

Machine Learning & Data Mining

CS/CNS/EE 155

Lecture 13:
Hidden Markov Models

Sequence Prediction (POS Tagging)

- $x = \text{"Fish Sleep"}$
- $y = (\text{N}, \text{V})$

- $x = \text{"The Dog Ate My Homework"}$
- $y = (\text{D}, \text{N}, \text{V}, \text{D}, \text{N})$

- $x = \text{"The Fox Jumped Over The Fence"}$
- $y = (\text{D}, \text{N}, \text{V}, \text{P}, \text{D}, \text{N})$

Challenges

- Multivariable Output
 - Make multiple predictions simultaneously
- Variable Length Input/Output
 - Sentence lengths not fixed

Multivariate Outputs

- $x = \text{"Fish Sleep"}$
- $y = (N, V)$
- Multiclass prediction:

POS Tags:
Det, Noun, Verb, Adj, Adv, Prep

Replicate Weights:

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_K \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_K \end{bmatrix}$$

Score All Classes:

$$f(x | w, b) = \begin{bmatrix} w_1^T x - b_1 \\ w_2^T x - b_2 \\ \vdots \\ w_K^T x - b_K \end{bmatrix}$$

Predict via Largest Score:

$$\operatorname{argmax}_k \begin{bmatrix} w_1^T x - b_1 \\ w_2^T x - b_2 \\ \vdots \\ w_K^T x - b_K \end{bmatrix}$$

- How many classes?

Multiclass Prediction

- $x = \text{"Fish Sleep"}$
- $y = (N, V)$
- Multiclass prediction:
 - All possible length-M sequences as different class
 - (D, D), (D, N), (D, V), (D, Adj), (D, Adv), (D, Pr)
(N, D), (N, N), (N, V), (N, Adj), (N, Adv), ...
- L^M classes!
 - Length 2: $6^2 = 36!$

POS Tags:
Det, Noun, Verb, Adj, Adv, Prep
 $L=6$

Multiclass Prediction

- $x = \text{"Fish Sleep"}$

- $y = (N, V)$

- M

POS Tags:
Det, Noun, Verb, Adj, Adv, Prep

L=6

Exponential Explosion in #Classes!

(Not Tractable for Sequence Prediction)

- L^m classes:

– Length 2: $6^2 = 36!$

Why is Naïve Multiclass Intractable?

$x = \text{"I fish often"}$

POS Tags:
Det, Noun, Verb, Adj, Adv, Prep

- (D, D, D), (D, D, N), (D, D, V), (D, D, Adj), (D, D, Adv), (D, D, Pr)
- (D, N, D), (D, N, N), (D, N, V), (D, N, Adj), (D, N, Adv), (D, N, Pr)
- (D, V, D), (D, V, N), (D, V, V), (D, V, Adj), (D, V, Adv), (D, V, Pr)
- ...
- (N, D, D), (N, D, N), (N, D, V), (N, D, Adj), (N, D, Adv), (N, D, Pr)
- (N, N, D), (N, N, N), (N, N, V), (N, N, Adj), (N, N, Adv), (N, N, Pr)
- ...

Assume pronouns are nouns for simplicity.

Why is Naïve Multiclass Intractable?

$x = \text{"I fish often"}$

POS Tags:
Det, Noun, Verb, Adj, Adv, Prep

- (D, D, D), (D, D, N), (D, D, V), (D, D, Adj), (D, D, Adv), (D, D, Pr)

Treats Every Combination As Different Class
(Learn model for each combination)

Exponentially Large Representation!
(Exponential Time to Consider Every Class)
(Exponential Storage)

Assume pronouns are nouns for simplicity.

Independent Classification

$x = \text{"I fish often"}$

POS Tags:
Det, Noun, Verb, Adj, Adv, Prep

- Treat each word independently (assumption)
 - Independent multiclass prediction per word
 - Predict for $x = \text{"I"}$ independently
 - Predict for $x = \text{"fish"}$ independently
 - Predict for $x = \text{"often"}$ independently
 - Concatenate predictions.

Assume pronouns are nouns for simplicity.

Independent Classification

$x = \text{"I fish often"}$

POS Tags:
Det, Noun, Verb, Adj, Adv, Prep

- Treat each word independently (assumption)
 - Independent multiclass prediction per word

#Classes = #POS Tags
(6 in our example)

Solvable using standard multiclass prediction.

Independent Classification

$x = \text{"I fish often"}$

POS Tags:
Det, Noun, Verb, Adj, Adv, Prep

- Treat each word independently
 - Independent multiclass prediction per word

| $P(y x)$ | $x = \text{"I"}$ | $x = \text{"fish"}$ | $x = \text{"often"}$ |
|----------|------------------|---------------------|----------------------|
| y="Det" | 0.0 | 0.0 | 0.0 |
| y="Noun" | 1.0 | 0.75 | 0.0 |
| y="Verb" | 0.0 | 0.25 | 0.0 |
| y="Adj" | 0.0 | 0.0 | 0.4 |
| y="Adv" | 0.0 | 0.0 | 0.6 |
| y="Prep" | 0.0 | 0.0 | 0.0 |

Prediction: (N, N, Adv)

Correct: (N, V, Adv)

Why the mistake?

Assume pronouns are nouns for simplicity.

Context Between Words

x = “I fish often”

POS Tags:
Det, Noun, Verb, Adj, Adv, Prep

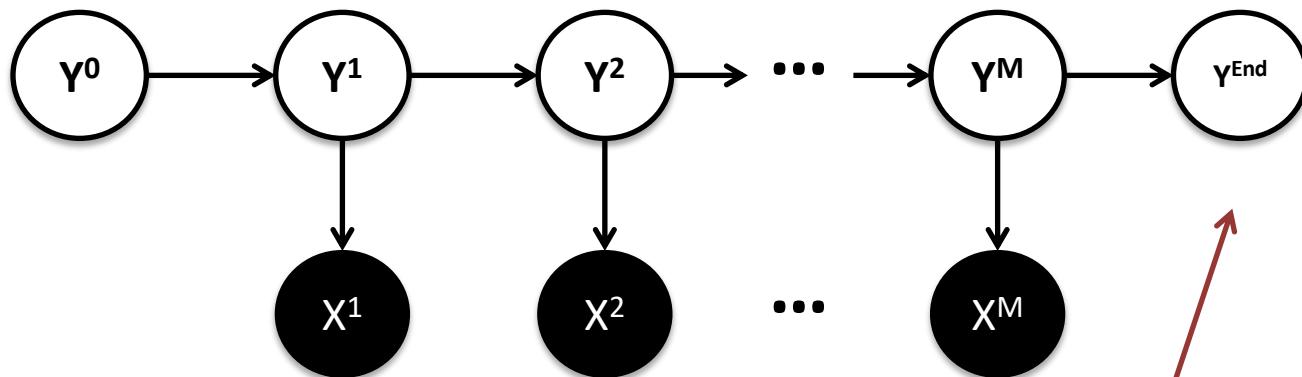
- Independent Predictions Ignore Word Pairs
 - In Isolation:
 - “Fish” is more likely to be a Noun
 - But Conditioned on Following a (pro)Noun...
 - “Fish” is more likely to be a Verb!
 - **“1st Order” Dependence (Model All Pairs)** 
 - 2nd Order Considers All Triplets
 - Arbitrary Order = Exponential Size (Naïve Multiclass)

Assume pronouns are nouns for simplicity.

1st Order Hidden Markov Model

- $x = (x^1, x^2, x^3, x^4, \dots, x^M)$ (sequence of words)
- $y = (y^1, y^2, y^3, y^4, \dots, y^M)$ (sequence of POS tags)
- $P(x^i | y^i)$ Probability of state y^i generating x^i
- $P(y^{i+1} | y^i)$ Probability of state y^i transitioning to y^{i+1}
- $P(y^1 | y^0)$ y^0 is defined to be the Start state
- $P(\text{End} | y^M)$ Prior probability of y^M being the final state
 - Not always used

Graphical Model Representation



$$P(x, y) = P(End \mid y^M) \prod_{i=1}^M P(y^i \mid y^{i-1}) \prod_{i=1}^M P(x^i \mid y^i)$$

1st Order Hidden Markov Model

$$P(x, y) = P(\text{End} \mid y^M) \prod_{i=1}^M P(y^i \mid y^{i-1}) \prod_{i=1}^M P(x^i \mid y^i)$$

“Joint Distribution”

- $P(x^i \mid y^i)$ Probability of state y^i generating x^i
- $P(y^{i+1} \mid y^i)$ Probability of state y^i transitioning to y^{i+1}
- $P(y^1 \mid y^0)$ y^0 is defined to be the Start state
- $P(\text{End} \mid y^M)$ Prior probability of y^M being the final state

1st Order Hidden Markov Model

$$P(x|y) = \prod_{i=1}^M P(x^i | y^i)$$

Given a POS Tag Sequence y :
Can compute each $P(x^i|y)$ independently!
(x^i conditionally independent given y^i)

“Conditional Distribution on x given y ”

- $P(x^i | y^i)$ Probability of state y^i generating x^i
- $P(y^{i+1} | y^i)$ Probability of state y^i transitioning to y^{i+1}
- $P(y^1 | y^0)$ y^0 is defined to be the Start state
- $P(\text{End} | y^M)$ Prior probability of y^M being the final state

1st Order Hidden Markov Model

Models All State-State Pairs

(all POS Tag-Tag pairs)

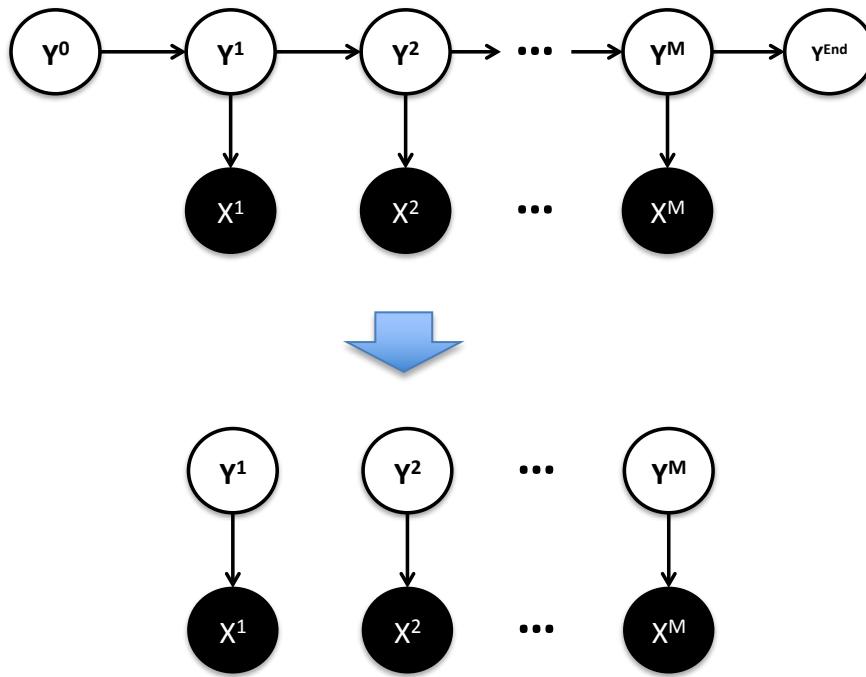
Models All State-Observation Pairs

(all Tag-Word pairs)

Same Complexity as Independent Multiclass

- $P(x^i | y^i)$ Probability of state y^i generating x^i
 - $P(y^{i+1} | y^i)$ Probability of state y^i transitioning to y^{i+1}
 - $P(y^1 | y^0)$ y^0 is defined to be the Start state
 - $P(\text{End} | y^M)$ Prior probability of y^M being the final state

Relationship to Naïve Bayes



Reduces to a sequence of disjoint Naïve Bayes models
(if we ignore transition probabilities)

$P(\text{word} \mid \text{state/tag})$

- Two-word language: “fish” and “sleep”
- Two-tag language: “Noun” and “Verb”

| $P(x y)$ | $y = \text{“Noun”}$ | $y = \text{“Verb”}$ |
|-----------|---------------------|---------------------|
| x=“fish” | 0.8 | 0.5 |
| x=“sleep” | 0.2 | 0.5 |

