VIT-AP UNIVERSITY

Natural Language Processing

(Course Code: CSE 3015)

Module-2:Lecture-3: Hidden Markov Model

Gundimeda Venugopal, Professor of Practice, SCOPE

Generative vs Discriminative Models

Both of them are used to predict the conditional probability **P(y|x)**, Generally used in supervised learning problems

1. Discriminative Model
   - A Discriminative Model models the **decision boundary between the classes**
   - A Discriminative model learns the conditional probability distribution p(y|x)

2. Generative Model
   - A Generative Model explicitly models the **actual distribution of each class**
   - It learns the **joint probability distribution P(x,y)**
   - It predicts the conditional probability P(y|x) with the help of Bayes theorem

Generative vs Discriminative Models

Generative Classifiers

- Assume some functional form for $P(Y)$, $P(X|Y)$
- Estimate parameters of $P(X|Y)$, $P(Y)$ directly from training data
- Use Bayes Theorem to calculate $P(Y|X)$
- Examples
  - Naive Bayes
  - Hidden Markov Model
  - Markov Random Fields

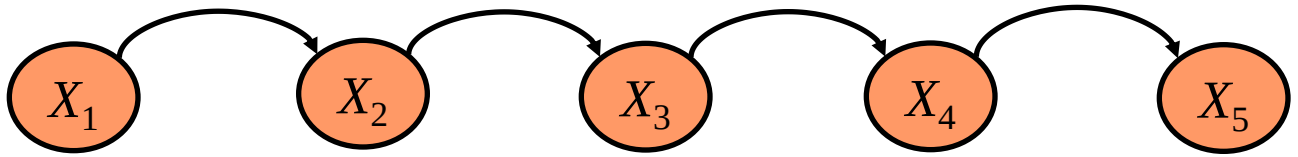
Discriminative Classifiers

- Assume some functional form for $P(Y|X)$
- Estimate parameters of $P(Y|X)$ directly from training data
- Examples
  - Logistic Regression
  - Nearest Neighbour
  - Conditional Random Fields
  - Maximum Entropy Models

Markov Process

- **Markov Property:** The state of the system at time $t+1$ depends only on the state of the system at time t

$$\Pr[X_{t+1} = x_{t+1} \mid X_1 \cdots X_t = x_1 \cdots x_t] = \Pr[X_{t+1} = x_{t+1} \mid X_t = x_t]$$



- **Stationary Assumption:** Transition probabilities are independent of time (t)

$$\Pr[X_{t+1} = b \mid X_t = a] = p_{ab}$$

Bounded memory transition model

Defining a Markov Model

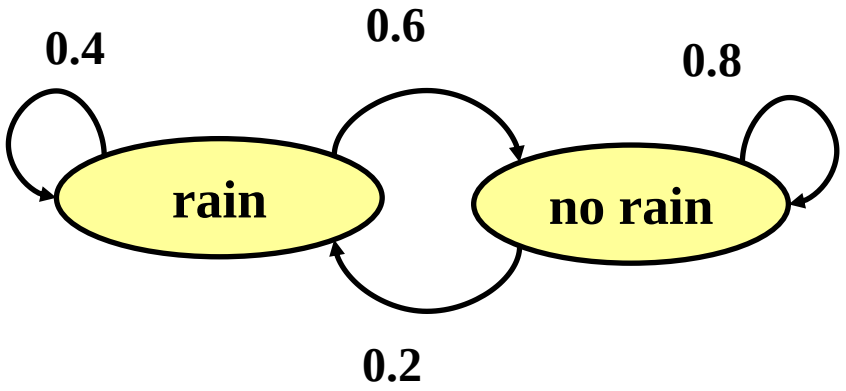
- Define the initial states
- Determine the cycle length
- Consider possible transitions among states
- Determine transition probabilities
- Determine utilities, and costs (if cost-effectiveness analysis), for each state

Markov Process – A Simple Example

Weather:

- raining today → 40% rain tomorrow
  → 60% no rain tomorrow
- not raining today → 20% rain tomorrow
  → 80% no rain tomorrow

Stochastic FSM:



The transition matrix:

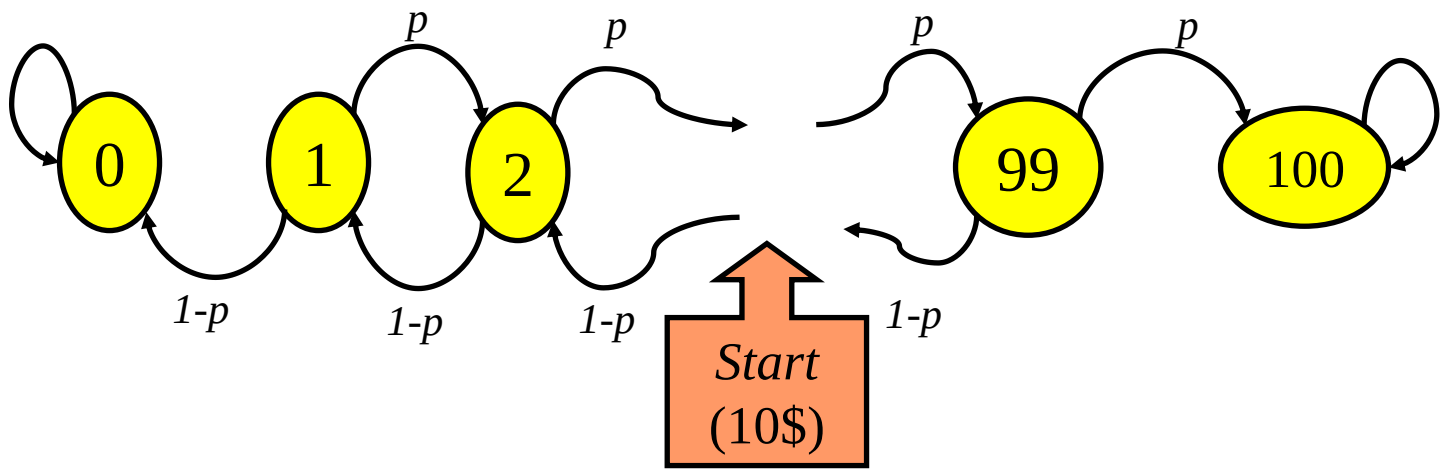
$$P = \begin{pmatrix} 0.4 & 0.6 \\ 0.2 & 0.8 \end{pmatrix}$$

- Stochastic matrix:
  Rows sum up to 1
- Double stochastic matrix:
  Rows and columns sum up to 1

Markov Process: A Gambler's Example

- Gambler starts with $10
- At each play we have one of the following:
  - Gambler wins $1 with probability p
  - Gambler looses $1 with probability $1\text{-}p$
- Game ends when gambler goes broke, or gains a fortune of $100

  (Both 0 and 100 are absorbing states)

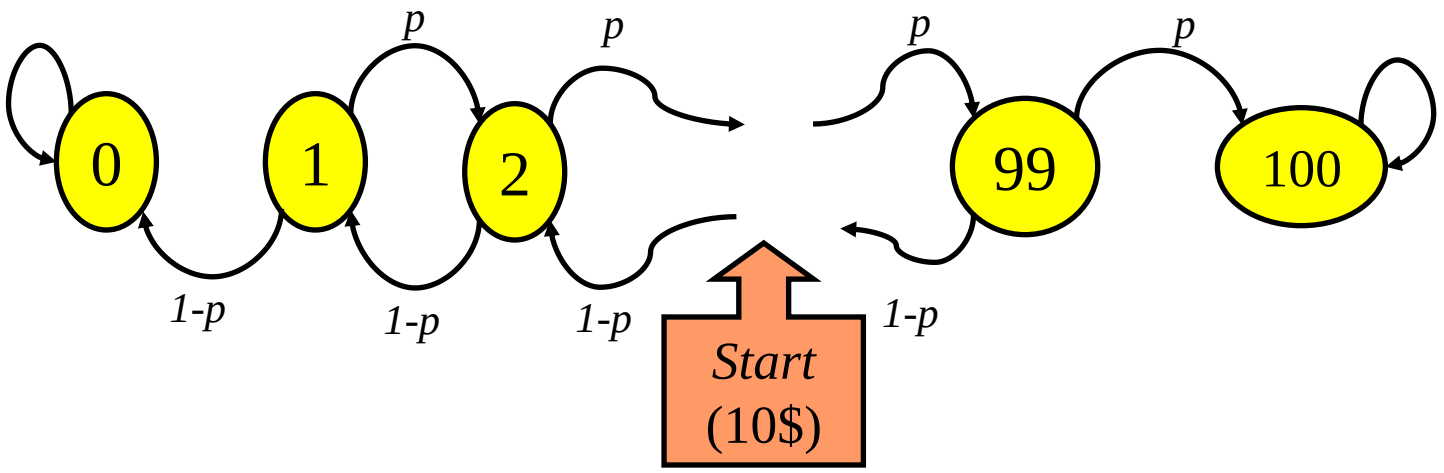


Markov Process

- **Markov process** - described by a stochastic FSM
- **Markov chain** - a random walk on this graph

(distribution over paths)

- Edge-weights give us $\Pr[X_{t+1} = b \mid X_t = a] = p_{ab}$
- We can ask more complex questions, like $\Pr[X_{t+2} = a \mid X_t = b] = ?$

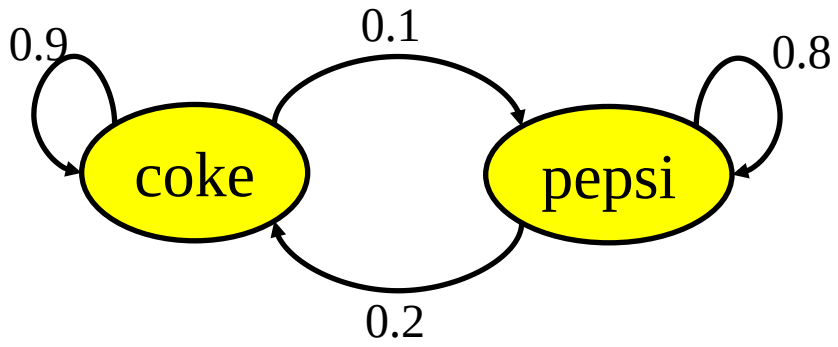


Markov Process: Coke vs. Pepsi Example

- Given that a person's last cola purchase was Coke, there is a 90% chance that his next cola purchase will also be Coke.
- If a person's last cola purchase was Pepsi, there is an 80% chance that his next cola purchase will also be Pepsi.

transition matrix:

$$P = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}$$



Markov Process: Coke vs. Pepsi Example (continued)

Given that a person is currently a Pepsi purchaser, what is the probability that he will purchase Coke two purchases from now?

Pr[Pepsi→?→Coke] =

Pr[Pepsi→Coke→Coke] + Pr[Pepsi→ Pepsi →Coke] =

0.2 * 0.9 + 0.8 * 0.2 = 0.34

$$P^2 = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix} \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix} = \begin{bmatrix} 0.83 & 0.17 \\ 0.34 & 0.66 \end{bmatrix}$$

Pepsi → ? ? → Coke

Markov Process: Coke vs. Pepsi Example (continued)

Given that a person is currently a Coke purchaser, what is the probability that he will purchase Pepsi **three** purchases from now?

$$P^3 = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix} \begin{bmatrix} 0.83 & 0.17 \\ 0.34 & 0.66 \end{bmatrix} = \begin{bmatrix} 0.781 & 0.219 \\ 0.438 & 0.562 \end{bmatrix}$$

Markov Process: Coke vs. Pepsi Example (continued)

- Assume each person makes one cola purchase per week
- Suppose 60% of all people now drink Coke, and 40% drink Pepsi
- What fraction of people will be drinking Coke three weeks from now?

$$P = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix} \qquad P^3 = \begin{bmatrix} 0.781 & 0.219 \\ 0.438 & 0.562 \end{bmatrix}$$

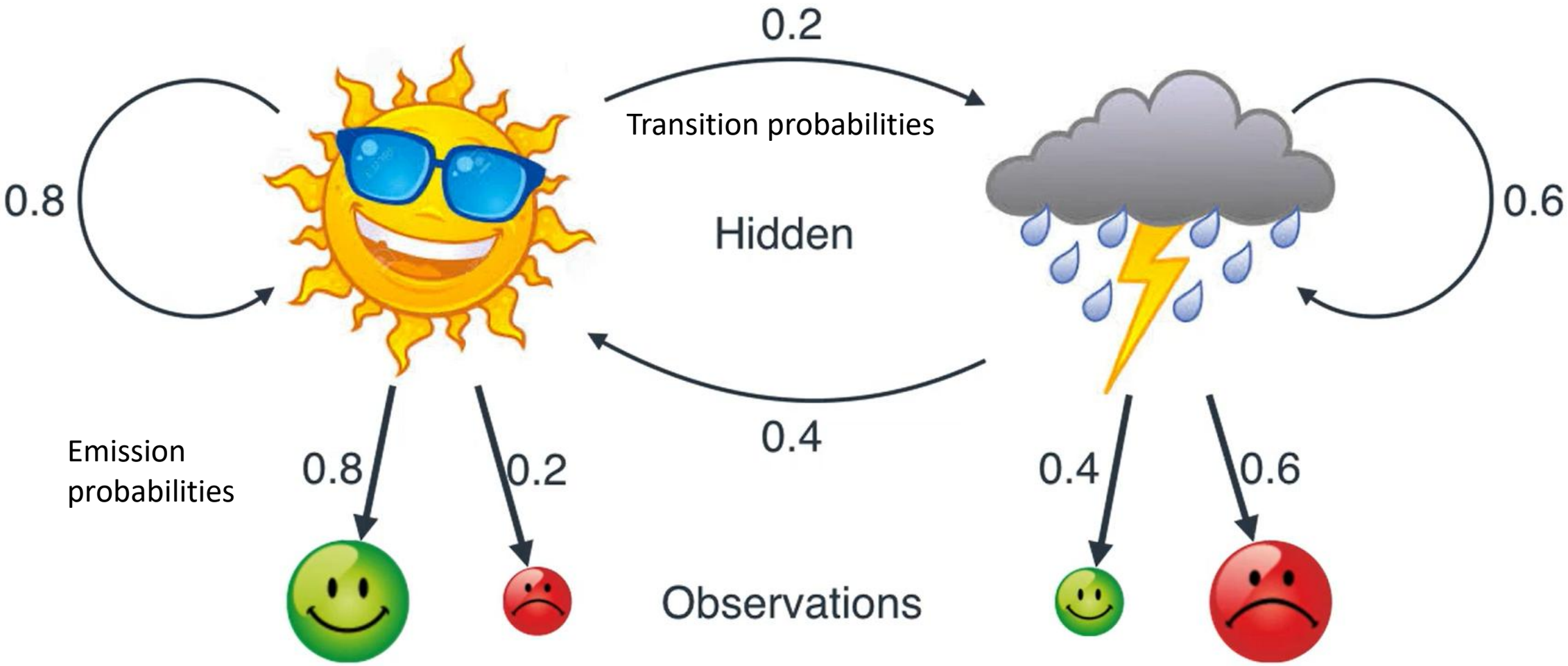
$$\Pr[X_3=\text{Coke}] = 0.6 * 0.781 + 0.4 * 0.438 = 0.6438$$

