Natural Language Processing

Sequence labeling

Yulia Tsvetkov

yuliats@cs.washington.edu

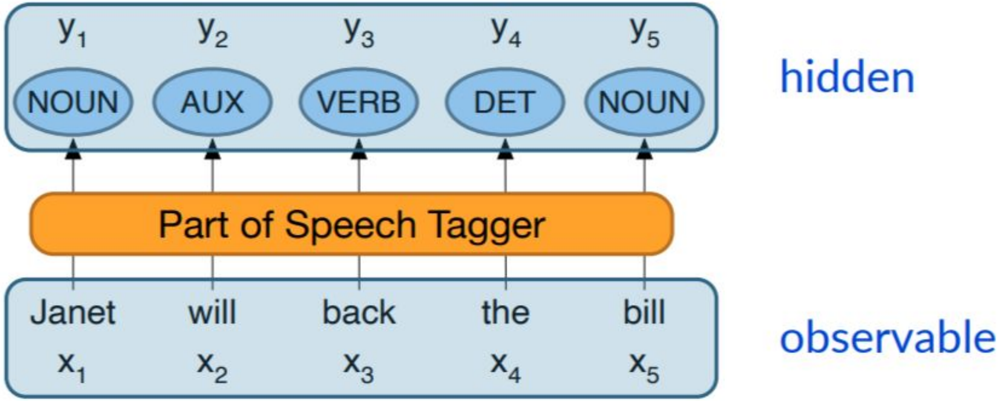
PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

Announcements

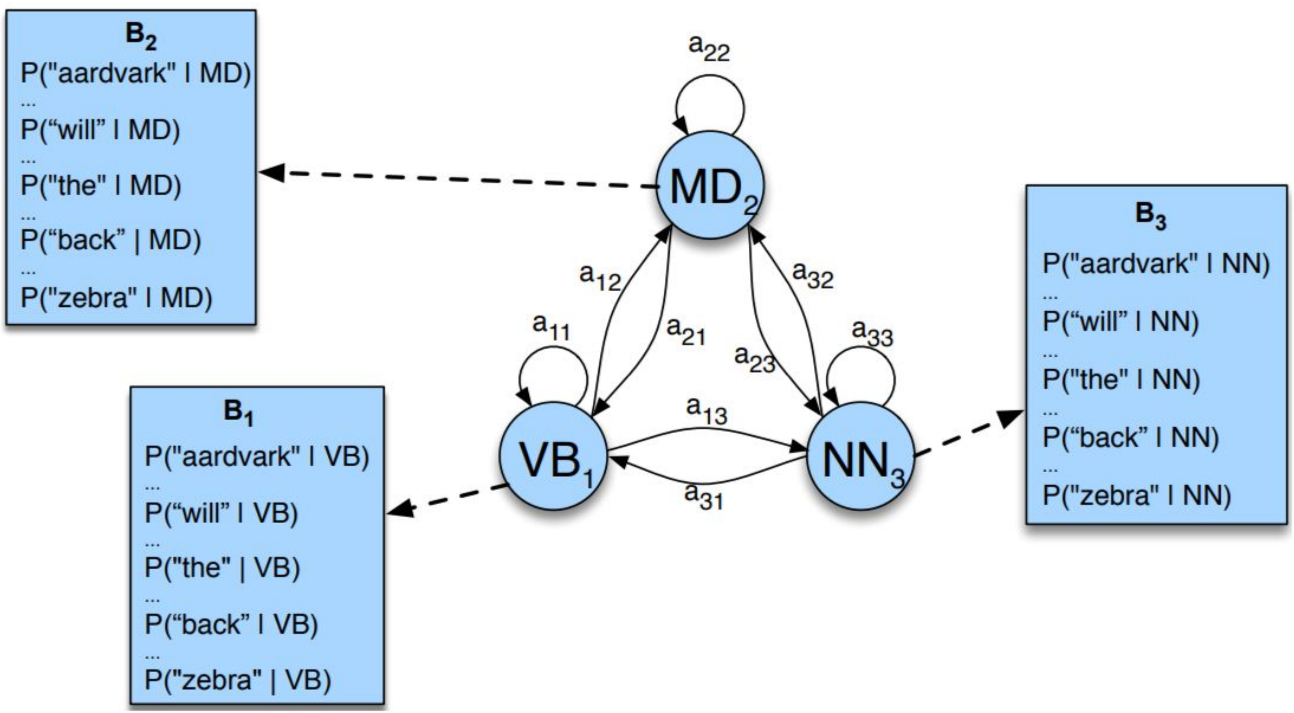
- HW 2 overview
- Wednesday’s lecture - CRFs (please don’t skip, it is closely related to HW2)
- Wednesday’s quiz - not in class. Quiz will be open from 9am to 9pm.
- Friday - Veterans day. No class, no OHs

