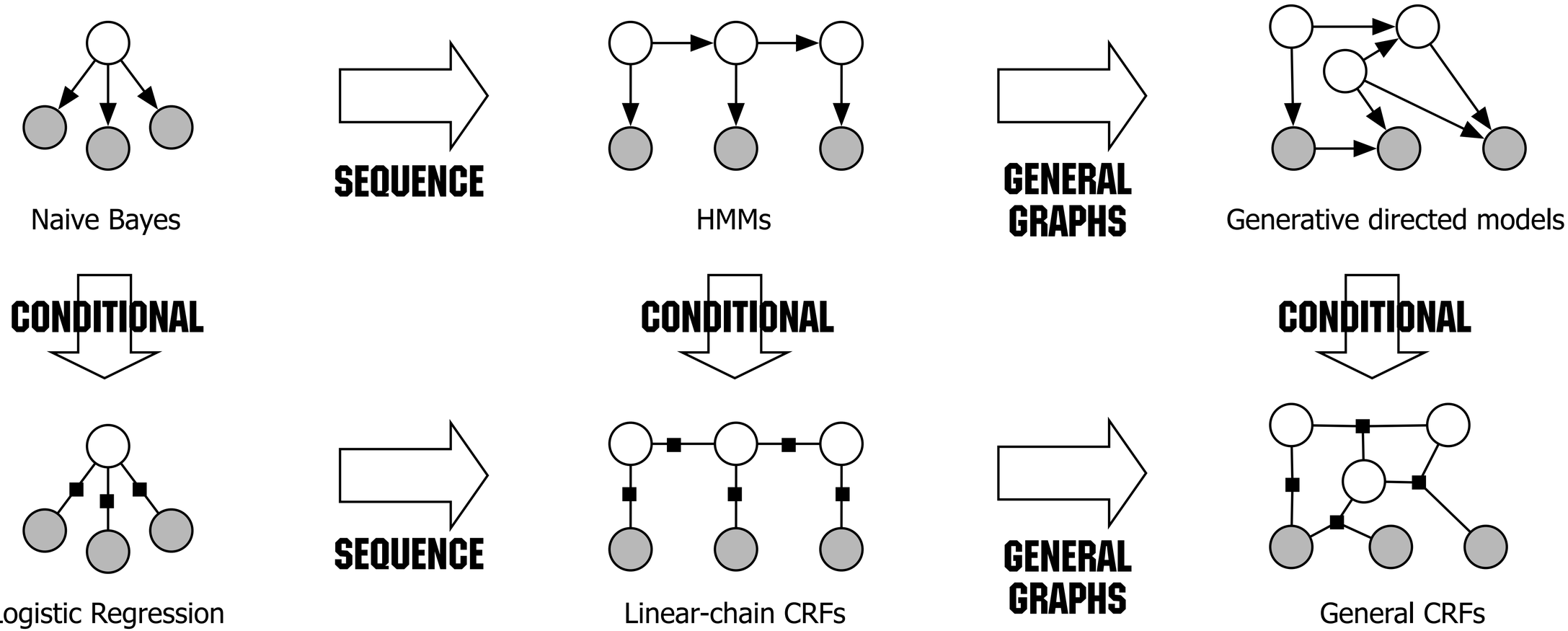


Image from: Sutton and McCallum. An Introduction to Conditional Random Fields for Relational Learning. In *Introduction to Statistical Relational Learning*. 2012.

Conditional random fields (CRFs)

- **CRFs**: Globally-normalized discriminative sequence labeling models!

$$P(\mathbf{y} \mid \mathbf{w}) = \frac{\exp(\Psi(\mathbf{w}, \mathbf{y}))}{\sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{w})} \exp(\Psi(\mathbf{w}, \mathbf{y}'))} \qquad \Psi(\mathbf{w}, \mathbf{y}) = \sum_{m=1}^{M+1} \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m)$$

Conditional random fields (CRFs)

- **CRFs**: Globally-normalized discriminative sequence labeling models!

$$P(\mathbf{y} \mid \mathbf{w}) = \frac{\exp(\Psi(\mathbf{w}, \mathbf{y}))}{\sum_{\mathbf{y'} \in \mathcal{Y}(\mathbf{w})} \exp(\Psi(\mathbf{w}, \mathbf{y'}))}$$

$$\Psi(\mathbf{w}, \mathbf{y}) = \sum_{m=1}^{M+1} \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m)$$

y' is an entire sequence

Conditional random fields (CRFs)

