Natural Language Processing

DSECL ZG565

Session 3: Part-of-Speech Tagging

Dr. Ashish Kulkarni

BITS Pilani

April 24, 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	*and, but, or*	PDT	predeterminer	*all, both*	VBP	verb non-3sg present	*eat*
CD	cardinal number	*one, two*	POS	possessive ending	*'s*	VBZ	verb 3sg pres	*eats*
DT	determiner	*a, the*	PRP	personal pronoun	*I, you, he*	WDT	wh-determ.	*which, that*
EX	existential 'there'	*there*	PRP$	possess. pronoun	*your, one's*	WP	wh-pronoun	*what, who*
FW	foreign word	*mea culpa*	RB	adverb	*quickly*	WP$	wh-possess.	*whose*
IN	preposition/ subordin-conj	*of, in, by*	RBR	comparative adverb	*faster*	WRB	wh-adverb	*how, where*
JJ	adjective	*yellow*	RBS	superlatv. adverb	*fastest*	$	dollar sign	$
JJR	comparative adj	*bigger*	RP	particle	*up, off*	#	pound sign	#
JJS	superlative adj	*wildest*	SYM	symbol	*+,%, &*	“	left quote	‘ or “
LS	list item marker	*1, 2, One*	TO	“to”	*to*	”	right quote	’ or ”
MD	modal	*can, should*	UH	interjection	*ah, oops*	(	left paren	[, (, {, <
NN	sing or mass noun	*llama*	VB	verb base form	*eat*	)	right paren	],), }, >
NNS	noun, plural	*llamas*	VBD	verb past tense	*ate*	,	comma	,
NNP	proper noun, sing.	*IBM*	VBG	verb gerund	*eating*	.	sent-end punc	. ! ?
NNPS	proper noun, plu.	*Carolinas*	VBN	verb past part.	*eaten*	:	sent-mid punc	: ; ... – -

POS Labeled Datasets

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

Brown corpus	million words of samples from 500 written texts from different genres published in the United States in 1961
WSJ corpus	a million words published in the Wall Street Journal in 1989
Switchboard corpus	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.*

		WSJ		Brown	
Types:					
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%**)

book that flight *v.s.* hand me that *book*

Does *that* flight serve dinner *v.s.* I thought *that* your flight was earlier

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 $q_1, q_2, \ldots, q_i$,
Markov assumption:
$P(q_i = a | q_1 \ldots q_{i-1}) = P(q_i = a | q_{i-1})$

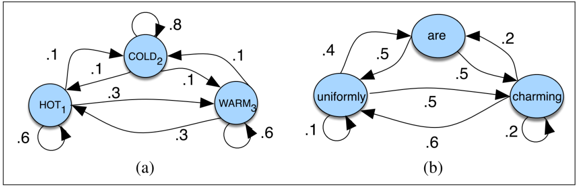


Figure: A Markov chain for weather (a) and one for words (b); $\pi = [0.1, 0.7, 0.2]$ for (a)

$Q = q_1 q_2 \ldots q_N$	a set of N states
$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{mn}$	a transition probability matrix A, each a_{ij} is the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \forall i$
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an initial probability distribution over states. π_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

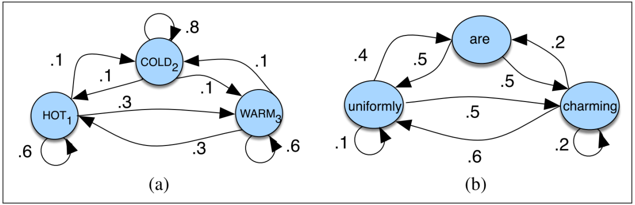


Figure: A Markov chain for weather (a) and one for words (b); $\pi = [0.1, 0.7, 0.2]$ for (a)

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
$\pi = \pi_i, \pi_2, \dots, \pi_N$	an initial probability distribution over states; π_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;

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

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;

$$P(o_i|q_1 \ldots q_i \ldots q_T, o_1 \ldots o_i \ldots o_T) = P(o_i|q_i) \tag{2}$$

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.

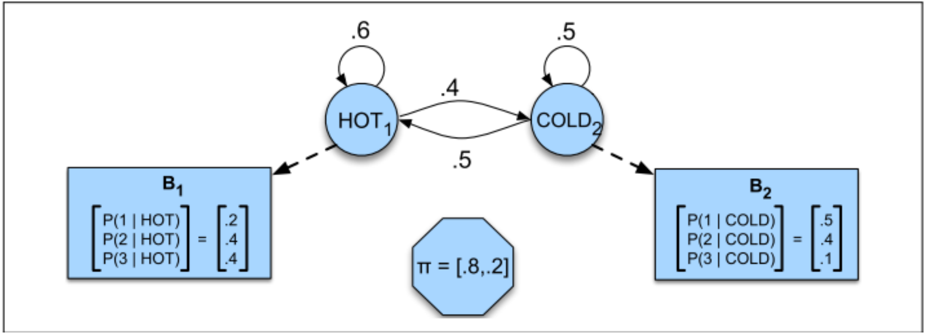


Figure: A hidden Markov model for relating numbers of ice creams eaten by Jason (the observations) to the weather (H or C, the hidden variables).

