

# Statistical Methods in Natural Language Processing (NLP)

*Class 16: Statistical Tagging*



Charalambos (Haris) Themistocleous

*Department of Philosophy, Linguistics and Theory  
of Science, Centre for Linguistic Theory and Studies  
in Probability*

# Introduction

- ▶ Hidden Markov Models.
- ▶ Viterbi Algorithm.

# Part of Speech Tagger

- Let us start with a sequence of words: ['The', 'man', 'walks']
- our task is to predict a list of tags such as ['DT', 'N', 'V'] for these words, namely the Parts of Speech

- We start with the Bayes Rule.  $P(T)$  provides the tag sequences.

$$\begin{aligned}
 \bullet \frac{\arg \max}{T} P(T|W) &= \frac{\arg \max}{T} \frac{P(W|T) P(T)}{P(W)} \\
 &= \frac{\arg \max}{T} \frac{P(W|T) P(T)}{P(W)}
 \end{aligned}$$

- To analyze the sequence we should estimate  $P(T)$  and  $P(W|T)$
- **Transition probability:** when we rely on the previous tag as in a bigram tagger, Markov assumption, to predict the next tag:

$$P(t_n | t_{n-1}) \approx \\ P(t_n | t_1, \dots, t_{n-1})$$

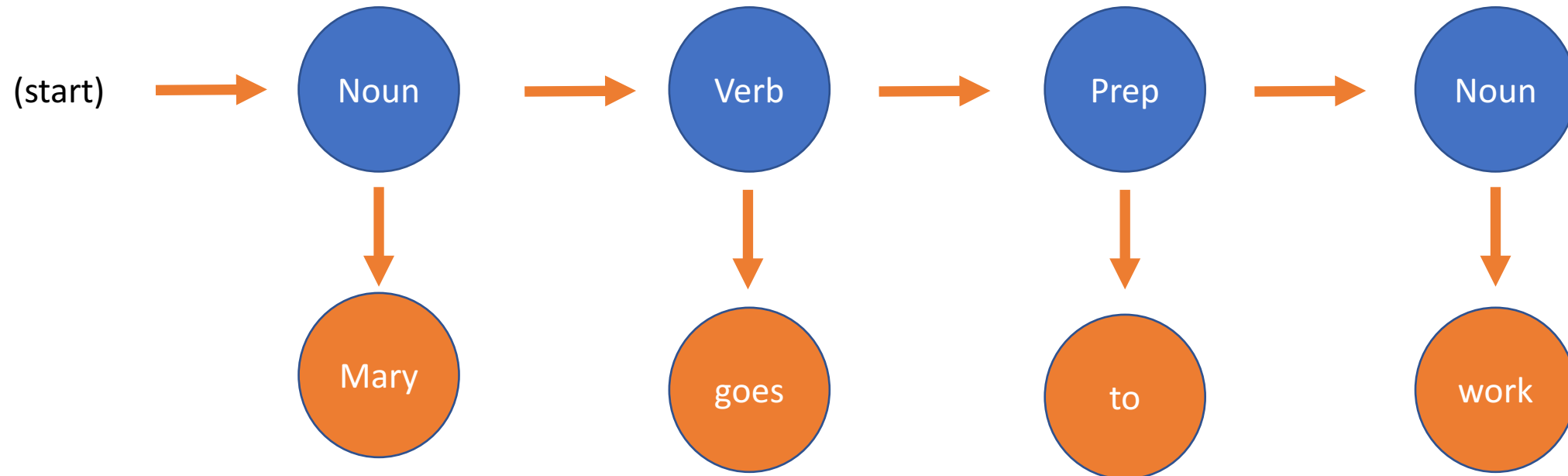
**Emission probability:** the probability of a word depends only on its tag:

$$P(w_n | t_n) \approx \\ P(w_n | \text{tags, other words})$$

# Hidden Markov models

- **Hidden Markov model (HMM)** are models where we have an unknown underlying sequence.

# Hidden Markov models: generative grammar



# Estimating probabilities

- From an annotated corpus (by humans), we estimate  $P(t_n | t_{n-1})$  and  $P(w_n | t_n)$ .

Maria	N
went	V
to	Prep
the	DT
cinema	N
.	.
In	Prep
my	Pro
opinion,	N
she	Pro
...	



