Hidden Markov Models

CIS 530

University of Pennsylvania

Parts of Speech (POS)

- Parts of speech (POS) classify words into different syntactic categories

- Dionysius Thrax of Alexandria (c. 100 B.C.) gave description of 8 categories
  - Noun, Verb, Pronoun, Preposition, Adverb, Conjunction, Participle & Article

- Today, Penn TreeBank (Mitch Marcus et al. ’93) define 45 word classes

- Gives information about the word and also its neighbors
  - Nouns are likely to be preceded by determiners or adjectives
  - Verbs are mostly preceded by nouns

- Influences morphological affixes and pronunciation
  - Eg, content as Noun: *CONtent* vs. Adjective: *conTENT*

Parts of Speech tagging

- Computational method to assign POS tags to words in a sentence

- Challenges:
  - Ambiguity: Same words can have different POS tags in different contexts.
    - Eg. I can can a can

  - Open Worldness: New words are invented whose POS tags are to be determined from context
    - iPhone, Google
    - gonna, ikr, fb, lolololol …

Hidden Markov Models (HMM)

- **Sequence Model**

Input:	Let	us	learn	about	HMMs
Output Labels	VB	PRP	VB	IN	NNP

- **Probabilistic Model**
    - Compute a probability distribution over all possible sequence of labels

Let	us	learn	about	HMMs	
VB	PRP	VB	IN	NNP	p=0.45
IN	VB	VB	NN	DT	p=0.03
		...			
PRP	.	NN	IN	WP	p=0.00006

Motivation for using HMMs

Why not build a classifier that maps input sequences to output sequences?

- Input sequences are of different lengths
- Output at different time steps are not independent

Let	us	learn	about	HMMs
VB	PRP	VB	IN	NNP

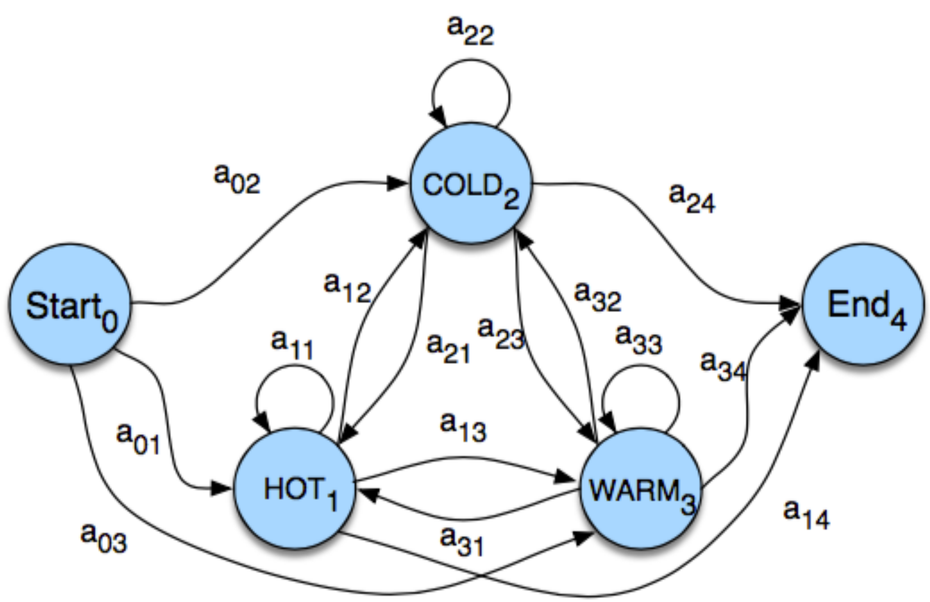
- Important in practice
  - Text Processing
  - Speech Processing
  - DNA Analysis
  - ..... basically any application requiring modeling sequences

Markov Chains
aka
Observed Markov Models

Observed Markov Models

- Special case of **Weighted Finite Automaton**
  - Set of States
  - Set of transitions between states

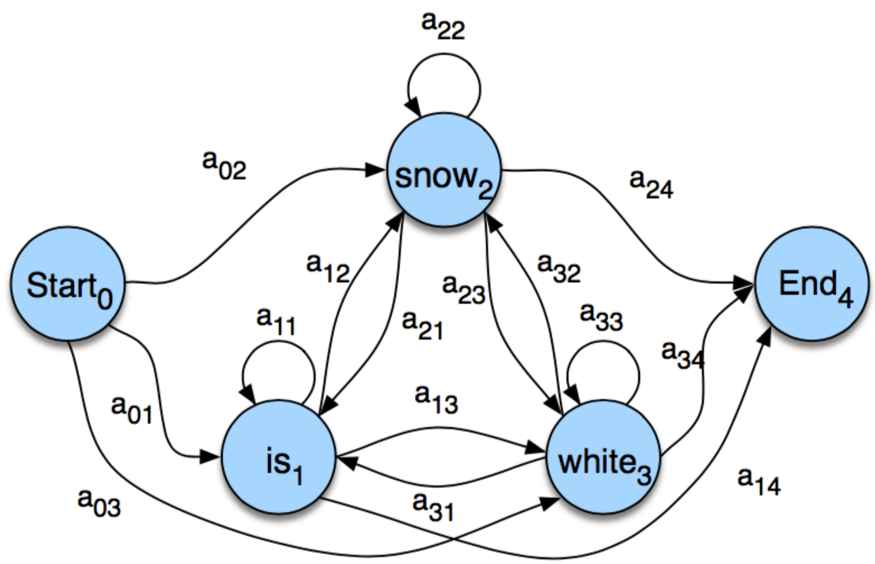
- In **Markov Chains**
  - Weights are probabilities
  - Therefore, sum of all probabilities on arcs leaving a node = 1



$$a_{11} + a_{12} + a_{13} = 1$$
$$a_{21} + a_{22} + a_{23} = 1$$
$$\ldots$$

For eg.
- Cold, Warm, Hot, Hot, Hot
- Warm, Hot, Hot, Cold, Cold

Observed Markov Models



Markov chain assigns probabilities to words $w_1 \dots w_n$

is white snow
white is snow
is snow white

Bigram Language Model!

Markov Chains are Probabilistic Graphical Models

$Q = q_1 q_2 \dots q_N$	a set of N **states**
$A = a_{01} a_{02} \dots a_{n1} \dots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
q_0, q_F	a special **start state** and **end (final) state** that are not associated with observations

Observed Markov Models

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{01} a_{02} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
q_0, q_F	a special **start state** and **end (final) state** that are not associated with observations

Probability of state (at any given time-step) only depends on the previous state:

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

Probability of state q_i given all previous states

Probability of state q_i only depends on previous state, q_{i-1}

Hidden Markov Models

- Used when events in the world are not directly observable
  - Parts-of-Speech (POS) tags
  - Reason about both, **observed** (ex. words) and **hidden** (ex. POS tags)

Observed Events	Let	us	learn	about	HMMs
Hidden Events	VB	PRP	VB	IN	NNP

- HMMs model this as a process where hidden events cause the occurrence of observed events

