

## Hidden Markov Models (HMMs) for POS Tagging

# Probabilistic Tagging

- $W = w_1 \dots w_n$  - words in the corpus (observed)
- $T = t_1 \dots t_n$  - the corresponding tags (unknown)

**In this lecture, we will study how a probabilistic generative model can be used to identify POS tags of a sequence of words.**

# Probabilistic Tagging

- $W = w_1 \dots w_n$  - words in the corpus (observed)
- $T = t_1 \dots t_n$  - the corresponding tags (unknown)

## Tagging: Probabilistic View (Generative Model)

Find

$$\hat{T} = \operatorname{argmax}_T P(T|W)$$

**Among all the possible sequences of (POS) tags that the given word-sequence can take, find the tag-sequence with the highest probability.**

# Probabilistic Tagging

- $W = w_1 \dots w_n$  - words in the corpus (observed)
- $T = t_1 \dots t_n$  - the corresponding tags (unknown)

## Tagging: Probabilistic View (Generative Model)

Find

$$\begin{aligned}\hat{T} &= \operatorname{argmax}_T P(T|W) \\ &= \operatorname{argmax}_T \frac{P(W|T)P(T)}{P(W)}\end{aligned}$$

We have inverted the direction of the probabilities. Instead of finding probability of the tags given the words, we will now find the probability of the words given the tags.

# Probabilistic Tagging

- $W = w_1 \dots w_n$  - words in the corpus (observed)
- $T = t_1 \dots t_n$  - the corresponding tags (unknown)

## Tagging: Probabilistic View (Generative Model)

Find

$$\begin{aligned}\hat{T} &= \operatorname{argmax}_T P(T|W) \\ &= \operatorname{argmax}_T \frac{P(W|T)P(T)}{P(W)} \\ &= \operatorname{argmax}_T P(W|T)P(T)\end{aligned}$$

For a given word sequence,  $P(W)$  is constant for all  $T$ .

# Probabilistic Tagging

- $W = w_1 \dots w_n$  - words in the corpus (observed)
- $T = t_1 \dots t_n$  - the corresponding tags (unknown)

## Tagging: Probabilistic View (Generative Model)

Find

$$\begin{aligned}\hat{T} &= \operatorname{argmax}_T P(T|W) \\ &= \operatorname{argmax}_T \frac{P(W|T)P(T)}{P(W)} \\ &= \operatorname{argmax}_T P(W|T)P(T) \\ &= \operatorname{argmax}_T \prod_i P(w_i|w_1 \dots w_{i-1}, t_1 \dots t_i)P(t_i|t_1 \dots t_{i-1})\end{aligned}$$

## Further simplifications

$$\hat{T} = \operatorname{argmax}_T \prod_i P(w_i | w_1 \dots w_{i-1}, t_1 \dots t_i) P(t_i | t_1 \dots t_{i-1})$$

- The probability of a word appearing depends only on its own POS tag

**Remember: a generative model assumes that first the POS tags are decided, and then the words are generated.**

## Further simplifications

$$\hat{T} = \operatorname{argmax}_T \prod_i P(w_i | w_1 \dots w_{i-1}, t_1 \dots t_i) P(t_i | t_1 \dots t_{i-1})$$

- The probability of a word appearing depends only on its own POS tag  
 $P(w_i | w_1 \dots w_{i-1}, t_1 \dots t_i) \approx P(w_i | t_i)$
- Bigram assumption: the probability of a tag appearing depends only on the previous tag



## Further simplifications

$$\hat{T} = \operatorname{argmax}_T \prod_i P(w_i | w_1 \dots w_{i-1}, t_1 \dots t_i) P(t_i | t_1 \dots t_{i-1})$$

- The probability of a word appearing depends only on its own POS tag  
 $P(w_i | w_1 \dots w_{i-1}, t_1 \dots t_i) \approx P(w_i | t_i)$
- Bigram assumption: the probability of a tag appearing depends only on the previous tag  
 $P(t_i | t_1 \dots t_{i-1}) \approx P(t_i | t_{i-1})$