Hidden Markov Models

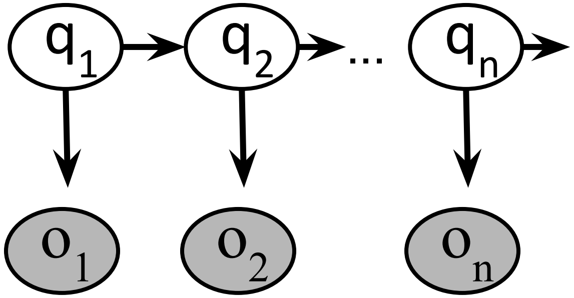
- We use a Markov chain for computing P for a sequence of observable events
- In many cases the events we are interested in are hidden
  - e.g., we don't observe POS tags in a text

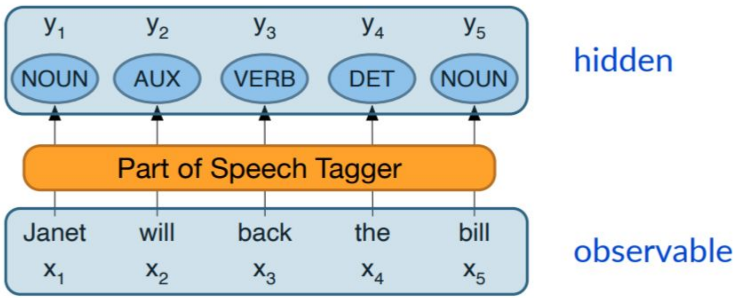


Hidden Markov Models



Hidden Markov Models





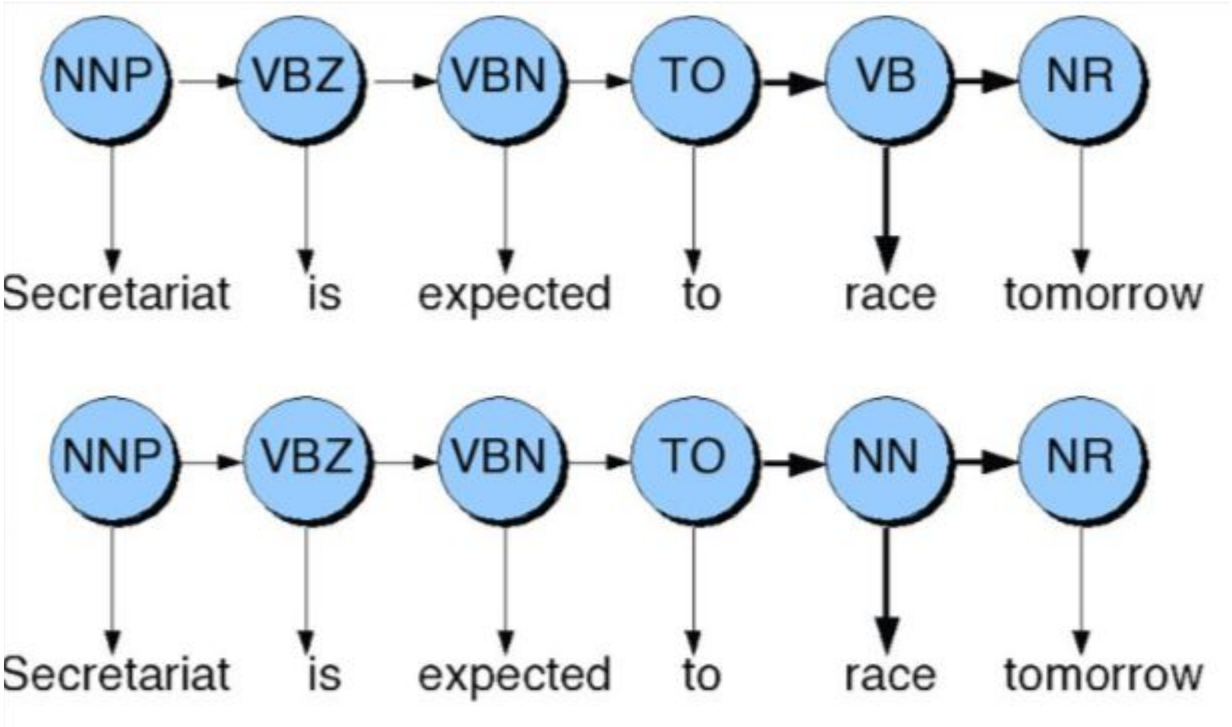
Markov Assumption: $P(q_i|q_1...q_{i-1}) = P(q_i|q_{i-1})$