Q_i - the distribution in week i

$Q_0=(0.6,0.4)$ - initial distribution

$Q_3 = Q_0 * P^3 = (0.6438, 0.3562)$

A Hidden Markov Model Example

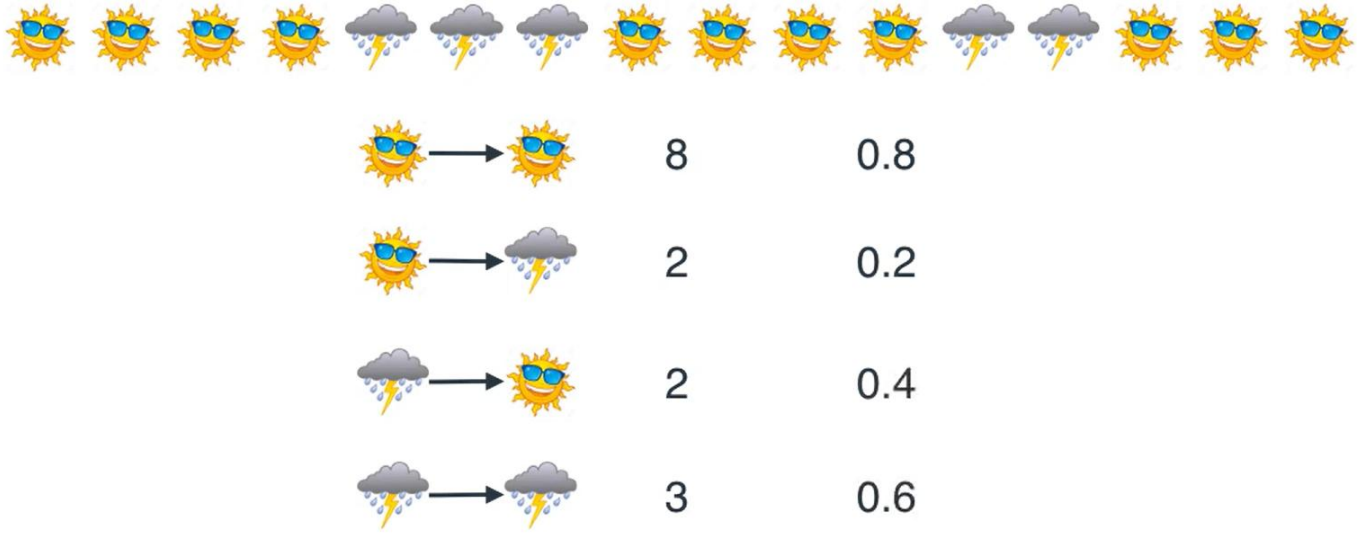


Ram (at Location A) and Laxman (at Location B) are two friends who communicate with each other over phone daily. Ram tries to guess the location B's weather (hidden variable) by observing Laxman's mood.

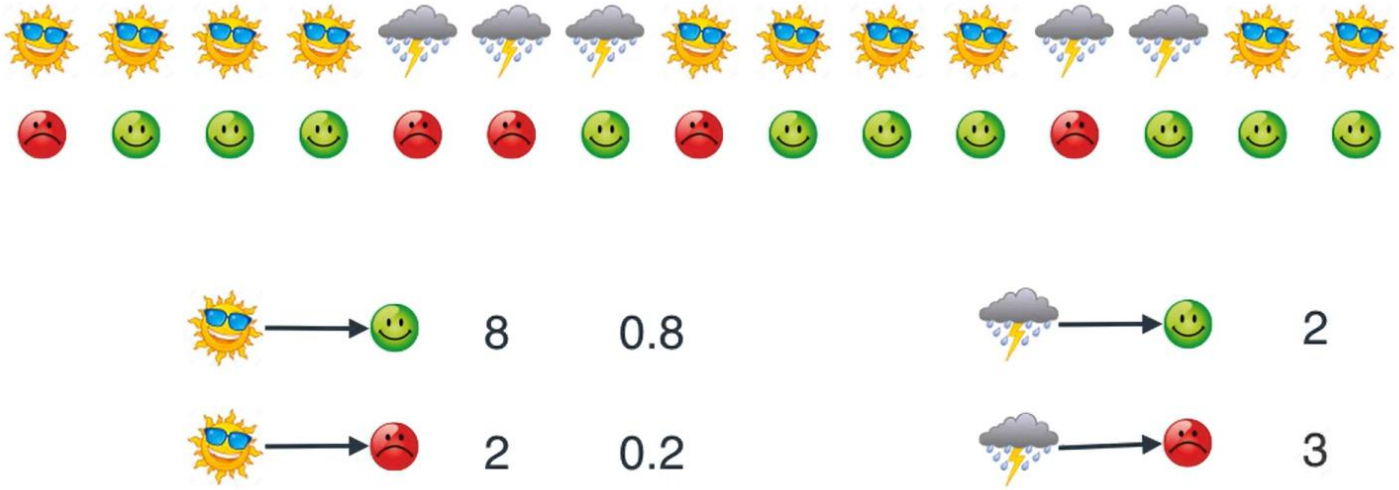
Questions

1. How did we find these probabilities?
2. What's the probability that a random day is Sunny or Rainy?
3. If Bob is Happy today, what's the probability that it's Sunny or Rainy?
4. If for three days Bob is Happy, Grumpy, Happy, what was the weather?

How do we find the probabilities?



Transition Probabilities

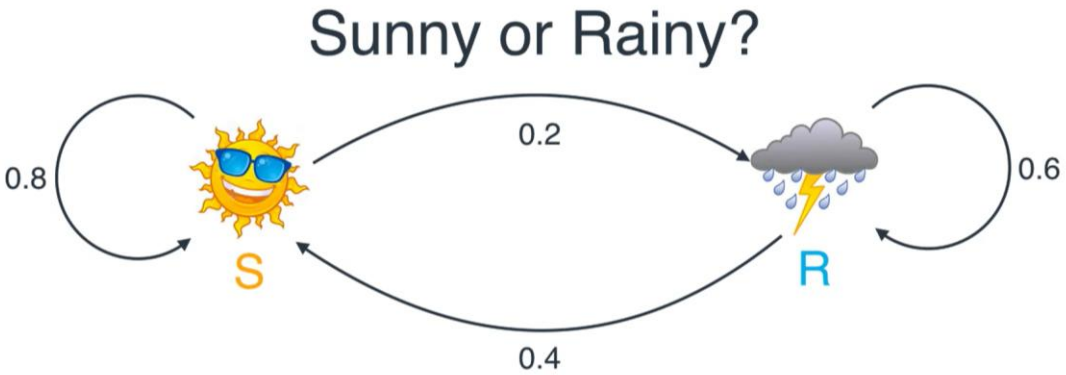


Emission Probabilities

Probability that a random day is Sunny or Rainy?

☀	10	2/3
⛈	5	1/3

Initial (prior) Probabilities



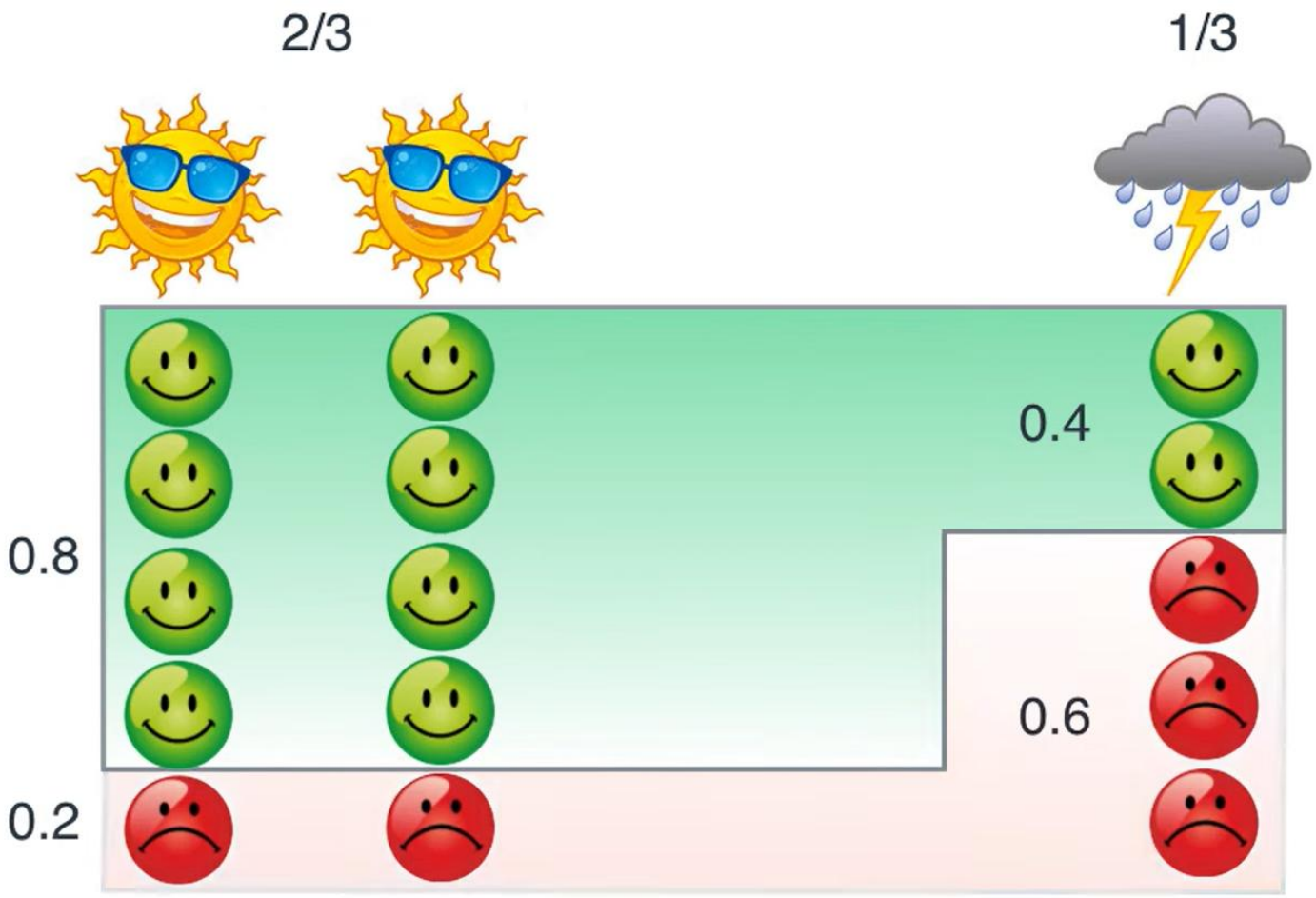
$S = 0.8S + 0.4R$

$S + R = 1$

$R = 0.2S + 0.6R$

$S = 2/3$
$R = 1/3$

If Bob is happy today, what is the probability that it is Sunny or Rainy?