## Further simplifications

$$\hat{T} = \operatorname{argmax}_T \prod_i P(w_i | w_1 \dots w_{i-1}, t_1 \dots t_i) P(t_i | t_1 \dots t_{i-1})$$

- The probability of a word appearing depends only on its own POS tag  
 $P(w_i | w_1 \dots w_{i-1}, t_1 \dots t_i) \approx P(w_i | t_i)$
- Bigram assumption: the probability of a tag appearing depends only on the previous tag

$$P(t_i | t_1 \dots t_{i-1}) \approx P(t_i | t_{i-1})$$

- Using these simplifications:

$$\hat{T} = \operatorname{argmax}_T \prod_i P(w_i | t_i) P(t_i | t_{i-1})$$

**How to compute the probability values?**

**Assume we have a large corpus of text, where the POS tags have been marked for all words.**

# Computing the probability values

Tag Transition probabilities  $p(t_i|t_{i-1})$

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

$$P(NN|DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = 0.49$$

# Computing the probability values

*Tag Transition probabilities  $p(t_i|t_{i-1})$*

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

$$P(NN|DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = 0.49$$

*Word Likelihood probabilities  $p(w_i|t_i)$*

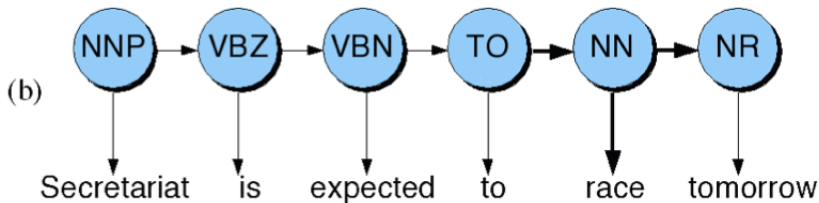
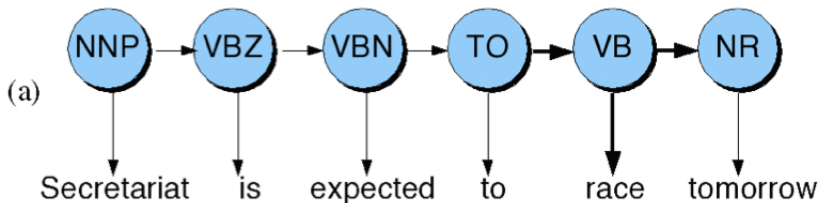
$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

$$P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = 0.47$$

Suppose now we have a way to compute such probabilities, based on a corpus.

How can we disambiguate the POS tag for a particular instance of a word that can have multiple tags?

# Disambiguating “race”



# Disambiguating “race”

*Difference in probability due to*

- $P(VB|TO)$  vs.  $P(NN|TO)$
- $P(race|VB)$  vs.  $P(race|NN)$
- $P(NR|VB)$  vs.  $P(NR|NN)$

These probabilities can be computed from a large text corpus where every word is marked with its POS tag.

The product of these three terms will determine which has higher probability  
- “race” being a VB or a NN.



# Disambiguating “race”

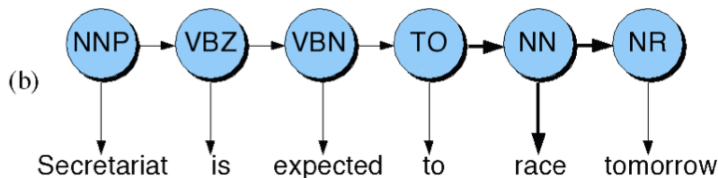
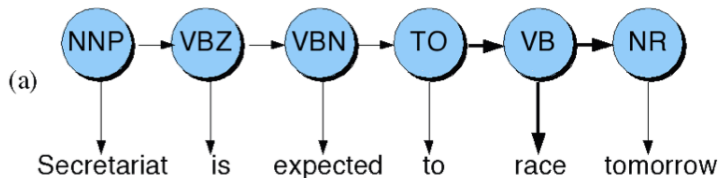
## Difference in probability due to

