19CSE453 – Natural Language Processing

Parts of Speech tagging

By

Ms. Kavitha C.R.

Dept. of Computer Science and Engineering

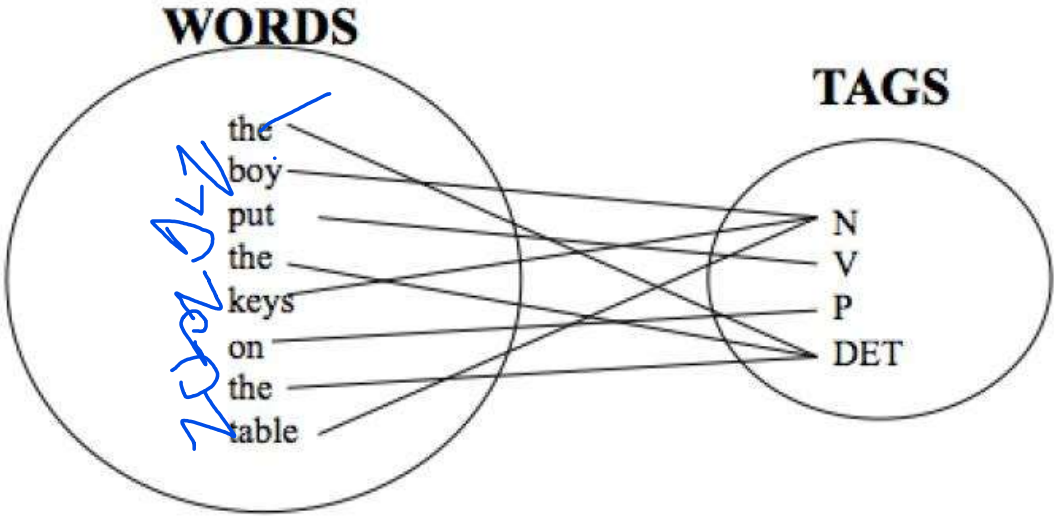
Amrita School of Computing, Bengaluru

Amrita Vishwa Vidyapeetham

Part-of-Speech (POS) tagging

Task

Given a text of English, identify the parts of speech of each word



Parts of Speech: How many?

Open class words (content words)

- nouns, verbs, adjectives, adverbs
- mostly content-bearing: they refer to objects, actions, and features in the world
- *open class*, since new words are added all the time

Closed class words

- pronouns, determiners, prepositions, connectives, ...
- there is a limited number of these
- *mostly functional:* to tie the concepts of a sentence together

POS Examples

N	noun	chair, bandwidth, pacing
V	verb	study, debate, munch
ADJ	adj	purple, tall, ridiculous
ADV	adverb	unfortunately, slowly,
P	preposition	of, by, to
PRO	pronoun	I, me, mine
DET	determiner	the, a, that, those

POS Tagging: Choosing a tag set

- To do POS tagging, a standard set needs to be chosen
- Could pick very coarse tagsets
  N, V, Adj, Adv
- More commonly used set is finer grained, "UPenn TreeBank tagset", 45 tags

A Nice Tutorial on POS tags

https://sites.google.com/site/partofspeechhelp/

UPenn Treebank POS Tag set

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	"to"	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential 'there'	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	*$*
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	#
PDT	Predeterminer	*all, both*	"	Left quote	(' or ")
POS	Possessive ending	*'s*	"	Right quote	(' or ")
PRP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PRP$	Possessive pronoun	*your, one's*	)	Right parenthesis	( ],), }, >)
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	(: ; ... – -)
RP	Particle	*up, off*			

Using the UPenn tag set

Example Sentence

The grand jury commented on a number of other topics.

POS tagged sentence

The/DT grand/JJ jury/NN commmented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

DT–Determiner, JJ-Adjective, VBD-verb past, IN-preposition, NN-noun singular or mass, NNS-plural

Why is POS tagging hard?

Words often have more than one POS: back

- The back door: *back/JJ* adjective
- On my back: *back/NN* noun in singular
- Win the voters back: *back/RB* adverb
- Promised to back the bill: *back/VB* Verb base form

POS tagging problem

To determine the POS tag for a particular instance of a word

Ambiguous word types in the Brown Corpus

Ambiguity in the Brown corpus

- 40% of word tokens are ambiguous
- 12% of word types are ambiguous
- Breakdown of ambiguous word types:

Unambiguous (1 tag)	35,340
Ambiguous (2–7 tags)	4,100
2 tags	3,760
3 tags	264
4 tags	61
5 tags	12
6 tags	2
7 tags	1 ("still")

How bad is the ambiguity problem?

- One tag is usually more likely than the others.
  In the Brown corpus, *race* is a noun 98% of the time, and a verb 2% of the time
- A tagger for English that simply chooses the most likely tag for each word can achieve good performance
- Any new approach should be compared against the unigram baseline (assigning each token to its most likely tag)

Deciding the correct POS

Can be difficult even for people

- Mrs./NNP Shaefer/NNP never/RB got/VBD around/_ to/TO joining/VBG.
- All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/_ the/DT corner/NN.
- Chateau/NNP Petrus/NNP costs/VBZ around/_ 2500/CD.

Can be difficult even for people

- Mrs./NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG.
- All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN.
- Chateau/NNP Petrus/NNP costs/VBZ around/RB 2500/CD.

NNP-proper noun singular
PRP-personal pronoun
RB-adverb
TO-to
VBD-verb past
VBG-verb gerund or present participle
VBN-verb past participle
VBZ-verb 3 sing present
CD-cardinal no
NN-noun singular or mass

RP – Particle
VB – verb base form
IN - preposition

Relevant Knowledge for POS tagging

The word itself

- Some words may only be nouns, e.g. *arrow*
- Some words are ambiguous, e.g. *like, flies*
- Probabilities may help, if one tag is more likely than another

Local context

- Two determiners rarely follow each other
- Two base form verbs rarely follow each other
- Determiner is almost always followed by adjective or noun

POS Tagging: Two Approaches

Rule-based Approach

- Assign each word in the input a list of potential POS tags
- Then winnow down this list to a single tag using hand-written rules

Statistical tagging

- Get a training corpus of tagged text, learn the transformation rules from the most frequent tags (TBL tagger) **Transformation Based Taggers**
- Probabilistic: Find the most likely sequence of tags T for a sequence of words W

TBL Tagger - Transformation Based Taggers

Label the training set with most frequent tags

- The can was rusted.
- The/DT can/MD was/VBD rusted/VBD.

MD-Modal
VBD- verb past

Add transformation rules to reduce training mistakes

- MD →NN: DT_
- VBD→VBN: VBD_

NN – noun in singular
VBN- verb past particle

TBL (Transformation-Based Learning) taggers, also known as Brill taggers, are a type of part-of-speech (POS) taggers in natural language processing (NLP). They are based on a machine learning algorithm that learns to assign POS tags to words in a sentence by iteratively applying a set of transformation rules.

The process of building a TBL tagger typically involves the following steps:

Initial Tagging: Each word in a training corpus is initially assigned a default tag based on some simple heuristic or baseline model.

Error Identification: The TBL tagger compares the initial tags with the correct tags in the training data and identifies the words that are incorrectly tagged.

Rule Generation: Transformation rules are generated to correct the tagging errors. These rules are usually in the form of "if context X is observed, change the tag of word Y to tag Z."

Rule Application: The generated rules are applied to the training data, and the tags of the words are updated accordingly.

Iterative Process: Steps 2-4 are repeated iteratively until the tagging accuracy reaches a satisfactory level or a predefined stopping criterion is met.

Tagging New Sentences: Once the TBL tagger is trained, it can be used to assign POS tags to new, unseen sentences by applying the learned transformation rules.