Given Tag Sequence y:

$$P(\text{“fish sleep”} \mid (\text{N,V})) = 0.8 * 0.5$$

$$P(\text{“fish fish”} \mid (\text{N,V})) = 0.8 * 0.5$$

$$P(\text{“sleep fish”} \mid (\text{V,V})) = 0.5 * 0.5$$

$$P(\text{“sleep sleep”} \mid (\text{N,N})) = 0.2 * 0.2$$

Sampling

- HMMs are “generative” models
 - Models joint distribution $P(x,y)$
 - Can generate samples from this distribution
 - First consider conditional distribution $P(x|y)$

| $P(x y)$ | $y = \text{“Noun”}$ | $y = \text{“Verb”}$ |
|-------------|---------------------|---------------------|
| x = “fish” | 0.8 | 0.5 |
| x = “sleep” | 0.2 | 0.5 |

Given Tag Sequence $y = (N,V)$:

Sample each word independently:

Sample $P(x^1 | N)$ (0.8 Fish, 0.2 Sleep)

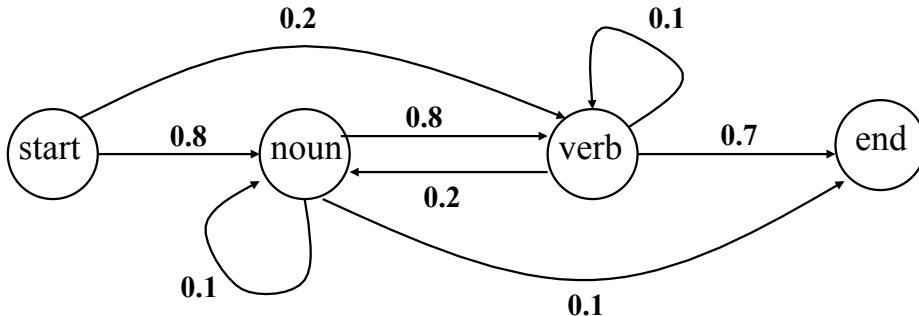
Sample $P(x^2 | V)$ (0.5 Fish, 0.5 Sleep)

– What about sampling from $P(x,y)$?

Forward Sampling of $P(y, x)$

$$P(x, y) = P(End \mid y^M) \prod_{i=1}^M P(y^i \mid y^{i-1}) \prod_{i=1}^M P(x^i \mid y^i)$$

| $P(x \mid y)$ | $y = \text{"Noun"}$ | $y = \text{"Verb"}$ |
|----------------------|---------------------|---------------------|
| $x = \text{"fish"}$ | 0.8 | 0.5 |
| $x = \text{"sleep"}$ | 0.2 | 0.5 |



Initialize $y^0 = \text{Start}$

Initialize $i = 0$

1. $i = i + 1$
2. Sample y^i from $P(y^i \mid y^{i-1})$
3. If $y^i == \text{End}$: Quit
4. Sample x^i from $P(x^i \mid y^i)$
5. Goto Step 1

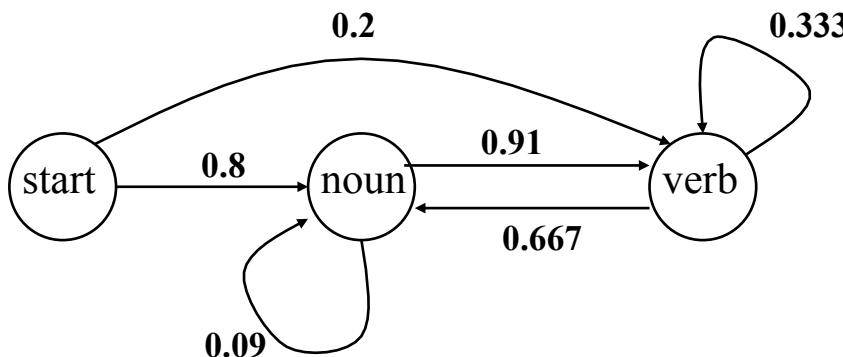
Exploits Conditional Ind.
Requires $P(\text{End} \mid y^i)$

Forward Sampling of $P(y, x | M)$

$$P(x, y | M) = P(\text{End} | y^M) \prod_{i=1}^M P(y^i | y^{i-1}) \prod_{i=1}^M P(x^i | y^i)$$

X

| $P(x y)$ | $y = \text{"Noun"}$ | $y = \text{"Verb"}$ |
|----------------------|---------------------|---------------------|
| $x = \text{"fish"}$ | 0.8 | 0.5 |
| $x = \text{"sleep"}$ | 0.2 | 0.5 |



```

Initialize  $y^0 = \text{Start}$ 
Initialize  $i = 0$ 

1.  $i = i + 1$ 
2. If( $i == M$ ): Quit
3. Sample  $y^i$  from  $P(y^i | y^{i-1})$ 
4. Sample  $x^i$  from  $P(x^i | y^i)$ 
5. Goto Step 1
  
```

Exploits Conditional Ind.
Assumes no $P(\text{End} | y^i)$

1st Order Hidden Markov Model

$$P(x^{k+1:M}, y^{k+1:M} | x^{1:k}, y^{1:k}) = P(x^{k+1:M}, y^{k+1:M} | y^k)$$

“Memory-less Model” – only needs y^k to model rest of sequence

- $P(x^i | y^i)$ Probability of state y^i generating x^i
- $P(y^{i+1} | y^i)$ Probability of state y^i transitioning to y^{i+1}
- $P(y^1 | y^0)$ y^0 is defined to be the Start state
- $P(\text{End} | y^M)$ Prior probability of y^M being the final state

Viterbi Algorithm

Most Common Prediction Problem

- Given input sentence, predict POS Tag seq.

$$\operatorname{argmax}_y P(y | x)$$

- Naïve approach:**
 - Try all possible y 's
 - Choose one with highest probability
 - Exponential time: L^M possible y 's**

Recall: Bayes's Rule

$$\begin{aligned}\operatorname{argmax}_y P(y \mid x) &= \operatorname{argmax}_y \frac{P(y, x)}{P(x)} \quad \blacksquare \\ &= \operatorname{argmax}_y P(y, x) \\ &= \operatorname{argmax}_y P(x \mid y)P(y)\end{aligned}$$

$$P(x|y) = \prod_{i=1}^M P(x^i|y^i)$$

$$P(y) = P(END|y^M) \prod_{i=1}^M P(y^i|y^{i-1})$$

$$\begin{aligned}
\underset{y}{\operatorname{argmax}} P(y, x) &= \underset{y}{\operatorname{argmax}} \prod_{i=1}^M P(y^i | y^{i-1}) \prod_{i=1}^M P(x^i | y^i) \\
&= \underset{y^M}{\operatorname{argmax}} \underset{y^{1:M-1}}{\operatorname{argmax}} \prod_{i=1}^M P(y^i | y^{i-1}) \prod_{i=1}^M P(x^i | y^i) \\
&= \underset{y^M}{\operatorname{argmax}} \underset{y^{1:M-1}}{\operatorname{argmax}} P(y^M | y^{M-1}) P(x^M | y^M) P(y^{1:M-1} | x^{1:M-1})
\end{aligned}$$

$$\begin{aligned}
P(y^{1:k} | x^{1:k}) &\stackrel{\text{?}}{=} P(x^{1:k} | y^{1:k}) P(y^{1:k}) & P(x^{1:k} | y^{1:k}) &= \prod_{i=1}^k P(x^i | y^i) \\
P(y^{1:k}) &= \prod_{i=1}^k P(y^{i+1} | y^i)
\end{aligned}$$

Exploit Memory-less Property:
The choice of y^M only depends on $y^{1:M-1}$ via $P(y^M | y^{M-1})$!

Dynamic Programming

- **Input:** $x = (x^1, x^2, x^3, \dots, x^M)$
- **Computed:** best length- k prefix ending in each Tag:
 - Examples:

$$\boxed{\text{Q}} \hat{Y}^k(V) = \left(\underset{y^{1:k-1}}{\operatorname{argmax}} P(y^{1:k-1} \oplus V, x^{1:k}) \right) \oplus V \quad \hat{Y}^k(N) = \left(\underset{y^{1:k-1}}{\operatorname{argmax}} P(y^{1:k-1} \oplus N, x^{1:k}) \right) \oplus N$$

Sequence Concatenation

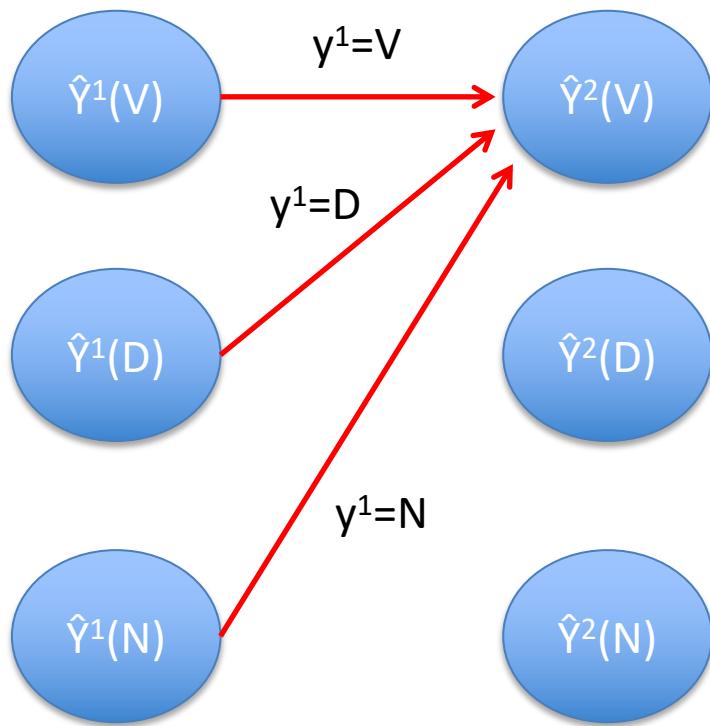
- **Claim:** $\hat{Y}^{k+1}(V) = \left(\underset{y^{1:k} \in \{\hat{Y}^k(T)\}_T}{\operatorname{argmax}} P(y^{1:k} \oplus V, x^{1:k+1}) \right) \oplus V$

Pre-computed

Recursive Definition!