- $P(VB|TO)$  vs.  $P(NN|TO)$
- $P(race|VB)$  vs.  $P(race|NN)$
- $P(NR|VB)$  vs.  $P(NR|NN)$

## After computing the probabilities

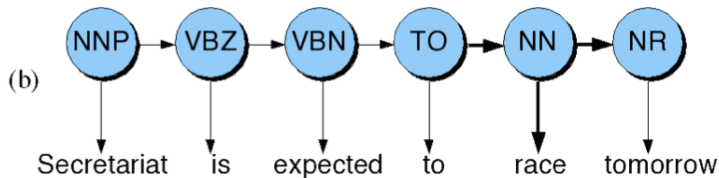
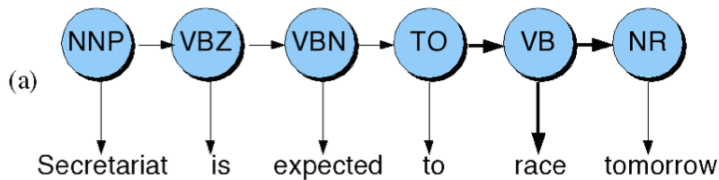
- $P(NN|TO)P(NR|NN)P(race|NN) = 0.0047 \times 0.0012 \times 0.00057 = 0.00000000032$
- $P(VB|TO)P(NR|VB)P(race|VB) = 0.83 \times 0.0027 \times 0.00012 = 0.000000027$

# What is this model?



We are observing a sequence of words. Also there is a set of tags. The model gives the probability of going from one tag (state) to another tag (state), and the probability of generating a word from a given tag (state).

# What is this model?



*This is a Hidden Markov Model*

# Hidden Markov Models

- Tag Transition probabilities  $p(t_i|t_{i-1})$
- Word Likelihood probabilities (emissions)  $p(w_i|t_i)$

- Tag Transition probabilities  $p(t_i|t_{i-1})$
- Word Likelihood probabilities (emissions)  $p(w_i|t_i)$
- What we have described with these probabilities is a hidden markov model.
- Let us quickly introduce the Markov Chain, or observable Markov Model.

# Markov Chain = First-order Markov Model

## Weather example

- Three types of weather: *sunny, rainy, foggy*

# Markov Chain = First-order Markov Model

## Weather example

- Three types of weather: *sunny, rainy, foggy*
- $q_n$ : variable denoting the weather on the  $n^{th}$  day

# Markov Chain = First-order Markov Model

## Weather example

- Three types of weather: *sunny, rainy, foggy*
- $q_n$ : variable denoting the weather on the  $n^{th}$  day
- We want to find the following conditional probabilities:

$$P(q_n | q_{n-1}, q_{n-2}, \dots, q_1)$$



# Markov Chain = First-order Markov Model

## Weather example

- Three types of weather: *sunny, rainy, foggy*
- $q_n$ : variable denoting the weather on the  $n^{th}$  day
- We want to find the following conditional probabilities:







$$P(q_n | q_{n-1}, q_{n-2}, \dots, q_1)$$

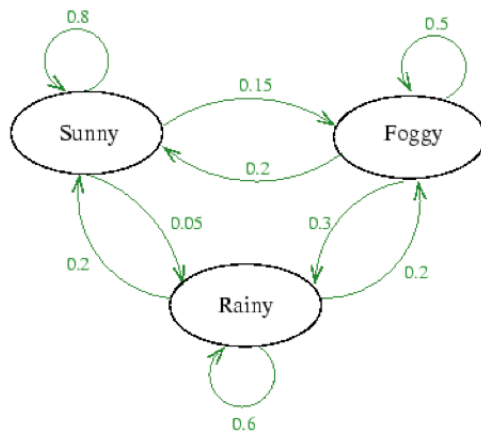
## First-order Markov Assumption

$$P(q_n | q_{n-1}, q_{n-2}, \dots, q_1) = P(q_n | q_{n-1})$$

# Markov Chain Transition Table

**Table 1:** Probabilities  $p(q_{n+1}|q_n)$  of tomorrow's weather based on today's weather

Today's weather	Tomorrow's weather		
			
	0.8	0.05	0.15
	0.2	0.6	0.2
	0.2	0.3	0.5



- Given that today the weather is sunny, what is the probability that tomorrow is sunny and day after is rainy?

