

# Natural Language Processing

DSECL ZG565

Session 3: Part-of-Speech Tagging

Dr. Ashish Kulkarni

BITS Pilani

April 18, 2021

# Session Content

1. Parts of Speech
2. Parts of Speech Tagging
3. HMM Part of Speech Tagging

# What are parts of speech?

Parts of speech are **word classes** or **syntactic categories** that group words that have similar neighboring words (distributional properties) or take similar affixes (morphological properties).  
*E.g.* noun, verb, pronoun, preposition, adverb, *etc.*

Parts of speech are useful:

- they tell us about the likely neighboring words and syntactic structure, making them a key aspect of parsing;
- they are useful features for labeling named entities like people or organizations in information extraction;
- they are useful in coreference resolution;
- play a role in speech recognition or synthesis, *e.g.* *CONtent* (noun) *v.s.* *conTENT* (adjective)

# English Word Classes

**Closed Class:** have relatively fixed membership

prepositions	on, under, over, near, by, at, from, to, with
particles	up, down, on, off, in, out, at, by
determiners	a, an, the
conjunctions	and, but, or, as, if, when
auxiliaries	may, can, will, had, been
pronouns	I, you, we, he, she, his, mine
numerals	one, two, three, first, second, third

**Open Class:** nouns, verbs, adjectives, and adverbs

# Open Class Words

**Nouns:** words for people, places, things, and others

- proper nouns: names of specific persons or entities, e.g. Bengaluru, IBM, Jurafsky;
- common nouns
  - count nouns: allow grammatical enumeration, occurring in both the singular and plural and they can be counted, e.g. goat/goats
  - mass nouns: used when something is conceptualized as a homogeneous group, e.g. snow, salt

**Verbs:**

- refer to actions and processes
- they have inflections: *eat, eats, eating, eaten*

**Adjectives:** refer to properties or qualities, like, *black, old, good*

**Adverbs:** modify a verb, adverb or a verb phrase

- directional adverbs: direction or location of action (*home, here*)
- degree adverbs: extent of some action (*extremely, somewhat*)
- manner adverbs: *slowly, delicately*
- temporal adverbs: describe the time of action (*yesterday, Monday*)

# The Penn Treebank Part-of-Speech Tagset

For POS tagging, we need to choose a standard set of tags to work with. An important tagset for English is the **45-tag Penn Treebank tagset**.

- The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.
- There/EX are/VBP 70/CD children/NNS there/RB

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	<i>and, but, or</i>	PDT	predeterminer	<i>all, both</i>	VBP	verb non-3sg present	<i>eat</i>
CD	cardinal number	<i>one, two</i>	POS	possessive ending	<i>'s</i>	VBZ	verb 3sg pres	<i>eats</i>
DT	determiner	<i>a, the</i>	PRP	personal pronoun	<i>I, you, he</i>	WDT	wh-determ.	<i>which, that</i>
EX	existential 'there'	<i>there</i>	PRP\$	possess. pronoun	<i>your, one's</i>	WP	wh-pronoun	<i>what, who</i>
FW	foreign word	<i>mea culpa</i>	RB	adverb	<i>quickly</i>	WP\$	wh-possess.	<i>whose</i>
IN	preposition/ subordin-conj	<i>of, in, by</i>	RBR	comparative adverb	<i>faster</i>	WRB	wh-adverb	<i>how, where</i>
JJ	adjective	<i>yellow</i>	RBS	superlatv. adverb	<i>fastest</i>	\$	dollar sign	<i>\$</i>
JJR	comparative adj	<i>bigger</i>	RP	particle	<i>up, off</i>	#	pound sign	<i>#</i>
JJS	superlative adj	<i>wildest</i>	SYM	symbol	<i>+, %, &amp;</i>	"	left quote	<i>' or "</i>
LS	list item marker	<i>1, 2, One</i>	TO	"to"	<i>to</i>	"	right quote	<i>' or "</i>
MD	modal	<i>can, should</i>	UH	interjection	<i>ah, oops</i>	(	left paren	<i>[, (, {, &lt;</i>
NN	sing or mass noun	<i>llama</i>	VB	verb base form	<i>eat</i>	)	right paren	<i>], ), }, &gt;</i>
NNS	noun, plural	<i>llamas</i>	VBD	verb past tense	<i>ate</i>	,	comma	<i>,</i>
NNP	proper noun, sing.	<i>IBM</i>	VBG	verb gerund	<i>eating</i>	.	sent-end punc	<i>. ! ?</i>
NNPS	proper noun, plu.	<i>Carolinas</i>	VBN	verb past part.	<i>eaten</i>	:	sent-mid punc	<i>: ; ... --</i>