Solve: $\hat{Y}^2(V) = \left(\underset{y^1 \in \{\hat{Y}^1(T)\}_T}{\operatorname{argmax}} P(y^1, x^1) P(y^2 = V | y^1) P(x^2 | y^2 = V) \right) \oplus V$

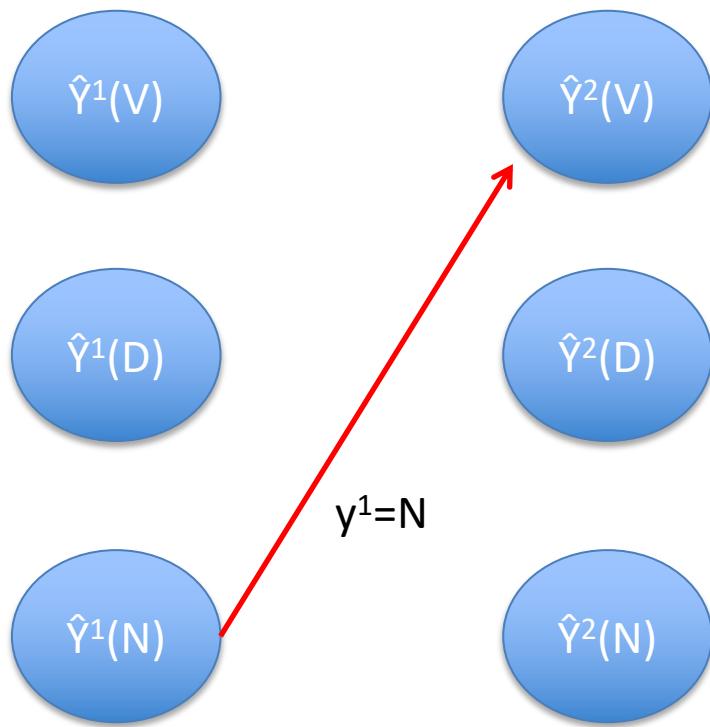
Store each
 $\hat{Y}^1(Z)$ & $P(\hat{Y}^1(Z), x^1)$



$\hat{Y}^1(Z)$ is just Z

Solve: $\hat{Y}^2(V) = \left(\underset{y^1 \in \{\hat{Y}^1(T)\}_T}{\operatorname{argmax}} P(y^1, x^1) P(y^2 = V | y^1) P(x^2 | y^2 = V) \right) \oplus V$

Store each
 $\hat{Y}^1(Z)$ & $P(\hat{Y}^1(Z), x^1)$



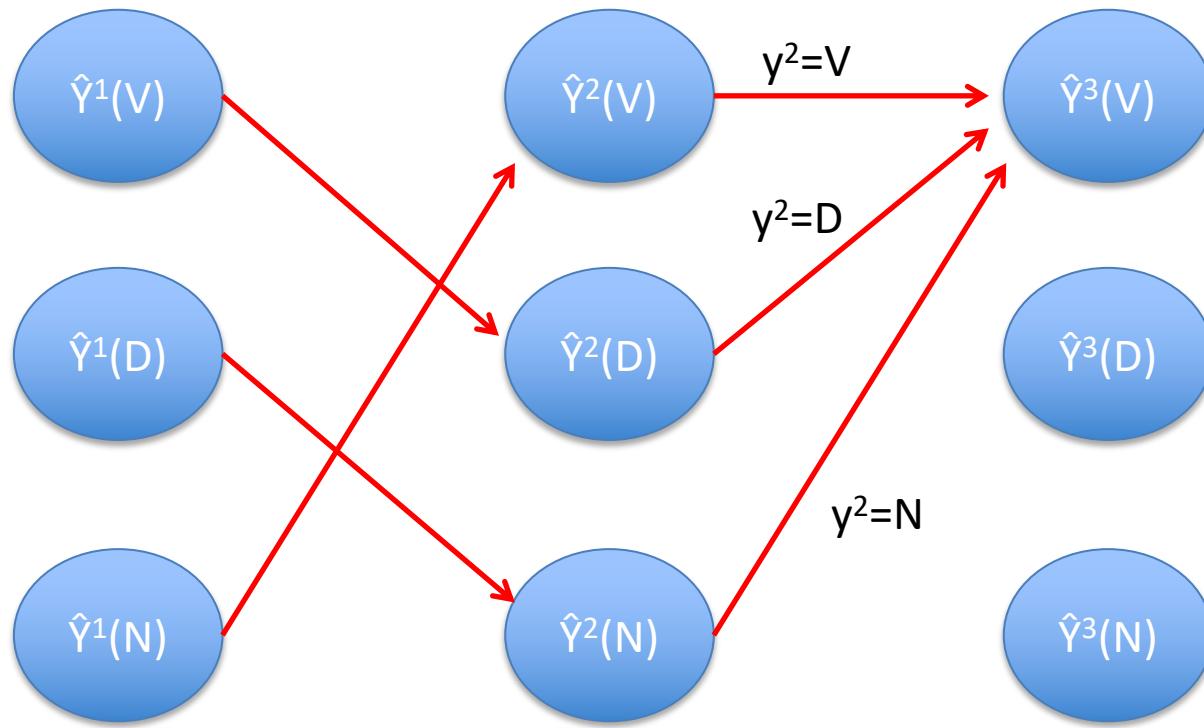
$\hat{Y}^1(Z)$ is just Z

Ex: $\hat{Y}^2(V) = (N, V)$

Solve: $\hat{Y}^3(V) = \left(\underset{y^{1:2} \in \{\hat{Y}^2(T)\}_T}{\operatorname{argmax}} P(y^{1:2}, x^{1:2}) P(y^3 = V | y^2) P(x^3 | y^3 = V) \right) \oplus V$

Store each
 $\hat{Y}^1(Z)$ & $P(\hat{Y}^1(Z), x^1)$

Store each
 $\hat{Y}^2(Z)$ & $P(\hat{Y}^2(Z), x^{1:2})$



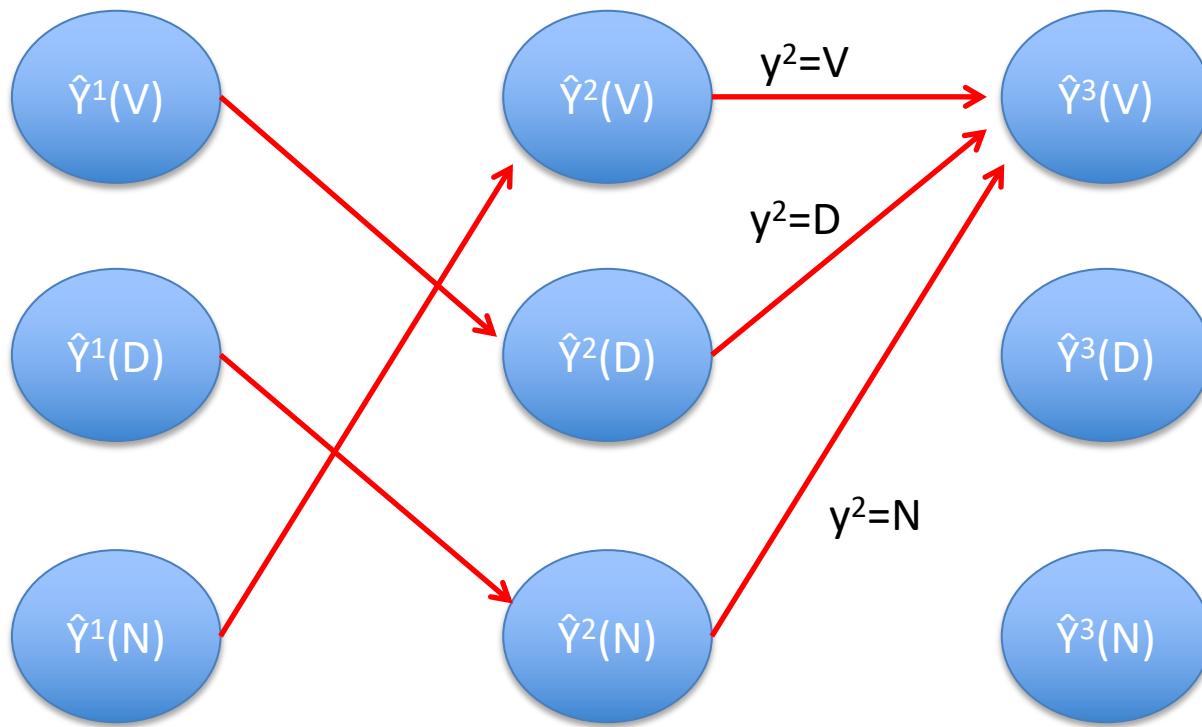
Ex: $\hat{Y}^2(V) = (N, V)$

Solve: $\hat{Y}^3(V) = \left(\underset{y^{1:2} \in \{\hat{Y}^2(T)\}_T}{\operatorname{argmax}} P(y^{1:2}, x^{1:2}) P(y^3 = V | y^2) P(x^3 | y^3 = V) \right) \oplus V$

Store each
 $\hat{Y}^1(Z)$ & $P(\hat{Y}^1(Z), x^1)$

Store each
 $\hat{Y}^2(Z)$ & $P(\hat{Y}^2(Z), x^{1:2})$

Claim: Only need to check
 solutions of $\hat{Y}^2(Z)$, $Z=V,D,N$

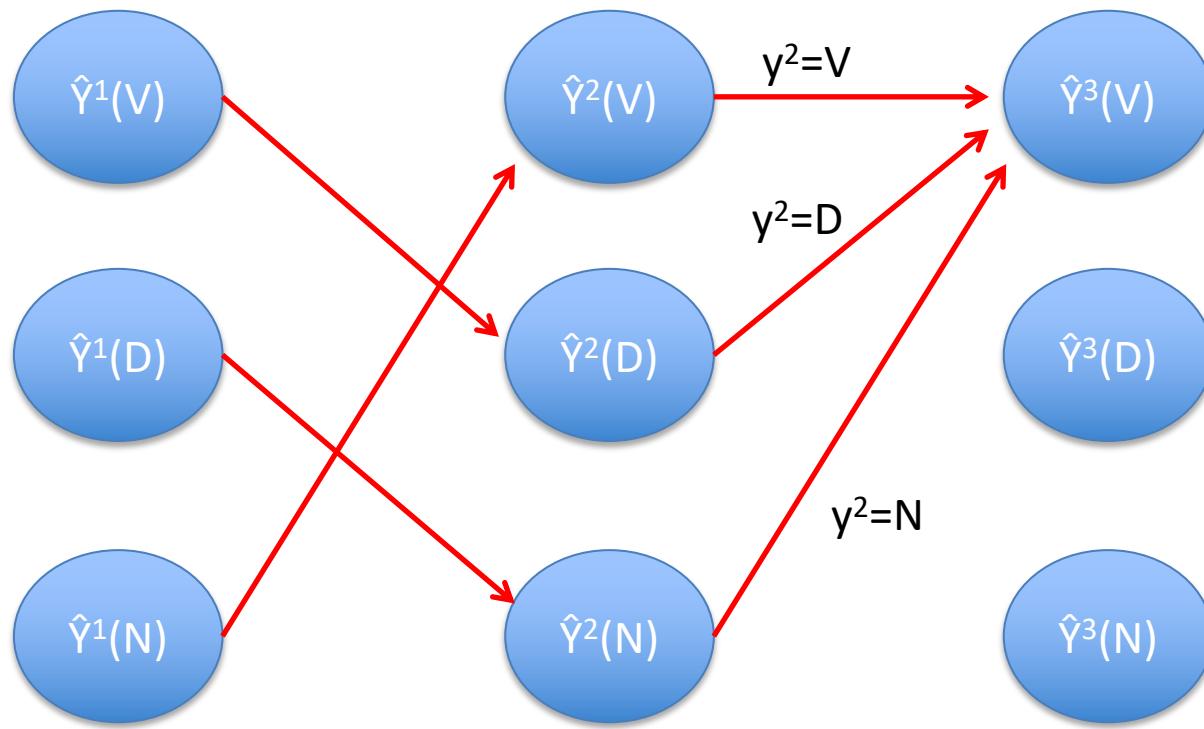


Ex: $\hat{Y}^2(V) = (N, V)$

Solve: $\hat{Y}^3(V) = \left(\underset{y^{1:2} \in \{\hat{Y}^2(T)\}_T}{\operatorname{argmax}} P(y^{1:2}, x^{1:2}) P(y^3 = V | y^2) P(x^3 | y^3 = V) \right) \oplus V$