TBL taggers are known for their simplicity and effectiveness, as they learn from the data itself rather than relying on explicit linguistic knowledge or complex statistical models. However, they can be sensitive to the quality and representativeness of the training data and may require careful tuning and evaluation to achieve optimal performance.

Probabilistic Tagging: Two different families of models

Problem at hand

We have some data $\{(d, c)\}$ of paired observations d and hidden classes c.

Different instances of d and c

- **Part-of-Speech Tagging**: words are observed and tags are hidden.
- **Text Classification**: sentences/documents are observed and the category is hidden.
  Categories can be positive/negative for sentiments ..
  sports/politics/business for documents ...

What gives rise to the two families?

Whether they generate the observed data from hidden stuff or the hidden structure given the data?

Generative vs Conditional Models

Generative (Joint) Models

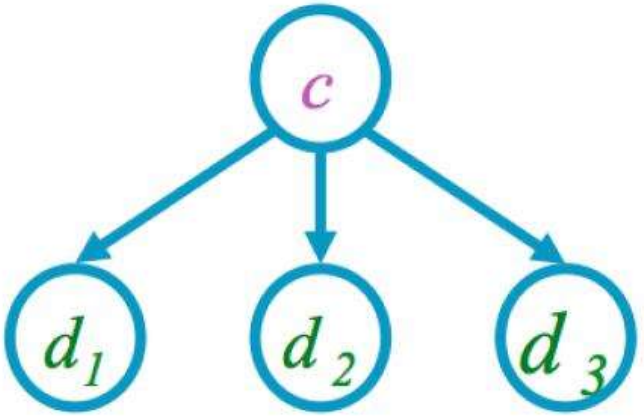
Generate the observed data from hidden stuff, i.e. put a probability over the observations given the class: $P(d, c)$ in terms of $P(d|c)$
e.g. Naïve Bayes' classifiers, Hidden Markov Models etc.

Discriminative (Conditional) Models

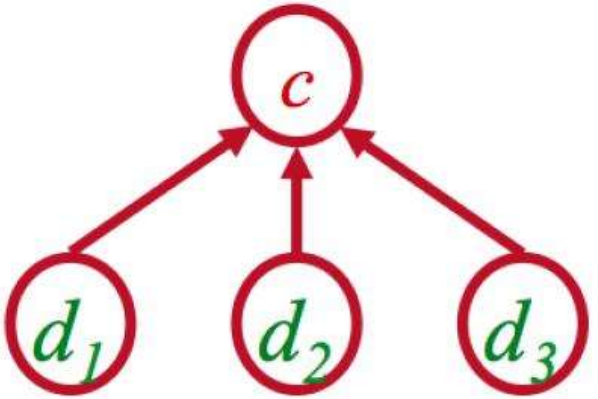
Take the data as given, and put a probability over hidden structure given the data: $P(c|d)$
e.g. Logistic regression, maximum entropy models, conditional random fields

SVMs, perceptron, etc. are discriminative classifiers but not directly probabilistic

Generative vs Discriminative Models



Naive Bayes



Logistic Regression

Joint vs. conditional likelihood

- A *joint* model gives probabilities $P(d, c)$ and tries to maximize this joint likelihood.
- A *conditional* model gives probabilities $P(c|d)$, taking the data as given and modeling only the conditional probability of the class.

Probabilistic Tagging

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

Tagging: Probabilistic View (Generative Model)

Find

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

By Bayesian Inference

Further Simplification

$$\hat{T} = argmax_T \prod_i P(w_i|w_1 \ldots w_{i-1}, t_1 \ldots t_i) P(t_i|t_1 \ldots t_{i-1})$$

- The probability of a word appearing depends only on its own POS tag
  $P(w_i|w_1 \ldots w_{i-1}, t_1 \ldots t_i) \approx P(w_i|t_i)$
- Bigram assumption: the probability of a tag appearing depends only on the previous tag
  $P(t_i|t_1 \ldots t_{i-1}) \approx P(t_i|t_{i-1})$
- Using these simplifications:
  $\hat{T} = argmax_T \prod_i P(w_i|t_i) P(t_i|t_{i-1})$

Computing The Probability Values

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

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

NN- noun singular

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

Word Likelihood probabilities $p(w_i|t_i)$

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

VBZ-verb 3 sing present

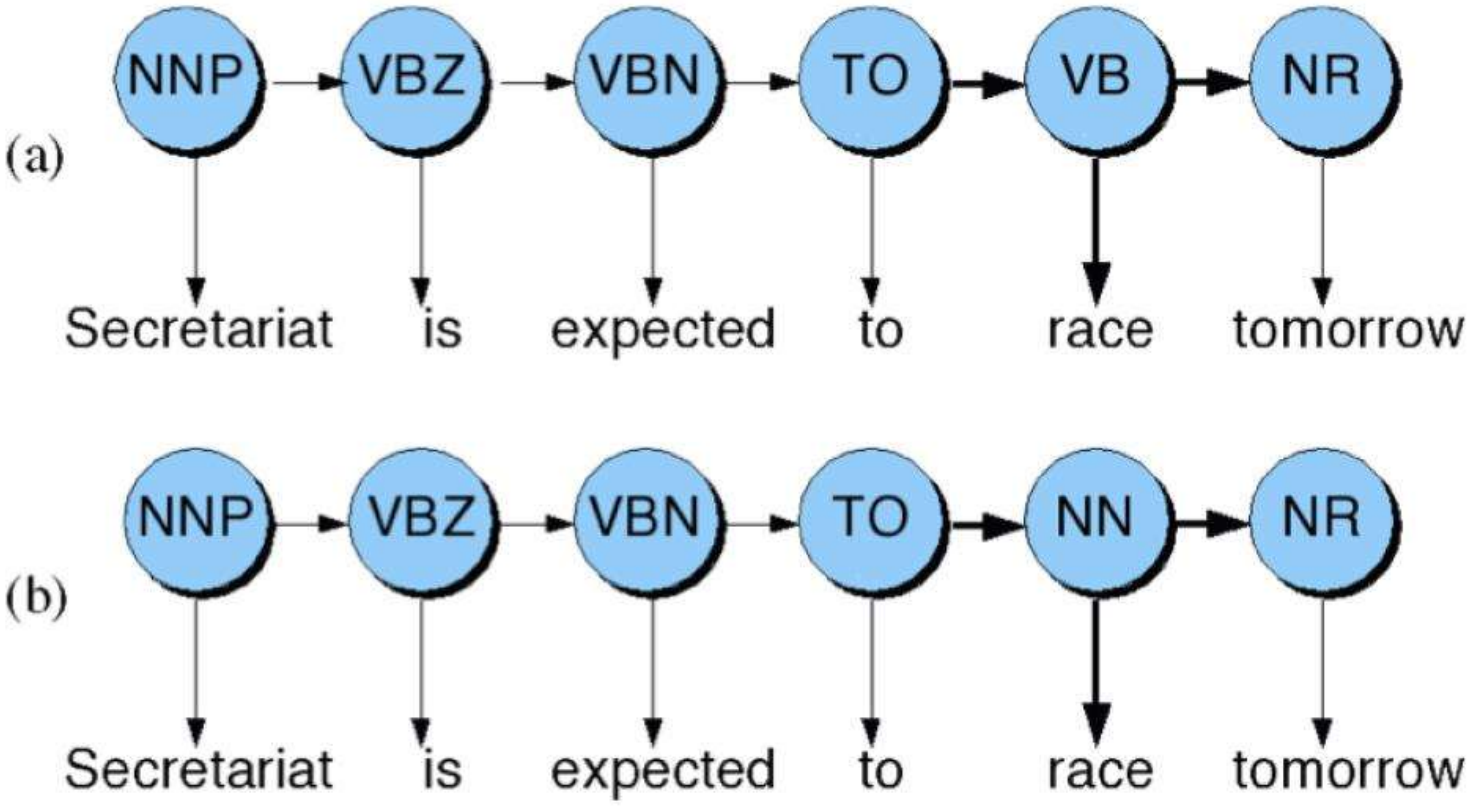
$$P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = 0.47$$