# POS Labeled Datasets

Corpora labeled with parts of speech are crucial training (and testing) sets for statistical tagging algorithms. Three main tagged corpora:

<b>Brown corpus</b>	million words of samples from 500 written texts from different genres published in the United States in 1961
<b>WSJ corpus</b>	a million words published in the Wall Street Journal in 1989
<b>Switchboard corpus</b>	2 million words of telephone conversations collected in 1990-1991

# What is POS Tagging?

**Part-of-speech tagging** is the process of assigning a part-of-speech marker to each word in an input text.

Words are ambiguous and have more than one possible part-of-speech. The goal is to find the correct tag for the situation. *E.g.*

<i>book</i> that flight	v.s.	hand me that <i>book</i>
Does <i>that</i> flight serve dinner	v.s.	I thought <i>that</i> your flight was earlier

Types:	WSJ	Brown
Unambiguous (1 tag)	44,432 (86%)	45,799 (85%)
Ambiguous (2+ tags)	7,025 (14%)	8,050 (15%)
Tokens:		
Unambiguous (1 tag)	577,421 (45%)	384,349 (33%)
Ambiguous (2+ tags)	711,780 (55%)	786,646 (67%)



# Most Frequent Class Baseline

In spite of the inherent ambiguity, many words are easy to disambiguate, because their different tags aren't equally likely.

**Most frequent class baseline:** given an ambiguous word, choose the tag which is **most frequent** in the training corpus.

**How good is this baseline?**

Achieves 92.34% accuracy on the WSJ corpus. By contrast, the state of the art in part-of-speech tagging on this dataset is around 97%.

# What is the Hidden Markov Model

The Hidden Markov Model (HMM) is a **sequence model**: given a sequence of observations, the model assigns a class label to each unit in the sequence.

It is a **probabilistic** model: given a sequence of units (words, letters, morphemes, sentences), it computes a probability distribution over possible sequences of labels and chooses the best label sequence.

It is a **generative** model: models the joint probability of the sequence of observations and labels.

# Understanding Markov Chains

A Markov chain is a model that tells us something about the probabilities of sequences of random variables, states, each of which can take on values from some set.

Assumes that prediction of future states in the sequence depend solely on the current state.

For a sequence of states  $q_1, q_2, \dots, q_i$ ,

**Markov assumption:**

$$P(q_i = a | q_1 \dots q_{i-1}) = P(q_i = a | q_{i-1})$$

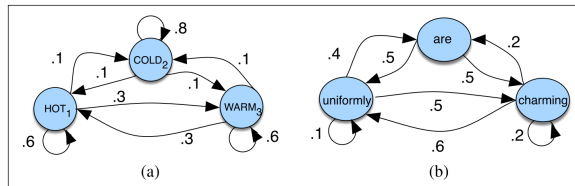


Figure: A Markov chain for weather (a) and one for words (b);  $\pi = [0.1, 0.7, 0.2]$  for (a)

$$Q = q_1 q_2 \dots q_N$$

$$A = a_{11} a_{12} \dots a_{n1} \dots a_{mn}$$

$$\pi = \pi_1, \pi_2, \dots, \pi_N$$

a set of  $N$  states

a transition probability matrix  $A$ , each  $a_{ij}$  is the probability of moving from state  $i$  to state  $j$ , s.t.  $\sum_{j=1}^n a_{ij} = 1 \forall i$

an initial probability distribution over states.  $\pi_i$  is the probability that the Markov chain will start in state  $i$ .  $\sum_{i=1}^n \pi_i = 1$