Store each
 $\hat{Y}^1(Z)$ & $P(\hat{Y}^1(Z), x^1)$

Store each
 $\hat{Y}^2(Z)$ & $P(\hat{Y}^2(Z), x^{1:2})$



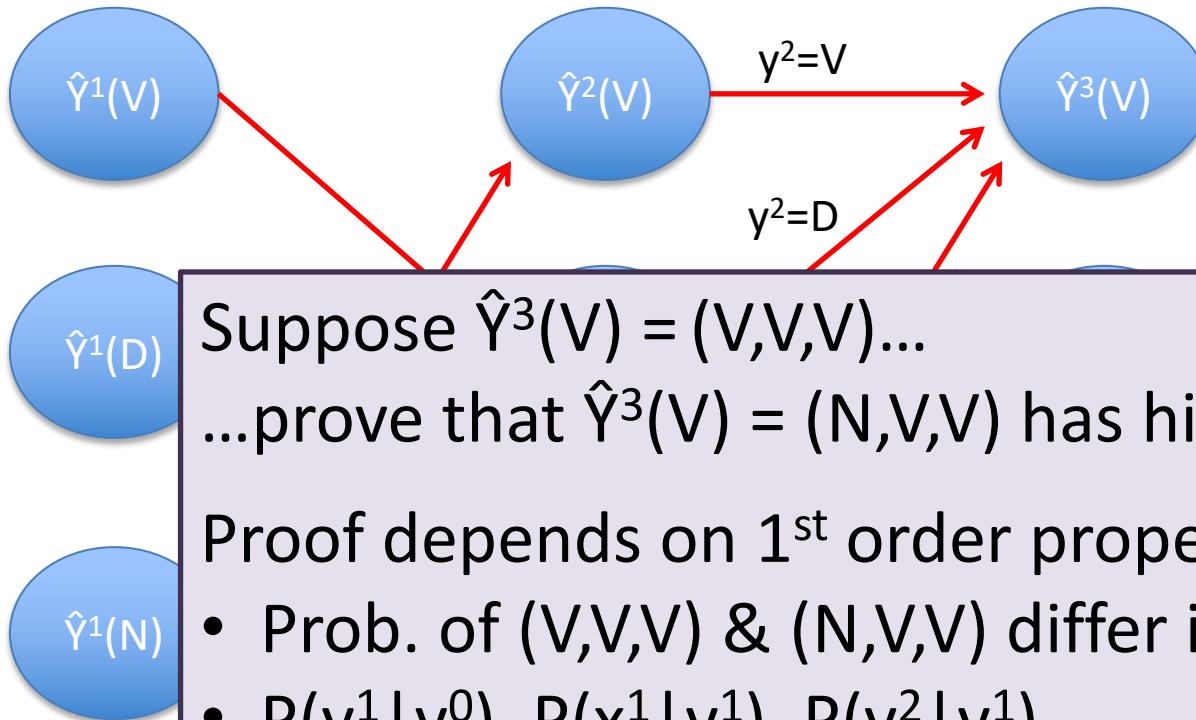
Ex: $\hat{Y}^2(V) = (N, V)$

Solve: $\hat{Y}^3(V) = \left(\underset{y^{1:2} \in \{\hat{Y}^2(T)\}_T}{\operatorname{argmax}} P(y^{1:2}, x^{1:2}) P(y^3 = V | y^2) P(x^3 | y^3 = V) \right) \oplus V$

Store each
 $\hat{Y}^1(Z)$ & $P(\hat{Y}^1(Z), x^1)$

Store each
 $\hat{Y}^2(Z)$ & $P(\hat{Y}^2(Z), x^{1:2})$

Claim: Only need to check
 solutions of $\hat{Y}^2(Z)$, $Z=V,D,N$



Suppose $\hat{Y}^3(V) = (V,V,V) \dots$
 ...prove that $\hat{Y}^3(V) = (N,V,V)$ has higher prob.

Proof depends on 1st order property

- Prob. of (V,V,V) & (N,V,V) differ in 3 terms
- $P(y^1 | y^0)$, $P(x^1 | y^1)$, $P(y^2 | y^1)$
- **None of these depend on y^3 !**

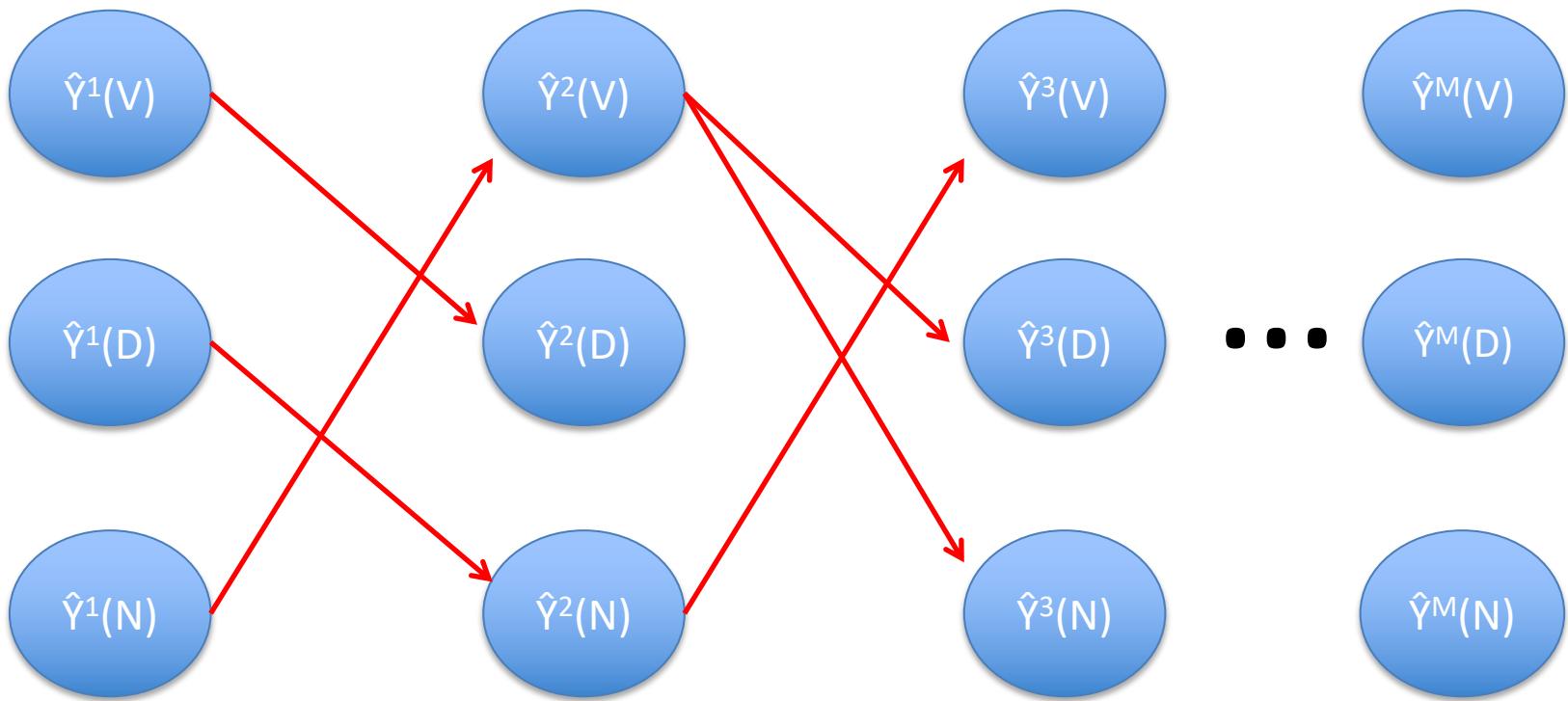
$$\hat{Y}^M(V) = \left(\underset{y^{1:M-1} \in \{\hat{Y}^{M-1}(T)\}_T}{\operatorname{argmax}} P(y^{1:M-1}, x^{1:M-1}) P(y^M = V \mid y^{M-1}) P(x^M \mid y^M = V) P(\text{End} \mid y^M = V) \right) \oplus V$$

Optional

Store each
 $\hat{Y}^1(Z) \& P(\hat{Y}^1(Z), x^1)$

Store each
 $\hat{Y}^2(Z) \& P(\hat{Y}^2(Z), x^{1:2})$

Store each
 $\hat{Y}^3(Z) \& P(\hat{Y}^3(Z), x^{1:3})$



Ex: $\hat{Y}^2(V) = (N, V)$

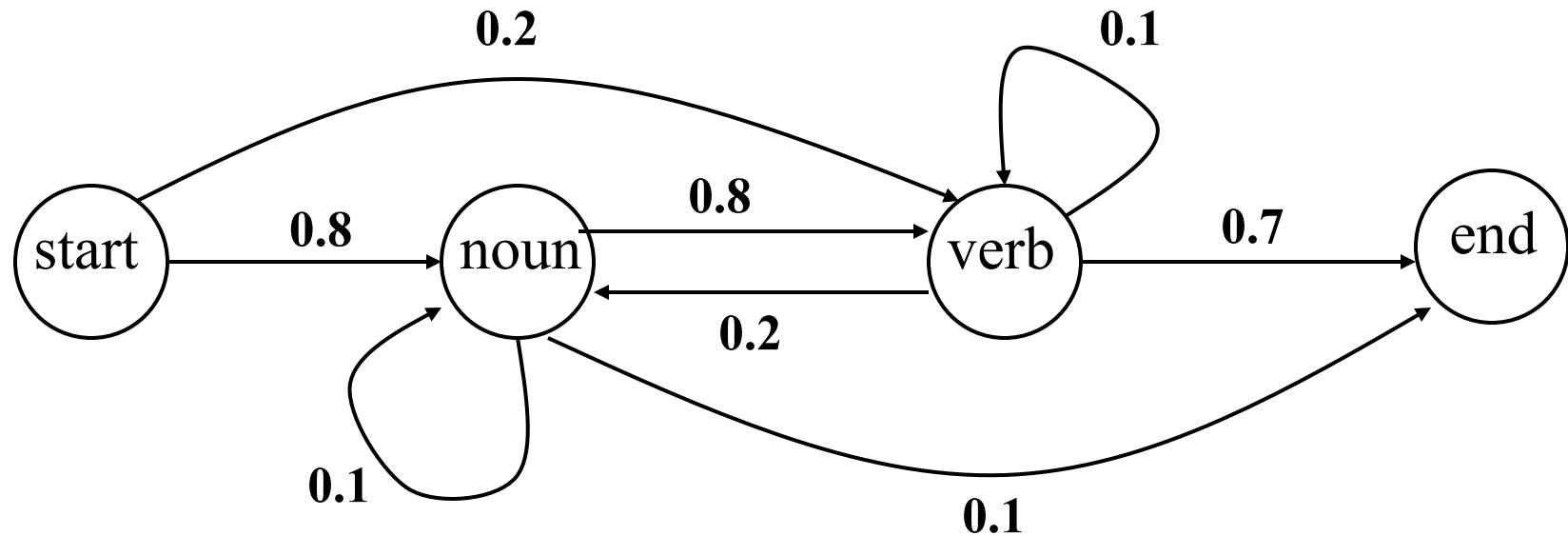
Ex: $\hat{Y}^3(V) = (D, N, V)$

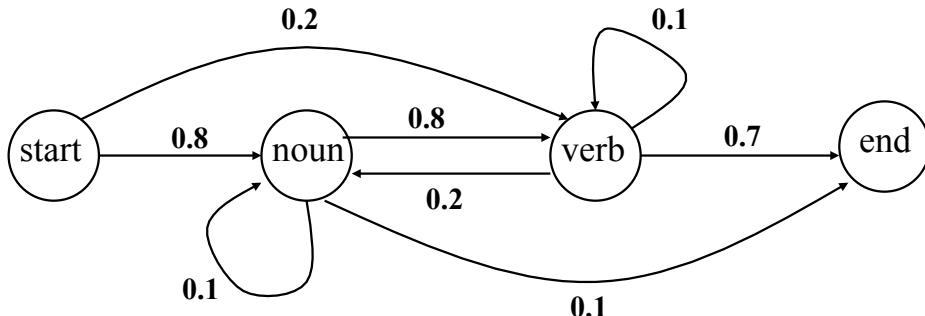
Viterbi Algorithm

- Solve:
$$\begin{aligned}\operatorname{argmax}_y P(y \mid x) &= \operatorname{argmax}_y \frac{P(y, x)}{P(x)} \\ &= \operatorname{argmax}_y P(y, x) \\ &= \operatorname{argmax}_y P(x \mid y)P(y)\end{aligned}$$
- For $k=1..M$
 - Iteratively solve for each $\hat{Y}^k(Z)$
 - Z looping over every POS tag.
- Predict best $\hat{Y}^M(Z)$
- Also known as Mean A Posteriori (MAP) inference

Numerical Example

$x = (\text{Fish Sleep})$





| $P(x y)$ | $y = \text{Noun}$ | $y = \text{Verb}$ |
|--------------------|-------------------|-------------------|
| $x = \text{fish}$ | 0.8 | 0.5 |
| $x = \text{sleep}$ | 0.2 | 0.5 |