- **CRFs**: Globally-normalized discriminative sequence labeling models!

$$P(\mathbf{y} \mid \mathbf{w}) = \frac{\exp(\Psi(\mathbf{w}, \mathbf{y}))}{\sum_{\mathbf{y'} \in \mathcal{Y}(\mathbf{w})} \exp(\Psi(\mathbf{w}, \mathbf{y'}))}$$

y' is an entire sequence

$$\Psi(\mathbf{w}, \mathbf{y}) = \sum_{m=1}^{M+1} \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m)$$

logistic regression,
plus weights over pairs of labels

Conditional random fields (CRFs)

- **CRFs**: Globally-normalized discriminative sequence labeling models!

$$P(\mathbf{y} \mid \mathbf{w}) = \frac{\exp(\Psi(\mathbf{w}, \mathbf{y}))}{\sum_{\mathbf{y'} \in \mathcal{Y}(\mathbf{w})} \exp(\Psi(\mathbf{w}, \mathbf{y'}))}$$

y' is an entire sequence

$$\Psi(\mathbf{w}, \mathbf{y}) = \sum_{m=1}^{M+1} \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m)$$

logistic regression,
plus weights over pairs of labels

- Decoding: Viterbi algorithm.

Conditional random fields (CRFs)

- **CRFs**: Globally-normalized discriminative sequence labeling models!

$$P(\mathbf{y} \mid \mathbf{w}) = \frac{\exp(\Psi(\mathbf{w}, \mathbf{y}))}{\sum_{\mathbf{y'} \in \mathcal{Y}(\mathbf{w})} \exp(\Psi(\mathbf{w}, \mathbf{y'}))}$$

y' is an entire sequence

$$\Psi(\mathbf{w}, \mathbf{y}) = \sum_{m=1}^{M+1} \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m)$$

logistic regression,
plus weights over pairs of labels

- Decoding: Viterbi algorithm.
- Likelihood: Forward algorithm.

Conditional random fields (CRFs)

- **CRFs**: Globally-normalized discriminative sequence labeling models!

$$P(\mathbf{y} \mid \mathbf{w}) = \frac{\exp(\Psi(\mathbf{w}, \mathbf{y}))}{\sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{w})} \exp(\Psi(\mathbf{w}, \mathbf{y}'))}$$

y' is an entire sequence

$$\Psi(\mathbf{w}, \mathbf{y}) = \sum_{m=1}^{M+1} \boldsymbol{\theta} \cdot \mathbf{f}(\mathbf{w}, y_m, y_{m-1}, m)$$

logistic regression,
plus weights over pairs of labels

- Decoding: Viterbi algorithm.
- Likelihood: Forward algorithm.
- Training: Forward-Backward (get backward for free w/ autodiff!)

Sequence labeling in practice: Named entity recognition

Named entity recognition (NER)

Citing high fuel prices, [ORG **United Airlines**] said [TIME **Friday**] it has increased fares by [MONEY **$6**] per round trip on flights to some cities also served by lower-cost carriers. [ORG **American Airlines**], a unit of [ORG **AMR Corp.**], immediately matched the move, spokesman [PER **Tim Wagner**] said. [ORG **United**], a unit of [ORG **UAL Corp.**], said the increase took effect [TIME **Thursday**] and applies to most routes where it competes against discount carriers, such as [LOC **Chicago**] to [LOC **Dallas**] and [LOC **Denver**] to [LOC **San Francisco**].

Named entity recognition: typical tags

Type	Tag	Sample Categories	Example sentences
People	PER	people, characters	**Turing** is a giant of computer science.
Organization	ORG	companies, sports teams	The **IPCC** warned about the cyclone.
Location	LOC	regions, mountains, seas	The **Mt. Sanitas** loop is in **Sunshine Canyon**.
Geo-Political Entity	GPE	countries, states, provinces	**Palo Alto** is raising the fees for parking.
Facility	FAC	bridges, buildings, airports	Consider the **Golden Gate Bridge**.
Vehicles	VEH	planes, trains, automobiles	It was a classic **Ford Falcon**.