Jason Eisner's Experiment

Year 2799; You are a climatologist

Job: Estimate the weather each day

Jason's diary: How many ice-creams he ate everyday in 2007

Unobserved Events

Observed Events

Jason Eisner’s Experiment

Input: Sequence of observations, O

Ice-creams eaten each day

Output: Sequence of *hidden* states, Q

Weather state each day

Unobserved Event

HOT → COLD → COLD → HOT → COLD

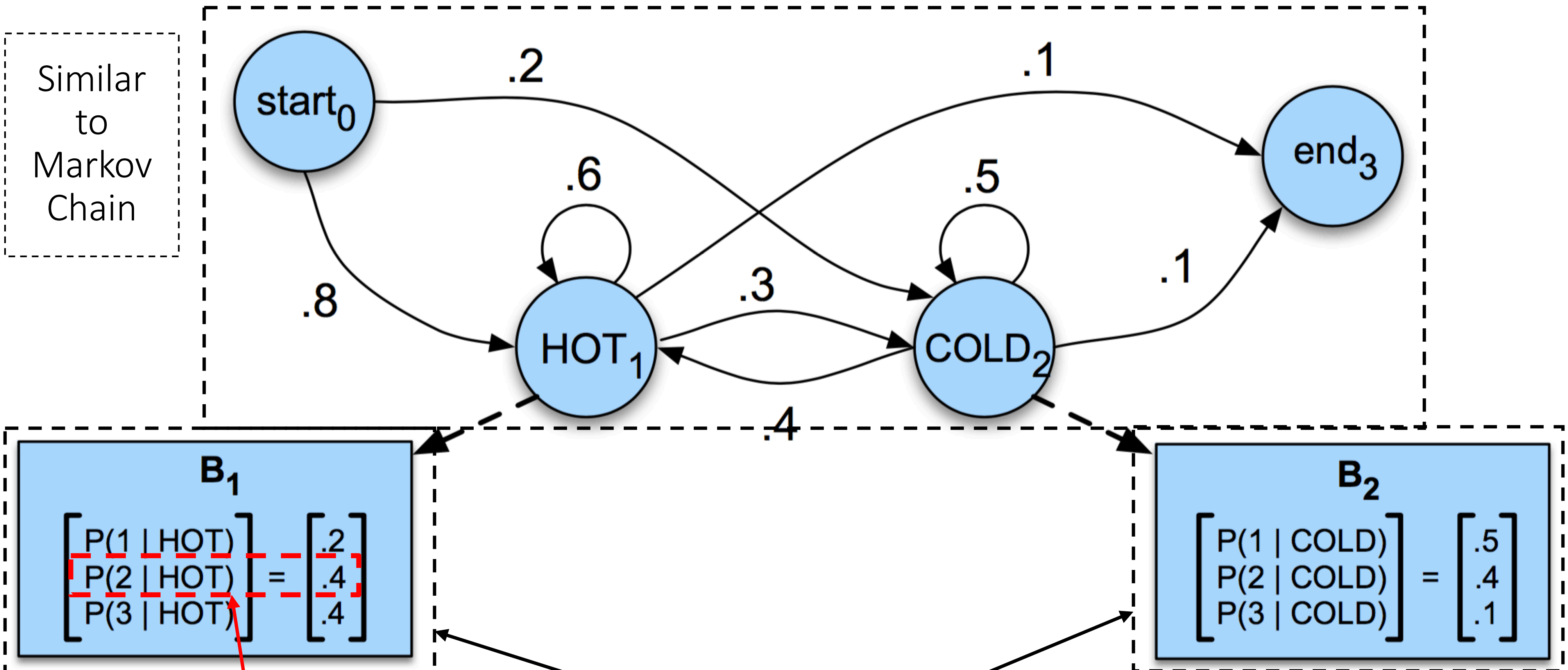
↓ ↓ ↓ ↓ ↓

2 2 1 3 2

Observed Event

Hidden Markov Models

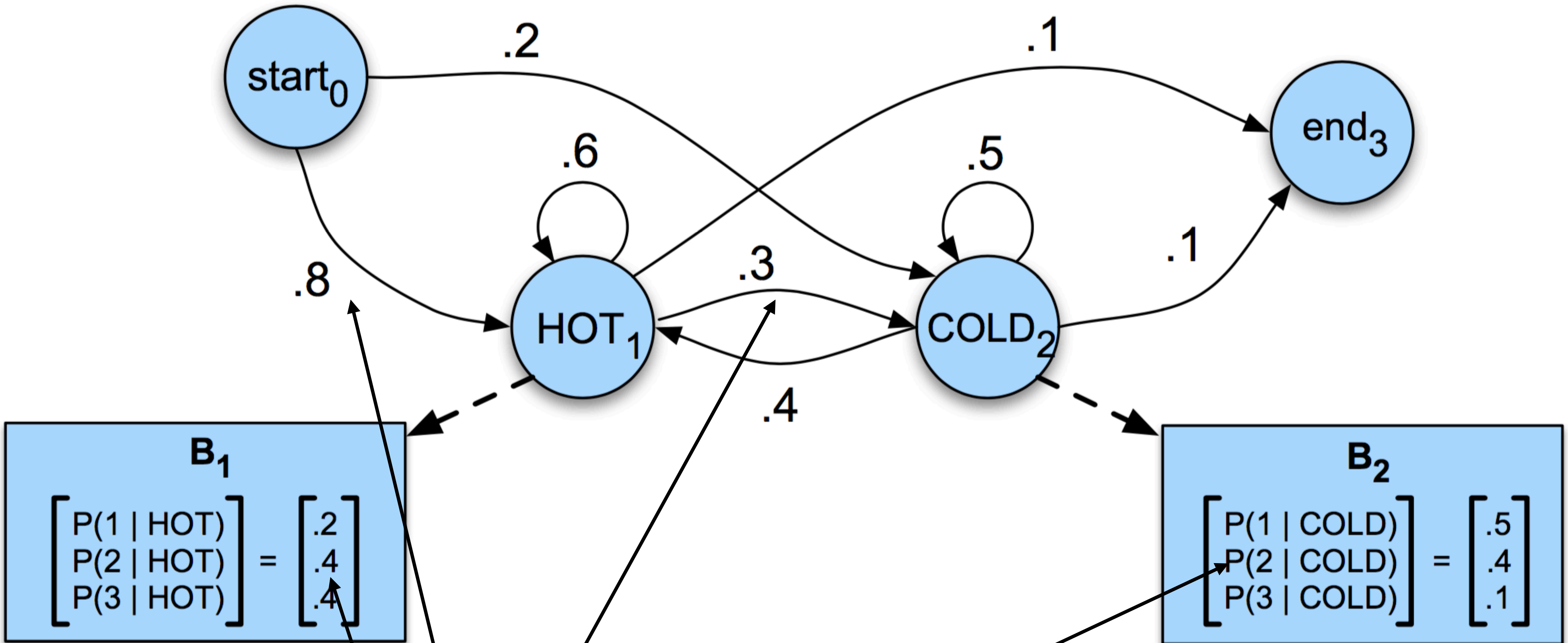
Similar to Markov Chain



Given the day is hot, the probability of Jason eating two ice-creams = 0.4

Emission Probabilities

Hidden Markov Models



HOT → COLD → COLD → HOT → COLD

↓ ↓ ↓ ↓ ↓

2 2 1 3 2

Hidden Markov Models

$Q = q_1 q_2 \dots q_N$	a set of N **states**
$A = a_{11} a_{12} \dots a_{n1} \dots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \dots o_T$	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 i
q_0, q_F	a special **start state** and **end (final) state** that are not associated with observations, together with transition probabilities $a_{01} a_{02} \dots a_{0n}$ out of the start state and $a_{1F} a_{2F} \dots a_{nF}$ into the end state

HMM is denoted by $\lambda = (A, B)$

Hidden Markov Models

$Q = q_1 q_2 \dots q_N$	a set of N **states**
$A = a_{11} a_{12} \dots a_{n1} \dots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \dots o_T$	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 i
q_0, q_F	a special **start state** and **end (final) state** that are not associated with observations, together with transition probabilities $a_{01} a_{02} \dots a_{0n}$ out of the start state and $a_{1F} a_{2F} \dots a_{nF}$ into the end state

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

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

Hidden Markov Models

Three fundamental problems:

1. **Likelihood Estimation**: Given an HMM, $\lambda = (A, B)$ and observation sequence O, determine $\mathrm{P}(\mathrm{O}|\lambda)$
2. **Decoding:** Given an HMM, $\lambda = (A, B)$ and observation sequence O, determine best hidden state sequence, Q^*
3. **Learning:** Given (many) observation sequences, O, and set of possible hidden states, estimate the parameters of HMM, $\lambda = (A, B)$

Likelihood Computation

Given an HMM, $\lambda = (A, B)$ and observation sequence O, determine $\mathrm{P}(\mathrm{O}|\lambda)$

Say, O = 3 1 3

For now, let us assume we also know the hidden state sequence, **Q = HOT HOT COLD**

$$\begin{array}{ccccc} \text{HOT} & \rightarrow & \text{HOT} & \rightarrow & \text{COLD} \\ {\scriptstyle 0.4}\downarrow & & {\scriptstyle 0.1}\downarrow & & {\scriptstyle 0.1}\downarrow \\ 3 & & 1 & & 3 \end{array}$$

Likelihood Computation

When hidden state sequence, Q is given:

$$
\begin{array}{ccccc}
\text{HOT} & \rightarrow & \text{HOT} & \rightarrow & \text{COLD} \\
\downarrow{\scriptstyle 0.4} & & \downarrow{\scriptstyle 0.1} & & \downarrow{\scriptstyle 0.1} \\
3 & & 1 & & 3
\end{array}
$$

$$P(O|Q) = \prod_{i=1}^{i=T} P(o_i|q_i)$$

P(3 1 3 | hot hot cold) = P(3 | hot) * P(1 | hot) * P(3 | cold)

Likelihood Computation

But, Q is not given. Possible options ...

$$\begin{array}{ccccc} \text{HOT} & \rightarrow & \text{COLD} & \rightarrow & \text{COLD} \\ \downarrow & & \downarrow & & \downarrow \\ 3 & & 1 & & 3 \end{array}$$

$$\begin{array}{ccccc} \text{HOT} & \rightarrow & \text{HOT} & \rightarrow & \text{HOT} \\ \downarrow & & \downarrow & & \downarrow \\ 3 & & 1 & & 3 \end{array}$$

$$\begin{array}{ccccc} \text{COLD} & \rightarrow & \text{HOT} & \rightarrow & \text{COLD} \\ \downarrow & & \downarrow & & \downarrow \\ 3 & & 1 & & 3 \end{array}$$

$$\begin{array}{ccccc} \text{HOT} & \rightarrow & \text{COLD} & \rightarrow & \text{HOT} \\ \downarrow & & \downarrow & & \downarrow \\ 3 & & 1 & & 3 \end{array}$$

How many Q sequences are possible?

Likelihood Computation

But, Q is not given. Possible options ...

$$\begin{array}{ccccc} \text{HOT} & \rightarrow & \text{COLD} & \rightarrow & \text{COLD} \\ \downarrow & & \downarrow & & \downarrow \\ 3 & & 1 & & 3 \end{array}$$

$$\begin{array}{ccccc} \text{HOT} & \rightarrow & \text{HOT} & \rightarrow & \text{HOT} \\ \downarrow & & \downarrow & & \downarrow \\ 3 & & 1 & & 3 \end{array}$$

$$\begin{array}{ccccc} \text{COLD} & \rightarrow & \text{HOT} & \rightarrow & \text{COLD} \\ \downarrow & & \downarrow & & \downarrow \\ 3 & & 1 & & 3 \end{array}$$

$$\begin{array}{ccccc} \text{HOT} & \rightarrow & \text{COLD} & \rightarrow & \text{HOT} \\ \downarrow & & \downarrow & & \downarrow \\ 3 & & 1 & & 3 \end{array}$$

$$2^3 = 8$$

Likelihood Computation

$$P(O) = \sum_{Q} P(O, Q)$$

Marginalize over all possible Q

$$P(O, Q) = P(O|Q)P(Q)$$

$$P(O) = \sum_{Q} P(O|Q)P(Q)$$

$$P(O|Q) * P(Q) \quad = \quad \prod_{i=1}^{i=T} P(o_i|q_i) \; * \prod_{i=1}^{i=T} P(q_i|q_{i-1})$$

Likelihood Computation

$$P(O) = \sum_{Q} P(O|Q)P(Q)$$

Marginalize over all possible Q

$$P(O|Q) * P(Q) \;=\; \prod_{i=1}^{i=T} P(o_i|q_i) \;*\; \prod_{i=1}^{i=T} P(q_i|q_{i-1})$$

Number of possible Qs: N^T

Consider POS tagging, N = 45, T ≈ 20

$N^T = 45^{20} =$
1,159,445,300,000,000,000,000,000,000,000,000,000

Likelihood Computation

$$P(O) = \sum_{Q} P(O|Q)P(Q)$$

Marginalize over all possible Q

$$P(O|Q) * P(Q) = \prod_{i=1}^{i=T} P(o_i|q_i) * \prod_{i=1}^{i=T} P(q_i|q_{i-1})$$

Number of possible Qs: N^T

Consider POS tagging, N = 45, T ≈ 20

$N^T =$ 1,159,445,300,000,000,000,000,000,000,000,000,000

Exponential Running Time: Too expensive!!

Likelihood Computation

inalize over
ossible Q

$l_{i-1})$

$N^T = $ 1,159,445,300,000,000,000,000,000,000,000,000

Exponential Running Time: Too expensive!!

Likelihood Computation: The Forward Algorithm

$$\begin{array}{ccccc} \text{HOT} & \rightarrow & \text{COLD} & \rightarrow & \text{COLD} \\ \downarrow & & \downarrow & & \downarrow \\ 3 & & 1 & & 3 \end{array} \qquad \begin{array}{ccccc} \text{HOT} & \rightarrow & \text{COLD} & \rightarrow & \text{HOT} \\ \downarrow & & \downarrow & & \downarrow \\ 3 & & 1 & & 3 \end{array}$$

$$P(O) \;\; = P(o_1|q_1)P(q_1|q_{1-1}) + P(o_2|q_2)P(q_2|q_{2-1}) \\ + P(o_3|q_3)P(q_3|q_2)$$

Likelihood Computation: The Forward Algorithm

HOT $\rightarrow$ COLD $\rightarrow$ COLD   HOT $\rightarrow$ COLD $\rightarrow$ HOT

$\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$

3   1   3   3   1   3

$$P(O) = P(o_1|q_1)P(q_1|q_{1-1}) + P(o_2|q_2)P(q_2|q_{2-1}) + P(o_3|q_3)P(q_3|q_2)$$

At t=3, only state at t=2 matters

Let's try to fold all computations before t=2 into single quantity

Dynamic Programming!!!

Dynamic Programming

- Coined by Richard Bellman in 1940s
  - “My boss, Secretary of Defense, actually had a pathological fear and hatred of the word research”
  - “*Dynamic* has a very interesting property as an adjective, and that it's impossible to use the word dynamic in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. It's impossible!”