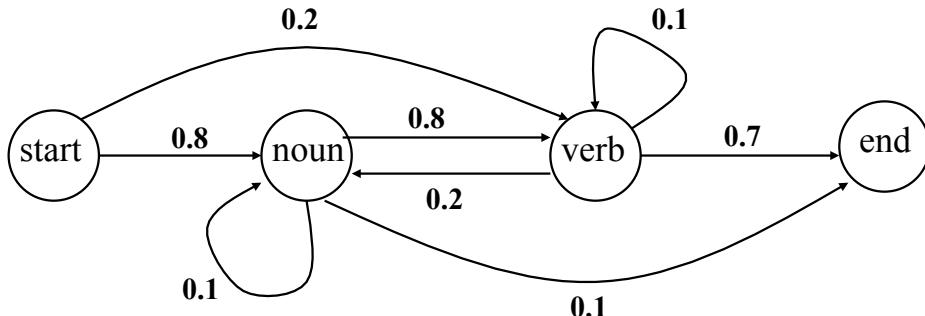
0 1 2 3

start 1

verb 0

noun 0

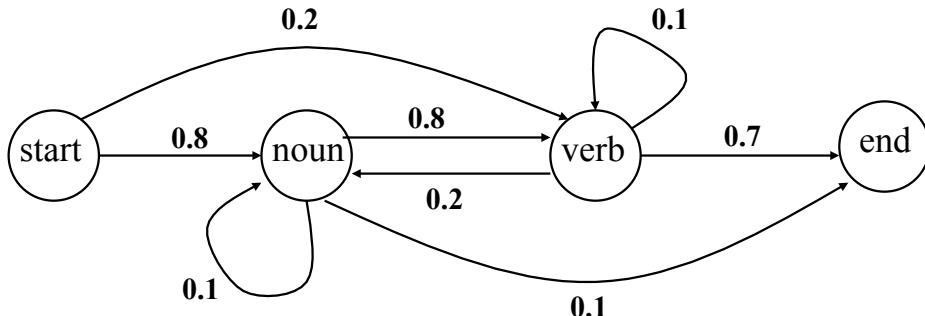
end 0



| $P(x y)$ | $y = \text{"Noun"}$ | $y = \text{"Verb"}$ |
|----------------------|---------------------|---------------------|
| $x = \text{"fish"}$ | 0.8 | 0.5 |
| $x = \text{"sleep"}$ | 0.2 | 0.5 |

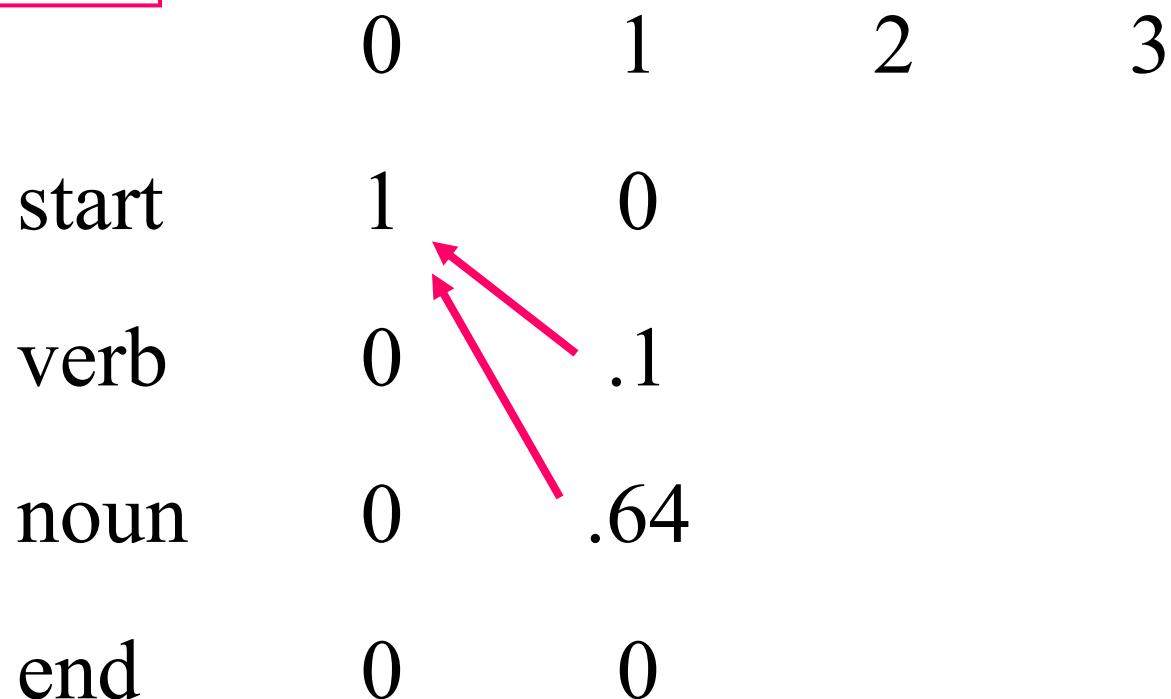
Token 1: fish

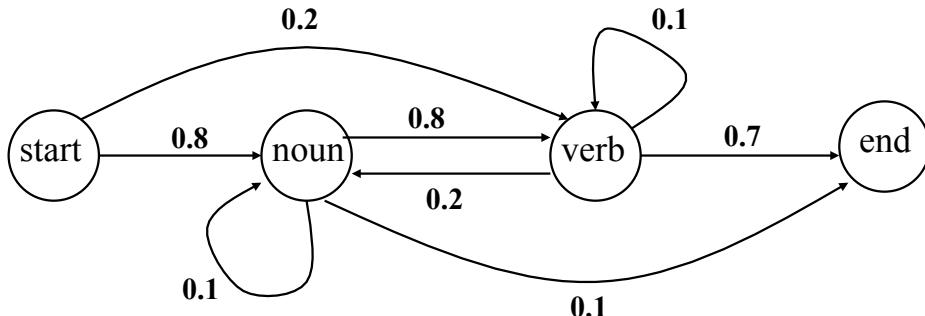
| | 0 | 1 | 2 | 3 |
|-------|---|---------|---|---|
| start | 1 | 0 | | |
| verb | 0 | .2 * .5 | | |
| noun | 0 | .8 * .8 | | |
| end | 0 | 0 | | |



| $P(x y)$ | $y = \text{"Noun"}$ | $y = \text{"Verb"}$ |
|----------------------|---------------------|---------------------|
| $x = \text{"fish"}$ | 0.8 | 0.5 |
| $x = \text{"sleep"}$ | 0.2 | 0.5 |

Token 1: fish





| $P(x y)$ | $y = \text{"Noun"}$ | $y = \text{"Verb"}$ |
|----------------------|---------------------|---------------------|
| $x = \text{"fish"}$ | 0.8 | 0.5 |
| $x = \text{"sleep"}$ | 0.2 | 0.5 |

Token 2: sleep

(if 'fish' is verb)

0 1 2 3

start

1 0 0

verb

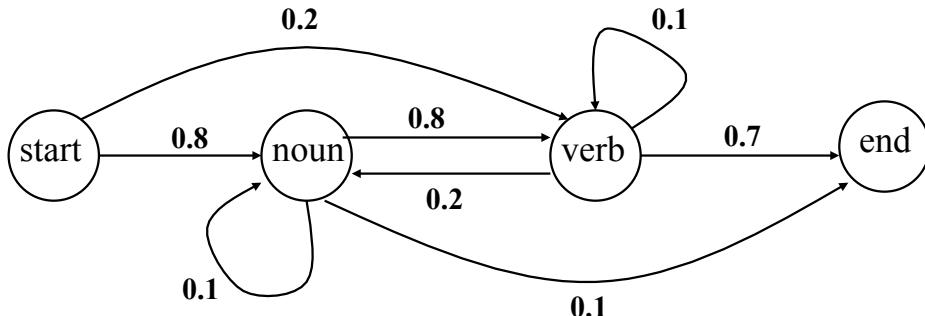
0 .1 $.1 * .1 * .5$

noun

0 .64 $.1 * .2 * .2$

end

0 0 -

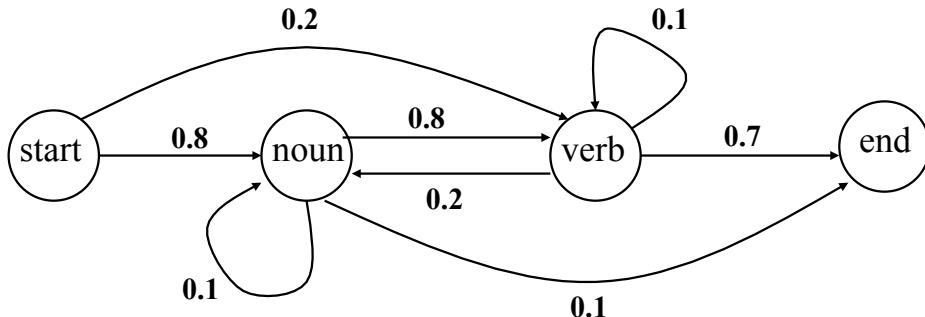


| $P(x y)$ | $y = \text{"Noun"}$ | $y = \text{"Verb"}$ |
|----------------------|---------------------|---------------------|
| $x = \text{"fish"}$ | 0.8 | 0.5 |
| $x = \text{"sleep"}$ | 0.2 | 0.5 |

Token 2: sleep

(if 'fish' is verb)

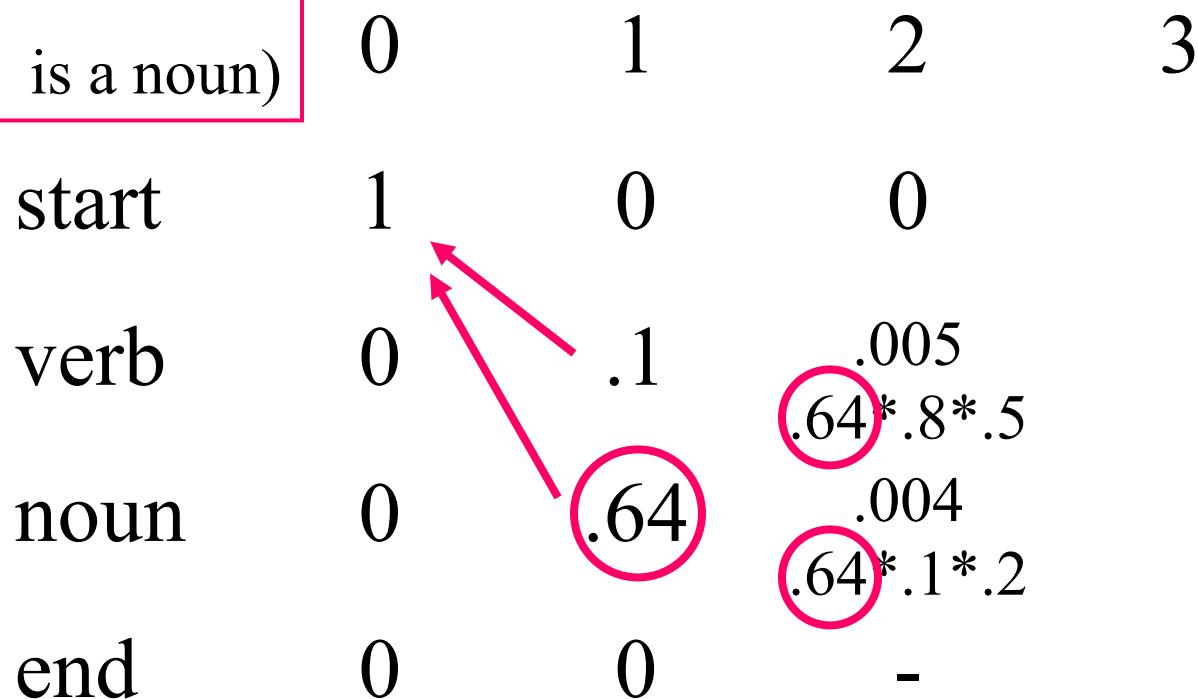
| | 0 | 1 | 2 | 3 |
|-------|---|-----|------|---|
| start | 1 | 0 | 0 | |
| verb | 0 | .1 | .005 | |
| noun | 0 | .64 | .004 | |
| end | 0 | 0 | - | |