- We estimate the probabilities by counting frequencies (**maximum likelihood estimation (MLE)**):

$$P_{MLE}(\text{noun} | \text{verb}) = \frac{\text{count}(\text{verb}, \text{noun})}{\text{count}(\text{verb})}$$

$$P_{MLE}(\text{table} | \text{noun}) = \frac{\text{count}(\text{noun}:\text{table})}{\text{count}(\text{noun})}$$

# Smoothing can be useful, especially when we have a small corpus:

**Laplace smoothing for transition probabilities:**

$$P(t_n | t_{n-1}) = \frac{\text{count}(t_{n-1}, t_n) + \lambda}{\text{count}(t_{n-1}) + \lambda \cdot T}$$

where

**T** is the number of distinct tags

**Laplace smoothing for emission probabilities:**

$$P(w | t) = \frac{\text{count}(w, t) + \lambda}{\text{count}(t) + \lambda \cdot V}$$

where

**V** is the number of distinct words

for the emission probability

$P(w_n | t_n)$  when  $w_n$  is unseen in the training corpus

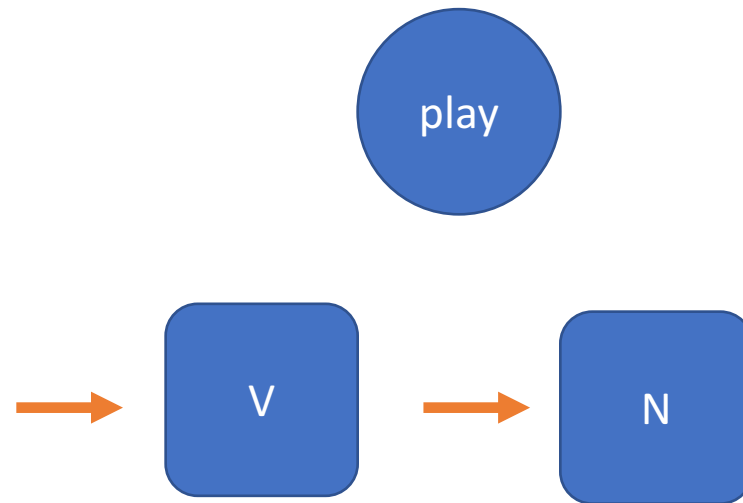
We can consider factors such as numbers, suffixes, capitalization, punctuation...

# Probabilities in tagging

1. enumerate all possible tag sequences;
2. use the probabilities to find the best one.
3. in long sentences, the number of possible tag sequences is very large, e.g., play: Noun or Verb
4. the Viterbi algorithm is used to find calculate the most probable underlying tags
5. Viterbi runs in linear time with respect to the length of the sentence

# the Viterbi algorithm

- for each possible tag  $t_i$  of a word  $w_i$ , we compute the best tag sequence leading to  $t_i$
- for instance: for the word *play*, we find the best sequence ending with *play* as a **verb**, and the best ending with *play* as a **noun**



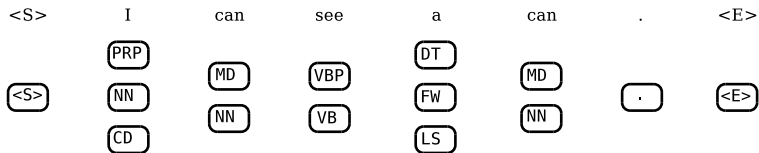
- to compute the best path ending with play as a verb, consider the best paths for the previous word and the transition probabilities
- assume the previous word is e.g. cooks, which can be a noun or a verb
- select the highest of the prob of the best path ending in cooks as a verb + the prob of the transition verb  $\rightarrow$  verb
- select the highest of the prob of the best path ending in cooks as a noun + the LP of the transition noun  $\rightarrow$  verb

- apply the Viterbi algorithm step by step
- after the last token of the sentence, add a special dummy end token
- this token will emit a dummy end tag with probability 1
- the best tag sequence for the whole sentence is the best path ending in the dummy tag
- finally, retrace your steps from the dummy item to get the tags so you need backpointers