- Method for solving complex problems by breaking them down into simpler sub-problems and storing their solutions

- Technique of storing solutions to sub-problems instead of recomputing them is called ”**memoization**”

Dynamic Programming

Fibonacci Series

$$fib(n) = fib(n-1) + fib(n-2)$$

- fib(5)

➢ fib(4) + fib(3)

➢ (fib(3) + fib(2)) + (fib(2) + fib(1))

➢ ((fib(2) + fib(1)) + (fib(1) + fib(0))) + ((fib(1) + fib(0)) + fib(1))

➢ (((fib(1) + fib(0)) + fib(1)) + (fib(1) + fib(0))) + ((fib(1) + fib(0)) + fib(1))

- Instead of calling fib(3) multiple times, we should store it and lookup instead of recomputing

Likelihood Computation: The Forward Algorithm

$$\begin{array}{ccccc} \text{HOT} & \rightarrow & \text{COLD} & \rightarrow & \text{COLD} \\ \downarrow & & \downarrow & & \downarrow \\ 3 & & 1 & & 3 \end{array} \qquad \begin{array}{ccccc} \text{HOT} & \rightarrow & \text{COLD} & \rightarrow & \text{HOT} \\ \downarrow & & \downarrow & & \downarrow \\ 3 & & 1 & & 3 \end{array}$$

$$P(O) = P(o_1|q_1)P(q_1|q_{1-1}) + P(o_2|q_2)P(q_2|q_{2-1}) + P(o_3|q_3)P(q_3|q_2)$$

At t=3, only state at t=2 matters

Let's try to fold all computations before t=2 into single quantity

Dynamic Programming!!!

Likelihood Computation: The Forward Algorithm

HOT → COLD → COLD
↓ ↓ ↓
3 1 3

HOT → COLD → HOT
↓ ↓ ↓
3 1 3

COLD → COLD → COLD
↓ ↓ ↓
3 1 3

COLD → COLD → HOT
↓ ↓ ↓
3 1 3

Probability of observing output till t=2 and the state being $q_2 = COLD$

$$P(o_1, o_2, q_2 = COLD|\lambda) = \alpha_2(COLD)$$

Probability of observing output till time=t and the state being $q_t = j$

$$\alpha_t(j) = P(o_1, o_2 \ldots, o_t, q_t = j|\lambda)$$

Likelihood Computation: The Forward Algorithm

Probability of observing output till t and the state being $q_2 = j$

$$\alpha_t(j) = P(o_1, o_2 \dots, o_t, q_t = j|\lambda)$$

Using independence assumption of our model (and recursion):

P(A,B) = P(A|B)P(B)

Sum over all possible values $\mathbf{q}_t$

$$\alpha_{t+1}(i) = \sum_{j=1}^{N} \alpha_t(j) * a_{ji} * b_i(o_{t+1})$$

$P(o_{t+1}|q_{t+1} = i)$

$P(o_1, o_2 \dots, o_t, o_{t+1}, q_{t+1} = i|\lambda)$

$P(q_{t+1} = i|q_t = j)$

$P(o_1, o_2 \dots, o_t, q_t = j|\lambda)$

Likelihood Computation: The Forward Algorithm

$$\alpha_t(i) = \sum_{j=1}^{N} \alpha_{t-1}(j) * a_{ji} * b_i(o_t)$$

$$\alpha_t(i) = \ P(o_1, o_2 \ldots, o_t, q_t = i|\lambda)$$

$$P(o_1, o_2 \ldots, o_T|\lambda) \ = \sum_{i=1}^{N} P(o_1, o_2 \ldots, o_T, q_T = i|\lambda) * a_{iF}$$

$$P(O) = \ P(o_1, o_2 \ldots, o_T|\lambda) \ = \sum_{i=1}^{N} \alpha_T(i) * a_{iF}$$

Likelihood Computation: The Forward Algorithm

Recap, to compute P(O):

$$\alpha_t(i) = P(o_1, o_2 \ldots, o_t, q_t = i|\lambda)$$

Initialization: $$\alpha_1(i) = a_{0i} b_i(o_1) \quad 1 \leq i \leq N$$

Recursion: $$\alpha_t(i) = \sum_{j=1}^{N} \alpha_{t-1}(j) * a_{ji} * b_i(o_t)$$

Termination: $$P(O) = P(o_1, o_2 \ldots, o_T|\lambda) = \sum_{i=1}^{N} \alpha_T(i) * a_{iF}$$

Decoding – Viterbi Algorithm

Given an HMM, $\lambda = (A, B)$ and observation sequence O, determine best hidden state sequence, Q^*

Let	us	learn	about	HMMs	
VB	PRP	VB	IN	NNP	p=0.45
IN	VB	VB	NN	DT	p=0.03
			...		
PRP	.	NN	IN	WP	p=0.00006

Decoding – Viterbi Algorithm

Given an HMM, $\lambda = (A, B)$ and observation sequence O, determine best hidden state sequence, Q^*

$$Q^* = arg\max_{Q} P(Q|O)$$

$$Q^* = arg\max_{Q} \frac{P(O|Q)P(Q)}{P(O)}$$

$$\frac{P(O|Q)P(Q)}{P(O)} = P(O|Q)P(Q)$$

$$Q^* = arg\max_{Q} P(O|Q)P(Q)$$

Decoding – Viterbi Algorithm

$$Q^* = \underset{Q}{arg\max} \, P(O|Q)P(Q)$$

Number of possible Qs: N^T

Remember,

$$P(O) = \sum_{Q} P(O|Q)P(Q)$$

We know how to compute this sum efficiently!

Can we use the same intuition to compute the argmax efficiently?

Decoding – Viterbi Algorithm

Let us first try to find the max instead of the argmax

$$P^* = \max_Q P(Q|O) = \max_Q \frac{P(O,Q)}{P(O)} = \max_Q P(O,Q)$$

(Not exactly) Probability of the best hidden sequence up till time = t, ending in $q_t = j$

$$v_t(j) = \max_{q_1,\ldots,q_{t-1}} P(q_1, \ldots, q_{t-1}, o_1, o_2, \ldots, o_t, q_t = j|\lambda)$$

Decoding – Viterbi Algorithm

Probability of the best hidden sequence up till time = t, ending in $q_t = j$

$$v_t(j) = \max_{q_1,\dots,q_{t-1}} P(q_1, \dots, q_{t-1}, o_1, o_2, \dots, o_t, q_t = j|\lambda)$$

Use independence assumption to recurse:

Max over transitions from all possible q_t

$$v_{t+1}(i) = \max_{j=1,\dots,N} v_t(j) * a_{ji} * b_i(o_{t+1})$$

$P(o_{t+1}|q_{t+1} = i)$

$P(o_1, o_2 \dots, o_t, o_{t+1}, q_{t+1} = i|\lambda)$

$P(q_{t+1} = i|q_t = j)$

$P(q_1, \dots, q_{t-1}, o_1, o_2, \dots, o_t, q_t = j|\lambda)$

Decoding – Viterbi Algorithm

$$v_T(j) = \max_{q_1,\ldots,q_{T-1}} P(q_1, \ldots, q_{T-1}, o_1, o_2, \ldots, o_T, q_T = j|\lambda)$$