Output Independence: $P(o_i|q_1 \ldots q_i, \ldots, q_T, o_1, \ldots, o_i, \ldots, o_T) = P(o_i|q_i)$

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, $\sum_{i=1}^{n} \pi_i = 1$

POS tagging with HMMs



HMM parameters

$Q = q_1 q_2 \ldots q_N$	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, $\sum_{i=1}^{n} \pi_i = 1$

HMMs: algorithms

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.

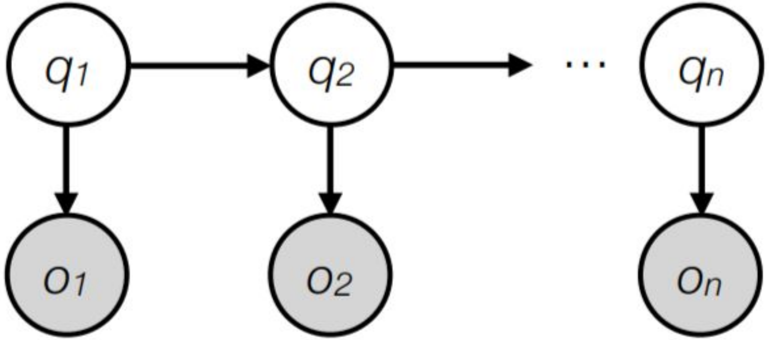
HMM tagging as decoding

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:

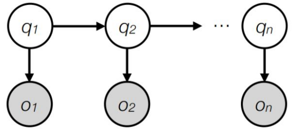
HMM tagging as decoding

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:

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

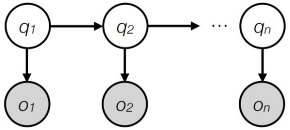
HMM tagging as decoding

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:

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

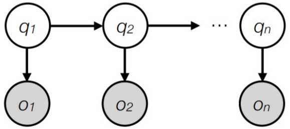
$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i \mid t_{i-1})$$

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 $Q = q_1, q_2, \ldots, q_n$

emission, B transition, A

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



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 $Q = q_1, q_2, \ldots, q_n$

emission, *B* transition, *A*

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

How many possible choices?