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:

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

But we don't know the hidden state sequence!

HMM Problem 1: Likelihood cont...

Apply the rule of **marginal probability**

$$P(O) = \sum_{Q} P(O, Q) = \sum_{Q} P(O|Q)P(Q) \tag{4}$$

where, from the HMM assumptions, we have:

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

For our running example:

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

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 $O(N^2T)$

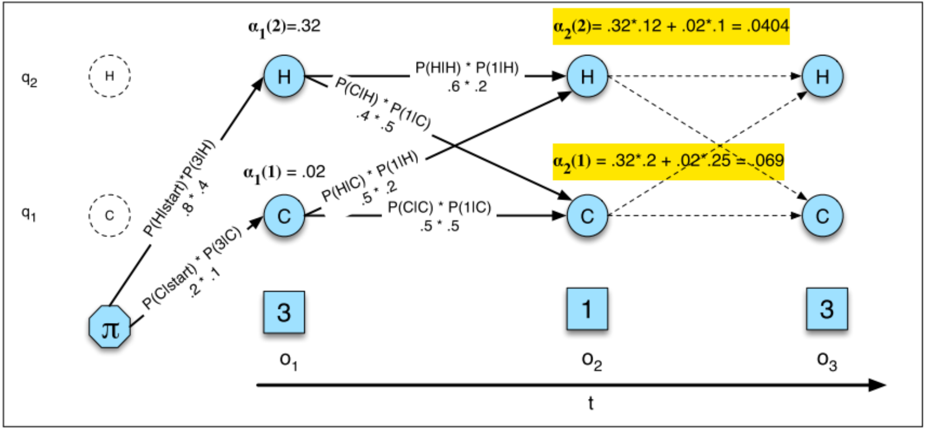


Figure: The forward trellis for computing the total observation likelihood for the ice-cream events 3 1 3.

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 $q_1, \ldots, q_{t-1}$

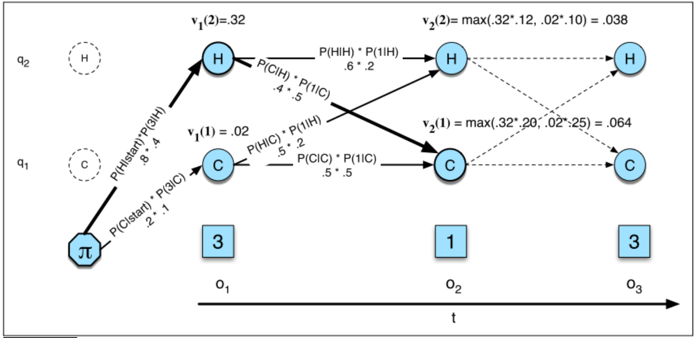
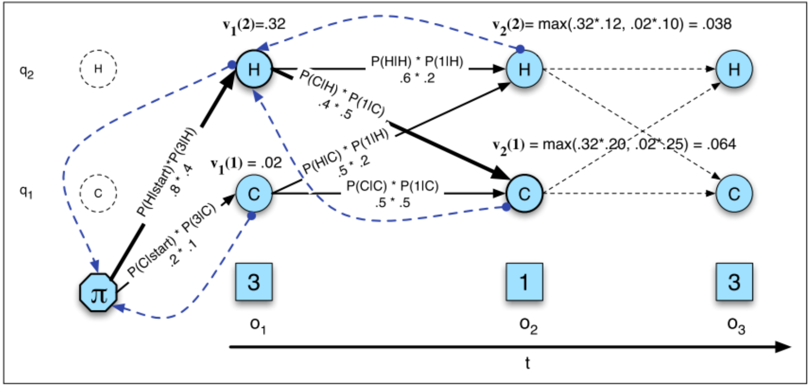


Figure: The Viterbi trellis for computing the best path through the hidden state space for the ice-cream eating events 3 1 3.

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

HMM Problem 3: Training

Given an observation sequence O and the set of possible states in the HMM, learn the HMM parameters A and B.

Problem of Unknown Words in HMM

Unknown words often arise owing to proper nouns, acronyms, new common nouns and verbs that might enter a language. These pose a challenge in HMM-based POS taggers.

Why?