Bayes Theorem

If 😊

$$P(☀ \mid 😊) = \frac{8}{10}$$

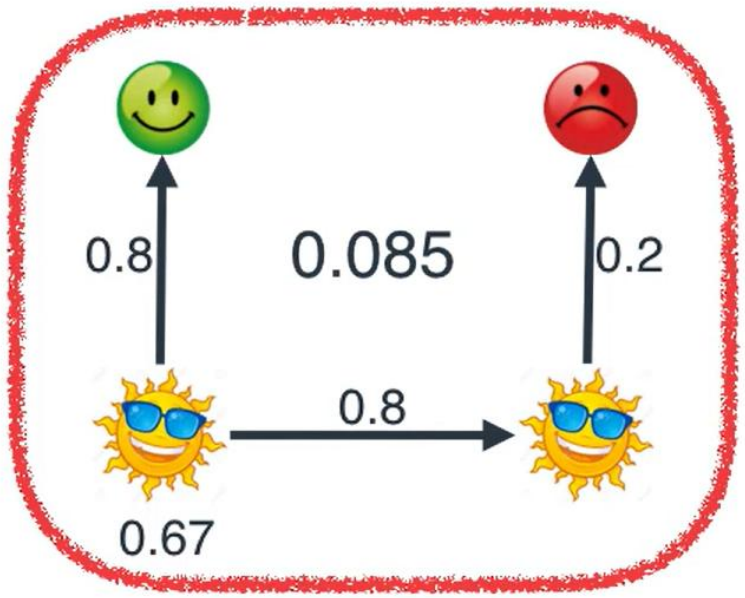
$$P(⛈ \mid 😊) = \frac{2}{10}$$

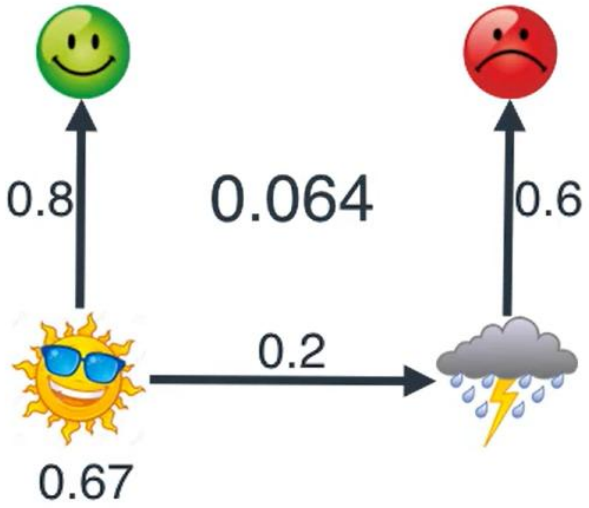
If ☹

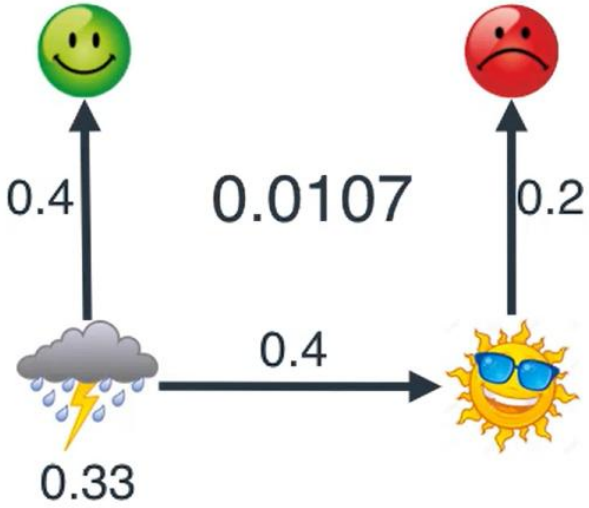
$$P(☀ \mid ☹) = \frac{2}{5}$$

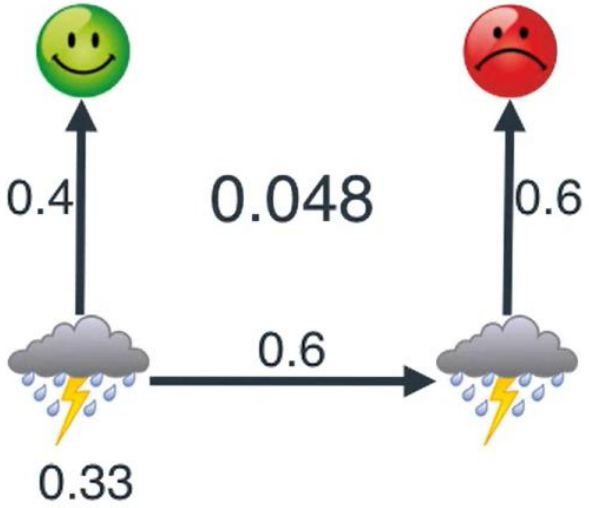
$$P(⛈ \mid ☹) = \frac{3}{5}$$

If Bob for three days Bob is happy grumpy and happy, what was the weather? 1/ 3

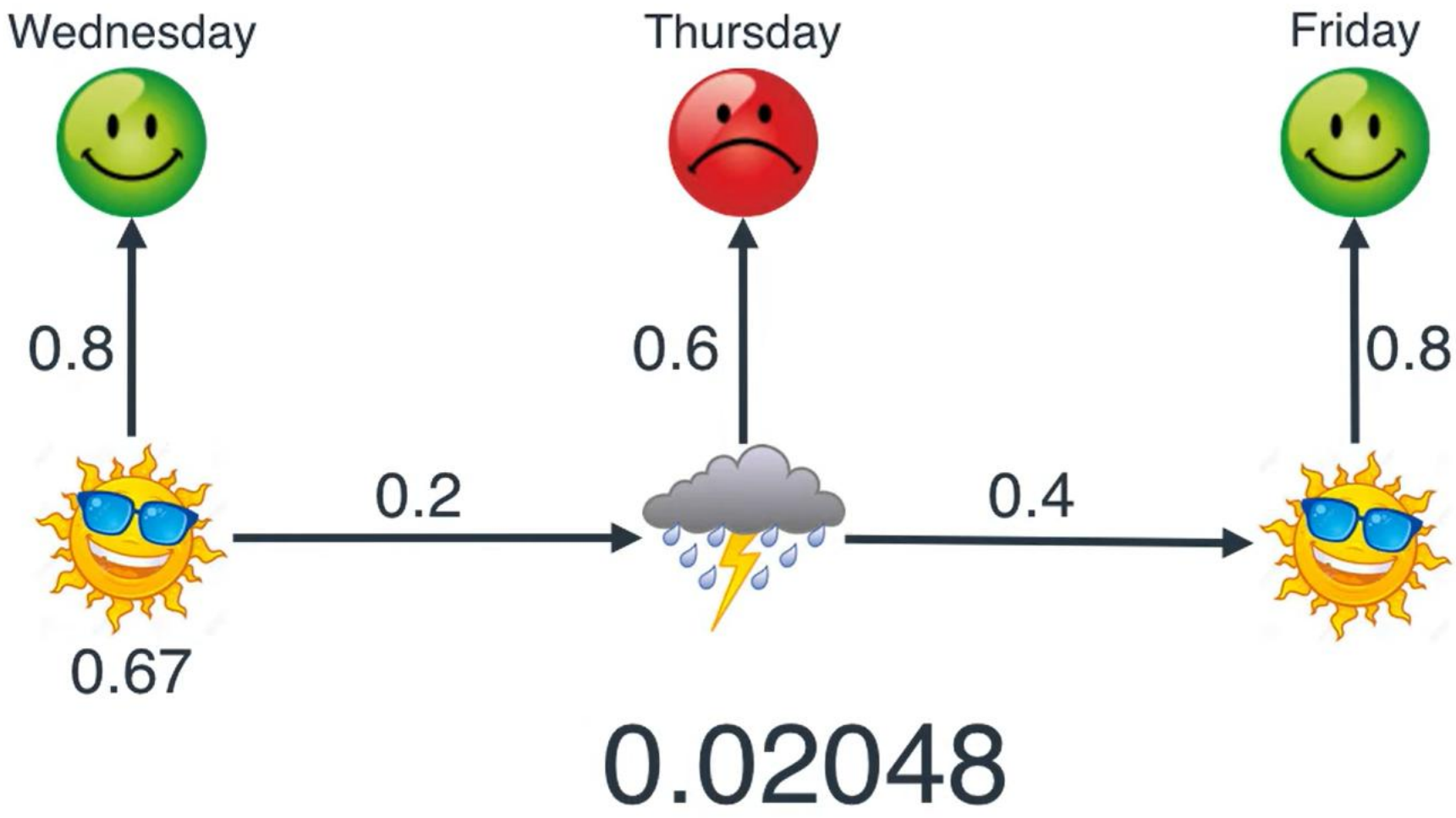




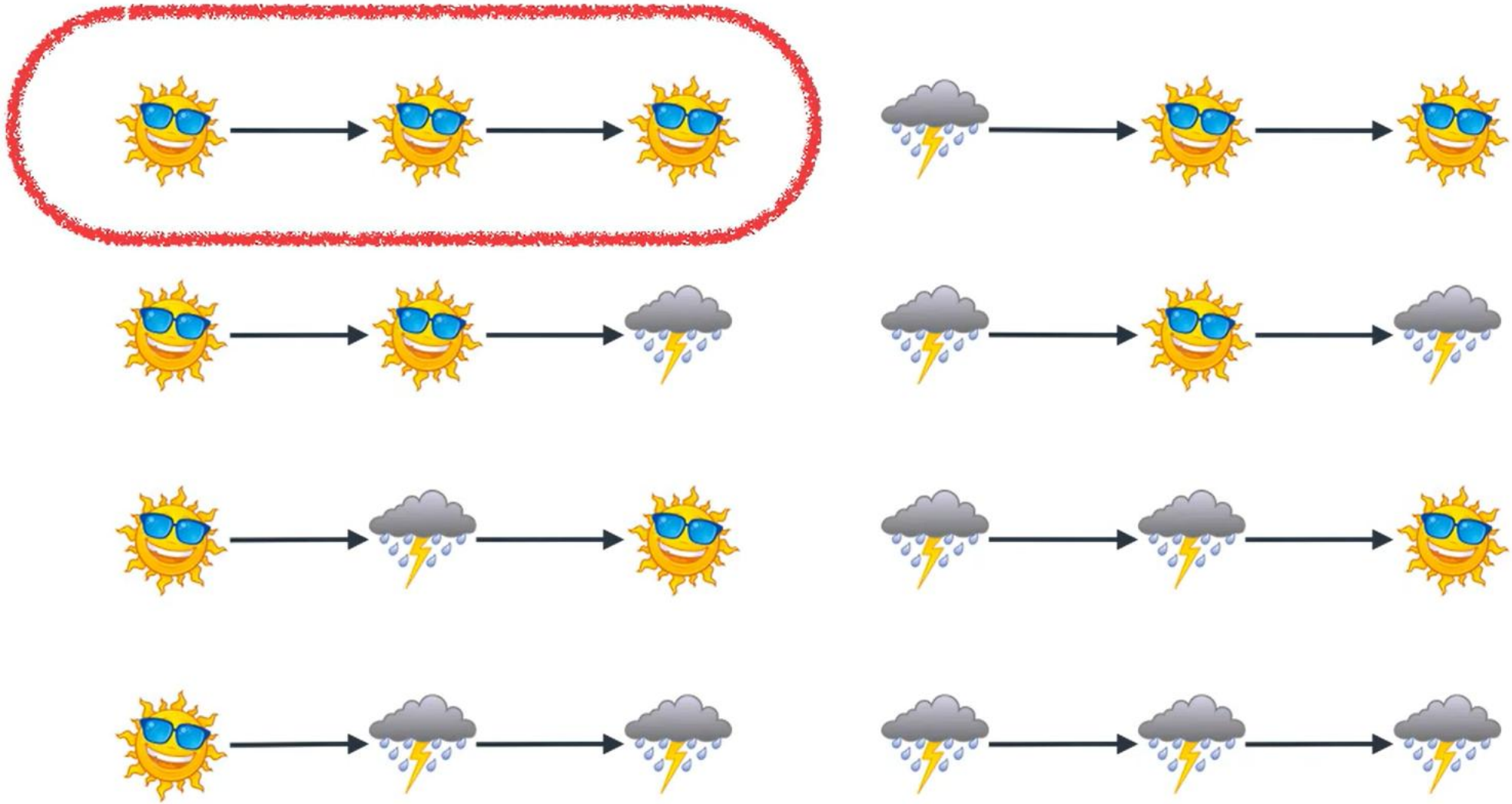




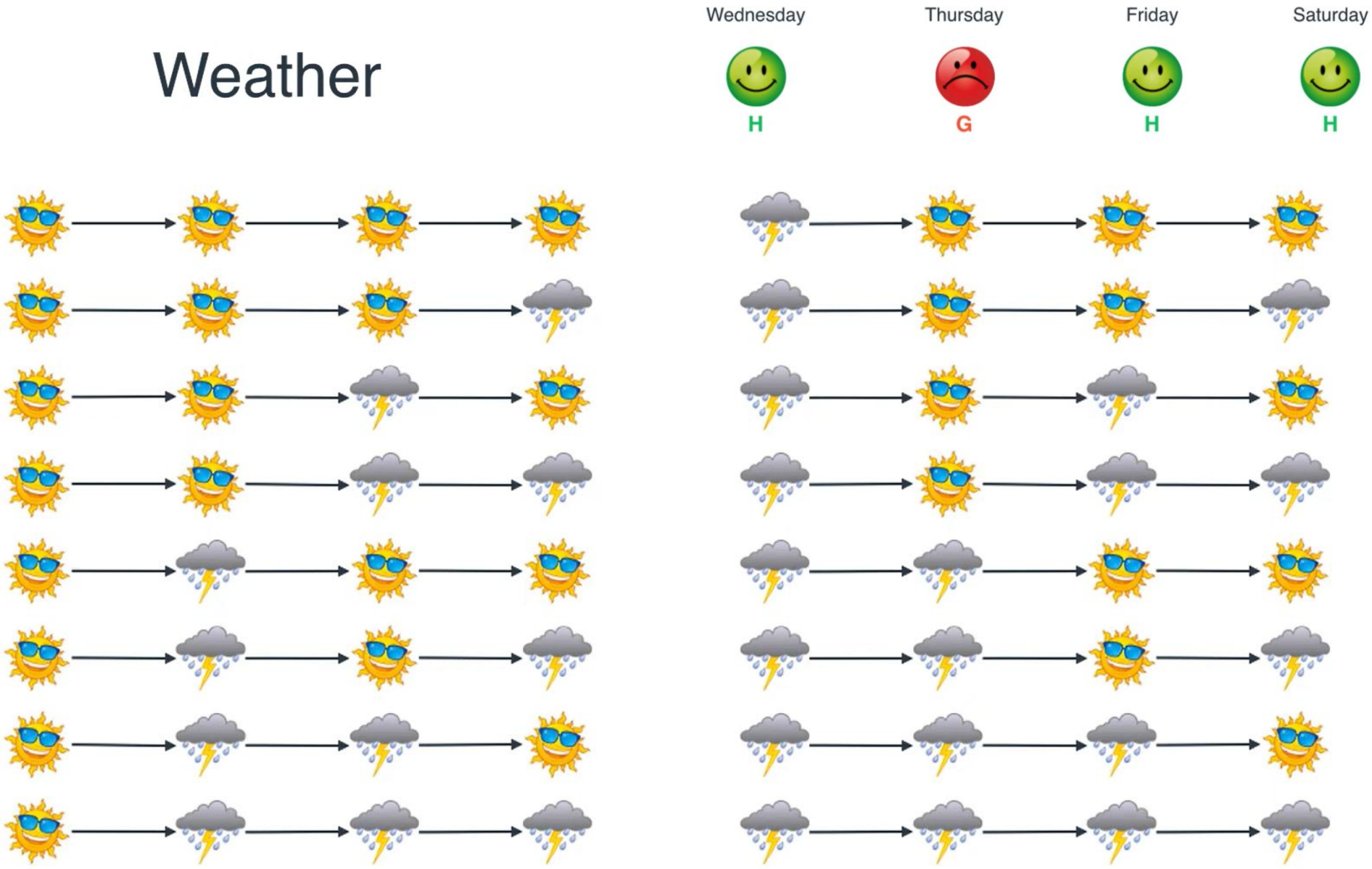
If Bob for three days Bob is happy grumpy and happy, what was the weather? 2/ 3