$$P^* = \max_{j=1,\ldots,N} v_T(j)$$

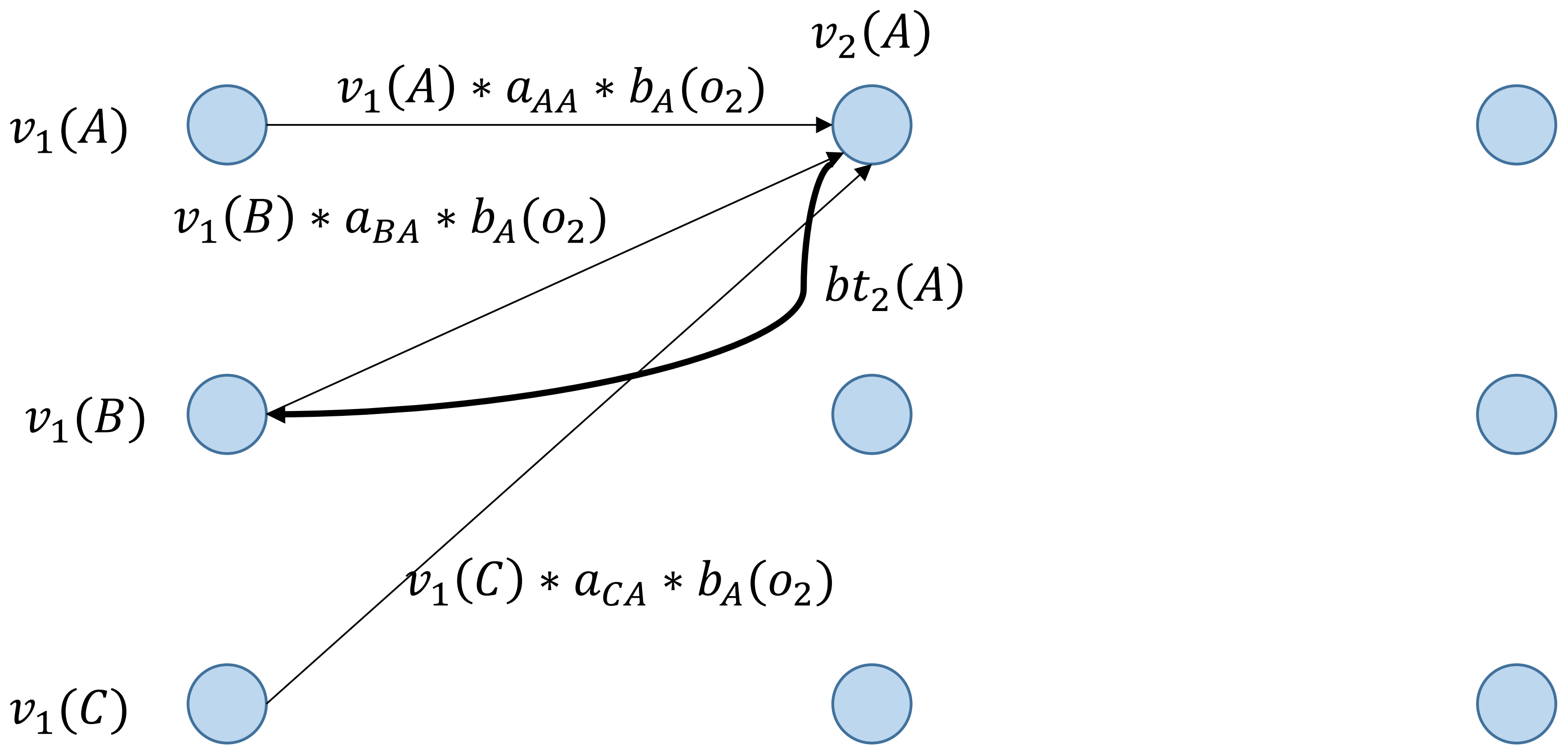
To find Q^*, add back-pointers

$$v_{t+1}(i) = \max_{j=1,\ldots,N} v_t(j) * a_{ji} * b_i(o_{t+1})$$

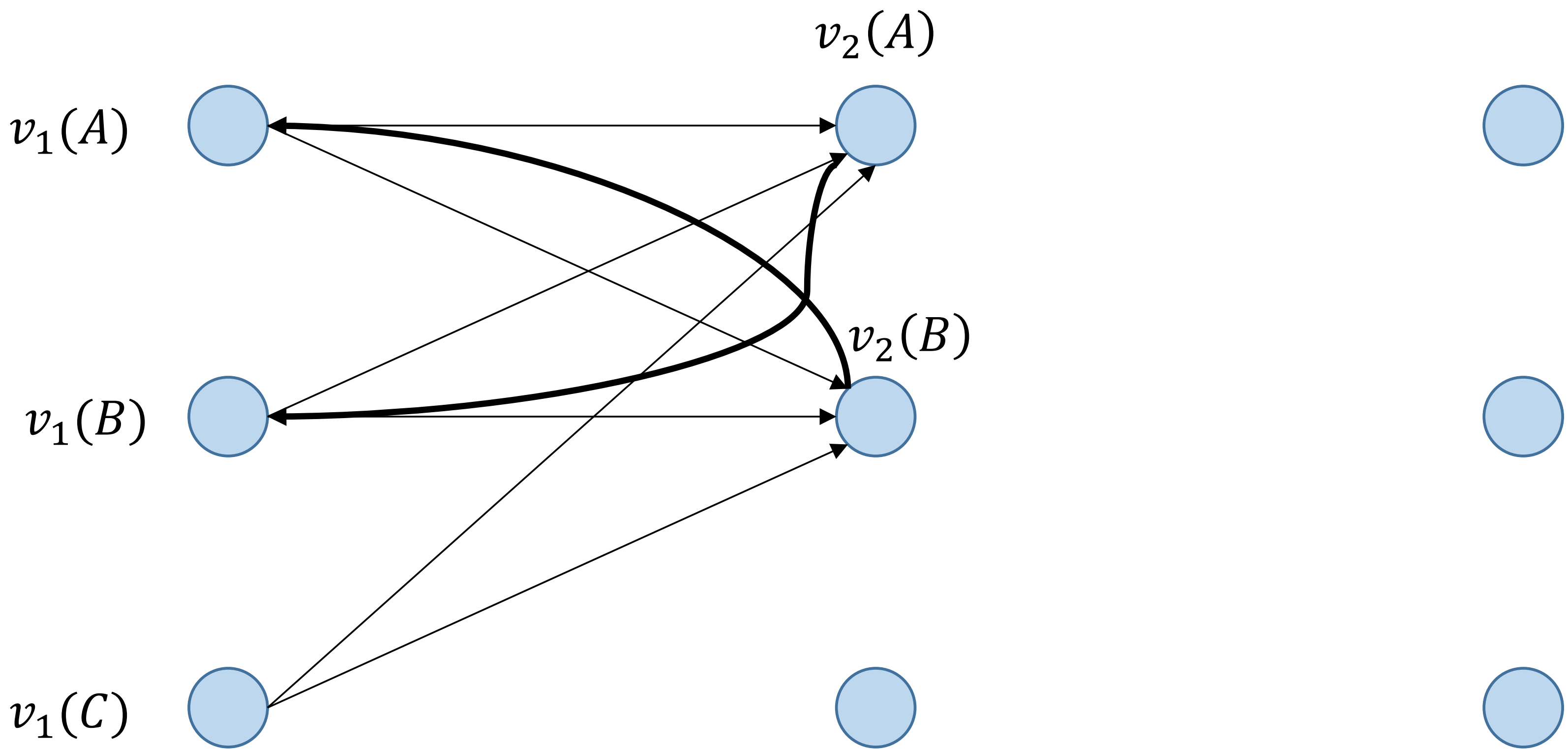
$$bt_{t+1}(i) = \underset{j=1,\ldots,N}{argmax}\, v_t(j) * a_{ji} * b_i(o_{t+1})$$