# Exercise: Markov Chains

Use the sample probabilities in the figure to compute the probability of each of the following sequences:

hot hot hot hot

cold hot cold hot

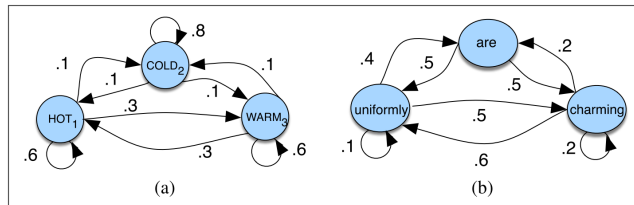


Figure: A Markov chain for weather (a) and one for words (b);  $\pi = [0.1, 0.7, 0.2]$  for (a)

# The Hidden Markov Model

Markov chains are useful to model a sequence of observable events.

In some cases, the events that we are interested in, are **hidden**: we don't observe them directly.

**POS tagging**: we only observe the sequence of words; tags are said to be hidden as they are not observed.

A **Hidden Markov Model (HMM)** allows us to jointly model both observed and hidden events.

$$Q = q_1 q_2 \dots q_n$$

a set of  $N$  states

$$A = a_{11} \dots a_{ij} \dots a_{NN}$$

a transition probability matrix  $A$ , each  $a_{ij}$  representing the probability of moving from state  $i$  to state  $j$  s.t.  $\sum_{j=1}^N a_{ij} = 1 \forall i$

$$O = o_1 \dots o_T$$

a sequence of  $T$  observations, each one drawn from a vocabulary  $V = v_1, v_2, \dots, v_V$

$$B = b_i(o_t)$$

observation likelihoods (or emission probabilities): the probability of an observation  $o_t$  being generated from state  $i$

$$\pi = \pi_1, \pi_2, \dots, \pi_N$$

an initial probability distribution over states;  $\pi_i$  is the probability that the Markov chain will start in state  $i$ .  $\sum_{i=1}^N \pi_i = 1$

# HMM Assumptions

A first-order HMM makes two simplifying assumptions

**Markov Assumption:** probability of a particular state depends only on the previous state;

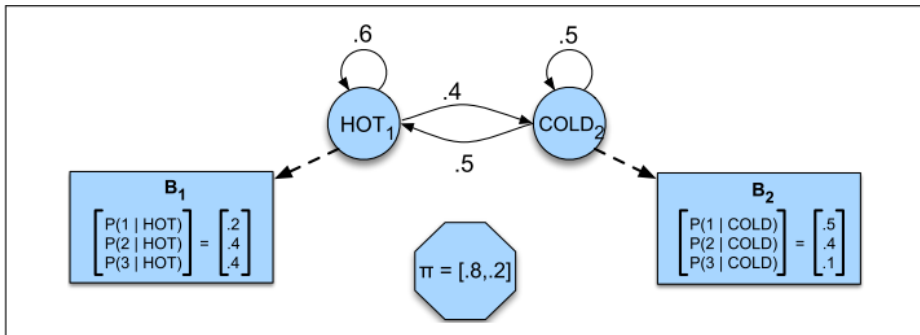
$$P(q_i | q_1 \dots q_{i-1}) = P(q_i | q_{i-1}) \quad (1)$$

**Output Independence:** probability of an output observation  $o_i$  depends only on the state that produced the observation  $q_i$  and not on any other observations or states;

$$P(o_i | q_1 \dots q_i \dots q_T, o_1 \dots o_i \dots o_T) = P(o_i | q_i) \quad (2)$$

# A Running Example: Eisner Task

Given a sequence of **observations**  $O$ , each an integer representing the number of ice creams eaten on a day, find the **hidden sequence**  $Q$  of **weather states**  $H$  and  $C$  which caused Jason to eat the ice creams.