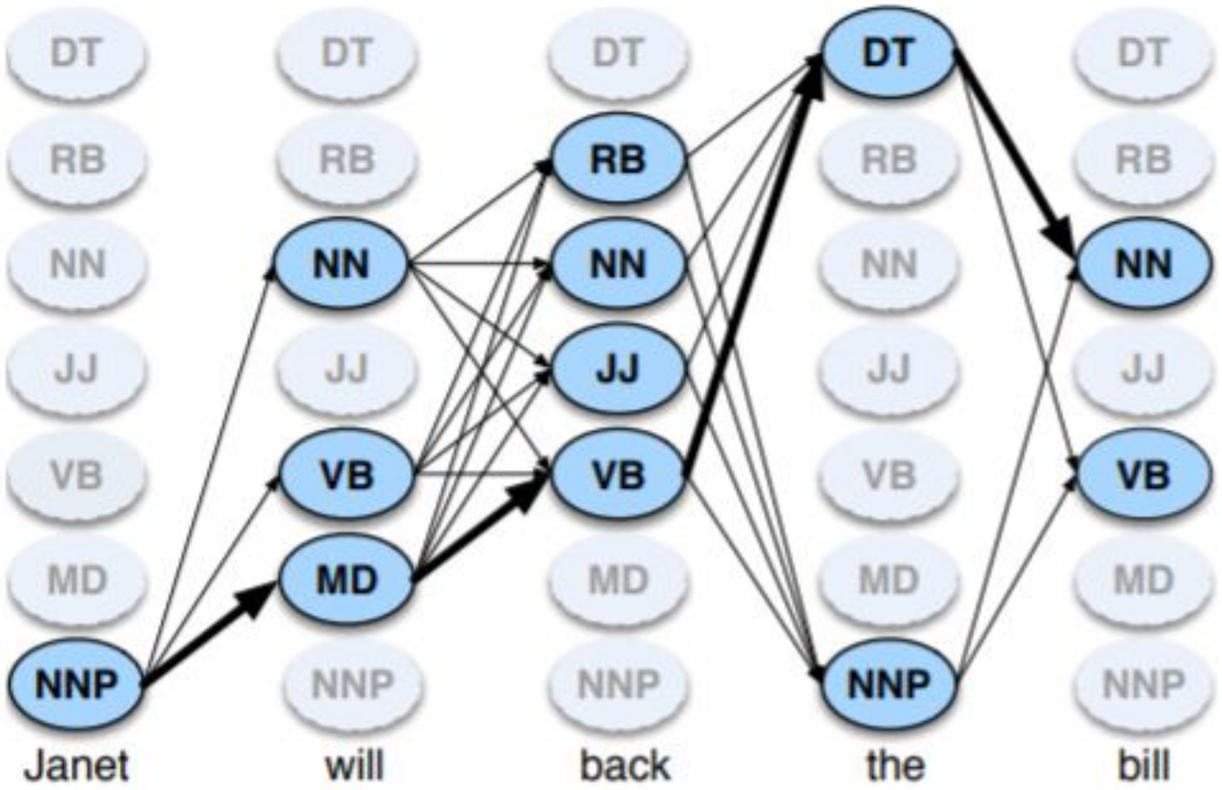
Part of speech tagging example

	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.)