Disambiguating “race”

Noun:

The horse race at the Kentucky Derby was highly anticipated.
She won the swimming race and set a new record.
The marathon is a grueling endurance race.
The participants lined up at the starting line for the bicycle race.
The Olympic Games feature a variety of track and field races.
Verb:

They raced each other to the finish line.
The cars raced down the highway, exceeding the speed limit.
The children raced to see who could reach the playground first.
She raced against time to complete the project before the deadline.
The dog raced after the ball, eager to retrieve it.
In the noun form, "race" refers to a competition or contest involving speed, usually between individuals or teams. As a verb, "race" means to compete or move



Disambiguating "race"

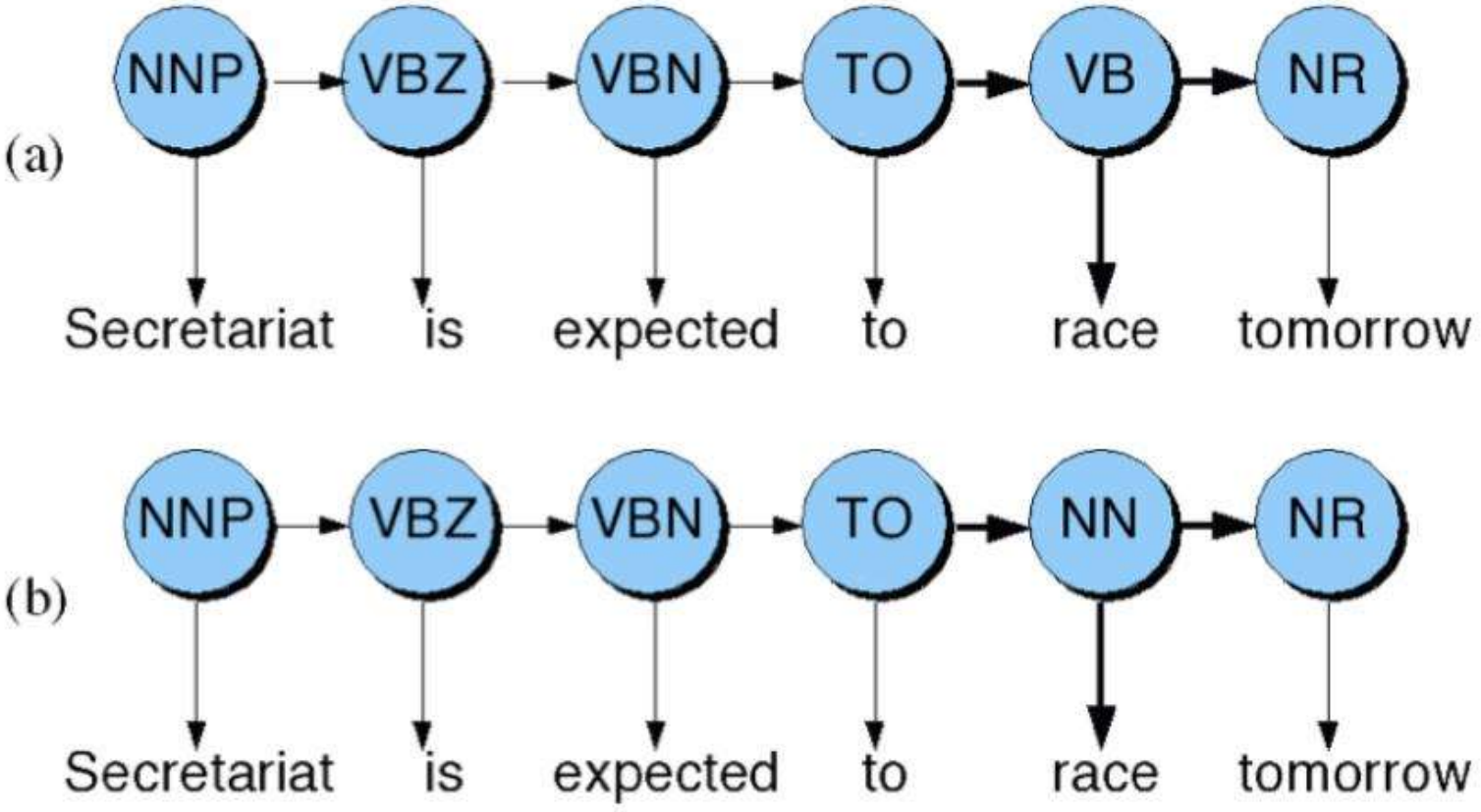
Difference in probability due to

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

After computing the probabilities

- $P(NN|TO)P(NR|NN)P(race|NN) = 0.0047 \times 0.0012 \times 0.00057 = 0.00000000032$
- $P(VB|TO)P(NR|VB)P(race|VB) = 0.83 \times 0.0027 \times 0.00012 = 0.00000027$

What is this model?



This is a Hidden Markov Model

Hidden Markov Model (HMM)

- 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*
- q_n: variable denoting the weather on the n^{th} day
- We want to find the following conditional probabilities:

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

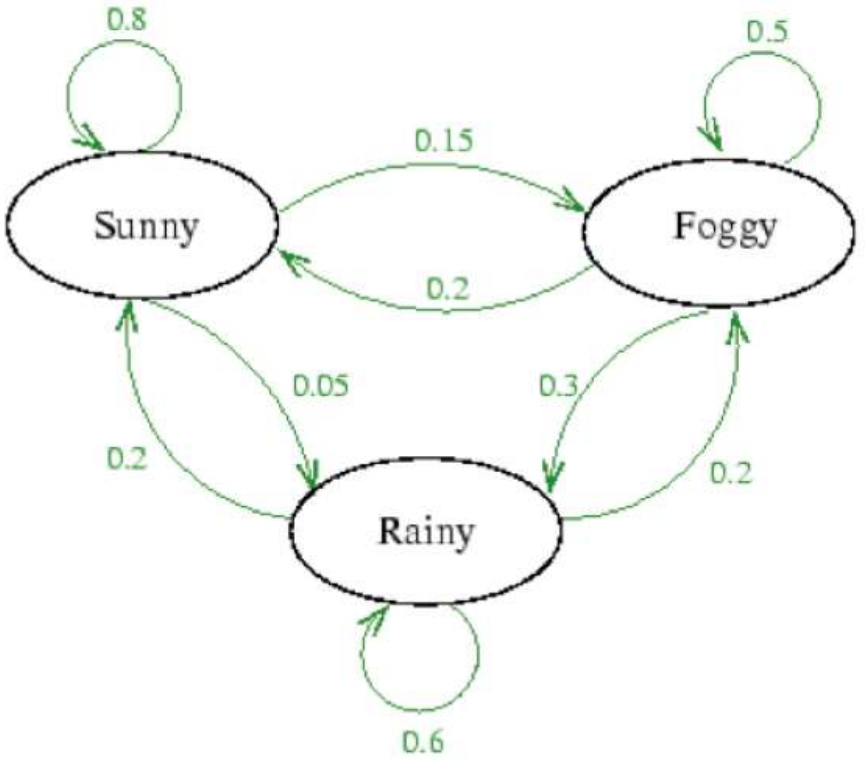
First-order Markov Assumption

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

Markov Chain Transition Table

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

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



Using Markov Chain

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

$$P(q_2 = sunny, q_3 = rainy | q_1 = sunny)$$

$$\begin{aligned} &= P(q_3 = rainy | q_2 = sunny, q_1 = sunny) \times P(q_2 = sunny | q_1 = sunny) \\ &= P(q_3 = rainy | q_2 = sunny) \times P(q_2 = sunny | q_1 = sunny) \\ &= 0.05 \times 0.8 \\ &= 0.04 \end{aligned}$$