**Figure:** A hidden Markov model for relating numbers of ice creams eaten by Jason (the observations) to the weather (H or C, the hidden variables).

# HMM Problem 1: Likelihood

Given an HMM  $\lambda = (A, B)$  and an observation sequence  $O$ , determine the likelihood  $P(O|\lambda)$ .

In our running example, what is the probability of the observation sequence “3 1 3”?

For a given hidden state sequence (e.g. *hot hot cold*), we have:

$$P(3\ 1\ 3|\text{hot hot cold}) = P(3|\text{hot}) \times P(1|\text{hot}) \times P(3|\text{cold}) \quad (3)$$

**But we don't know the hidden state sequence!**



# HMM Problem 1: Likelihood cont...

Apply the rule of **marginal probability**

$$P(O) = \sum_Q P(O, Q) = \sum_Q P(O|Q)P(Q) \quad (4)$$

where, from the HMM assumptions, we have:

$$P(O|Q) \times P(Q) = \prod_{i=1}^T P(o_i|q_i) \times \prod_{i=1}^T P(q_i|q_{i-1}) \quad (5)$$

For our running example:

$$P(3 \ 1 \ 3) = P(3 \ 1 \ 3, \text{cold cold cold}) + P(3 \ 1 \ 3, \text{cold hot cold}) + P(3 \ 1 \ 3, \text{hot cold hot}) + \dots$$

With  $N$  hidden states and  $T$  observations, there are  $N^T$  possible sequences! An exponential computational complexity!

# Forward Algorithm for Likelihood

A **dynamic programming** algorithm with complexity  $O(N^2 T)$

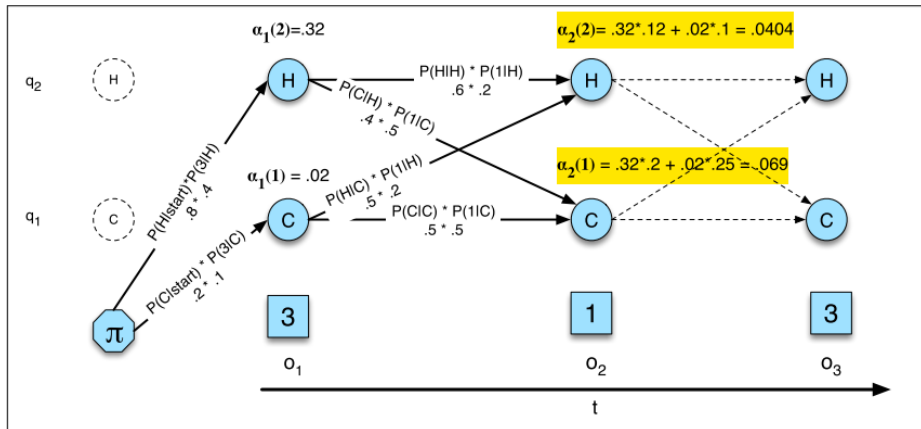


Figure: The forward trellis for computing the total observation likelihood for the ice-cream events 3 1 3.