If Bob for three days Bob is happy grumpy and happy, what was the weather? 3/ 3



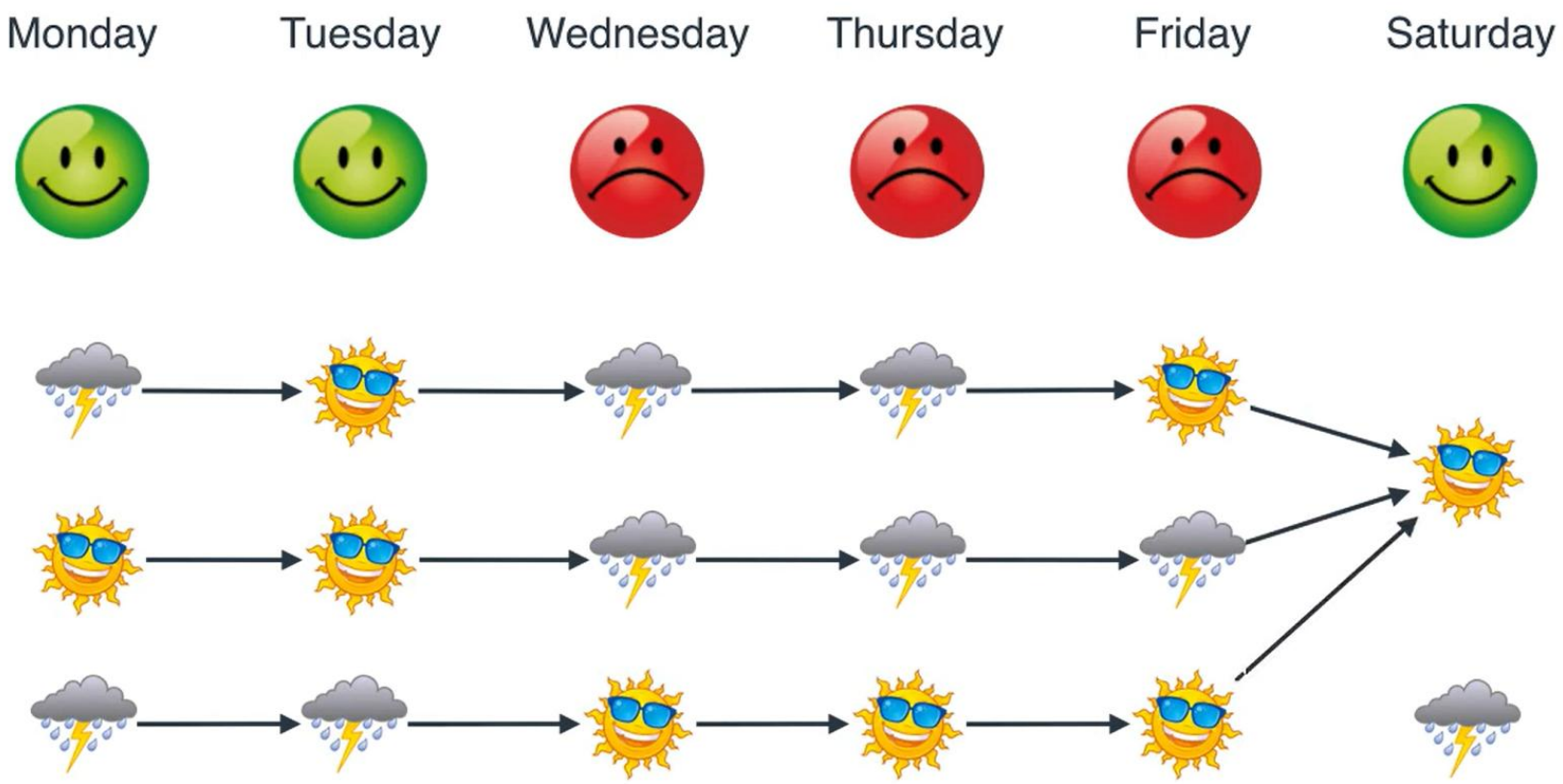
If Bob for four days Bob is happy grumpy & happy, what was the weather?



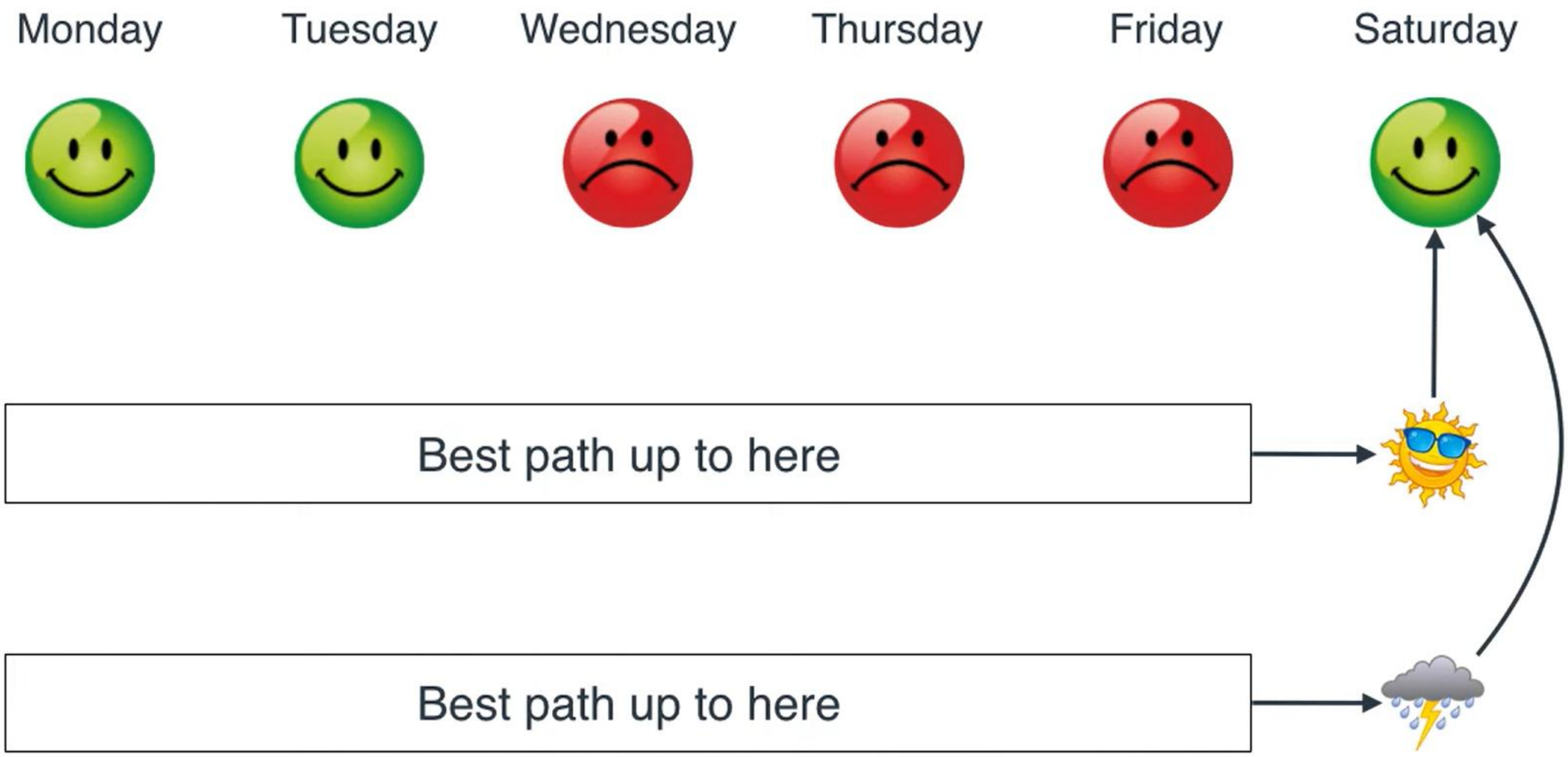
Viterbi Example

If we know Bob's mood for 7 days, what was the weather? (1/5)

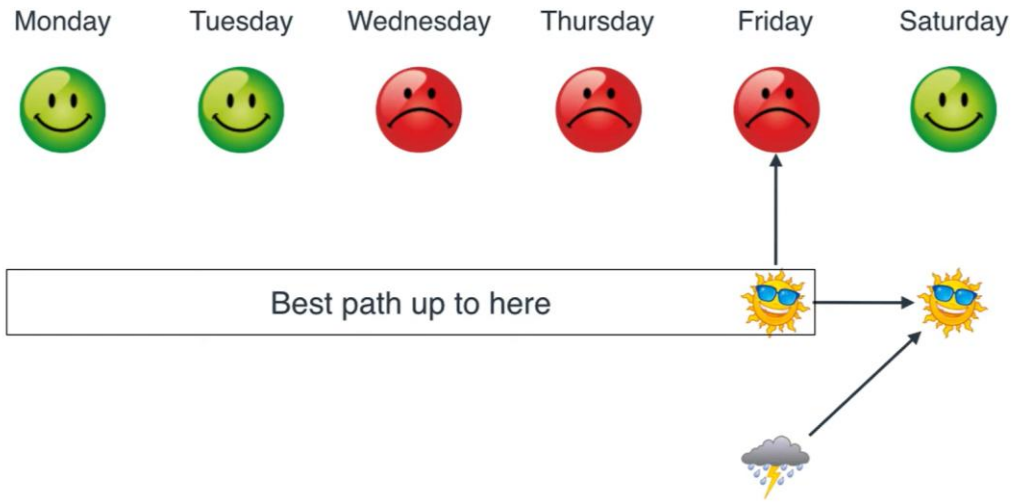
Complex: 2^7 options to check to find the optimum weather prediction



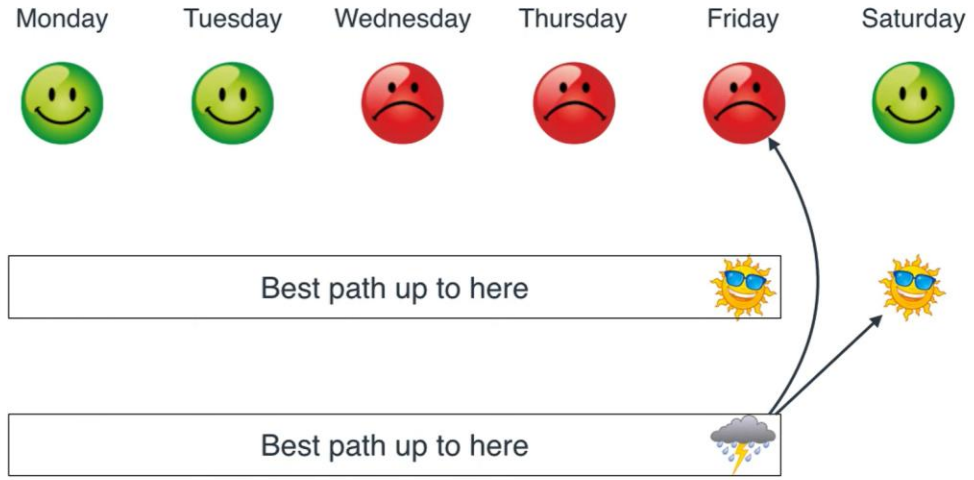
If we know Bob's mood for 7 days, what was the weather? (2/5)



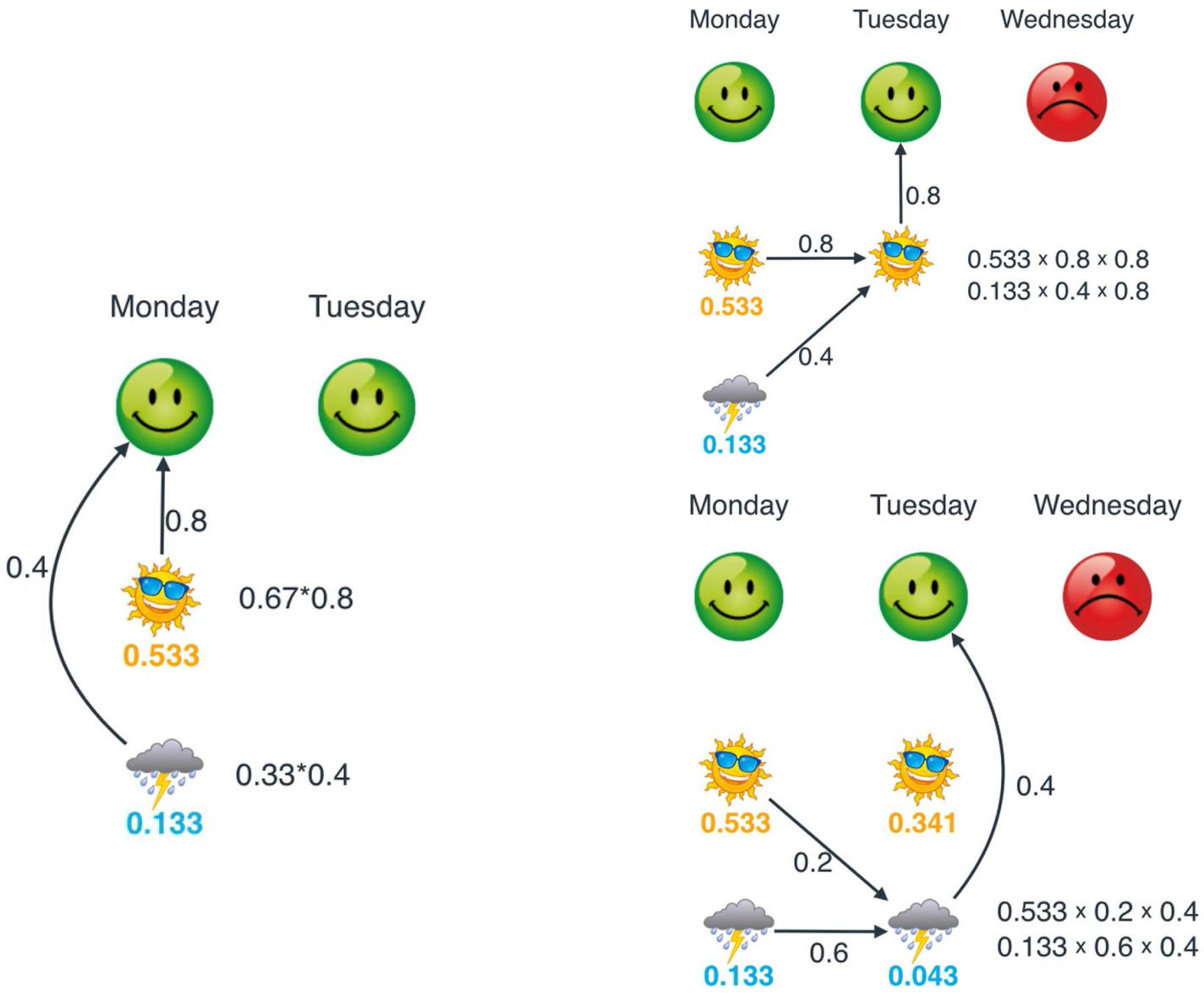
If we know Bob's mood for 7 days, what was the weather? (3/5)