Hidden Markov Model (HMM)

- The next level of complexity that can be introduced into a stochastic tagger combines the previous two approaches, using both tag sequence probabilities and word frequency measurements. This is known as the **Hidden Markov Model (HMM)**.
- Before proceeding with what is a **Hidden** Markov Model, let us first look at what is a Markov Model. That will better help understand the meaning of the term **Hidden** in HMMs

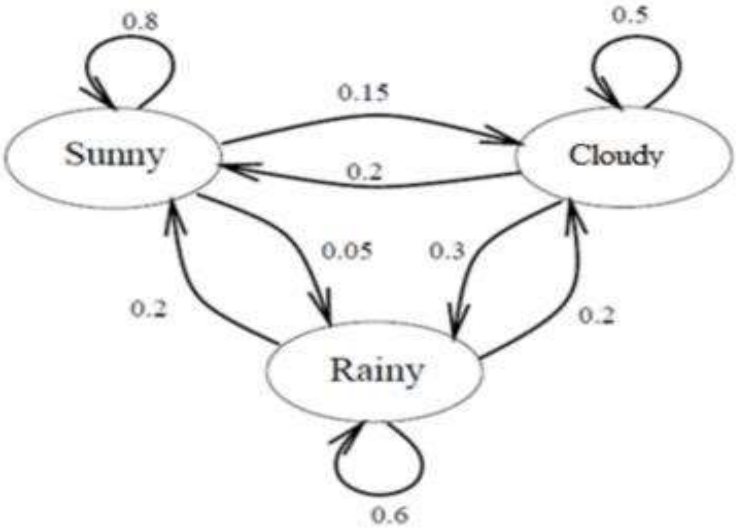
Example

Say that there are only three kinds of weather conditions, namely

- Rainy
- Sunny
- Cloudy
- How to make a prediction of the weather for today based on what the weather has been for the past N days?
- Say we have a sequence. Something like this:
- Sunny, Rainy, Cloudy, Cloudy, Sunny, Sunny, Sunny, Rainy
- So, the weather for any given day can be in any of the three states

Example

- Let's say we decide to use a Markov Chain Model to solve this problem. Now using the data that we have, we can construct the following state diagram with the labelled probabilities.



$$P(Sunny|Sunny) = 0.8$$
$$P(Rainy|Sunny) = 0.05$$
$$P(Cloudy|Sunny) = 0.15$$
$$\Big\} 1$$

$$P(Sunny|Rainy) = 0.2$$
$$P(Rainy|Rainy) = 0.6$$
$$P(Cloudyy|Rainy) = 0.2$$
$$\Big\} 1$$

$$P(Sunny|Cloudy) = 0.2$$
$$P(Rainy|Cloudy) = 0.3$$
$$P(Cloudy|Cloudy) = 0.5$$
$$\Big\} 1$$

In order to compute the probability of today's weather given N previous observations, we will use the Markov Property.

$$P(q_1, \ldots, q_n) = \prod_{i=1}^{n} P(q_i|q_{i-1})$$

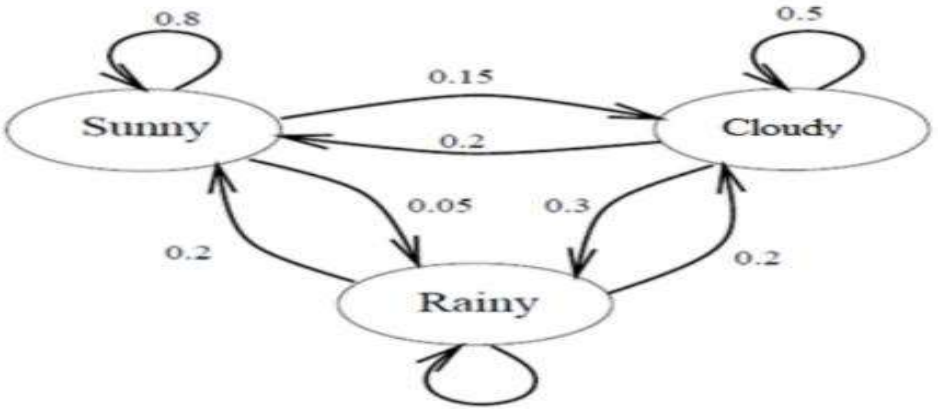
Markov Chain is essentially the simplest known Markov model, that it obeys the Markov property.

The Markov property suggests that the distribution for a random variable in the future depends solely only on its distribution in the current state, and none of the previous states have any impact on the future states.

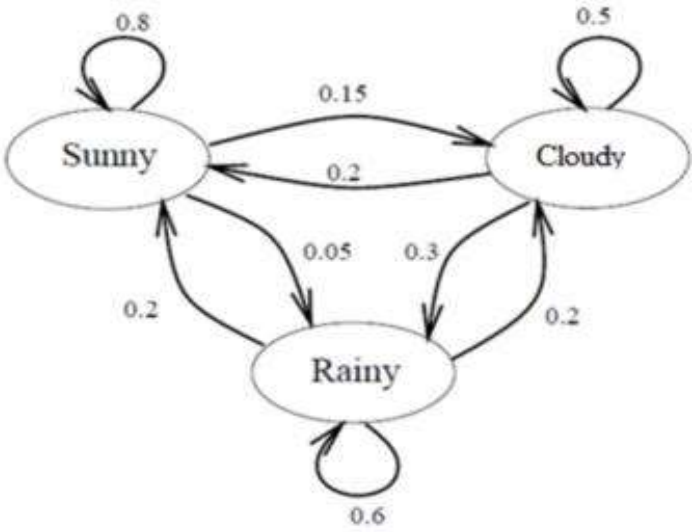
- Exercise 1: Given that today is Sunny, what's the probability that tomorrow is Sunny and the next day Rainy?

$$P(q_2, q_3|q_1) = P(q_2|q_1)P(q_3|q_1, q_2)$$

$$= P(q_2|q_1)\ P(q_3|q_2)$$
$$= P(Sunny|Sunny)\ P(Rainy|Sunny)$$
$$= (0.8)(0.05)$$
$$= 0.04$$



Exercise 2: Assume that yesterday's weather was Rainy, and today is Cloudy, what is the probability that tomorrow will be Sunny?



$$P(q_3|q_1, q_2) = P(q_3|q_2)$$

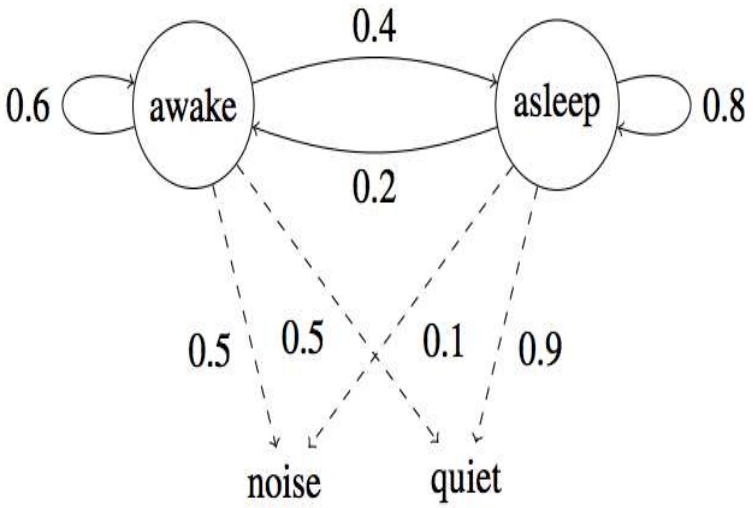
$$= P(Sunny|Cloudy)$$

$$= 0.2$$

Hidden Markov Model

There are two kinds of probabilities from the state diagram.