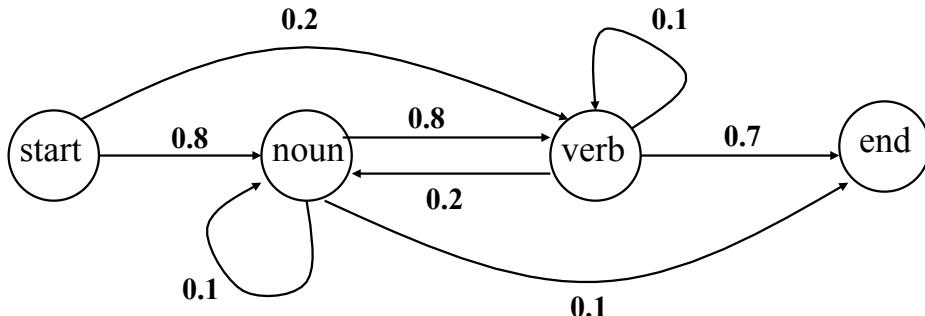


| $P(x y)$ | $y = \text{"Noun"}$ | $y = \text{"Verb"}$ |
|----------------------|---------------------|---------------------|
| $x = \text{"fish"}$ | 0.8 | 0.5 |
| $x = \text{"sleep"}$ | 0.2 | 0.5 |

Token 2: sleep

(if 'fish' is a noun)



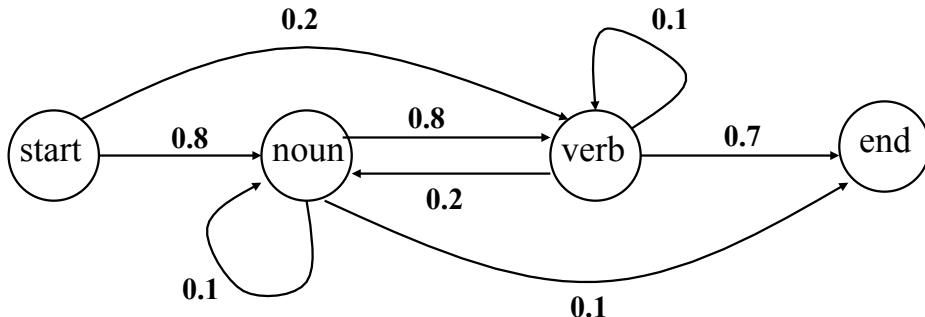


| $P(x y)$ | $y = \text{Noun}$ | $y = \text{Verb}$ |
|--------------------|-------------------|-------------------|
| $x = \text{fish}$ | 0.8 | 0.5 |
| $x = \text{sleep}$ | 0.2 | 0.5 |

Token 2: sleep

(if 'fish' is a noun)

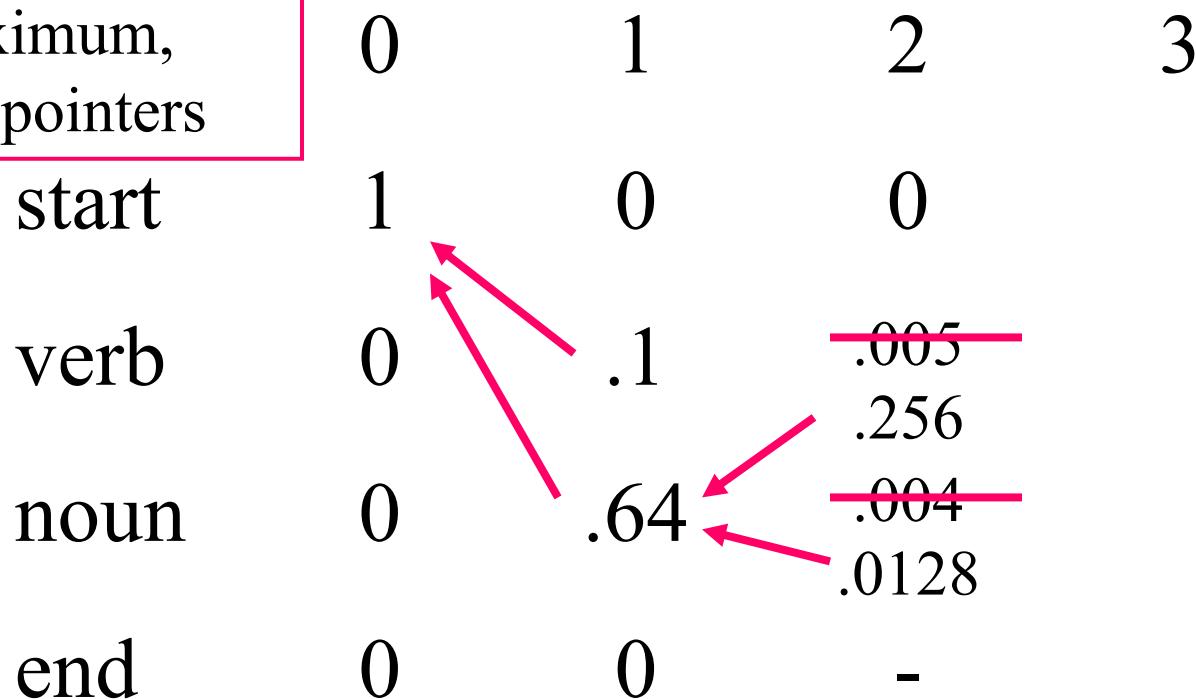
| | 0 | 1 | 2 | 3 |
|-------|---|-----|------|-------|
| start | 1 | 0 | 0 | |
| verb | 0 | .1 | .005 | .256 |
| noun | 0 | .64 | .004 | .0128 |
| end | 0 | 0 | - | |

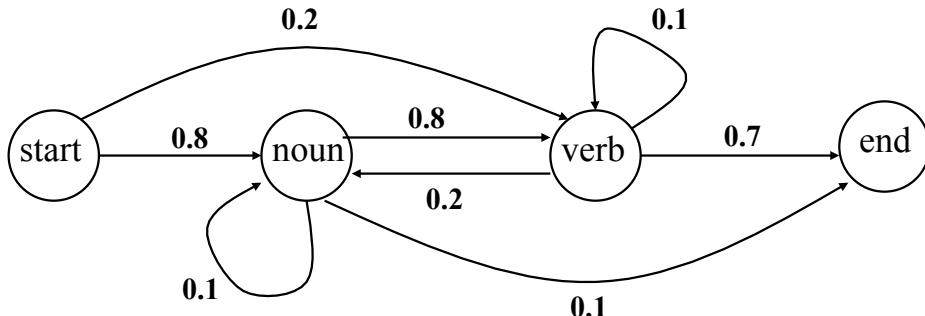


| $P(x y)$ | $y = \text{"Noun"}$ | $y = \text{"Verb"}$ |
|----------------------|---------------------|---------------------|
| $x = \text{"fish"}$ | 0.8 | 0.5 |
| $x = \text{"sleep"}$ | 0.2 | 0.5 |

Token 2: sleep

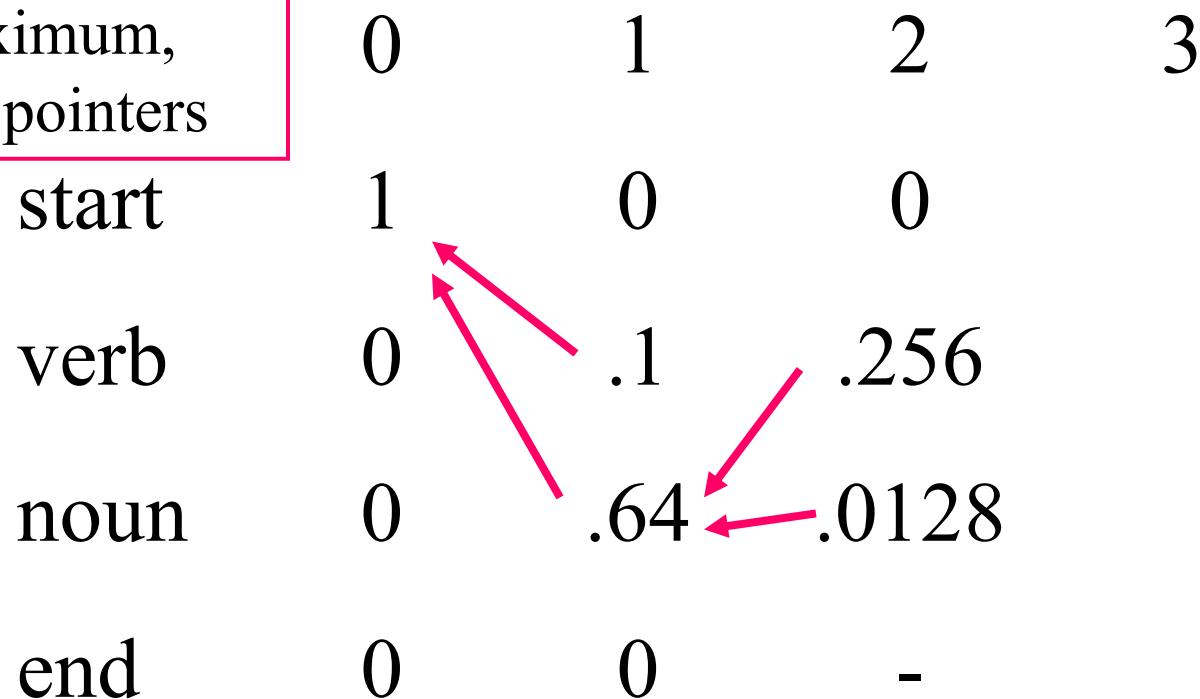
take maximum,
set back pointers

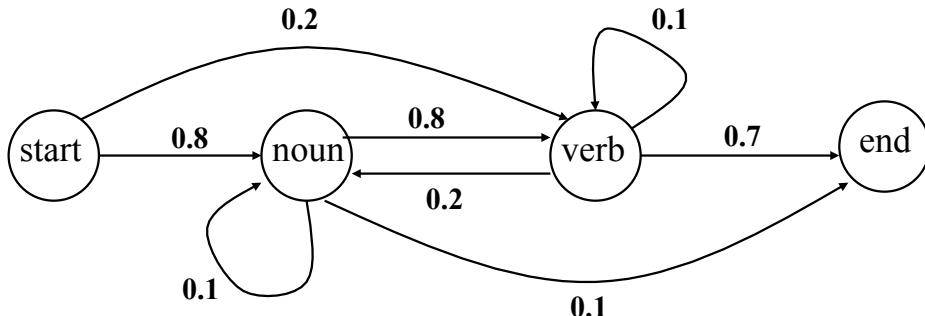




| $P(x y)$ | $y = \text{"Noun"}$ | $y = \text{"Verb"}$ |
|----------------------|---------------------|---------------------|
| $x = \text{"fish"}$ | 0.8 | 0.5 |
| $x = \text{"sleep"}$ | 0.2 | 0.5 |

Token 2: sleep
 take maximum,
 set back pointers



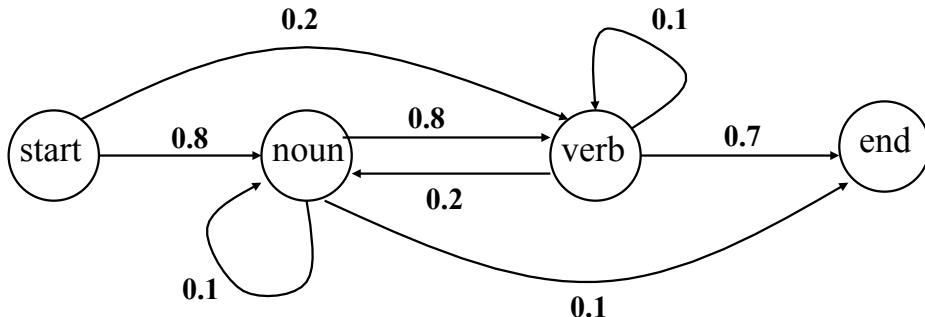


| $P(x y)$ | $y = \text{"Noun"}$ | $y = \text{"Verb"}$ |
|----------------------|---------------------|---------------------|
| $x = \text{"fish"}$ | 0.8 | 0.5 |
| $x = \text{"sleep"}$ | 0.2 | 0.5 |