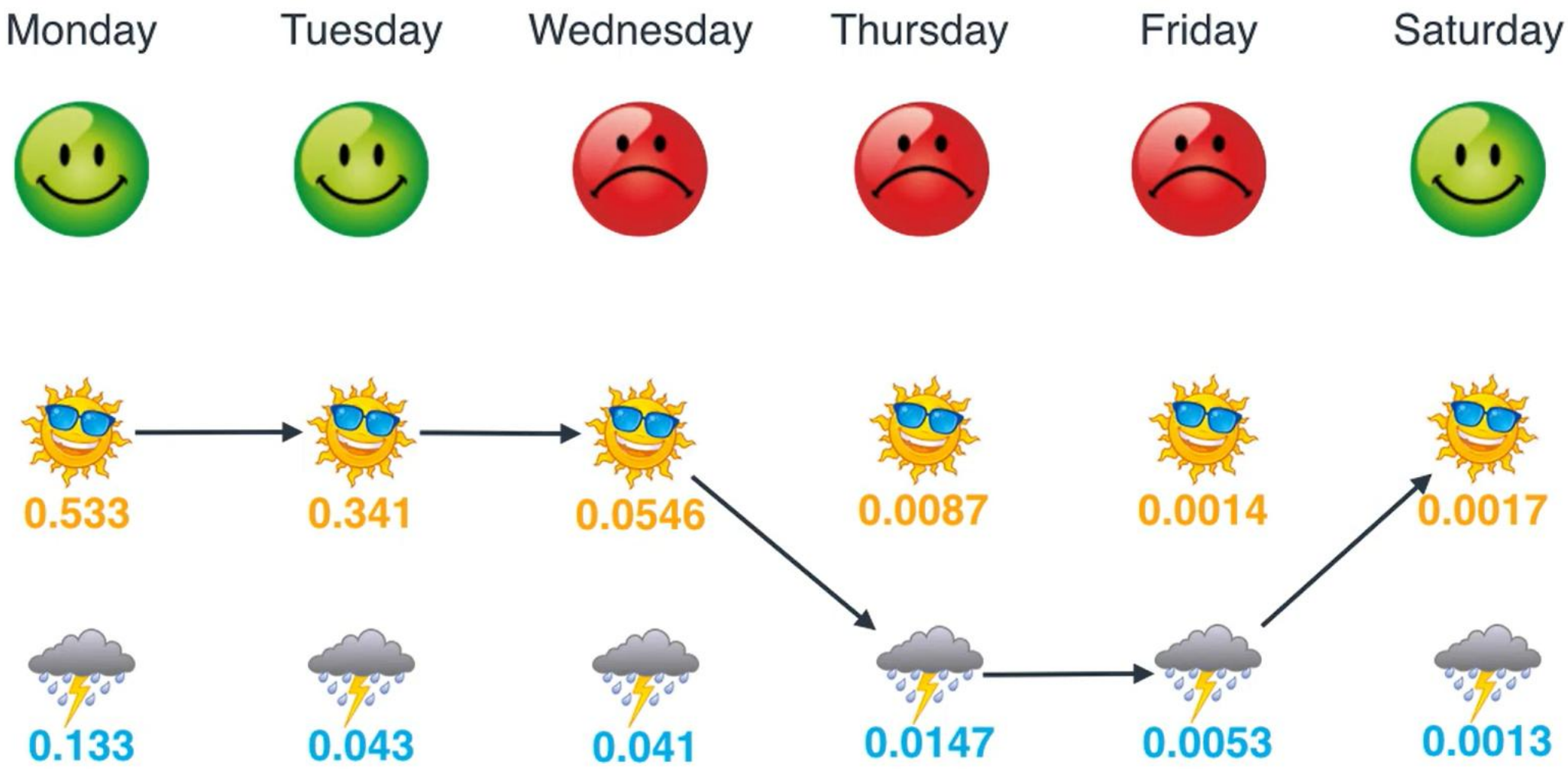
compare



If we know Bob's mood for 7 days, what was the weather? (4/5)



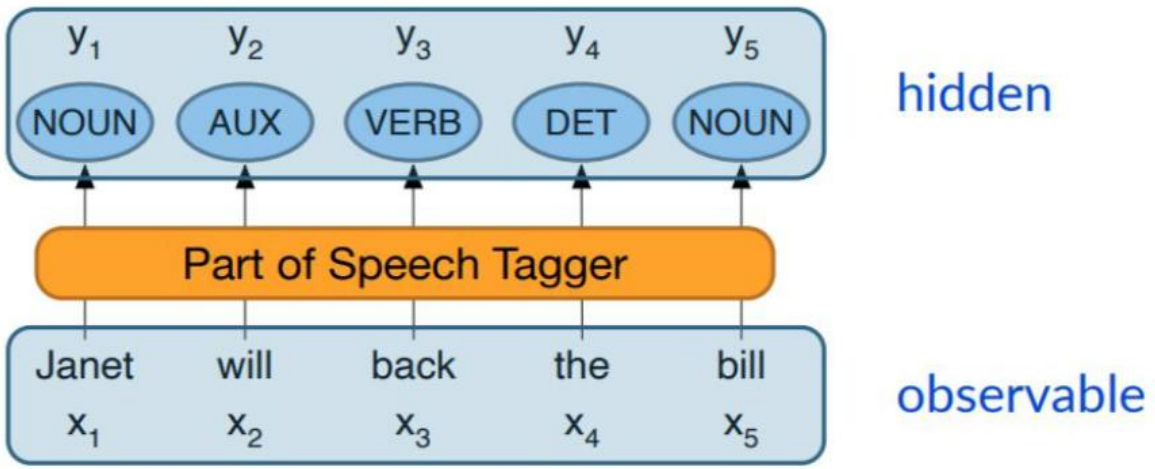
If we know Bob's mood for 7 days, what was the weather? (5/5)



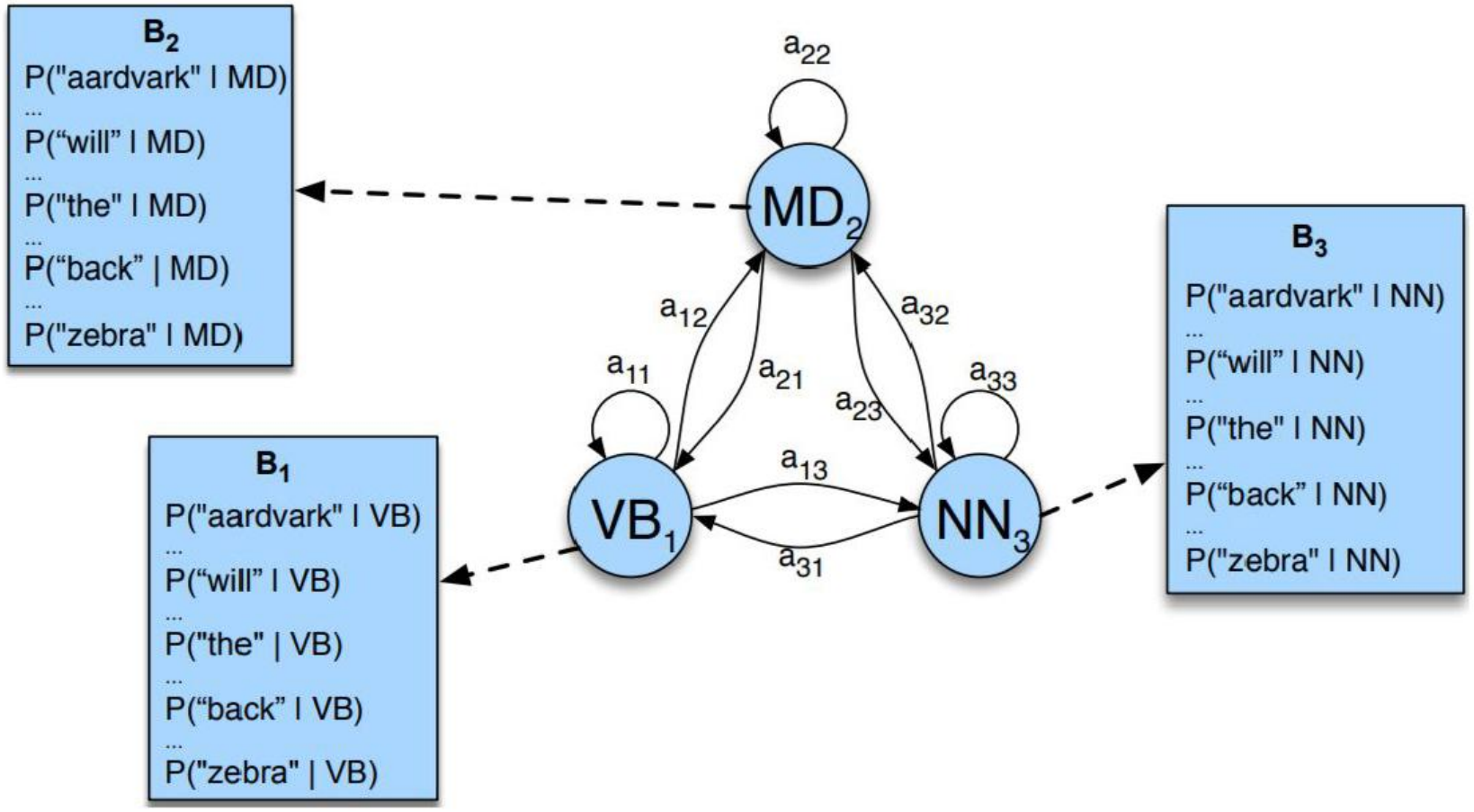
HMM for POS Tagging

Hidden Markov Models (HMMs)

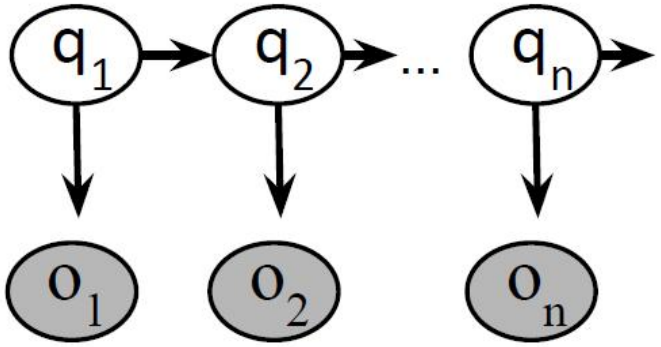
- We use a Markov chain for computing P for a sequence of observable events
- In many cases the events we are interested in are hidden
  - e.g., we don't observe POS tags in a text

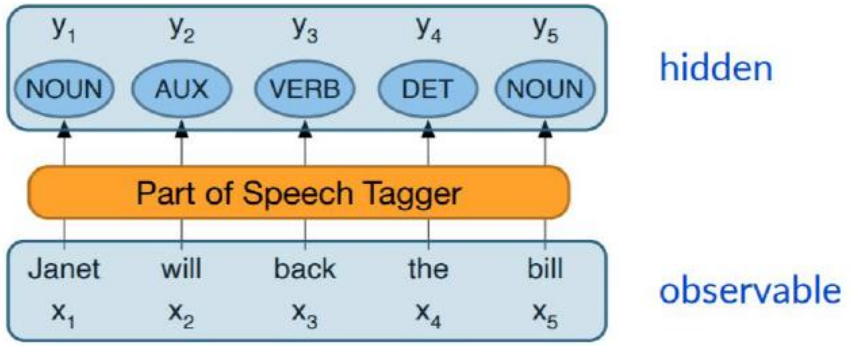


Hidden Markov Models (HMMs)



Hidden Markov Models (HMMs)