# Using Markov Chain

- Given that today the weather is sunny, what is the probability that tomorrow is sunny and day after is rainy?

$$P(q_2 = \text{sunny}, q_3 = \text{rainy} | q_1 = \text{sunny})$$

- Given that today the weather is sunny, what is the probability that tomorrow is sunny and day after is rainy?

$$P(q_2 = \text{sunny}, q_3 = \text{rainy} | q_1 = \text{sunny})$$

$$= P(q_3 = \text{rainy} | q_2 = \text{sunny}, q_1 = \text{sunny}) \times P(q_2 = \text{sunny} | q_1 = \text{sunny})$$

- Given that today the weather is sunny, what is the probability that tomorrow is sunny and day after is rainy?

$$P(q_2 = \text{sunny}, q_3 = \text{rainy} | q_1 = \text{sunny})$$

$$= P(q_3 = \text{rainy} | q_2 = \text{sunny}, q_1 = \text{sunny}) \times P(q_2 = \text{sunny} | q_1 = \text{sunny})$$

$$= P(q_3 = \text{rainy} | q_2 = \text{sunny}) \times P(q_2 = \text{sunny} | q_1 = \text{sunny})$$

$$= 0.05 \times 0.8$$

$$= 0.04$$

- For Markov chains, the output symbols are the same as the states  
*‘sunny’ weather is both observable and state*

Markov chain also called “Observable Markov Model”

# Hidden Markov Model

- For Markov chains, the output symbols are the same as the states  
*‘sunny’ weather is both observable and state*
- But in POS tagging



# Hidden Markov Model

- For Markov chains, the output symbols are the same as the states  
*'sunny' weather is both observable and state*
- But in POS tagging  
*The output symbols are words*

# Hidden Markov Model

- For Markov chains, the output symbols are the same as the states  
*'sunny' weather is both observable and state*
- But in POS tagging  
*The output symbols are words*  
*But the hidden states are POS tags*
- A Hidden Markov Model is an extension of a Markov chain in which the output symbols are not the same as the states
- We don't know which state we are in

**The outputs (e.g., words) are observed, the states (e.g., POS tags) are hidden.**

# Hidden Markov Models (HMMs)

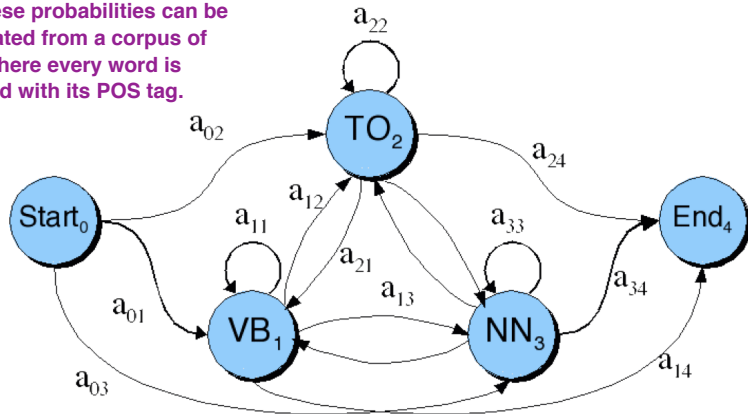
## *Elements of an HMM model*

- A set of states (here: the tags)
- An output alphabet (here: words)
- Initial state (here: beginning of sentence)
- State transition probabilities (here  $p(t_n|t_{n-1})$ )
- Symbol emission probabilities (here  $p(w_i|t_i)$ )