Token 3: end

| | 0 | 1 | 2 | 3 |
|-------|---|-----|-------|-----------------------------|
| start | 1 | 0 | 0 | 0 |
| verb | 0 | .1 | .256 | - |
| noun | 0 | .64 | .0128 | - |
| end | 0 | 0 | - | $.256 * .7$ $.0128 * .1$ |

Slides borrowed from Ralph Grishman



| $P(x y)$ | $y = \text{"Noun"}$ | $y = \text{"Verb"}$ |
|----------------------|---------------------|---------------------|
| $x = \text{"fish"}$ | 0.8 | 0.5 |
| $x = \text{"sleep"}$ | 0.2 | 0.5 |

Token 3: end

take maximum,
set back pointers

0 1 2 3

start

1 0 0 0

verb

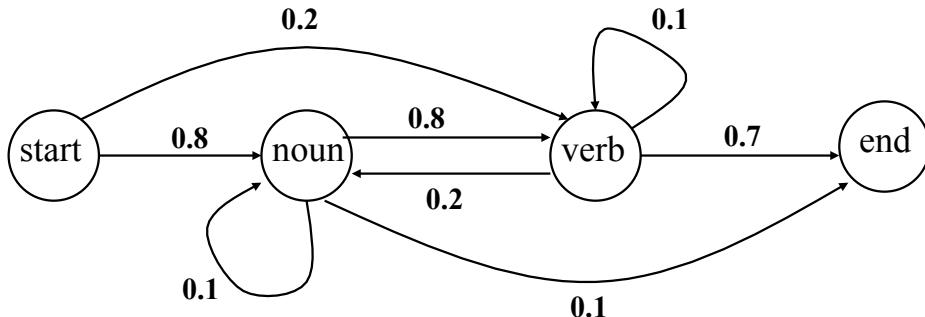
0 .1 .256 -

noun

0 .64 .0128 -

end

0 0 - $.256 * .7$
 ~~$.0128 * .1$~~

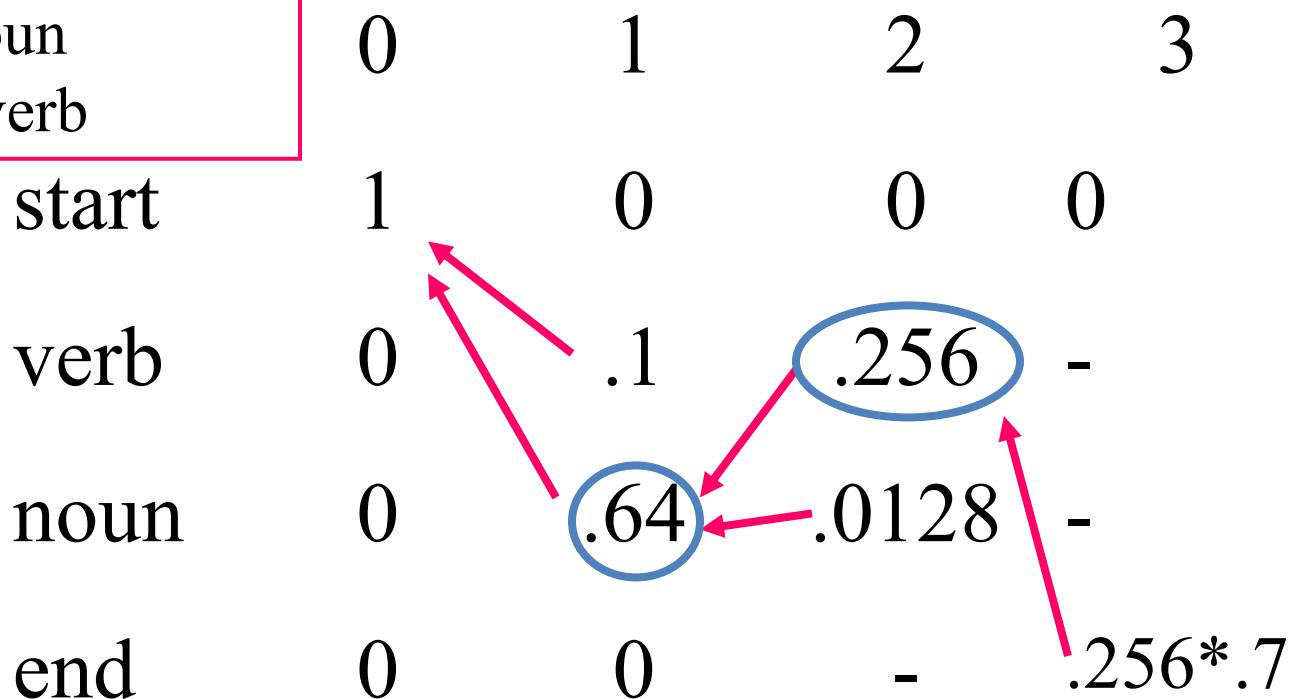


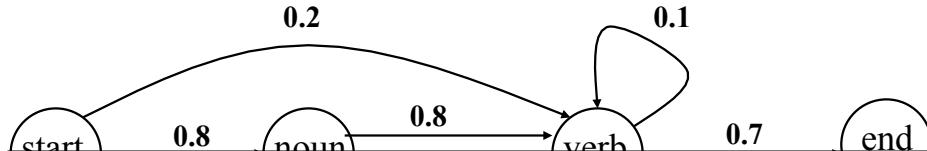
| $P(x y)$ | $y = \text{Noun}$ | $y = \text{Verb}$ |
|--------------------|-------------------|-------------------|
| $x = \text{fish}$ | 0.8 | 0.5 |
| $x = \text{sleep}$ | 0.2 | 0.5 |

Decode:

fish = noun

sleep = verb





| $P(x y)$ | $y = \text{"Noun"}$ | $y = \text{"Verb"}$ |
|---------------------|---------------------|---------------------|
| $x = \text{"fish"}$ | 0.8 | 0.5 |

What might go wrong for long sequences?

fish noun

sleep = verb

start



Underflow!

Small numbers get repeatedly multiplied together – exponentially small!

end



Viterbi Algorithm (w/ Log Probabilities)

- Solve: $\underset{y}{\operatorname{argmax}} P(y \mid x) = \underset{y}{\operatorname{argmax}} \frac{P(y, x)}{P(x)}$
 $= \underset{y}{\operatorname{argmax}} P(y, x)$
 $= \underset{y}{\operatorname{argmax}} \log P(x \mid y) + \log P(y)$
- For $k=1..M$
 - Iteratively solve for each $\hat{Y}^k(Z)$
 - Z looping over every POS tag.
- Predict best $\hat{Y}^M(Z)$
 - **Log($\hat{Y}^M(Z)$) accumulates additively, not multiplicatively**

Recap: Independent Classification

$x = \text{"I fish often"}$

POS Tags:
Det, Noun, Verb, Adj, Adv, Prep

- Treat each word independently
 - Independent multiclass prediction per word

| $P(y x)$ | $x = \text{"I"}$ | $x = \text{"fish"}$ | $x = \text{"often"}$ |
|----------|------------------|---------------------|----------------------|
| y="Det" | 0.0 | 0.0 | 0.0 |
| y="Noun" | 1.0 | 0.75 | 0.0 |
| y="Verb" | 0.0 | 0.25 | 0.0 |
| y="Adj" | 0.0 | 0.0 | 0.4 |
| y="Adv" | 0.0 | 0.0 | 0.6 |
| y="Prep" | 0.0 | 0.0 | 0.0 |

Prediction: (N, N, Adv)

Correct: (N, V, Adv)

**Mistake due to not
modeling multiple words.**

Assume pronouns are nouns for simplicity.

Recap: Viterbi

- Models pairwise transitions between states
 - Pairwise transitions between POS Tags
 - “1st order” model

$$P(x, y) = P(End \mid y^M) \prod_{i=1}^M P(y^i \mid y^{i-1}) \prod_{i=1}^M P(x^i \mid y^i)$$

x=“I fish often”

Independent: (N, N, Adv)

HMM Viterbi: (N, V, Adv)

*Assuming we defined $P(x,y)$ properly

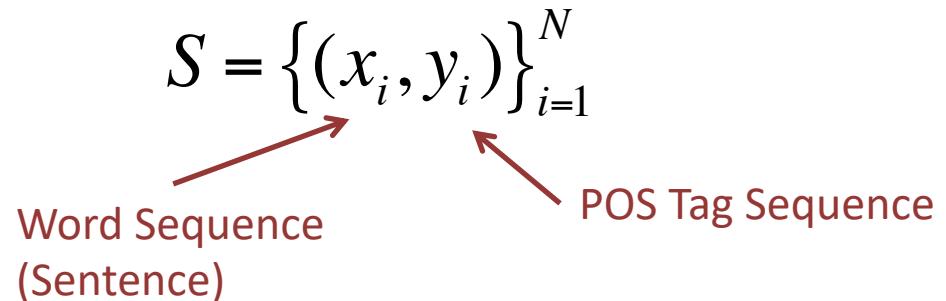
Training HMMs

Supervised Training

- **Given:**

$$S = \{(x_i, y_i)\}_{i=1}^N$$

Word Sequence (Sentence) POS Tag Sequence



- **Goal:** Estimate $P(x, y)$ using S

$$P(x, y) = P(End \mid y^M) \prod_{i=1}^M P(y^i \mid y^{i-1}) \prod_{i=1}^M P(x^i \mid y^i)$$

- **Maximum Likelihood!**

Aside: Matrix Formulation

- Define Transition Matrix: A

- $A_{ab} = P(y^{i+1}=a | y^i=b)$ or $-\log(P(y^{i+1}=a | y^i=b))$

| $P(y^{\text{next}} y)$ | $y = \text{"Noun"}$ | $y = \text{"Verb"}$ |
|-----------------------------------|---------------------|---------------------|
| $y^{\text{next}} = \text{"Noun"}$ | 0.09 | 0.667 |
| $y^{\text{next}} = \text{"Verb"}$ | 0.91 | 0.333 |

- Observation Matrix: O

- $O_{wz} = P(x^i=w | y^i=z)$ or $-\log(P(x^i=w | y^i=z))$

| $P(x y)$ | $y = \text{"Noun"}$ | $y = \text{"Verb"}$ |
|----------------------|---------------------|---------------------|
| $x = \text{"fish"}$ | 0.8 | 0.5 |
| $x = \text{"sleep"}$ | 0.2 | 0.5 |

Aside: Matrix Formulation

$$P(x, y) = P(\text{End} \mid y^M) \prod_{i=1}^M P(y^i \mid y^{i-1}) \prod_{i=1}^M P(x^i \mid y^i)$$

$$P(x, y) = P(\text{End} \mid y^M) \prod_{i=1}^M P(y^i \mid y^{i-1}) \prod_{i=1}^M P(x^i \mid y^i)$$

$$= A_{\text{End}, y^M} \prod_{i=1}^M A_{y^i, y^{i-1}} \prod_{i=1}^M O_{x^i, y^i}$$

$$-\log(P(x, y)) = \tilde{A}_{\text{End}, y^M} + \sum_{i=1}^M \tilde{A}_{y^i, y^{i-1}} + \sum_{i=1}^M \tilde{O}_{x^i, y^i}$$

Log prob. formulation
Each entry of \tilde{A} is
defined as $-\log(A)$

Maximum Likelihood

$$\operatorname{argmax}_{A,O} \prod_{(x,y) \in S} P(x,y) = \operatorname{argmax}_{A,O} \prod_{(x,y) \in S} P(\text{End} \mid y^M) \prod_{i=1}^M P(y^i \mid y^{i-1}) \prod_{i=1}^M P(x^i \mid y^i)$$

- Estimate each component separately:

$$A_{ab} = \frac{\sum_{j=1}^N \sum_{i=0}^{M_j} 1_{[(y_j^{i+1}=a) \wedge (y_j^i=b)]}}{\sum_{j=1}^N \sum_{i=0}^{M_j} 1_{[y_j^i=b]}}$$

$$O_{wz} = \frac{\sum_{j=1}^N \sum_{i=1}^{M_j} 1_{[(x_j^i=w) \wedge (y_j^i=z)]}}{\sum_{j=1}^N \sum_{i=1}^{M_j} 1_{[y_j^i=z]}}$$

- (Derived via minimizing neg. log likelihood)

Recap: Supervised Training

$$\operatorname{argmax}_{A,O} \prod_{(x,y) \in S} P(x,y) = \operatorname{argmax}_{A,O} \prod_{(x,y) \in S} P(\text{End} \mid y^M) \prod_{i=1}^M P(y^i \mid y^{i-1}) \prod_{i=1}^M P(x^i \mid y^i)$$

- Maximum Likelihood Training
 - Counting statistics
 - Super easy!
 - Why?
- What about unsupervised case?