Named entity recognition (NER)

Since PON1, a known antioxidant, has a peroxidase-like activity, it could prevent oxidative stress in hypothyroidism. However, several reports indicated elevated oxidative stress parameters and decreased PON1 activity in hypothyroid patients. It is well-known that increased level of ROS is related to many cancers including colorectal cancers.

gene

disease

species

chemical

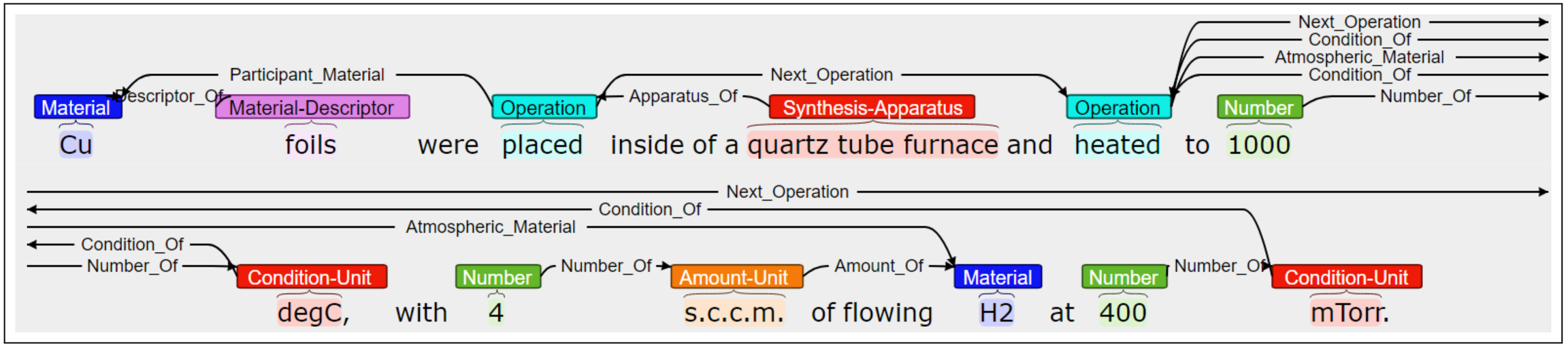
Example from PubTator: https://www.ncbi.nlm.nih.gov/research/pubtator/?view=publication&pmid=32939514

Named entity recognition (NER)

Since PON1, a known antioxidant, has a peroxidase-like activity, it could prevent oxidative stress in hypothyroidism. However, several reports indicated elevated oxidative stress parameters and decreased PON1 activity in hypothyroid patients. It is well-known that increased level of ROS is related to many cancers including colorectal cancers.

gene
disease
species
chemical

Example from PubTator: https://www.ncbi.nlm.nih.gov/research/pubtator/?view=publication&pmid=32939514



Ambiguity in NER

Name	Possible Categories
Washington	Person, Location, Political Entity, Organization, Vehicle
Downing St.	Location, Organization
IRA	Person, Organization, Monetary Instrument
Louis Vuitton	Person, Organization, Commercial Product

[$_{\text{ORG}}$ Washington] went up 2 games to 1 in the four-game series.
Blair arrived in [$_{\text{LOC}}$ Washington] for what may well be his last state visit.
In June, [$_{\text{GPE}}$ Washington] passed a primary seatbelt law.
The [$_{\text{VEH}}$ Washington] had proved to be a leaky ship, every passage I made...

NER as sequence labeling

[$_{ORG}$ **American Airlines**], a unit of [$_{ORG}$ **AMR Corp.**], immediately matched the move, spokesman [$_{PER}$ **Tim Wagner**] said.

(or, BIO)

Words	IOB Label	IO Label
American	B-ORG	I-ORG
Airlines	I-ORG	I-ORG
,	O	O
a	O	O
unit	O	O
of	O	O
AMR	B-ORG	I-ORG
Corp.	I-ORG	I-ORG
,	O	O
immediately	O	O

also, IOBES / BILOU:

Cu	B-Material
foils	L-Material
were	O
placed	U-Operation
inside	O
a	O
quartz	B-Apparatus
tube	I-Apparatus
furnace	L-Apparatus

Features for NER

identity of w_i, identity of neighboring words
embeddings for w_i, embeddings for neighboring words
part of speech of w_i, part of speech of neighboring words
base-phrase syntactic chunk label of w_i and neighboring words
presence of w_i in a **gazetteer**
w_i contains a particular prefix (from all prefixes of length ≤ 4)
w_i contains a particular suffix (from all suffixes of length ≤ 4)
w_i is all upper case
word shape of w_i, word shape of neighboring words
short word shape of w_i, short word shape of neighboring words
presence of hyphen

Features for NER

identity of w_i, identity of neighboring words
embeddings for w_i, embeddings for neighboring words
part of speech of w_i, part of speech of neighboring words
base-phrase syntactic chunk label of w_i and neighboring words
presence of w_i in a **gazetteer**
w_i contains a particular prefix (from all prefixes of length ≤ 4)
w_i contains a particular suffix (from all suffixes of length ≤ 4)
w_i is all upper case
word shape of w_i, word shape of neighboring words
short word shape of w_i, short word shape of neighboring words
presence of hyphen

Evaluating named entity recognition

- NER is typically evaluated using **segment-level F_1** score, meaning at the level of *multi-token entities*, not single words.