If ending the sequence at $q_{t+1}(i)$, what is the best q_t

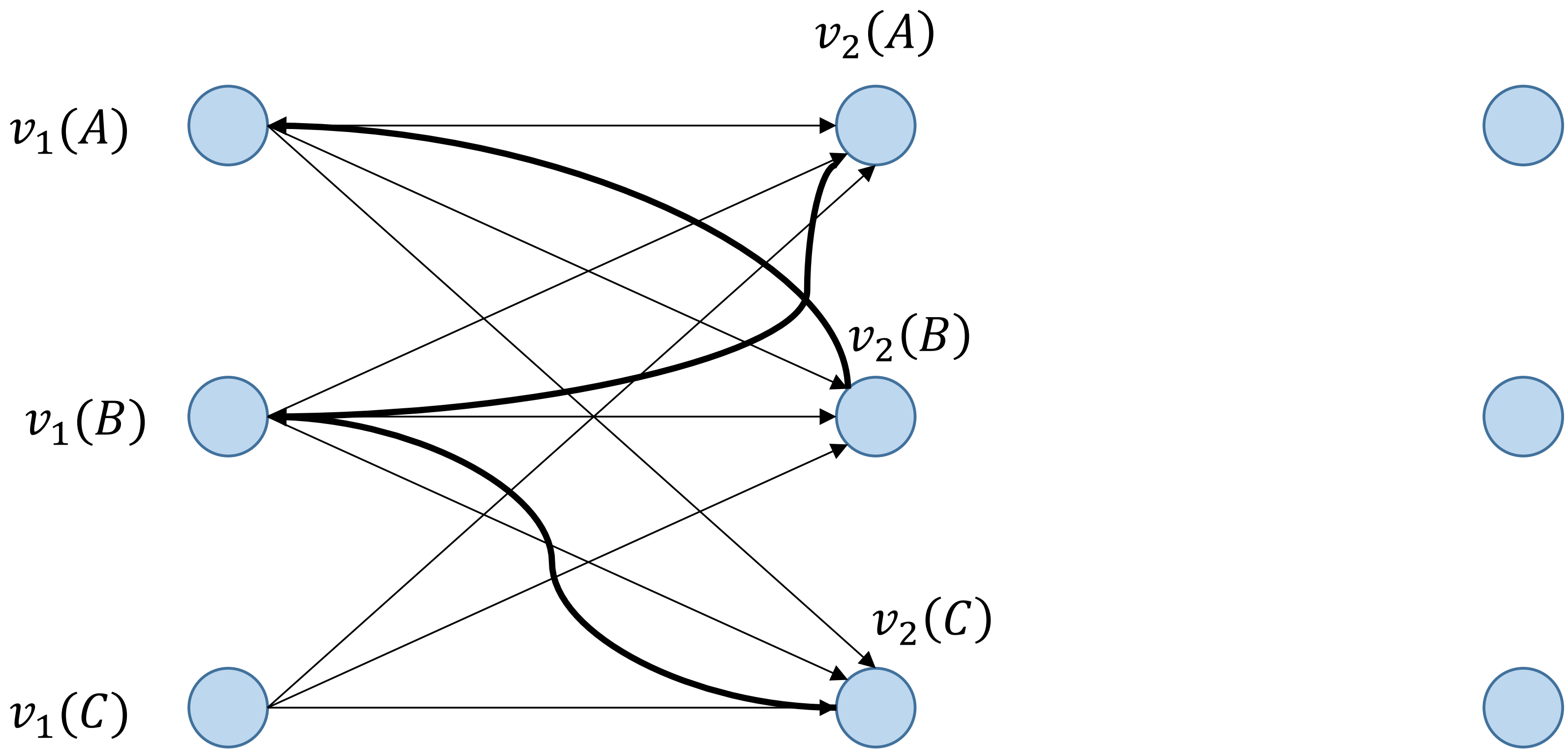
Decoding – Viterbi Algorithm



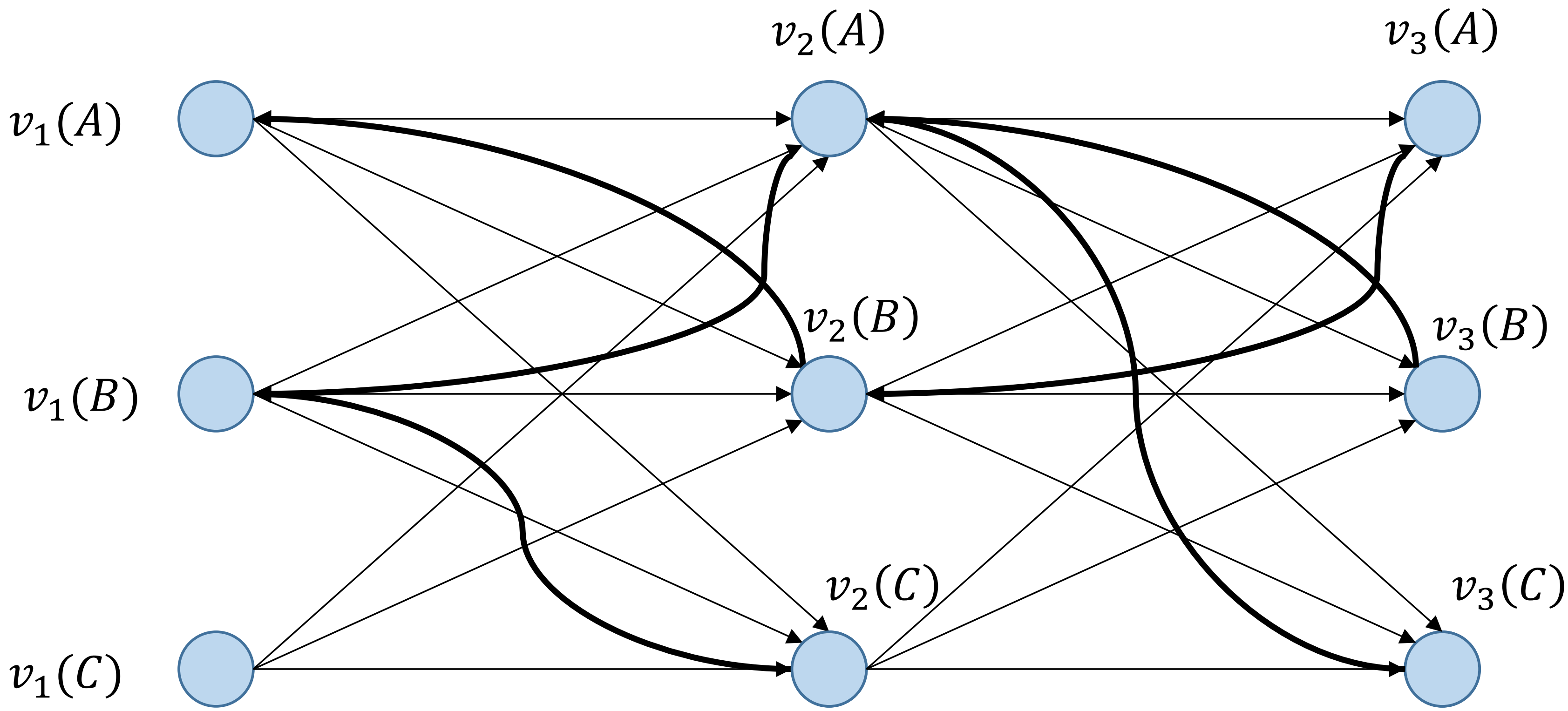
Decoding – Viterbi Algorithm