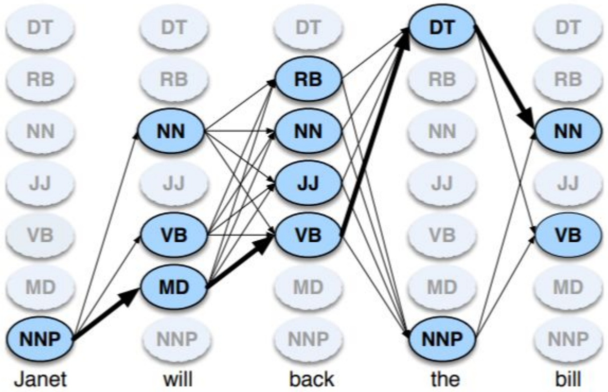
The Viterbi algorithm

The Viterbi algorithm



The Viterbi algorithm

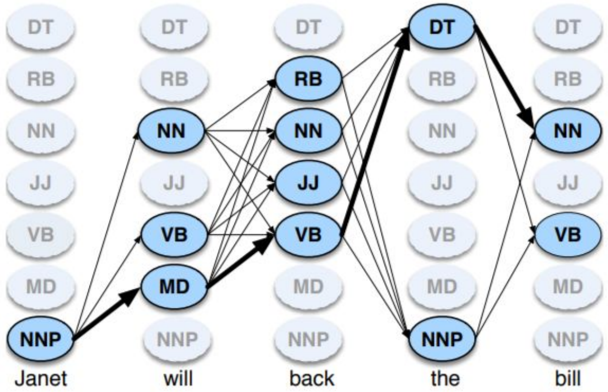
$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j



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

The Viterbi algorithm

$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

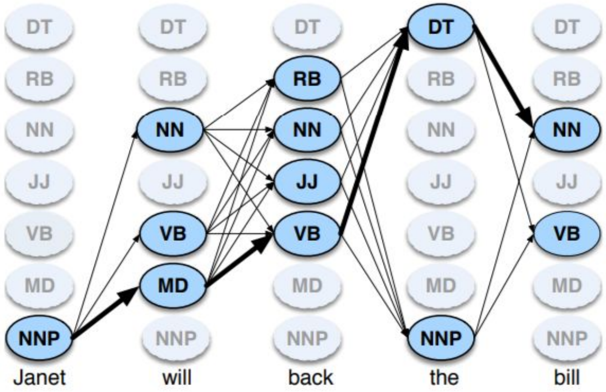


$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$$

previous
Viterbi path
probability

The Viterbi algorithm

$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j



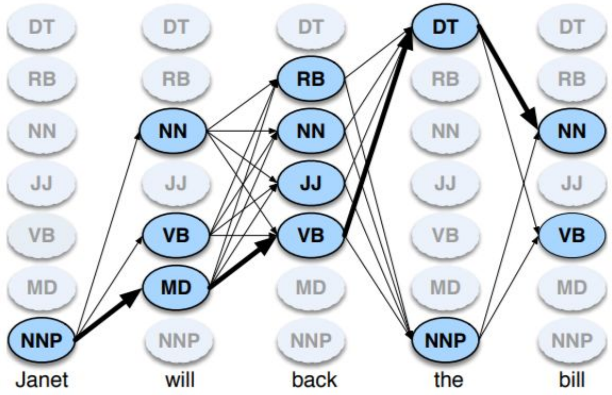
transition probability

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

previous Viterbi path probability

The Viterbi algorithm

$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j



transition probability

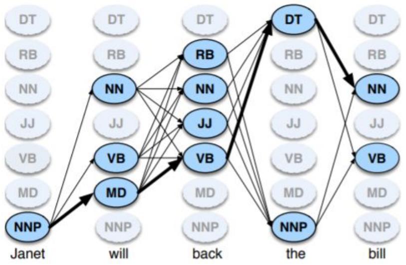
$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

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*



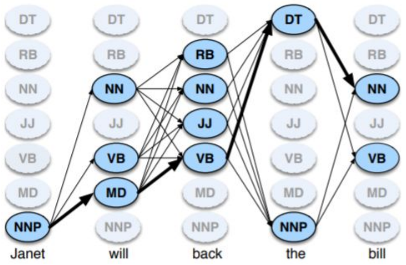
create a path probability matrix *viterbi[N,T]*
for each state *s* **from** 1 **to** *N* **do**
    $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
    $backpointer[s,1] \leftarrow 0$
for each time step *t* **from** 2 **to** *T* **do**
  for each state *s* **from** 1 **to** *N* **do**
    $viterbi[s,t] \leftarrow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

initialization



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**

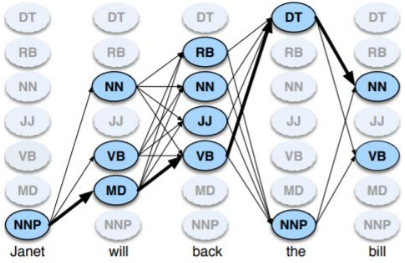
$$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$
$$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