# Graphical Representation

When tagging a sentence, we are walking through the state graph:

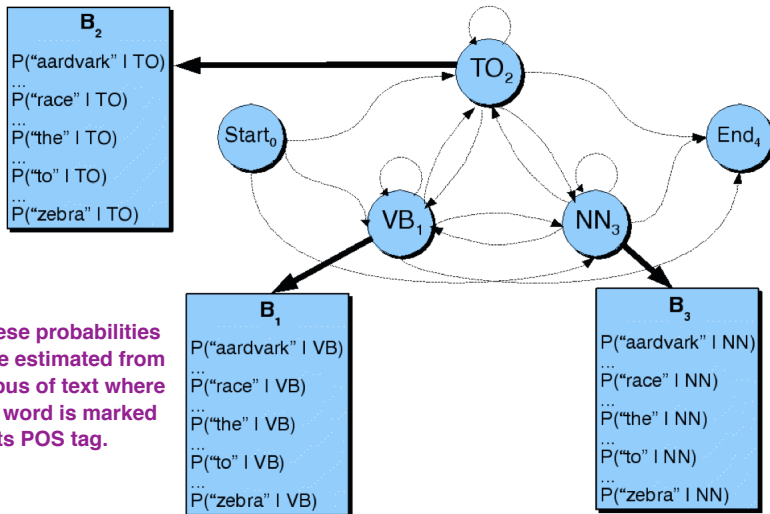
All these probabilities can be estimated from a corpus of text where every word is marked with its POS tag.



Edges are labeled with the state transition probabilities:  $p(t_n|t_{n-1})$

# Graphical Representation

At each state we emit a word:  $P(w_n|t_n)$



All these probabilities can be estimated from a corpus of text where every word is marked with its POS tag.

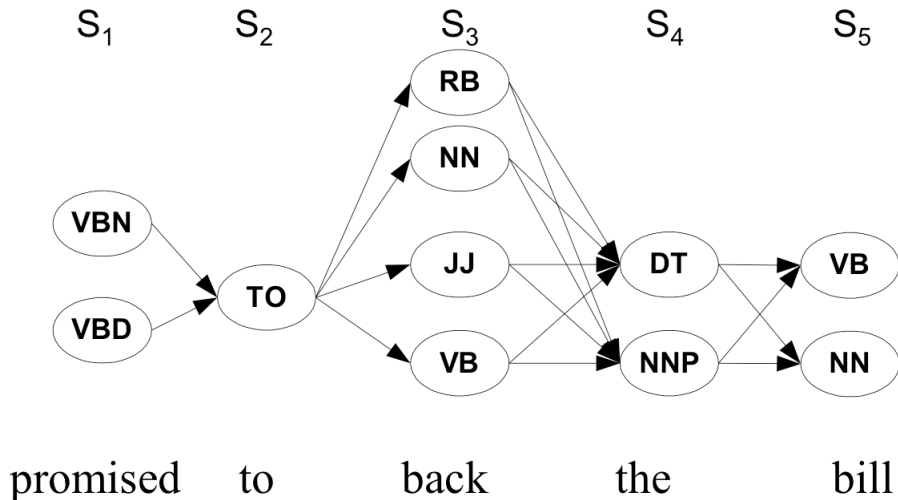
Now we have discussed the math behind HMMs.

Returning to the original problem: we are given a word sequence, and we need to find the most probable tag sequence.

## *Walking through the states: best path*

Given a sentence / phrase, there is a finite number of possible tag sequences, since each word can take only a fixed number of tags.

For the given example, there are  $2 \times 1 \times 4 \times 2 \times 2 = 32$  possible tag sequences.



## *Walking through the states: best path*

We can enumerate all possible tag sequences and compute their probabilities. But this approach will be very expensive for long word sequences. We will discuss an efficient algorithm for computing the most probable tag sequence.