## Viterbi example

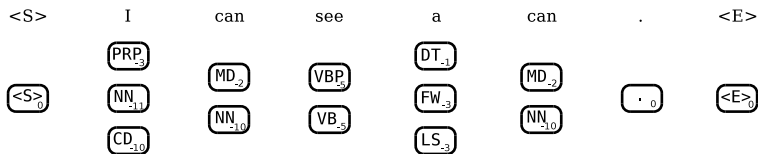
<S> I can see a can . <E>

## Viterbi example

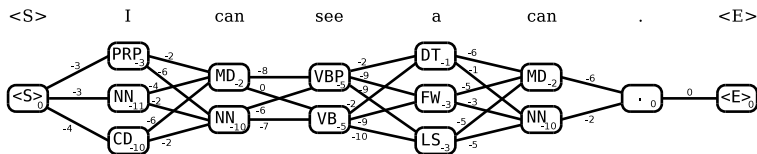




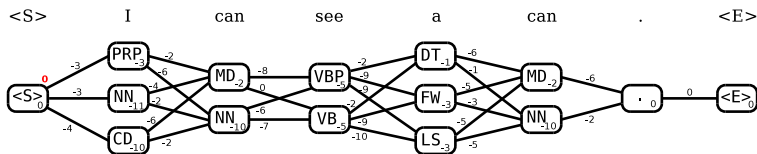
# Viterbi example



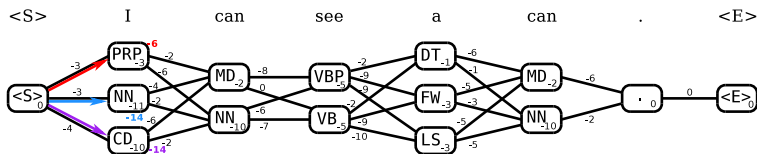
# Viterbi example



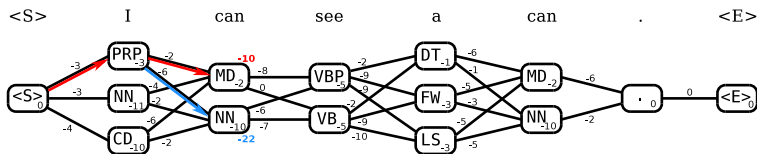
# Viterbi example



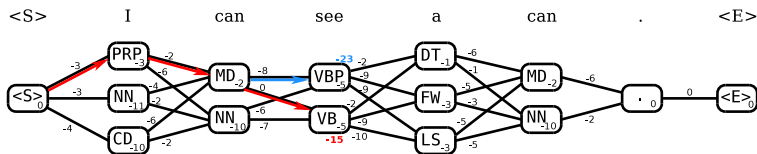
# Viterbi example



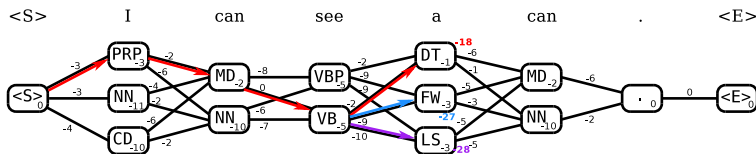
# Viterbi example



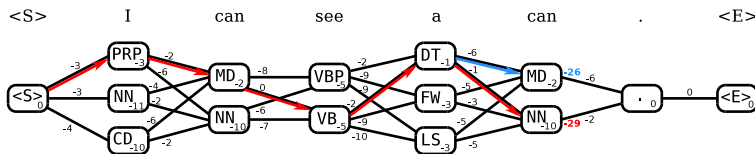
# Viterbi example



# Viterbi example

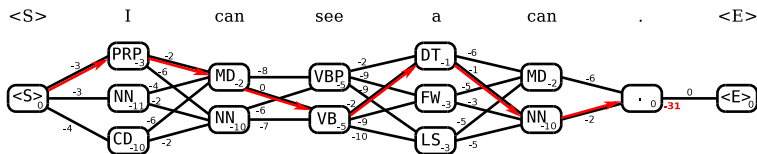


# Viterbi example

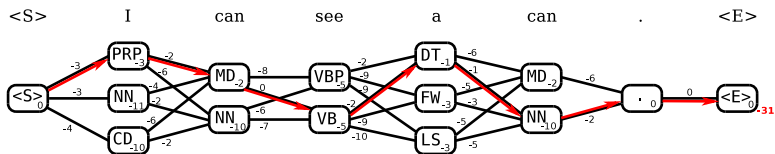




# Viterbi example



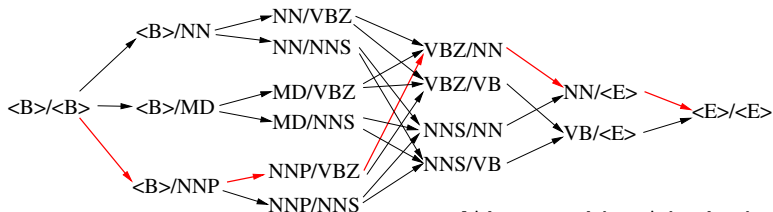
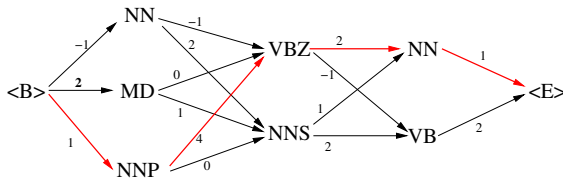
# Viterbi example





# Search spaces...

- ▶ example: *Will plays golf*



slide created by Richard Johansson