# Forward Algorithm for Likelihood cont...

$\alpha_{t-1}(i)$		the forward path probability from the previous time step
$a_{ij}$		the transition probability from previous state $q_i$ to current state $q_j$
$b_j(o_t)$		likelihood of the observation symbol $o_t$ given the current state $j$

## Initialization:

$$\alpha_1(j) = \pi_j b_j(o_1) 1 \leq j \leq N$$

## Recursion:

$$\alpha_t(j) = \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} b_j(o_t); 1 \leq j \leq N, 1 \leq t \leq T$$

## Termination:

$$P(O|\lambda) = \sum_{i=1}^N \alpha_T(i)$$

## HMM Problem 2: Decoding

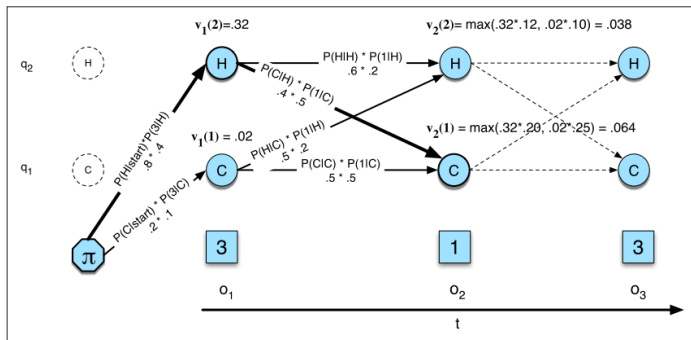
Given an input HMM  $\lambda = (A, B)$  and a sequence of observations  $O = o_1, o_2, \dots, o_T$ , find the most probable sequence of states  $Q = q_1 q_2 \dots q_T$ .

Naive solution: Compute likelihood of the observations for every possible hidden state sequence. Then choose the hidden state sequence with the maximum observation likelihood.

**Exponentially large number of state sequences!**

# Viterbi Algorithm for Decoding

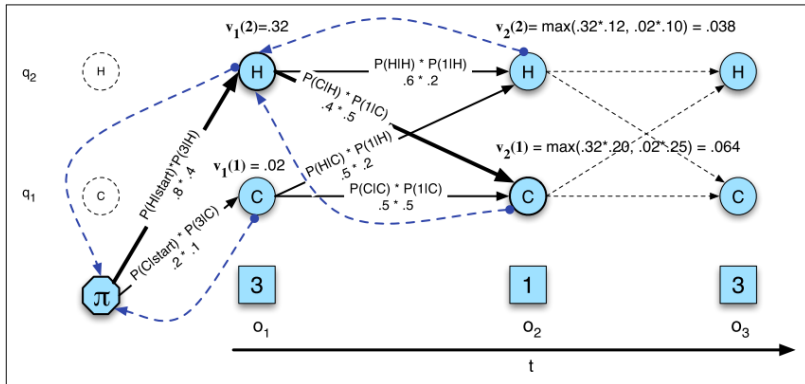
- process the observation sequence left to right, filling out the trellis
- each cell,  $v_t(j)$ , represents the probability that the HMM is in state  $j$  after seeing the first  $t$  observations and passing through the most probable state sequence  $q_1, \dots, q_{t-1}$



**Figure:** The Viterbi trellis for computing the best path through the hidden state space for the ice-cream eating events 3 1 3.

# Viterbi Algorithm for Decoding cont. . .

The Viterbi algorithm computes the best state sequence by keeping track of the path of the hidden states that led to each state - **Viterbi backtracing**



# Viterbi Algorithm for Decoding cont. . .

```
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                                ; initialization step
    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                            ; recursion step
    for each state  $s$  from 1 to  $N$  do
        viterbi[ $s, t$ ]  $\leftarrow \max_{s'=1}^N \text{viterbi}[s', t-1] * a_{s', s} * b_s(o_t)$ 
        backpointer[ $s, t$ ]  $\leftarrow \operatorname{argmax}_{s'=1}^N \text{viterbi}[s', t-1] * a_{s', s} * b_s(o_t)$ 
bestpathprob  $\leftarrow \max_{s=1}^N \text{viterbi}[s, T]$                         ; termination step
bestpathpointer  $\leftarrow \operatorname{argmax}_{s=1}^N \text{viterbi}[s, T]$                 ; termination step
bestpath  $\leftarrow$  the path starting at state bestpathpointer, that follows backpointer[] to states back in time
return bestpath, bestpathprob
```