Recap: Supervised Training

$$\operatorname{argmax}_{A,O} \prod_{(x,y) \in S} P(x,y) = \operatorname{argmax}_{A,O} \prod_{(x,y) \in S} P(\text{End} \mid y^M) \prod_{i=1}^M P(y^i \mid y^{i-1}) \prod_{i=1}^M P(x^i \mid y^i)$$

- Maximum Likelihood Training
 - Counting statistics
 - Super easy!
 - Why?
- What about unsupervised case?

Conditional Independence Assumptions

$$\operatorname{argmax}_{A,O} \prod_{(x,y) \in S} P(x,y) = \operatorname{argmax}_{A,O} \prod_{(x,y) \in S} P(\text{End} \mid y^M) \prod_{i=1}^M P(y^i \mid y^{i-1}) \prod_{i=1}^M P(x^i \mid y^i)$$

- Everything decomposes to products of pairs
 - I.e., $P(y^{i+1}=a \mid y^i=b)$ doesn't depend on anything else
- Can just estimate frequencies:
 - How often $y^{i+1}=a$ when $y^i=b$ over training set
 - Note that $P(y^{i+1}=a \mid y^i=b)$ is a common model across all locations of all sequences.

Conditional Independence Assumptions

$$\operatorname{argmax}_{A,O} \prod_{(x,y) \in S} P(x,y) = \operatorname{argmax}_{A,O} \prod_{(x,y) \in S} P(\text{End} \mid y^M) \prod_{i=1}^M P(y^i \mid y^{i-1}) \prod_{i=1}^M P(x^i \mid y^i)$$

Parameters:

Transitions A: #Tags²

Observations O: #Words x #Tags

Avoids directly model word/word pairings

#Tags = 10s

#Words = 10000s

Unsupervised Training

- What about if no y's?
 - Just a training set of sentences

$$S = \{x_i\}_{i=1}^N$$


Word Sequence
(Sentence)

- Still want to estimate $P(x,y)$
 - How?
 - Why?

$$\arg \max \prod_i P(x_i) = \operatorname{argmax} \prod_i \sum_y P(x_i, y)$$

Unsupervised Training

- What about if no y's?
 - Just a training set of sentences

$$S = \{x_i\}_{i=1}^N$$


Word Sequence
(Sentence)

- Still want to estimate $P(x,y)$
 - How?
 - Why?

$$\arg \max \prod_i P(x_i) = \operatorname{argmax} \prod_i \sum_y P(x_i, y)$$

Why Unsupervised Training?

- Supervised Data hard to acquire
 - Require annotating POS tags
- Unsupervised Data plentiful
 - Just grab some text!
- Might just work for POS Tagging!
 - Learn y's that correspond to POS Tags
- Can be used for other tasks
 - Detect outlier sentences (sentences with low prob.)
 - Sampling new sentences.

EM Algorithm (Baum-Welch)

- If we had y 's \rightarrow max likelihood.
- If we had (A, O) \rightarrow predict y 's

Chicken vs Egg!

1. Initialize A and O arbitrarily
2. Predict prob. of y 's for each training x
3. Use y 's to estimate new (A, O)
4. Repeat back to Step 1 until convergence

Expectation Step



Maximization Step



Expectation Step

- Given (A, O)
- For training $x=(x^1, \dots, x^M)$
 - Predict $P(y^i)$ for each $y=(y^1, \dots, y^M)$

| | x^1 | x^2 | ... | x^L |
|----------------------|-------|-------|-----|-------|
| $P(y^i=\text{Noun})$ | 0.5 | 0.4 | ... | 0.05 |
| $P(y^i=\text{Det})$ | 0.4 | 0.6 | ... | 0.25 |
| $P(y^i=\text{Verb})$ | 0.1 | 0.0 | ... | 0.7 |

- Encodes current model's beliefs about y
- “Marginal Distribution” of each y^i

Recall: Matrix Formulation

- Define Transition Matrix: A
 - $A_{ab} = P(y^{i+1}=a | y^i=b)$ or $-\text{Log}(P(y^{i+1}=a | y^i=b))$

| $P(y^{\text{next}} y)$ | $y=“\text{Noun}”$ | $y=“\text{Verb}”$ |
|---------------------------------|-------------------|-------------------|
| $y^{\text{next}}=“\text{Noun}”$ | 0.09 | 0.667 |
| $y^{\text{next}}=“\text{Verb}”$ | 0.91 | 0.333 |

- Observation Matrix: O
 - $O_{wz} = P(x^i=w | y^i=z)$ or $-\text{Log}(P(x^i=w | y^i=z))$

| $P(x y)$ | $y=“\text{Noun}”$ | $y=“\text{Verb}”$ |
|--------------------|-------------------|-------------------|
| $x=“\text{fish}”$ | 0.8 | 0.5 |
| $x=“\text{sleep}”$ | 0.2 | 0.5 |

Maximization Step

- Max. Likelihood over Marginal Distribution

Supervised: $A_{ab} = \frac{\sum_{j=1}^N \sum_{i=0}^{M_j} 1_{[(y_j^{i+1}=a) \wedge (y_j^i=b)]}}{\sum_{j=1}^N \sum_{i=0}^{M_j} 1_{[y_j^i=b]}}$

Unsupervised: $A_{ab} = \frac{\sum_{j=1}^N \sum_{i=0}^{M_j} P(y_j^i = b, y_j^{i+1} = a)}{\sum_{j=1}^N \sum_{i=0}^{M_j} P(y_j^i = b)}$

$$O_{wz} = \frac{\sum_{j=1}^N \sum_{i=1}^{M_j} 1_{[(x_j^i=w) \wedge (y_j^i=z)]}}{\sum_{j=1}^N \sum_{i=1}^{M_j} 1_{[y_j^i=z]}}$$

$$O_{wz} = \frac{\sum_{j=1}^N \sum_{i=1}^{M_j} 1_{[x_j^i=w]} P(y_j^i = z)}{\sum_{j=1}^N \sum_{i=1}^{M_j} P(y_j^i = z)}$$

Computing Marginals

(Forward-Backward Algorithm)

- Solving E-Step, requires compute marginals

| | x^1 | x^2 | ... | x^L |
|----------------------|-------|-------|-----|-------|
| $P(y^i=\text{Noun})$ | 0.5 | 0.4 | ... | 0.05 |
| $P(y^i=\text{Det})$ | 0.4 | 0.6 | ... | 0.25 |
| $P(y^i=\text{Verb})$ | 0.1 | 0.0 | ... | 0.7 |

- Can solve using Dynamic Programming!
 - Similar to Viterbi

Notation

Probability of observing prefix $x^{1:i}$ and having the i-th state be $y^i = Z$

$$\alpha_z(i) = P(x^{1:i}, y^i = Z | A, O)$$



Probability of observing suffix $x^{i+1:M}$ given the i-th state being $y^i = Z$

$$\beta_z(i) = P(x^{i+1:M} | y^i = Z, A, O)$$

Computing Marginals = Combining the Two Terms

$$P(y^i = z | x) = \frac{a_z(i)\beta_z(i)}{\sum_{z'} a_{z'}(i)\beta_{z'}(i)}$$

Notation

Probability of observing prefix $x^{1:i}$ and having the i-th state be $y^i = Z$

$$\alpha_z(i) = P(x^{1:i}, y^i = Z | A, O)$$

Probability of observing suffix $x^{i+1:M}$ given the i-th state being $y^i = Z$

$$\beta_z(i) = P(x^{i+1:M} | y^i = Z, A, O)$$

Computing Marginals = Combining the Two Terms

$$P(y^i = b, y^{i-1} = a | x) = \frac{a_a(i-1)P(y^i = b | y^{i-1} = a)P(x^i | y^i = b)\beta_b(i)}{\sum_{a', b'} a_{a'}(i-1)P(y^i = b' | y^{i-1} = a')P(x^i | y^i = b')\beta_{b'}(i)}$$

Forward (sub-)Algorithm

- Solve for every: $\alpha_z(i) = P(x^{1:i}, y^i = Z | A, O)$

- Naively: **Exponential Time!**

$$\alpha_z(i) = P(x^{1:i}, y^i = Z | A, O) = \sum_{y^{1:i-1}} P(x^{1:i}, y^i = Z, y^{1:i-1} | A, O)$$

- Can be computed recursively (like Viterbi)

$$\alpha_z(1) = P(y^1 = z | y^0) P(x^1 | y^1 = z) = O_{x^1, z} A_{z, start}$$

$$\alpha_z(i+1) = O_{x^{i+1}, z} \sum_{j=1}^L \alpha_j(i) A_{z, j}$$

Viterbi effectively replaces sum with max

Backward (sub-)Algorithm

- Solve for every: $\beta_z(i) = P(x^{i+1:M} \mid y^i = Z, A, O)$

- Naively: **Exponential Time!**

$$\beta_z(i) = P(x^{i+1:M} \mid y^i = Z, A, O) = \sum_{y^{i+1:L}} P(x^{i+1:M}, y^{i+1:L} \mid y^i = Z, A, O)$$

- Can be computed recursively (like Viterbi)

$$\beta_z(M) = 1$$

$$\beta_z(i) = \sum_{j=1}^L \beta_j(i+1) A_{j,z} O_{x^{i+1}, j}$$

Forward-Backward Algorithm

- Runs Forward $\alpha_z(i) = P(x^{1:i}, y^i = Z | A, O)$
- Runs Backward $\beta_z(i) = P(x^{i+1:M} | y^i = Z, A, O)$
- For each training $x = (x^1, \dots, x^M)$
 - Computes each $P(y^i)$ for $y = (y^1, \dots, y^M)$

$$P(y^i = z | x) = \frac{\alpha_z(i)\beta_z(i)}{\sum_{z'} \alpha_{z'}(i)\beta_{z'}(i)}$$

Recap: Unsupervised Training

- Train using only word sequences: $S = \{x_i\}_{i=1}^N$


Word Sequence
(Sentence)
- y's are “hidden states”
 - All pairwise transitions are through y's
 - Hence hidden Markov Model
- Train using EM algorithm
 - Converge to local optimum

Initialization

- How to choose #hidden states?
 - By hand
 - Cross Validation
 - $P(x)$ on validation data
 - Can compute $P(x)$ via forward algorithm:

$$P(x) = \sum_y P(x, y) = \sum_z \alpha_z(M) P(\text{End} \mid y^M = z)$$

Recap: Sequence Prediction & HMMs

- Models pairwise dependences in sequences

$x = \text{"I fish often"}$

POS Tags:

Det, Noun, Verb, Adj, Adv, Prep

Independent: (N, N, Adv)

HMM Viterbi: (N, V, Adv)

- Compact: only model pairwise between y 's
- **Main Limitation:** Lots of independence assumptions
 - Poor predictive accuracy

Next Lectures

- Thursday: Hidden Markov Models
 - (Unstructured Lecture)
- Next Week: Deep Generative Models
- Recitation Thursday
 - Recap of Viterbi and Forward/Backward