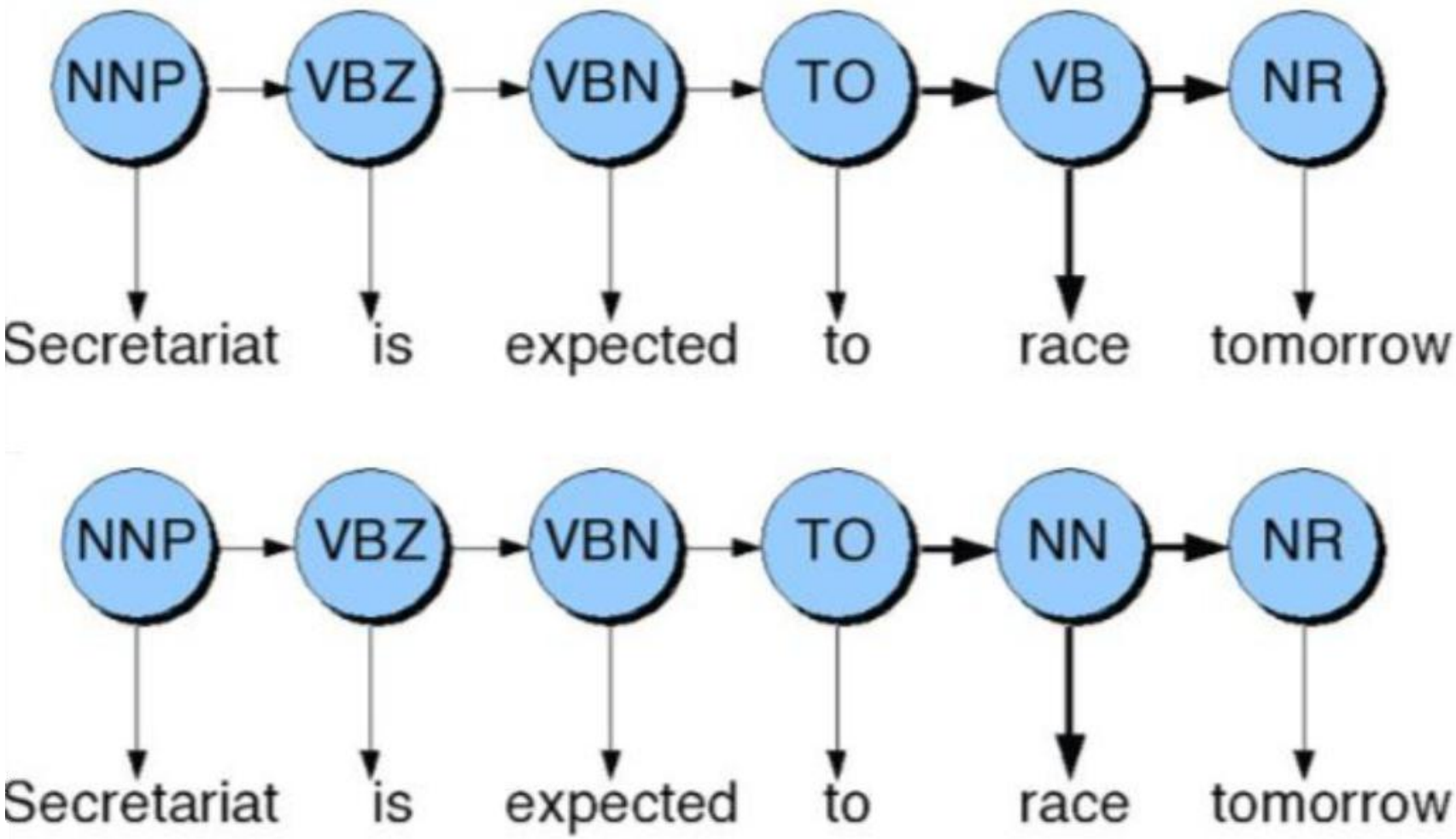
Markov Assumption: $P(q_i|q_1 \ldots q_{i-1}) = P(q_i|q_{i-1})$

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 $V = v_1, v_2, \ldots, v_V$
$B = b_i(o_t)$	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, $\sum_{i=1}^{n} \pi_i = 1$

POS tagging with HMMs



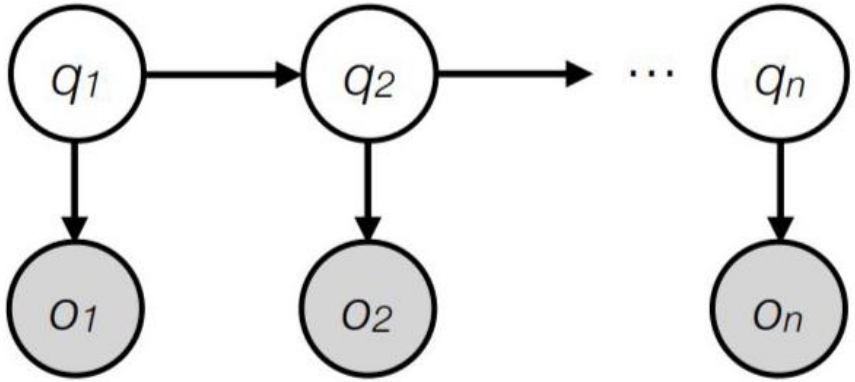
HMM: Algorithms

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

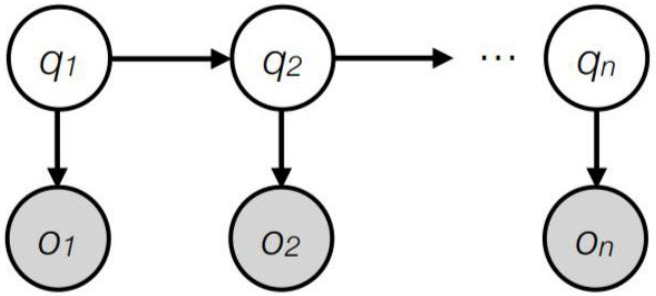
$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n \mid w_1^n)$$



Find the maximum probable sequence of **POS tags 1..n** given **Observed Words 1..n**

HMM: POS 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:

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

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

emission, B transition, A

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

Find the maximum probable sequence of **POS tags 1..n** given **Observed Words 1..n**

Parts Of Speech tagging Example

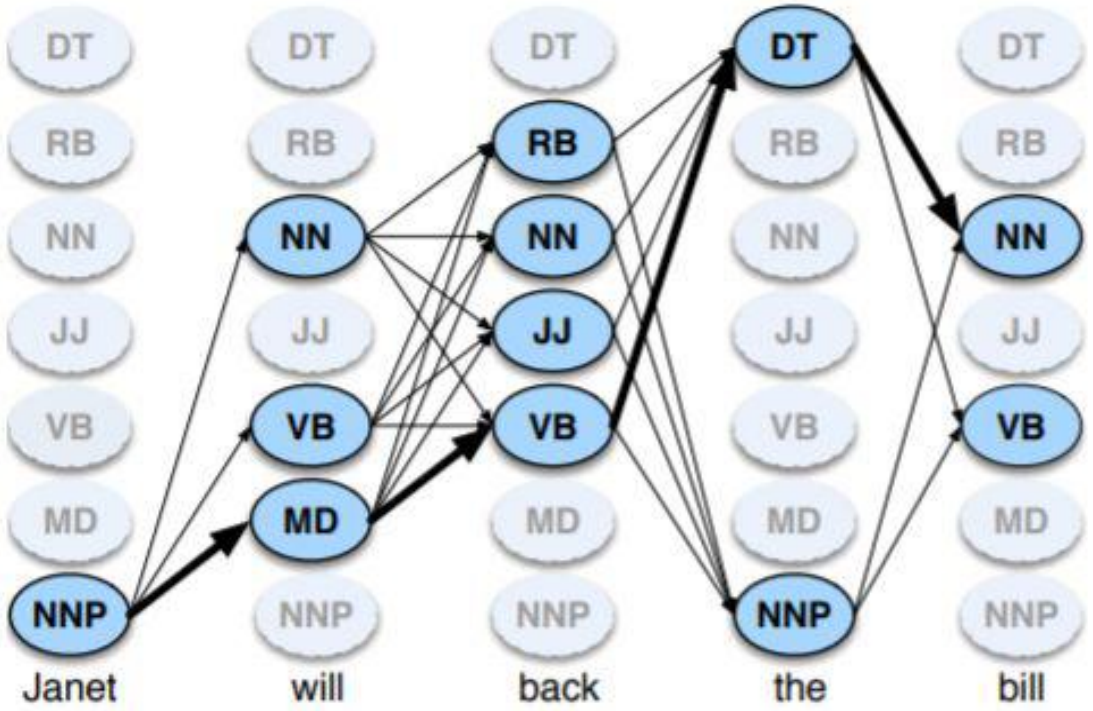
	I	suspect	the	present	forecast	is	pessimistic	.
noun	•	•	•	•	•	•		
adj.		•		•	•		•	
adv.				•				
verb		•		•	•	•		
num.	•							
det.			•					
punc.								•

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

$v_{t-1}(i)$	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

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

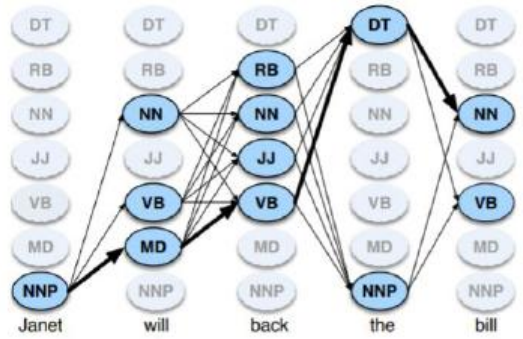
state observation likelihood

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 $\leftarrow$ 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*

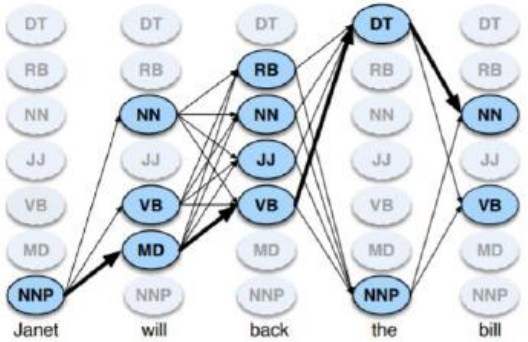
create a path probability matrix *viterbi*$[N,T]$
for each state s **from** 1 **to** N **do**
    $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
    $backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do**
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N}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