- One is the emission probabilities, which represent the probabilities of making certain observations given a particular state.
  For example, we have P(noise | awake) = 0.5 .
  This is an emission probability.
- The other ones is transition probabilities, which represent the probability of transitioning to another state given a particular state.
  For example, we have P(asleep | awake) = 0.4 .
  This is a transition probability.



- We usually observe longer stretches of the child being awake and being asleep.

If Peter is awake now, the probability of him staying awake is higher than of him going to sleep. Hence, the 0.6 and 0.4 in the diagram. P (awake | awake) = 0.6 and P (asleep | awake) = 0.4

- The Transition probabilities matrix

	awake	asleep
awake	0.6	0.4
asleep	0.2	0.8

- The Emission probabilities matrix

	noise	quiet
awake	0.5	0.5
asleep	0.1	0.9

- Before actually trying to solve the problem at hand using HMMs, let's relate this model to the task of Part of Speech Tagging

HMMs for Part of Speech Tagging

- We know that to model any problem using a Hidden Markov Model we need a set of observations and a set of possible states. The states in an HMM are hidden.
- In the parts of speech tagging problem, the **observations** are the words themselves in the given sequence.
- As for the **states**, which are hidden, these would be the POS tags for the words.
- The **transition probabilities** would be somewhat like P(VP | NP) that is, what is the probability of the current word having a tag of Verb Phrase given that the previous tag was a Noun Phrase.
- **Emission probabilities** would be P(john | NP) or P(will | VP) that is, what is the probability that the word is, say, John given that the tag is a Noun Phrase.
- Note that this is just an informal modeling of the problem to provide a very basic understanding of how the Parts of Speech tagging problem can be modeled using an HMM

Why do we need HMM for POS Tagger?

- If you notice closely, we can have the **words in a sentence as Observable States (given to us in the data) but their POS Tags as Hidden states and hence we use HMM for estimating POS tags**.

Note: we call **Observable states as 'Observation' & Hidden states as 'States'.**

- A Hidden Markov Model has the following components:

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, \ldots, v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state q_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. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

HMM components example

- **Q**: Set of possible Tags
- **A:** The *A* matrix contains the tag transition probabilities $P(t_i|t_{i-1})$ which represent the probability of a tag occurring given the previous tag.

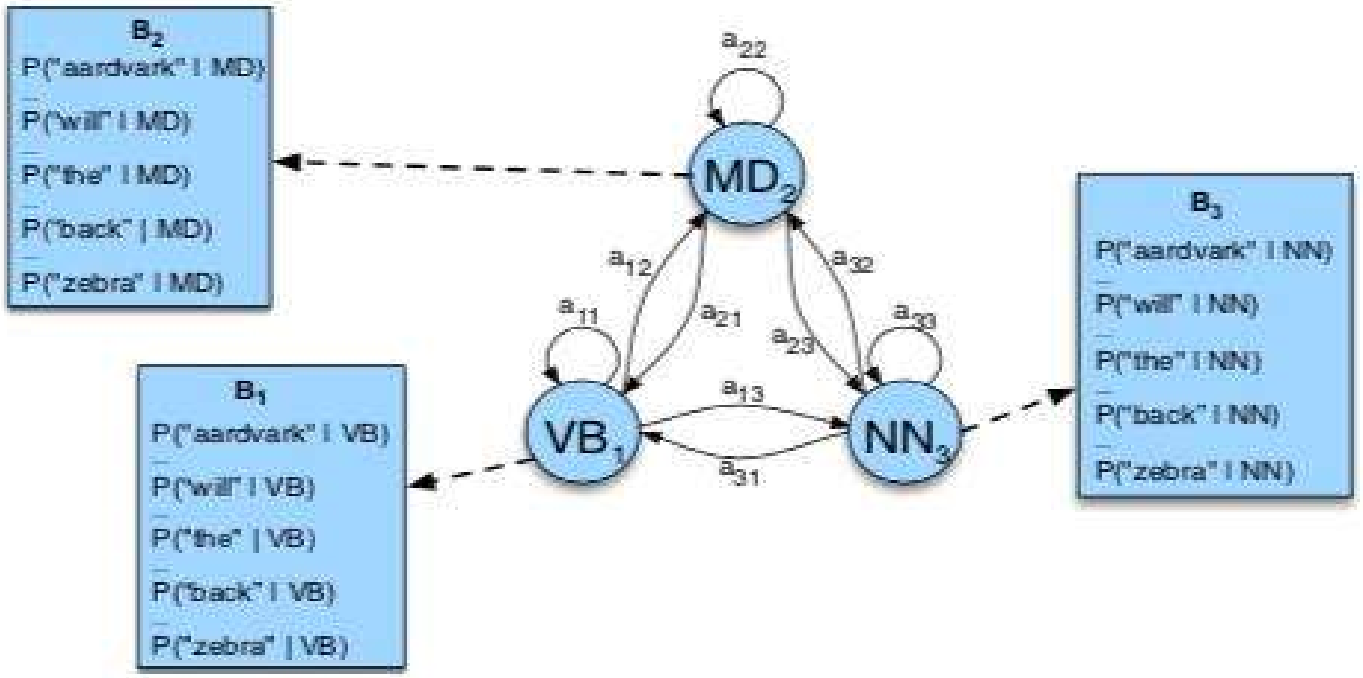
Example: Calculating A[Verb][Noun]:

- *P (Noun|Verb): Count(Verb, Noun)/Count(Verb)*
- **O**: Sequence of observation (words in the sentence)
- **B:** The *B* emission probabilities, $P(w_i|t_i)$, represent the probability, given a tag (say Verb), that it will be associated with a given word (say Playing).

The emission probability B[Verb][Playing] is calculated using:

- P(Playing | Verb): Count (Verb, Playing) / Count (Verb)
- It must be noted that we get all these Count() from the corpus itself used for training.

A sample HMM with both ‘A’ & ‘B’ matrix will look like this :



MD–Modal can, should
VB- verb base form
NN-Noun singular, mass

Here, the black, continuous arrows represent values of Transition matrix ‘A’ while the dotted black arrow represents Emission Matrix ‘B’ for a system with Q: {MD, VB, NN}.

Decoding using HMM

- Given an input as HMM (Transition Matrix, Emission Matrix) and a sequence of observations $O = o1, o2, \ldots, oT$ *(Words in sentences of a corpus)*, find the most probable sequence of states $Q = q1q2q3 \ldots qT$ *(POS Tags in our case)*

The 2 major assumptions followed while decoding tag sequence using HMMs:

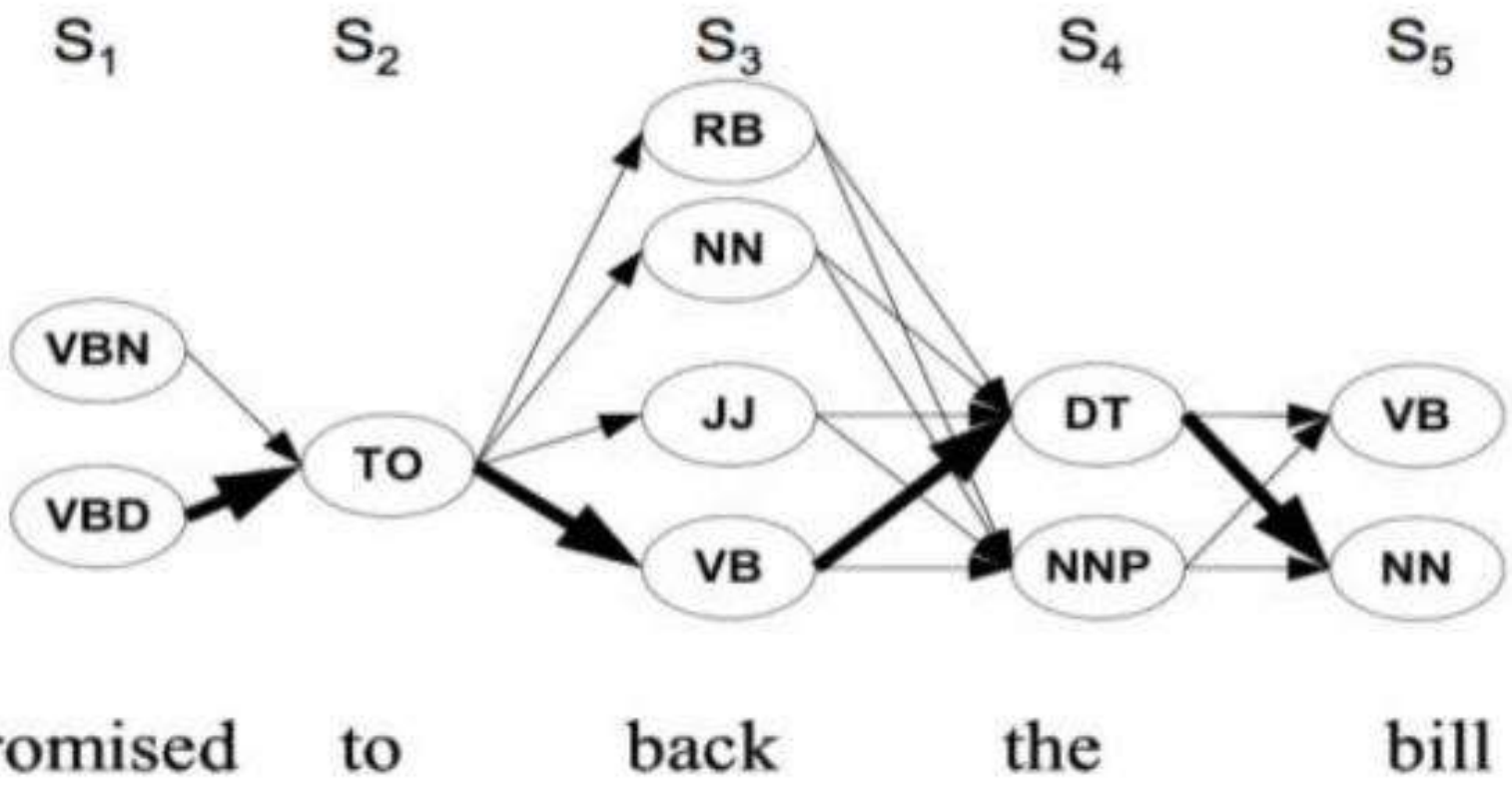
- The probability of a word appearing depends only on its **own tag** and is independent of neighboring words and tags.
- The probability of a tag depends only on the **previous tag(bigram HMM)** occurred rather than the entire previous tag sequence i.e. shows Markov Property. Though we can be flexible with this

- So the problem is to find out what is the sequence of that would be used for this sentence.
- approach to this problem, given the state transition graph that already exists
- To simplify , find out what are all the possible part of speech text a word can take.
- Already discussed, that some words can take multiple part of speech tags, but none of the words went beyond 7 part of speech tags. And mostly the words were having if they were ambiguous they were having 2 or 3 part of space tags.
- 1st of all, need to set up a **probability matrix called lattice** where we have **columns as our observables** (words of a sentence in the same sequence as in sentence) & **rows as hidden states**(all possible POS Tags are known).

Walking through the states: best path

Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*
CD	Cardinal number	*one, two, three*
DT	Determiner	*a, the*
EX	Existential 'there'	*there*
FW	Foreign word	*mea culpa*
IN	Preposition/sub-conj	*of, in, by*
JJ	Adjective	*yellow*
JJR	Adj., comparative	*bigger*
JJS	Adj., superlative	*wildest*
LS	List item marker	*1, 2, One*
MD	Modal	*can, should*
NN	Noun, sing. or mass	*llama*
NNS	Noun, plural	*llamas*
NNP	Proper noun, singular	*IBM*
NNPS	Proper noun, plural	*Carolinas*
PDT	Predeterminer	*all, both*
POS	Possessive ending	*'s*
PRP	Personal pronoun	*I, you, he*
PRP$	Possessive pronoun	*your, one's*
RB	Adverb	*quickly, never*
RBR	Adverb, comparative	*faster*
RBS	Adverb, superlative	*fastest*
RP	Particle	*up, off*

Tag	Description	Example
SYM	Symbol	+,%, &
TO	"to"	*to*
UH	Interjection	*ah, oops*
VB	Verb, base form	*eat*
VBD	Verb, past tense	*ate*
VBG	Verb, gerund	*eating*
VBN	Verb, past participle	*eaten*
VBP	Verb, non-3sg pres	*eat*
VBZ	Verb, 3sg pres	*eats*
WDT	Wh-determiner	*which, that*
WP	Wh-pronoun	*what, who*
WP$	Possessive wh-	*whose*
WRB	Wh-adverb	*how, where*
$	Dollar sign	*$*
#	Pound sign	*#*
"	Left quote	(' or ")
"	Right quote	(' or ")
(	Left parenthesis	([, (, {, <)
)	Right parenthesis	( ],), }, >)
,	Comma	,
.	Sentence-final punc	(. ! ?)
:	Mid-sentence punc	(: ; ... – -)



- There are many possibilities. Now each word can take any of these possibilities.
- So now, there are 32 possibilities of the part of speech tag sequences.
- Among these , we have to find out which is the most likely sequence of participation tags that is being used.
- Enumerate all the possibilities and for each possibility compute the probability separately. So for 32 possibilities, compute 32 different probability.
- That is one possibility, but this is not very efficient, and think of some sentences which might have say 15-20 words. This will just go exponentially with the number of words.

So this is not a good solution.

- So, need to have a solution where it does not grow exponentially with the size of the sentence. So what will be a good algorithm for doing that?
- So ideally, want to come up with the actual part of speech tag sequence like here it will be VBD TO VB DT and NN from all the possibilities.

- So this is the Viterbi algorithm

Viterbi Decoding for HMM

- The decoding algorithm used for HMMs is called the Viterbi algorithm penned down by the Founder of Qualcomm, an American MNC

- The Viterbi algorithm is **a dynamic programming algorithm for obtaining the maximum a posteriori probability estimate of the most likely sequence of hidden states**—called the Viterbi path—that results in a sequence of observed events, especially in the context of Markov information sources and hidden Markov models (HMM).
- Now, how to compute the probability by using the HMM.

Finding the best path: Viterbi Algorithm

Intuition

Optimal path for each state can be recorded. We need

- Cheapest cost to state j at step s: $\delta_j(s)$
- Backtrace from that state to best predecessor $\psi_j(s)$

Computing these values

- $\delta_j(s+1) = max_{1 \leq i \leq N} \delta_i(s) p(t_j|t_i) p(w_{s+1}|t_j)$
- $\psi_j(s+1) = argmax_{1 \leq i \leq N} \delta_i(s) p(t_j|t_i) p(w_{s+1}|t_j)$

Best final state is $argmax_{1 \leq i \leq N} \delta_i(|S|)$, we can backtrack from there

Example

Suppose you want to use a HMM tagger to tag the phrase, “the light book”, where we have the following probabilities:

P(the|Det) = 0.3, P(the|Noun) = 0.1, P(light|Noun) = 0.003, P(light|Adj) = 0.2, P(light|Verb) = 0.06, P(book|Noun) = 0.03, P(book|Verb) = 0.001