Decoding – Viterbi Algorithm

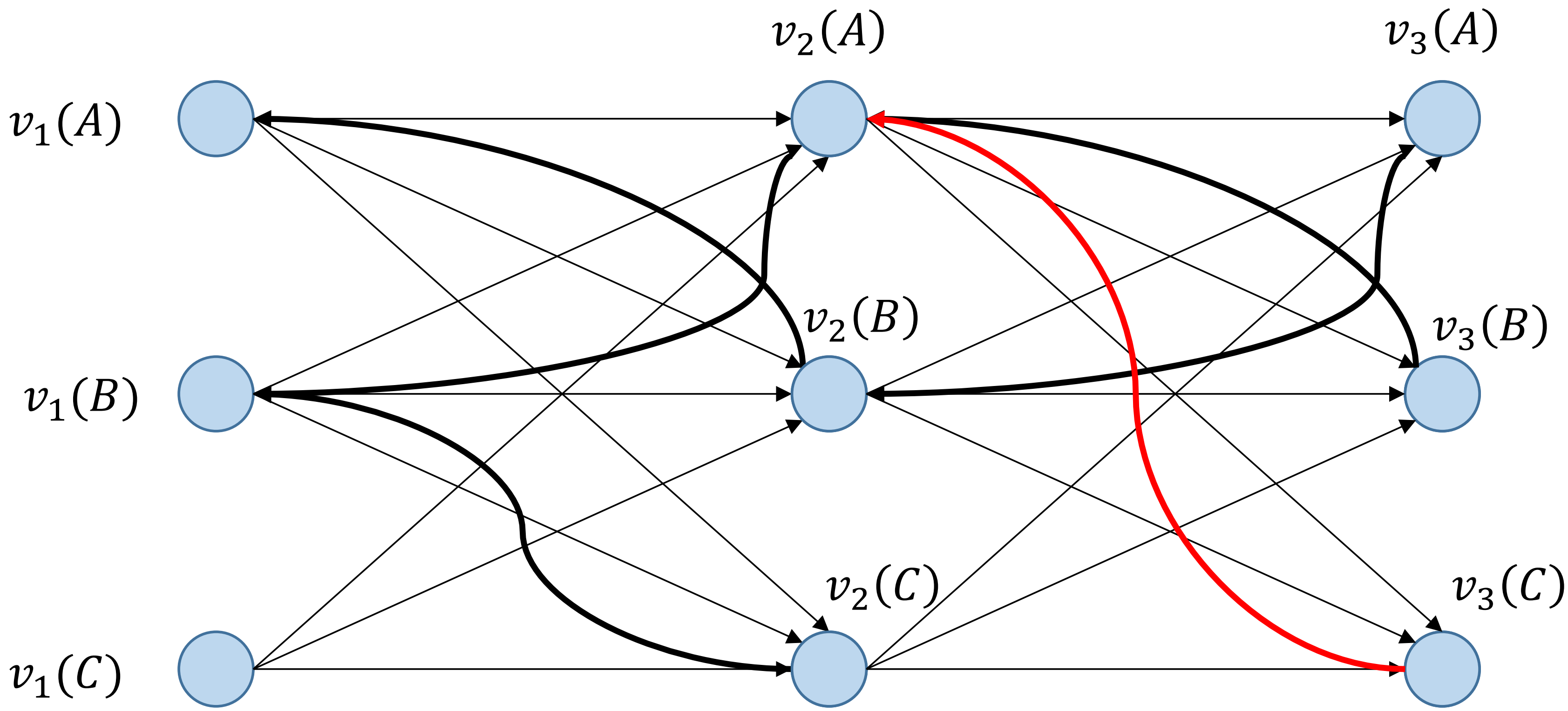


Decoding – Viterbi Algorithm



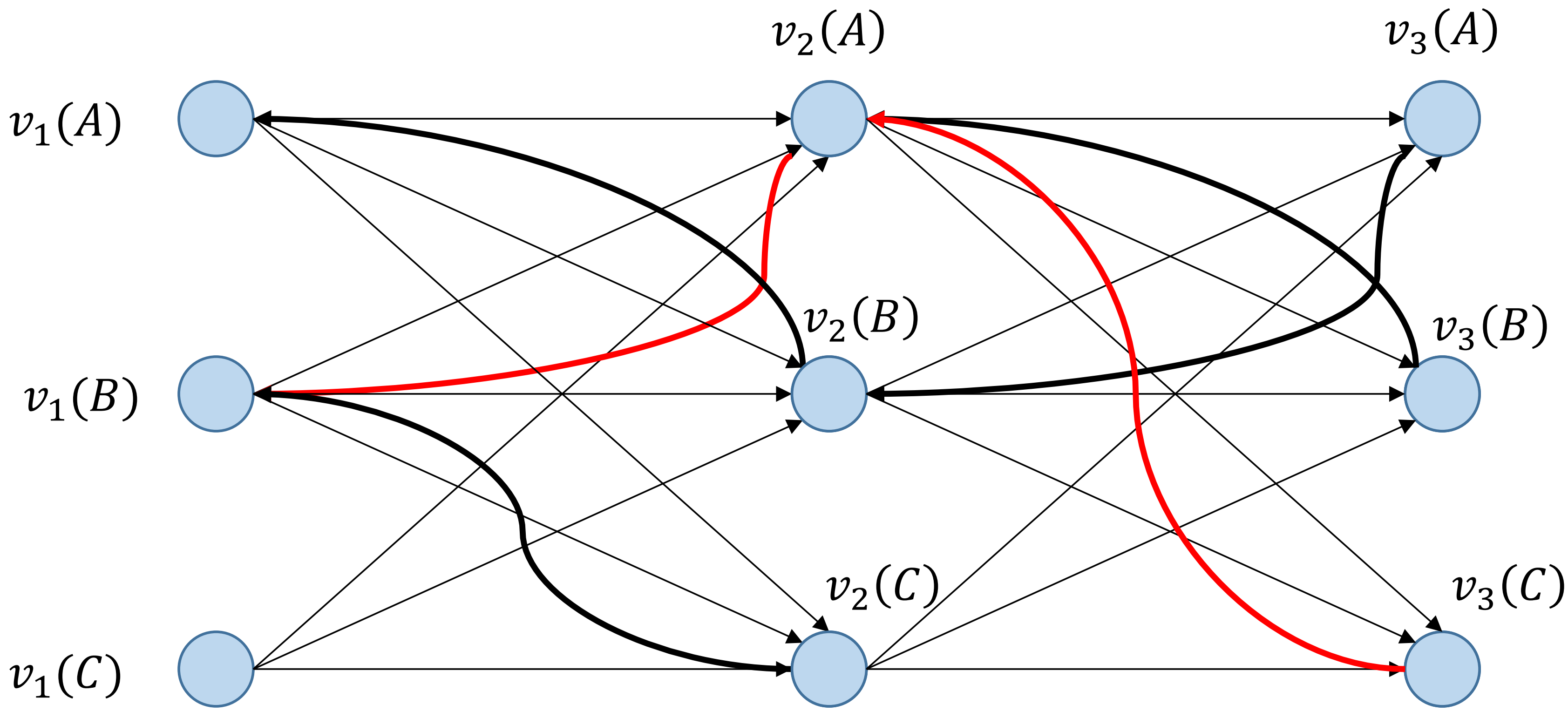
$$q^*_{T=3} = \underset{j=1,\ldots,N}{\operatorname{argmax}} v_{T=3}(j) = C$$