initialization

recursion

termination

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



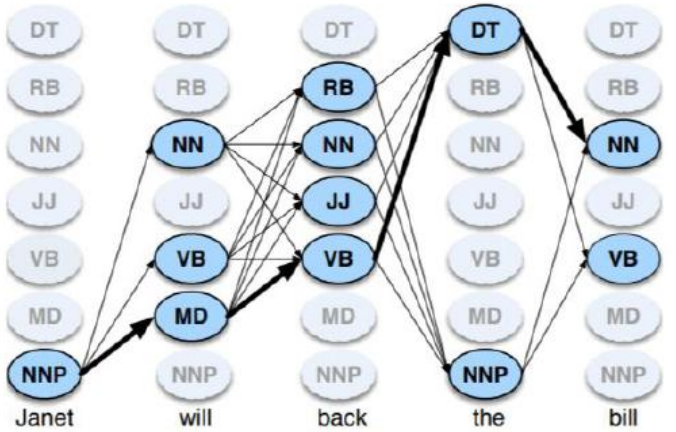
The Viterbi Algorithm

Transition Probabilities

	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

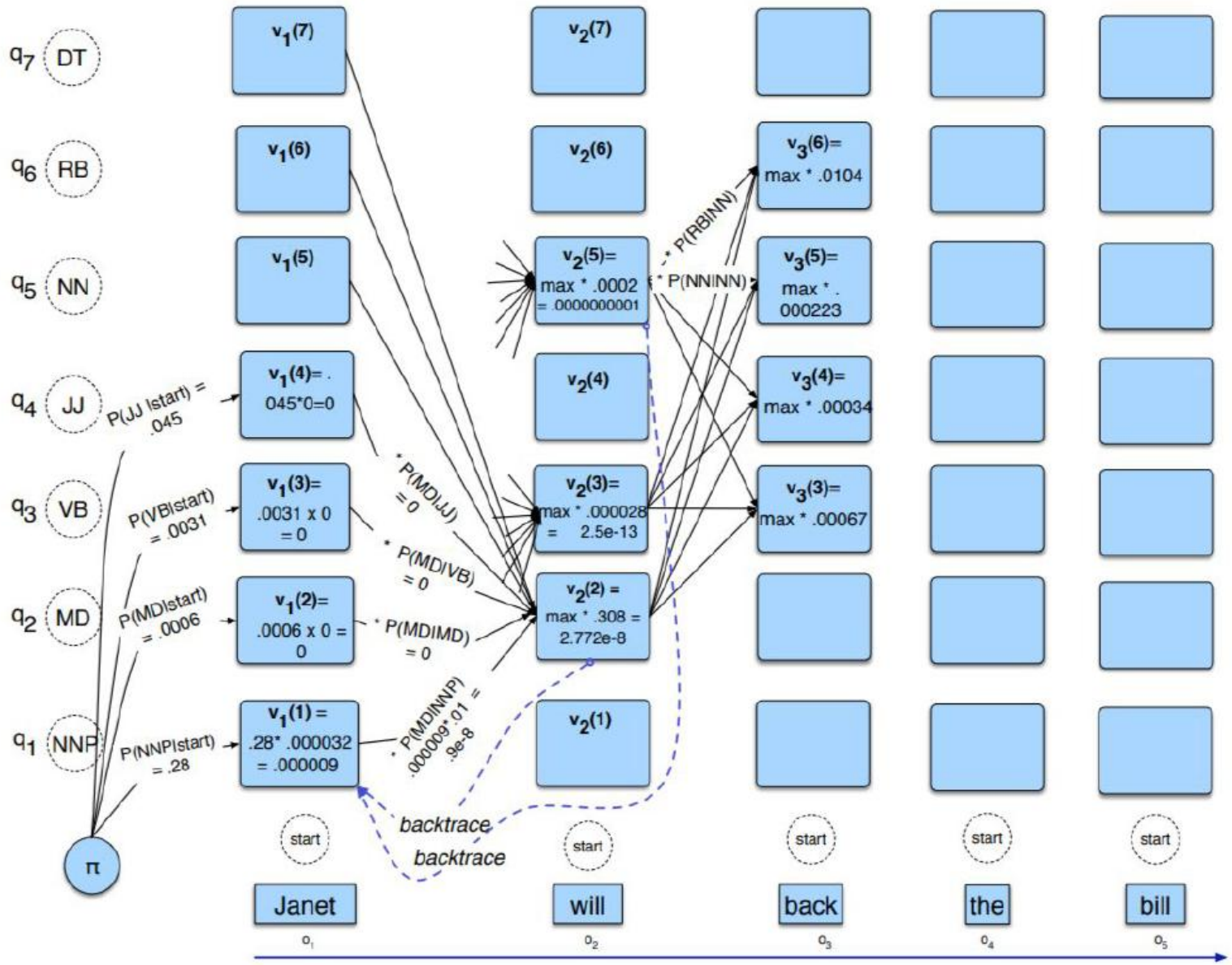
State Observation Likelihoods

	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



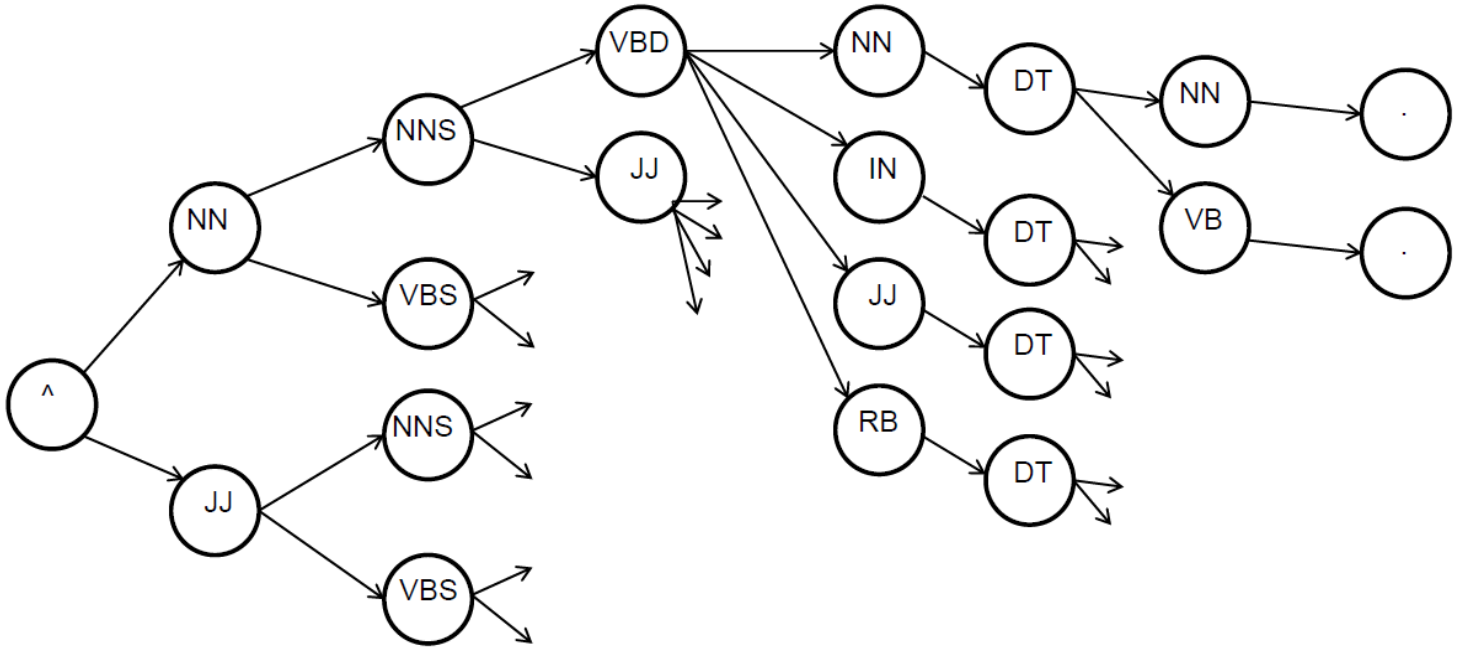
$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$$

The Viterbi Algorithm



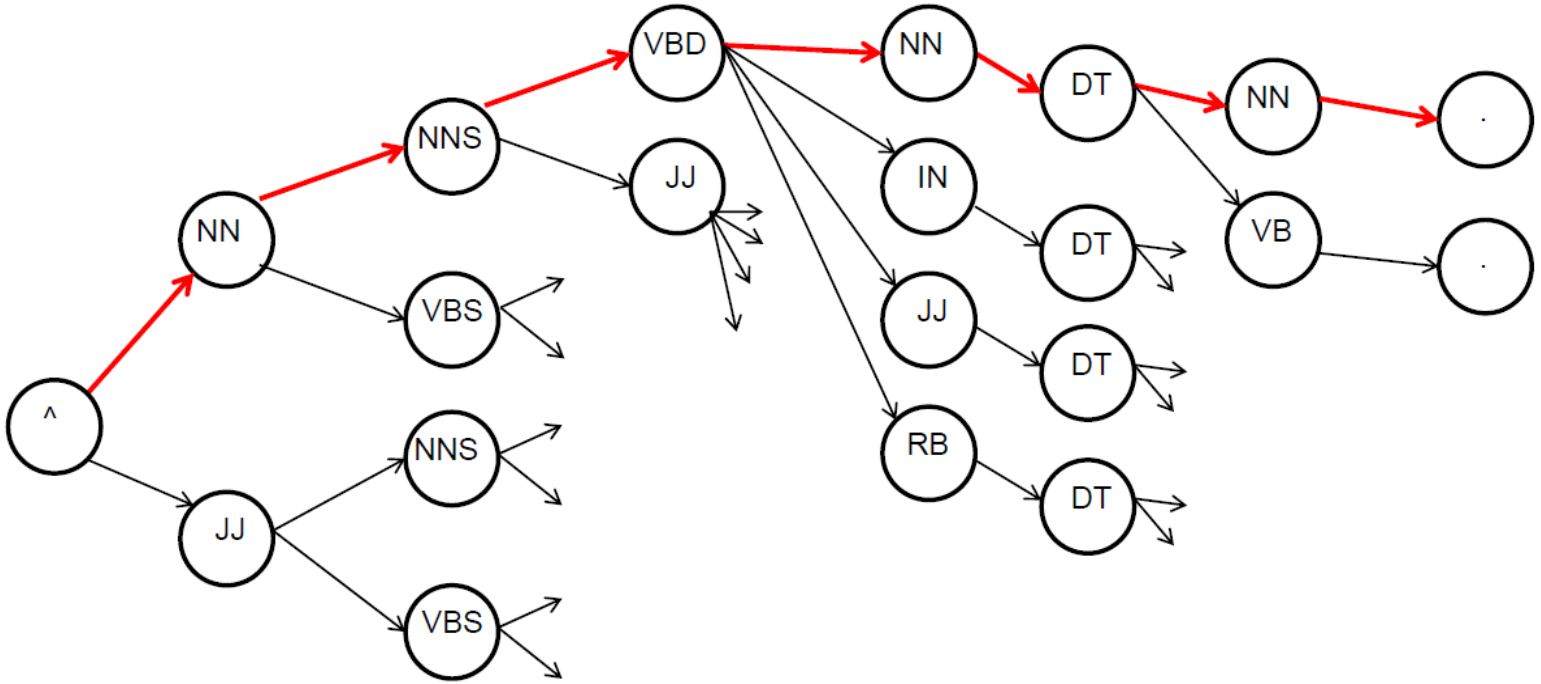
Viterbi Decoding Example - 1

W:	^	Brown	foxes	jumped	over	the	fence	.
T:	^	JJ	NNS	VBD	NN	DT	NN	.
		NN	VBS	JJ	IN		VB	
					JJ			
					RB			



^ Brown foxes jumped over the fence .

Viterbi Decoding Example – 1 (continued)



Probability of a path (e.g. Top most path) = *P*(T) * *P*(W|T)

$$P(\hat{}) \cdot P(NN|\hat{}) \cdot P(NNS|NN) \cdot P(VBD|NNS) \cdot P(NN|VBD) \cdot P(DT|NN) \cdot P(NN|DT) \cdot P(.|NN) \cdot P(.)$$

*

$$P(\hat{}|\hat{}) \cdot P(brown|NN) \cdot P(foxes|NNS) \cdot P(jumped|VBD) \cdot P(over|NN) \cdot P(the|DT) \cdot P(fence|NN) \cdot P(.|.)$$

Viterbi Decoding Example – 2: Sentence "People Dance"

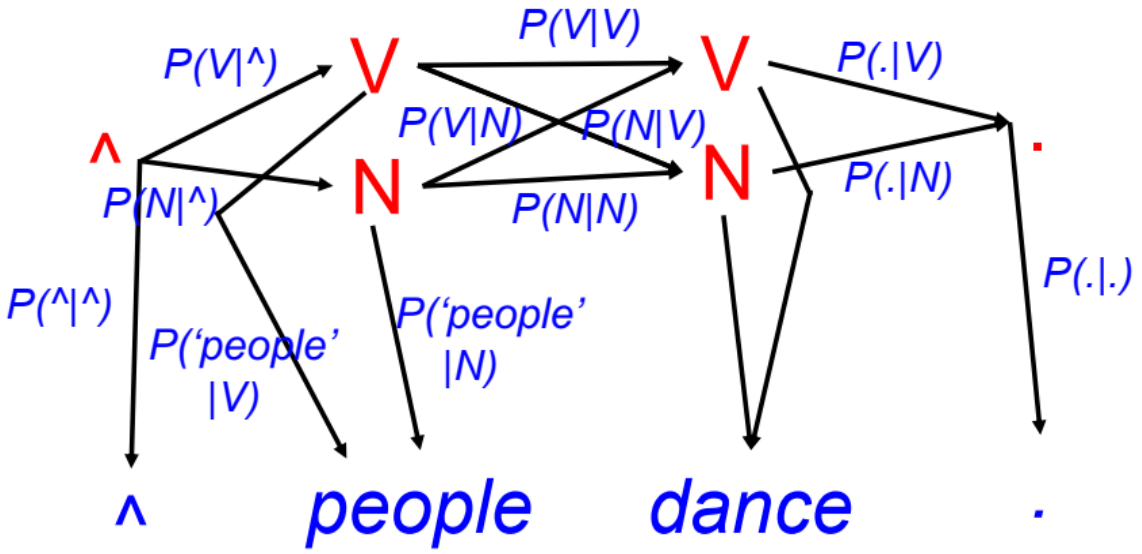
- '*people*' and '*dance*' can both be both nouns and verbs, as in
  - "*old_JJpeople_NNS*" ('*people*' as noun)
  - "*township_NNpeopled_VBNwith soldiers_NNS*" ('*people*' as verb)
- as well as
  - "*rules_NNSof_INclassical_JJdance_NN*" ('*dance*' as noun)
  - "*will_*VAUX*dance_VBwell_RB*"('*dance*' as verb)

- Possible Tags for "People Dance": for simplicity we take single letter tags-*N: noun, V: verb*:
  - *^ N N.*
  - *^ N V.*
  - *^ V N.*
  - *^ V V.*
- •We know that out of these, the second option *^ N V.* is the correct one. How do we get this sequence?

Viterbi Decoding Example 2: Sentence "People Dance" (continued)

❖Step 1: Trellis

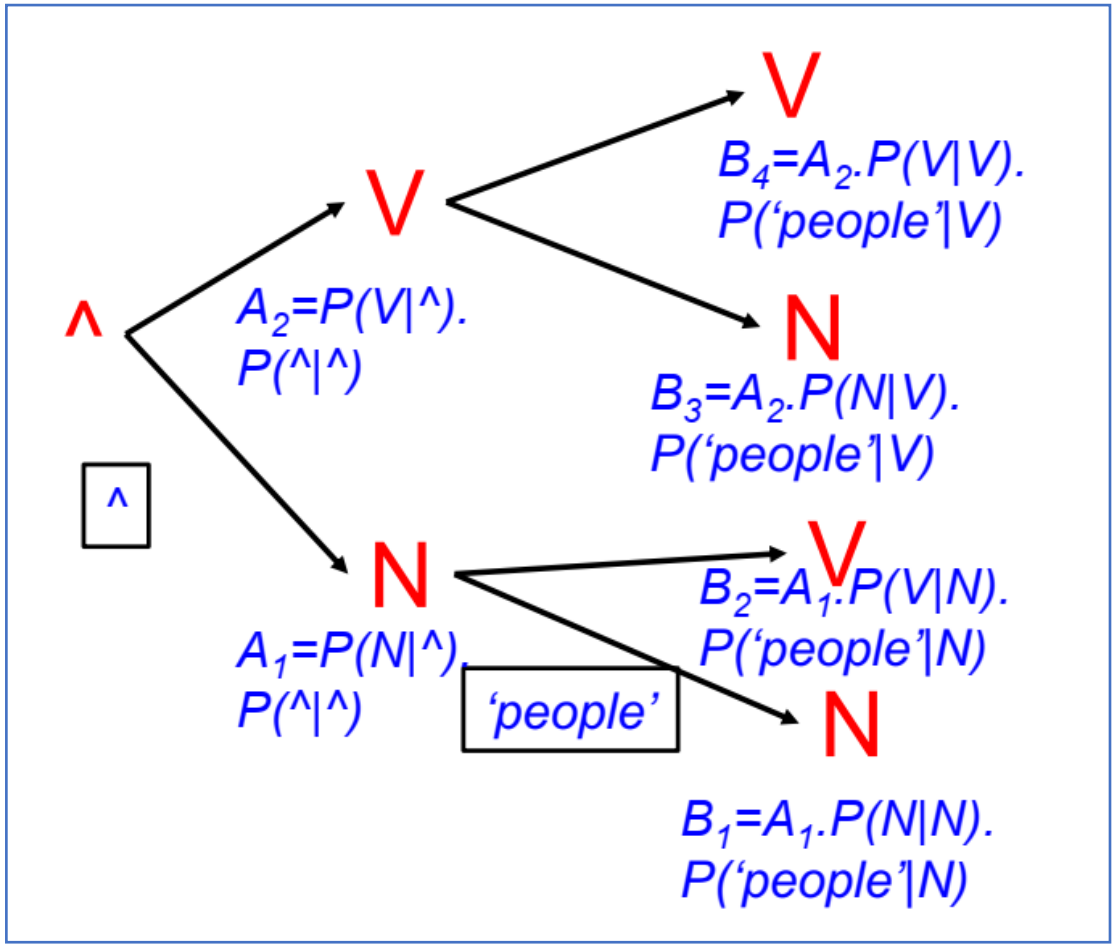
Columns of tags on each input word with transition arcs going from tags (states) to tags in consecutive columns and output arcs going from tags to words (observations)



Viterbi Decoding Example 2: Sentence "People Dance" (continued)

❖Aim: select the highest probability path

From 4 possibilities; A s and B s are accumulated probabilities



Some numerical values: hypothetical but not unrealistic

- Calculations:
- When it comes to the start of the sentence, most sentences start with a noun. So lets have

$P(N|^)=0.8$, $P(V|^)=0.2$ and of course $P('^'|^)=1.0$

- Then

$A_1=0.8$, $A_2=0.2$

Viterbi Decoding Example 2: Sentence "People Dance" (continued)