P(Verb|Det) = 0.00001, P(Noun|Det) = 0.5, P(Adj|Det) = 0.3,

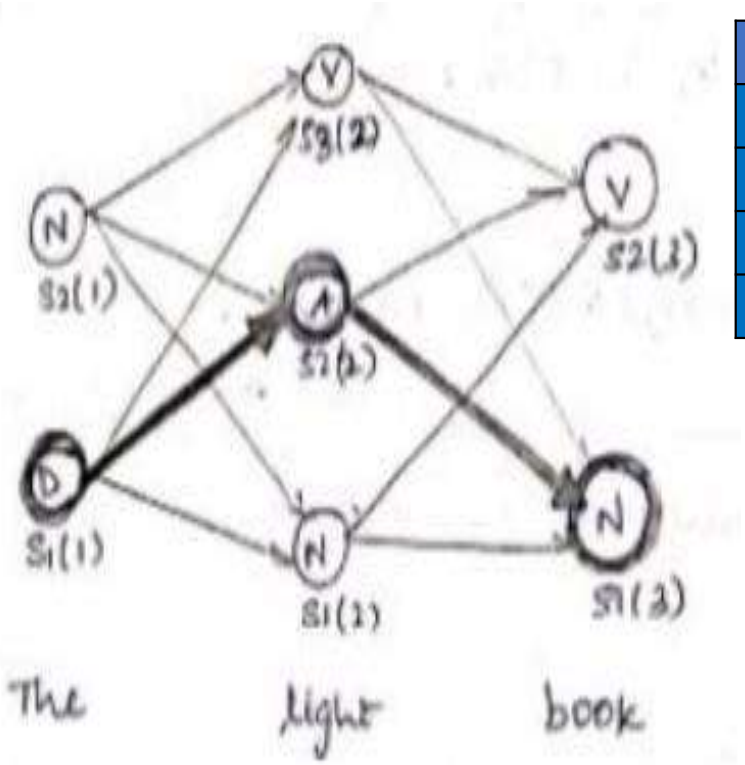
P(Noun|Noun) =0.2, P(Adj|Noun) = 0.002, P(Noun|Adj) = 0.2,

P(Noun|Verb) = 0.3, P(Verb|Noun) = 0.3, P(Verb|Adj) = 0.001,

P(Verb|Verb) = 0.1

Work out in details the steps of the Viterbi algorithm. You can use a Table to show the steps. Assume all other conditional probabilities, not mentioned to be zero. Also, assume that all tags have the same probabilities to appear in the beginning of a sentence.

Example



	Det	Noun	Adj	Verb
Det	0	0.5	0.3	0.00001
Noun	0	0.2	0.002	0.3
Adj	0	0.2	0	0.001
Verb	0	0.3	0	0.1

Tag Transition Probability

$P(t_i/t_{i-1})$

	The	light	book
D	0.3	0	0
N	0.1	0.003	0.03
A	0	0.2	0
V	0	0.06	0.001

Word Emission Probability
Word Probability matrix
$P(w_i/t_i)$

Assumption

All tags have same probabilities to appear in the beginning of a Sentence

So Prob (Det | <s>) , P(N | <s>), P (Adj | <s>), P (V | <s>)

= 0.25 (Sum of Prob = 1)

Level 1.

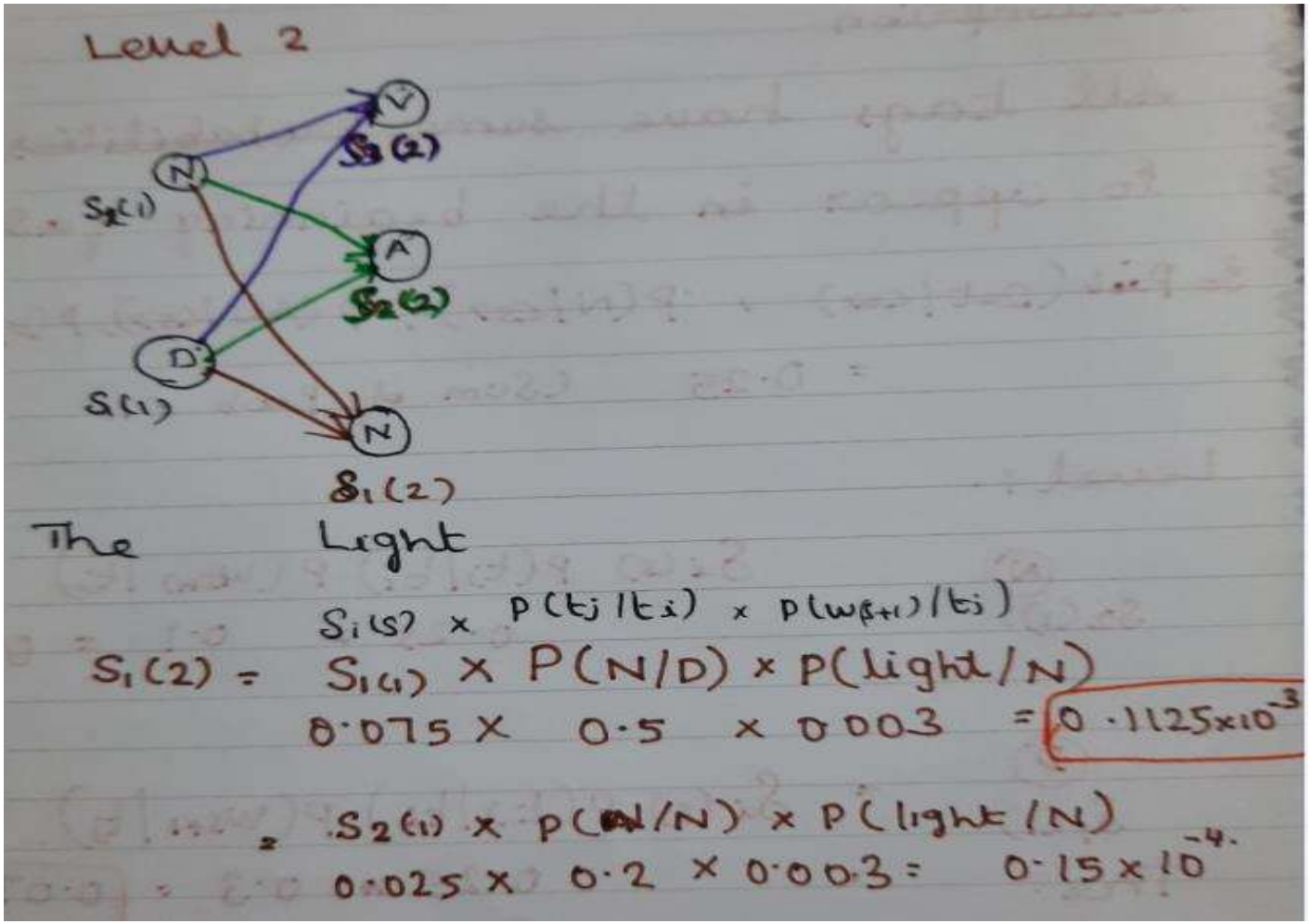
$S_2(1)$ (N) $\delta_i(s)\ P(t_j|t_i)\ P(w_{s+1}|t_j)$

$0.25 * 0.1 = 0.025$

$S_1(1)$ (D) $= \delta_i(s)\ P(t_j|t_i)\ P(w_{s+1}|t_j)$.

$0.25 * 0.3 = 0.075$

The .