Decoding – Viterbi Algorithm



$$q^*_{T=3} = \underset{j=1,\ldots,N}{\operatorname{argmax}}\, v_{T=3}(j) = C$$

Decoding – Viterbi Algorithm



$$q^*_{T=3} = \underset{j=1,\dots,N}{\operatorname{argmax}}\, v_{T=3}(j) = C$$

$$Q^* = B\ A\ C$$

Learning an HMM

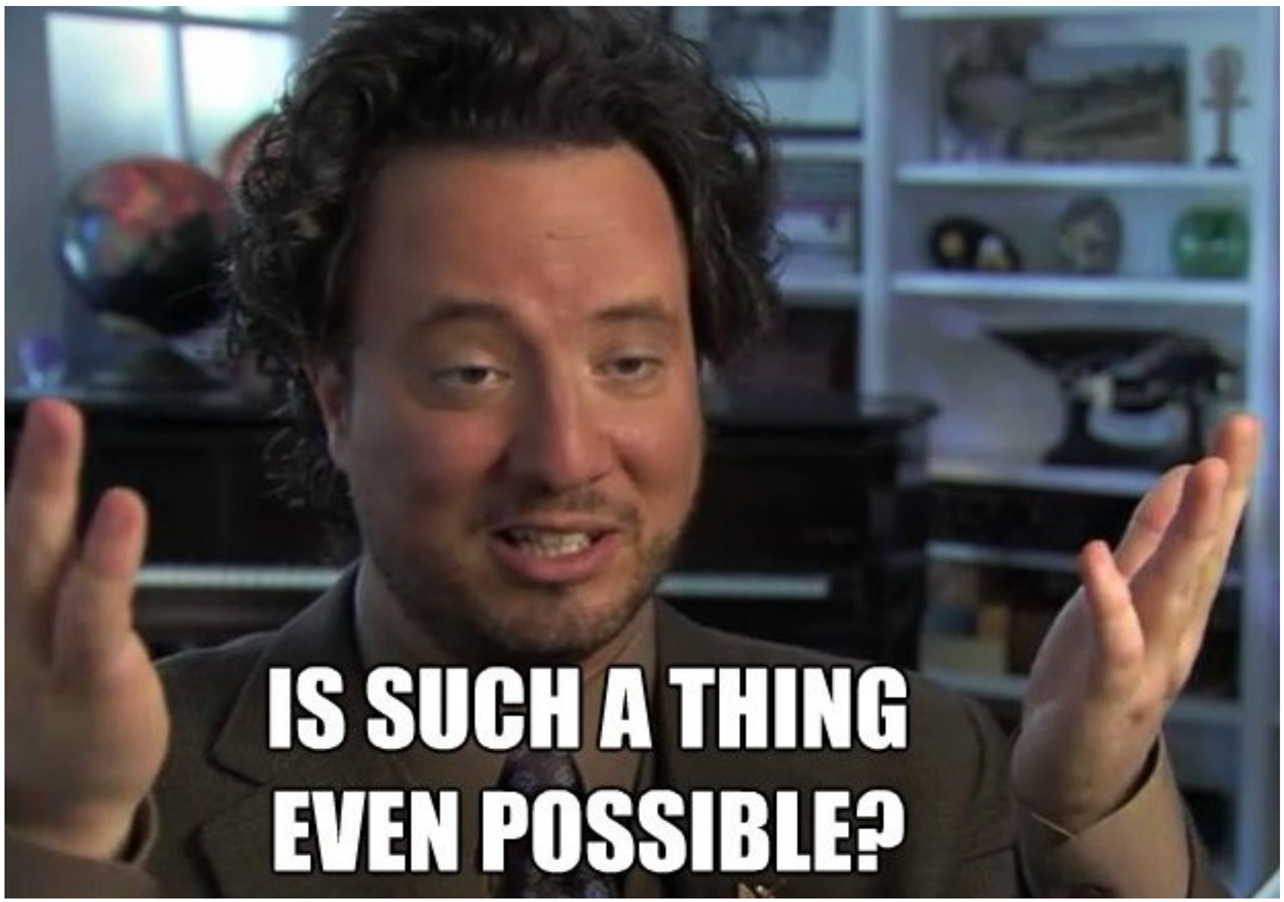
Given (many) observation sequences, $\{O_i\}$ and the set of possible hidden states, estimate the parameters of HMM, $\lambda = (A, B)$

$$A = [a_{ij}] = P(q_t = j | q_{t-1} = i)$$

$$B = [b_j(o_t)] = P(o_t | q_t = j)$$

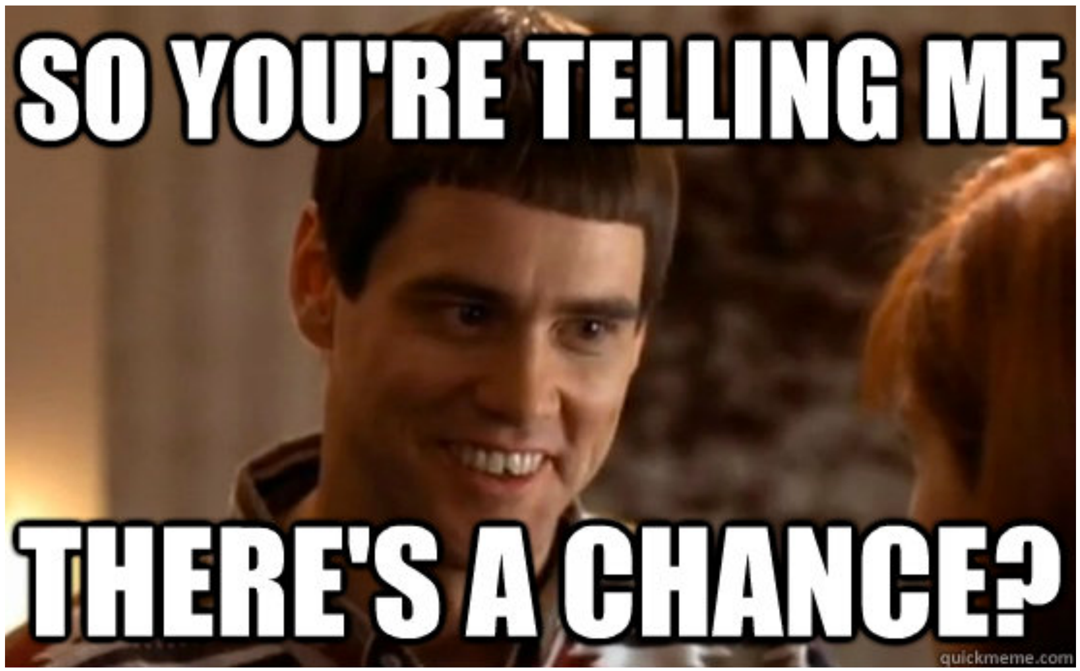
Learning an HMM

So just by looking at sentences, can we learn a POS tagging model?



Learning an HMM

Turns out, we can!



This is an example of **unsupervised learning**.

Learning an HMM

Let's consider a simpler case, where we are given $\{O_i, Q_i\}$ sequence pairs

Similar, to a language model, counting should've worked!

$$a_{ij} = \frac{C(i \to j)}{\sum_{q \in Q} C(i \to q)}$$

$$b_j(o) = \frac{C(j \to o)}{\sum_{o' \in Vocab(o')} C(j \to o')}$$

Learning an HMM

But, we only have unlabeled sequences, $\{O_i\}$.

Why do we even expect to learn (A,B)?

Consider, multiple pairs of (A,B) =
$\{(A_1, B_1), (A_2, B_2), (A_3, B_3),\ \dots\ ,\ (A_N, B_N)\}$

For each (A_x, B_x), we can compute

$$Likelihood_x = \prod_i P(O_i | (A_x, B_x))$$

Choose best (A_x, B_x)

Learning an HMM

Infinite number of possibilities for (A, B)

Can we do something smarter?

Expectation – Maximization is an algorithm to find parameters of a statistical model containing hidden variables

Learning an HMM

EM is an iterative algorithm. Let's draw a sketch of the algorithm:

1. Start with some estimate of parameters, (A, B)

2. Compute an expectation of the likelihood of the observed data using current parameter estimates

3. Update parameters (A, B) so as to maximize the expectation computed in Step 2

4. Repeat 2. & 3. until some stopping criterion

Learning an HMM

EM is an iterative algorithm. Let's draw a sketch of the algorithm:

1. Start with some estimate of parameters, (A, B)
2. Compute an expectation of the likelihood of the observed data using current parameter estimates
3. Update parameters (A, B) so as to maximize the expectation computed in Step 2
4. Repeat 2. & 3. until some stopping criterion

Optimum (A, B) are found!

What we learned..

Hidden Markov Models allow us to model sequences with hidden latent variables

1. Likelihood: Given a observation sequence, computing its likelihood

2. Decoding: Given an observation sequence, decoding the best hidden sequence of labels

3. Learning: Using unlabeled sequence examples, learning the parameters of a HMM.