Encounter "people": more probabilities

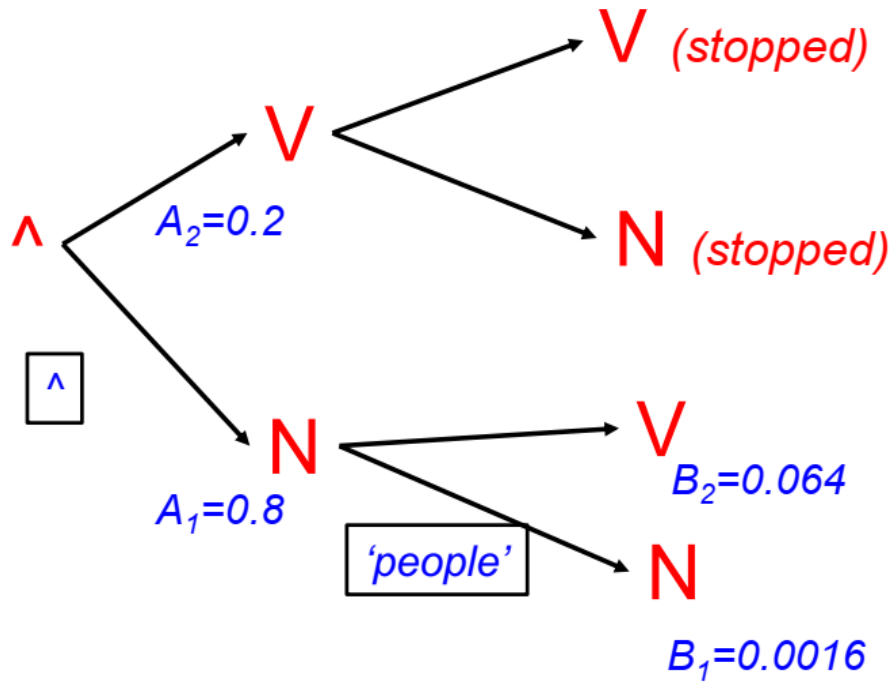
- Transition from *N* to *N* is less common than to *V*.
- Transition from *V* to *V* - as in auxiliary verb to main verb-is quite common (e.g., *is going*).
- *V* to *N* too is common - as in case of a nominal object following the verb *(going home).*
- Following plausible transition probabilities:
  - ***P(N|N)=0.2, P(V|N)=0.8, P(V|V)=0.4, P(N|V)=0.6***
- We also need lexical probabilities. 'people' appearing as verb is much less common than its appearing as noun. So let us have
  - ***P('people'|N)=0.01, P('people'|V)=0.001***
- Note: *N N*: golf club, cricket bat, town people
  - town people ambiguity "The **town people** visited was deserted"/ "**Town People** will not be able to live here"
- V V combination examples:
  - I **avoid eating** after 10 PM.
  - He **regrets losing** his temper at the meeting.
  - I **recommend visiting** Central Park.

Calculate *Bs*

B_1=0.8.0.2.0.01=0.0016 (approx.)
B_2=0.8.0.8.0.01=0.064 (approx.)
B_3=0.2.0.6.0.001=0.00012
B_4=0.2.0.4.0.001=0.00008

Reduced Viterbi Configuration

Heart of Decoding ⍰linear time



Viterbi Decoding Example 2: Sentence "People Dance" (continued)

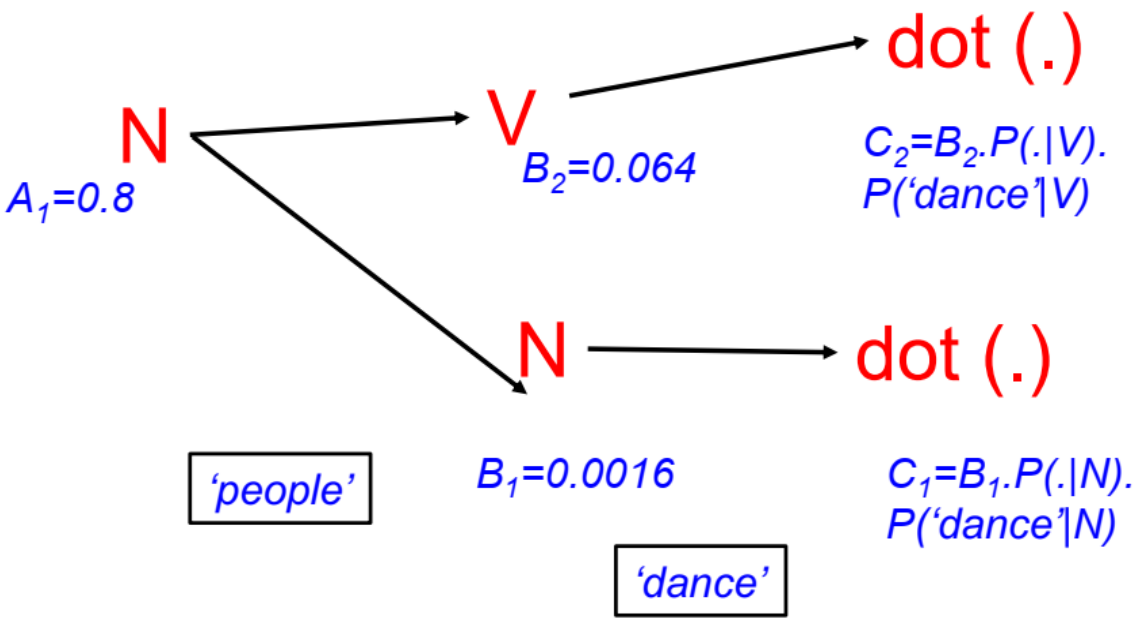
More probabilities needed

We can give equal probabilities to sentences ending in noun and verb. Also, 'dance' as verb is more common than noun.
P(.|N)=0.5=P(.|V)
P('dance'|N)=0.001
P('dance'|V)=0.01

Examples:

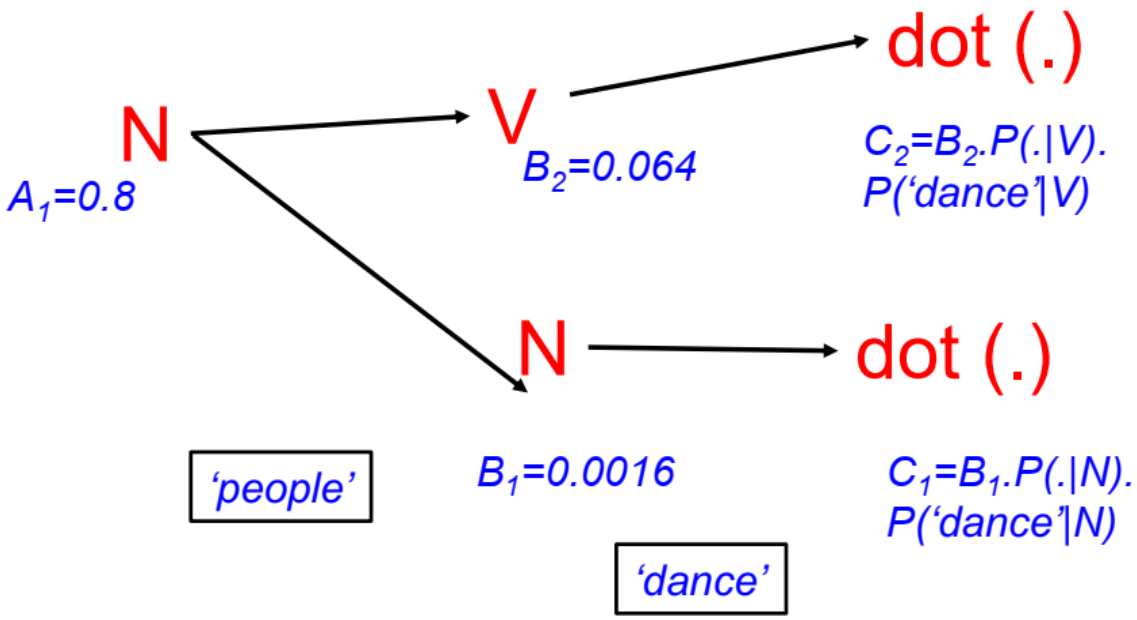
Dance (Noun) is a way of life.
Kamal Hasan dances (verb) well.

Next word: '*dance*'



Viterbi Decoding Example 2: Sentence "People Dance" (continued)

Best Path: ^ N V .

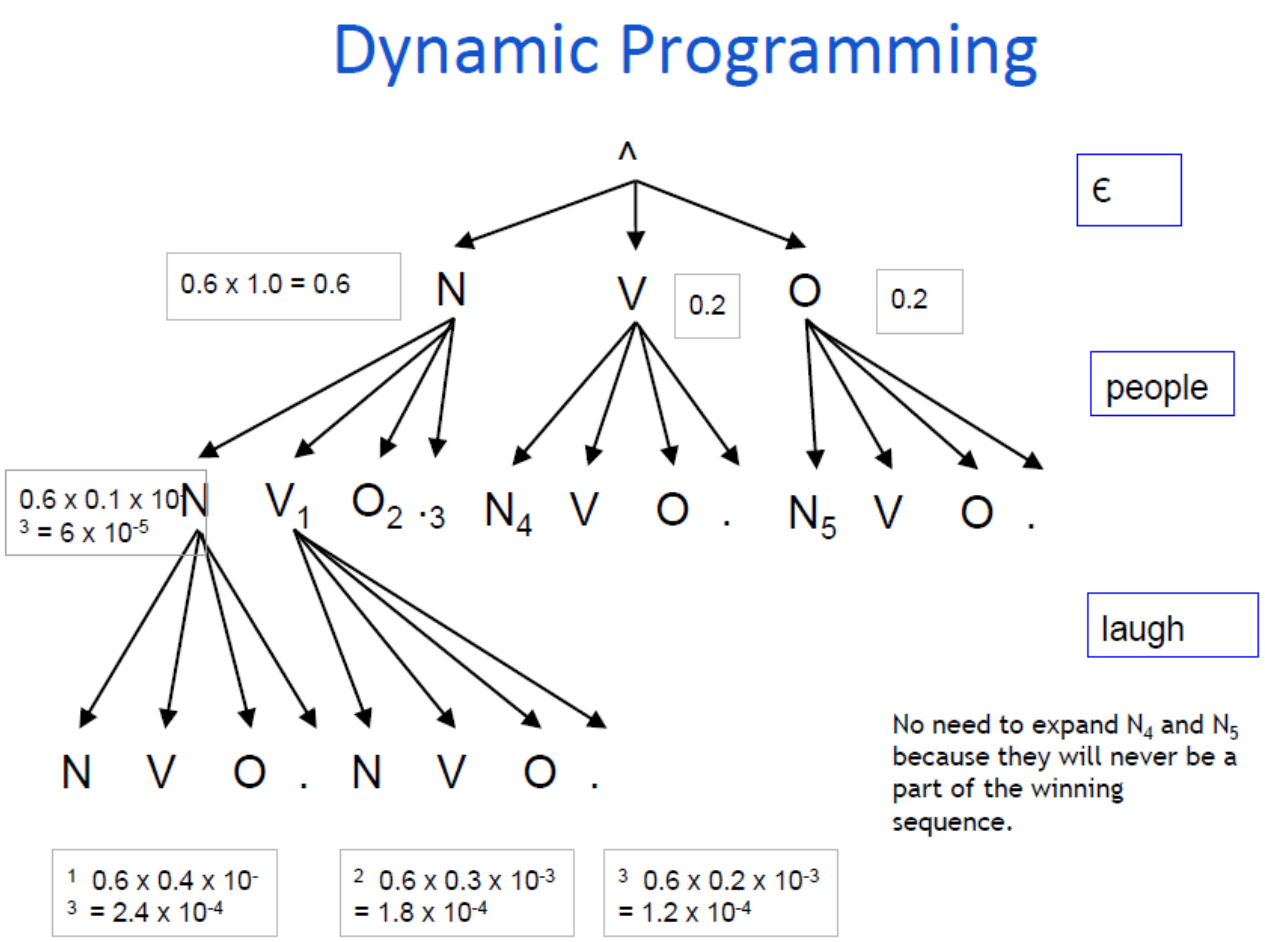


- *C1=0.0016.0.5.0.001=0.0000008*
- *C2=0.064.0.5.0.01=0.00032*

Decoding: Computational Complexity

- If we have to get the probability of each sequence and then find maximum among them, we would run into exponential number of computations.
- If |s| = #states (tags + ^ + .)
- and |o| = length of sentence (words + ^ + .)
- Then, #sequences = $s^{(|o|-2)}$
- But, a large number of partial computations can be reused using Dynamic Programming.

Dynamic Programming



Markov Model Applications

- **Parts of Speech Tagging**
- **Named Entity Recognition**
- **Application in communication:** message sent is $(s_1,\ldots,s_m)$ but we receive $(x_1,\ldots,x_m)$. Compute what is the most likely message sent ?
- **Application in speech recognition:** word said is $(s_1,\ldots,s_m)$ but we recorded $(x_1,\ldots,x_m)$. Compute what is the most likely word said ?

Problems faced in HMM based POS tagging

All problems are due to SPARSITY; following are the kinds of sparsity

- (a) unseen words (‘delay’ in training corpus, but not ‘procrastination’)
- (b) Different form of the word-morphology (corpus has ‘predictable’, but not ‘unpredictable’)
- (c) has code mixing (‘aap mujhe advice di jiye’-‘you give me advice)
- (d) The corpus does not have the particular word-tag combination (‘people’ as verb)

Limitations of HMM

- Unknown Words
  - The emission probabilities are not known
    - So, Its not possible to use viterbi decoding
  - Possible Solution
    - Use some morphological clues (capitalization, suffix) to assign a more calculated guess
    - eg. word ending with 'ed' etc.

- Limited Context
  - Assuming first order Markov assumption, each item depends on the previous item only
    1. "its clearly **marked**" : verb, past participle
    2. "he clearly **marked**" : verb, past tense
  - Just the previous context is not sufficient for tagging the word "marked"
  - Possible Solution
    - Use higher order model, combine various n-gram models to avoid sparseness problem

References

- A tutorial on Hidden Markov Models and selected applications in Speech Recognition by Lawrence R. Rabiner (highly cited tutorial)
- A friendly introduction to Bayes Theorem and Hidden Markov Models by Serrano Academy