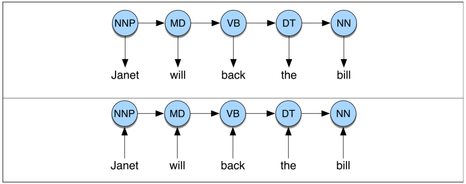
For unknown words, $P(o_i|q_i) = 0$!

Thus, to achieve high accuracy with POS taggers, it is also important to have a good model for dealing with unknown words.

Although, there are workarounds, there is no elegant way to achieve this in a generative model like HMM

Maximum Entropy Markov Model (MEMM)

MEMM is a **discriminative model**. It directly models $P(T|W)$.
Let the sequence of words be $W = w_i^n$ and the sequence of tags $T = t_1^n$



HMM

$$\hat{T} = \arg\max_T P(T|W) = \arg\max_T P(W|T)P(T) = \arg\max_T \prod_i P(w_i|t_i) \prod_i P(t_i|t_{i-1}) \quad (6)$$

MEMM

$$\hat{T} = \arg\max_T P(T|W) = \arg\max_T \prod_i P(t_i|w_i, t_{i-1}) \quad (7)$$

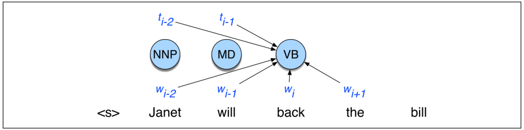
Features in a MEMM

It is much more easier to incorporate features in a discriminative sequence model like MEMM.

We can use **feature templates** like:

$\langle t_i, w_{i-2}\rangle, \langle t_i, w_{i-1}\rangle, \langle t_i, w_i\rangle, \langle t_i, w_{i+1}\rangle, \langle t_i, w_{i+2}\rangle$
$\langle t_i, t_{i-1}\rangle, \langle t_i, t_{i-2}, t_{i-1}\rangle,$
$\langle t_i, t_{i-1}, w_i\rangle, \langle t_i, w_{i-1}, w_i\rangle, \langle t_i, w_i, w_{i+1}\rangle$

w_i contains a particular prefix (from all prefixes of length ≤ 4)
w_i contains a particular suffix (from all suffixes of length ≤ 4)
w_i contains a number
w_i contains an upper-case letter
w_i contains a hyphen
w_i is all upper case
w_i's word shape
w_i's short word shape
w_i is upper case and has a digit and a dash (like *CFC-12*)
w_i is upper case and followed within 3 words by Co., Inc., etc.



We can also add **features to deal with unknown words**

Word shape features are used by mapping lower-case letters to 'x', upper-case letters to 'X', numbers to 'd' and retaining punctuation. *e.g.* DC10-30 maps to XXdd-dd.

Decoding MEMMs

The most likely sequence of tags is then computed by combining these features as:

$$
\begin{aligned}
\hat{T} &= \arg\max_{T} P(T|W) \\
&= \arg\max_{T} \prod_{i} P(t_i | w_{i-l}^{i+l}, t_{i-k}^{i-1}) \\
&= \arg\max_{T} \prod_{i} \frac{exp\left(\sum_j \theta_j f_j(t_i, w_{i-l}^{i+l}, t_{i-k}^{i-1})\right)}{\sum_{t' \in tagset} exp\left(\sum_j \theta_j f_j(t', w_{i-l}^{i+l}, t_{i-k}^{i-1})\right)}
\end{aligned}
\tag{8}
$$

here, $f(\cdot)$ are the feature functions and θ are the corresponding feature weights.

How should we decode to find this optimal tag sequence $\hat{T}$?

Decoding MEMM cont...

Greedy sequence decoding:

- greedily choose the best tag for each word;
- very fast;
- can't use evidence from future decisions leading to low performance.

function GREEDY SEQUENCE DECODING(words W, model P) **returns** tag sequence T

for $i = 1$ **to** *length(W)*
$\hat{t}_i = \underset{t' \in T}{\operatorname{argmax}} P(t' \mid w_{i-l}^{i+l}, t_{i-k}^{i-1})$

Viterbi decoding: finds the sequence of POS tags that is optimal for the whole sentence. Similar to HMM, except that, the recursive step of the Viterbi equation takes the form:

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) P(s_j | s_i, o_t) \quad 1 \le j \le N, 1 < t \le T \tag{9}$$

Conditional Random Field (CRF)

Both HMM and MEMM models cannot directly use the information from future tags. They also suffer from **label bias** or **observation bias**.

$$will/NN\ to/TO\ fight/VB$$

$P(t_{will}|\langle s \rangle)$ prefers the 'modal' tag MD, and, because $P(TO|to, t_{will}) \approx 1$ regardless of t_{will}, the model cannot make use of the transition probability and incorrectly chooses MD.

Conditional Random Field (CRF) is a more powerful model that implements ‘bidirectionality’ by modeling the problem as an **undirected graphical model**.