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

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

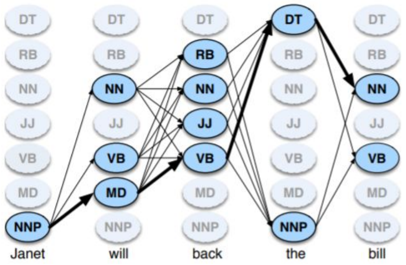
initialization

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**
$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$
bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

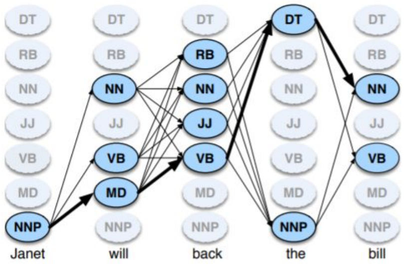
initialization

recursion

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

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**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$ ← $v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$

recursion

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

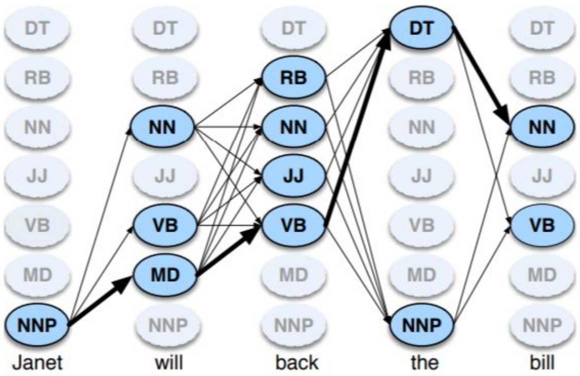
termination

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

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

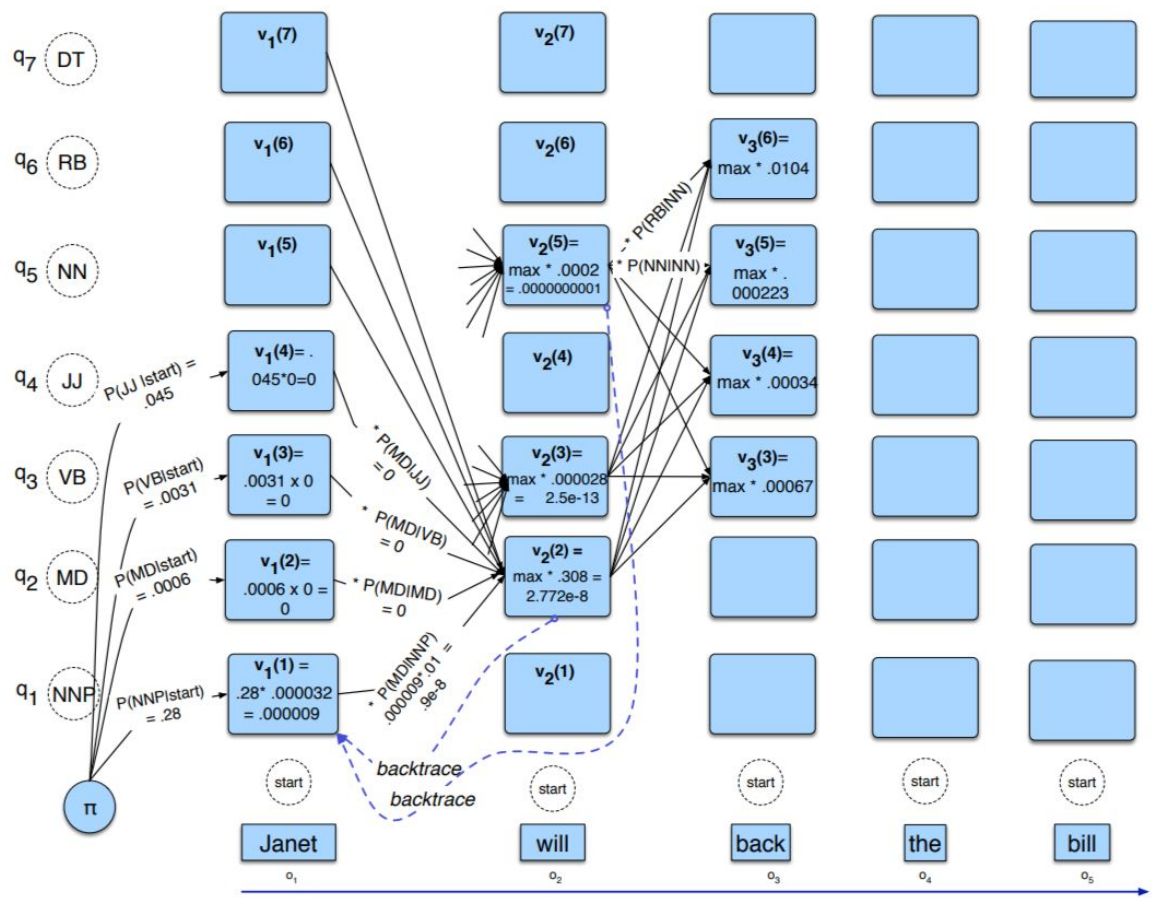
	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0