American Airlines said Friday …

Evaluating named entity recognition

- NER is typically evaluated using **segment-level F_1** score, meaning at the level of *multi-token entities*, not single words.

American	Airlines	said	Friday	…
B-ORG	O	O	O	

Neural sequence labeling

- Next class: CRFs, neural network models for sequence labeling, and how to combine them.

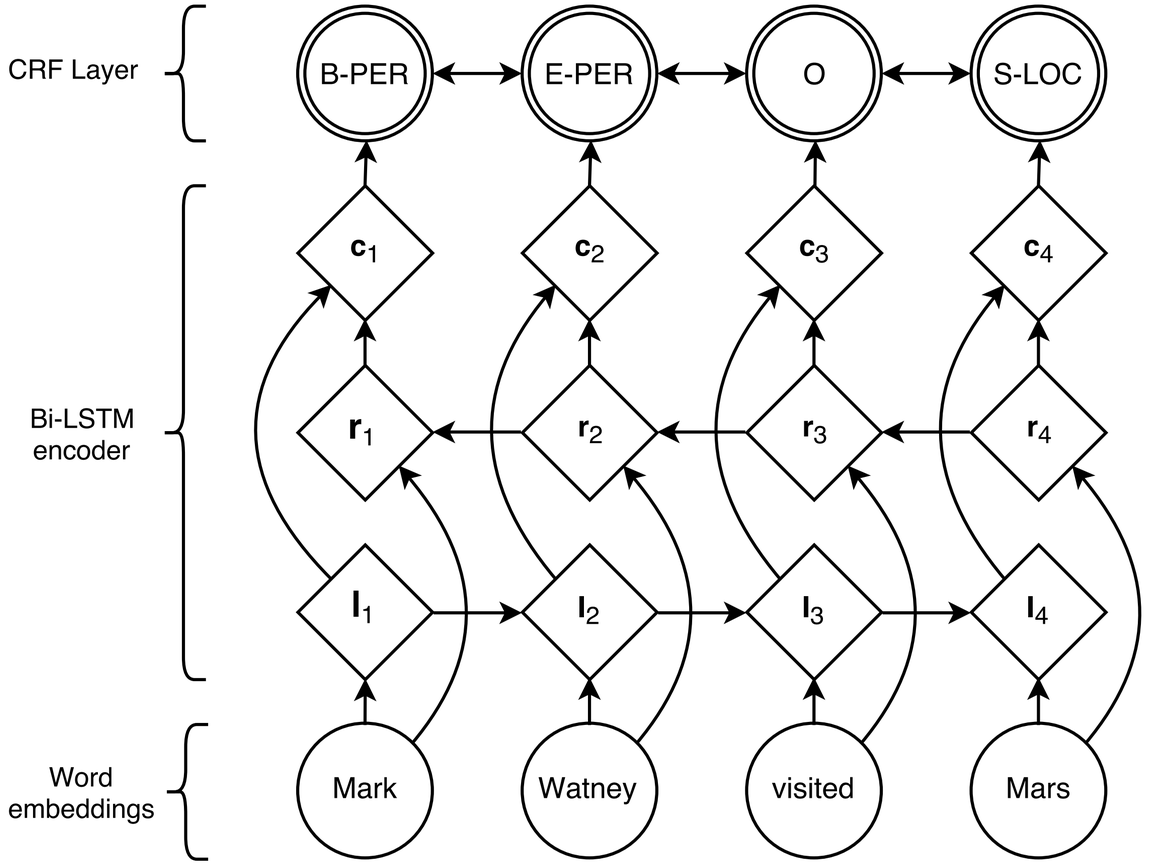


Image from: Lample et al. Neural Architectures for Named Entity Recognition. NAACL 2016.

Announcements

- Project 1 is due tomorrow! You may submit up to 3 days late (out of a budget of 5 total for the semester).
- No recitation tomorrow (Friday). Do your homework.