A

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

B


q7 DT
q6 RB
q5 NN
q4 JJ
q3 VB
q2 MD
q1 NNP
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
π
start
start
start
start
start
Janet
will
back
the
bill
o1
o2
o3
o4
o5

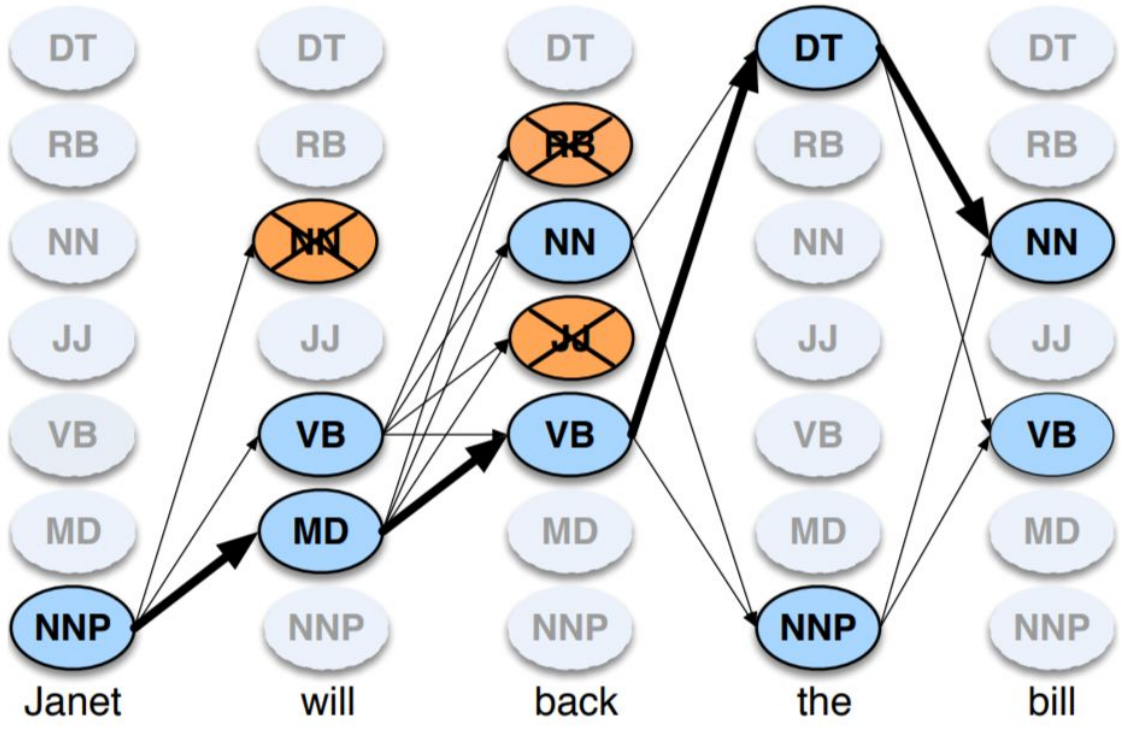
The Viterbi algorithm

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 ← 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



HMMs: algorithms

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.

From J&M

HMMs: algorithms

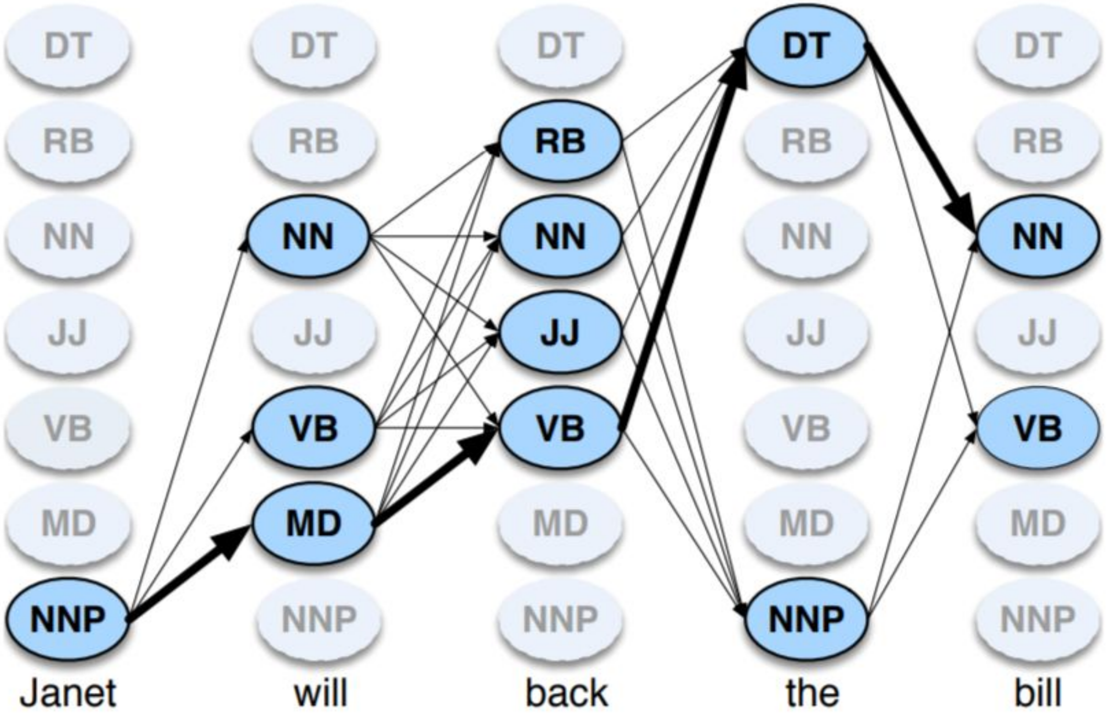
Forward	**Problem 1 (Likelihood):**	Given an HMM $\lambda = (A,B)$ and an observation sequence O, determine the likelihood $P(O\|\lambda)$.
Viterbi	**Problem 2 (Decoding):**	Given an observation sequence O and an HMM $\lambda = (A,B)$, discover the best hidden state sequence Q.
Forward-backward; Baum-Welch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

From J&M

The Forward algorithm

- Just sum instead of max!

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



Viterbi

- n-best decoding
- relationship to sequence alignment

Citation	Field
Viterbi (1967)	information theory
Vintsyuk (1968)	speech processing
Needleman and Wunsch (1970)	molecular biology
Sakoe and Chiba (1971)	speech processing
Sankoff (1972)	molecular biology
Reichert et al. (1973)	molecular biology
Wagner and Fischer (1974)	computer science