$S_i(s) \times P(t_j/t_i) \times P(w_{(t+1)}/t_i)$

$S_1(2) = S_1(1) \times P(N/D) \times P(light/N)$

$0.075 \times 0.5 \times 0.003 = 0.1125 \times 10^{-3}$

$= S_2(1) \times P(N/N) \times P(light/N)$

$0.025 \times 0.2 \times 0.003 = 0.15 \times 10^{-4}$

$S_2(2) = S_1(1) \times P(A/D) \times P(light/A)$

$0.075 \times 0.3 \times 0.2 = 0.45 \times 10^{-2}$

$= S_2(1) \times P(A/N) \times P(light/A)$

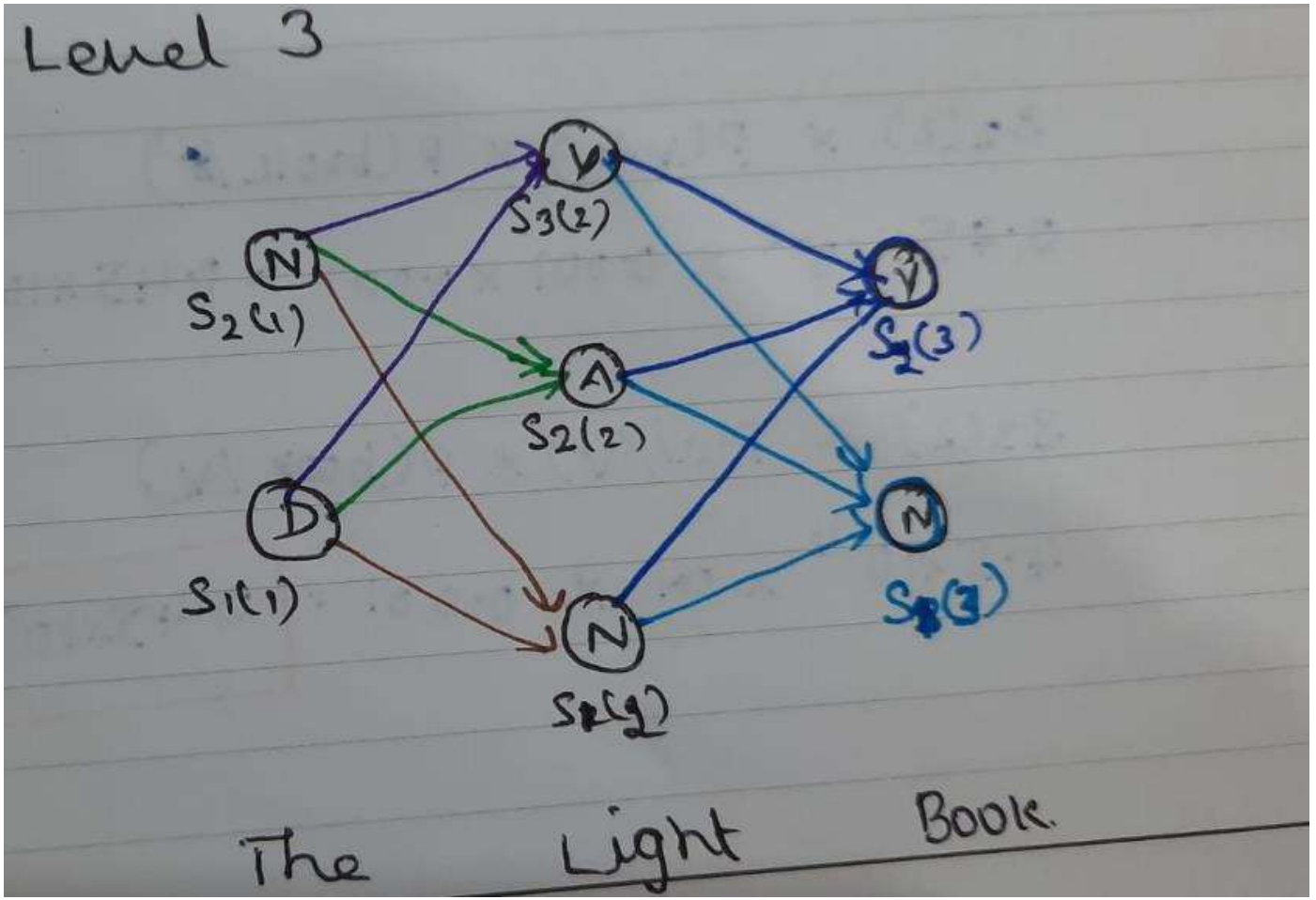
$0.025 \times 0.002 \times 0.2 = 10^{-5}$

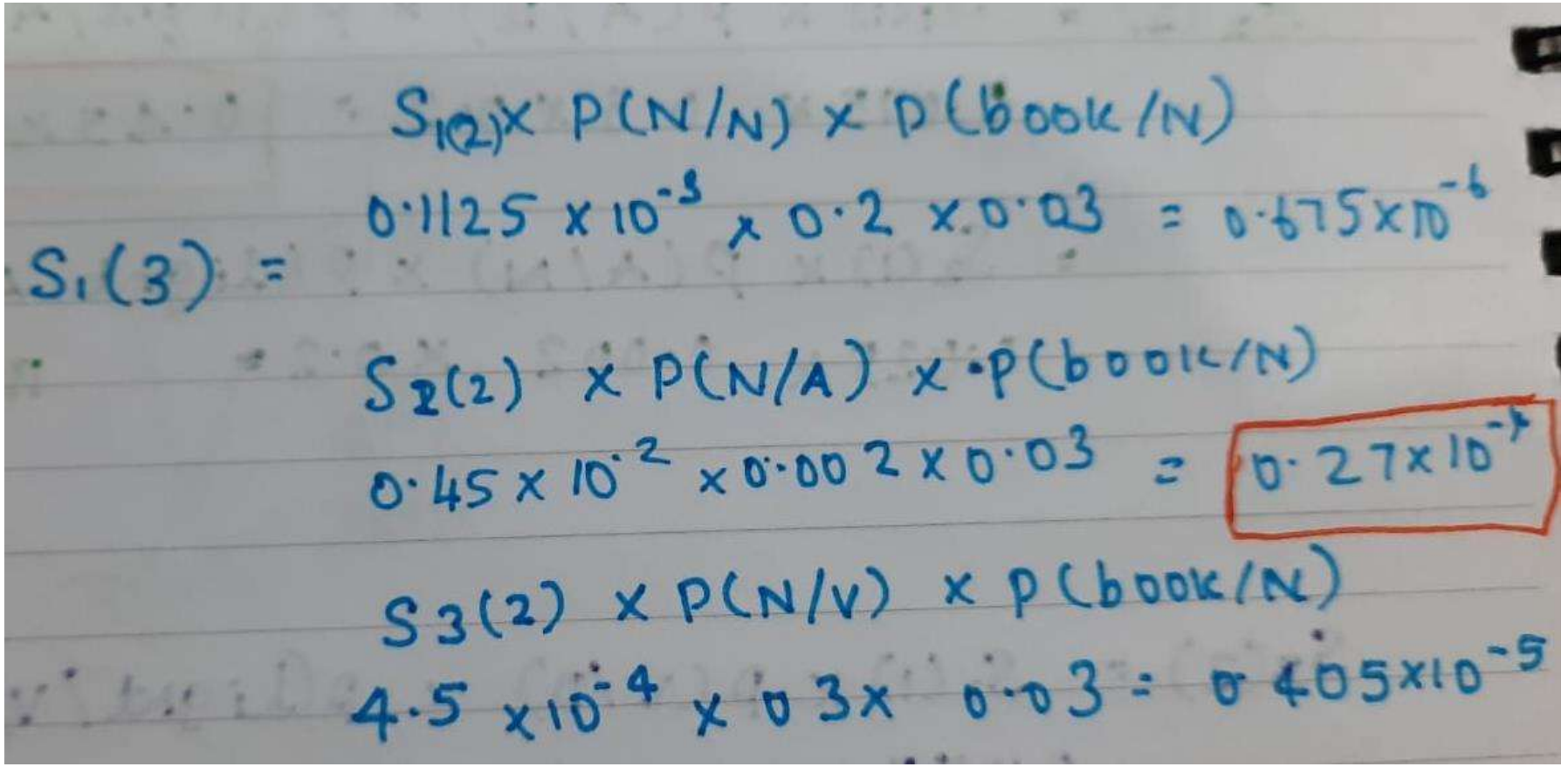
$S_3(2) = S_1(1) \times P(V/D) \times P(light/V)$

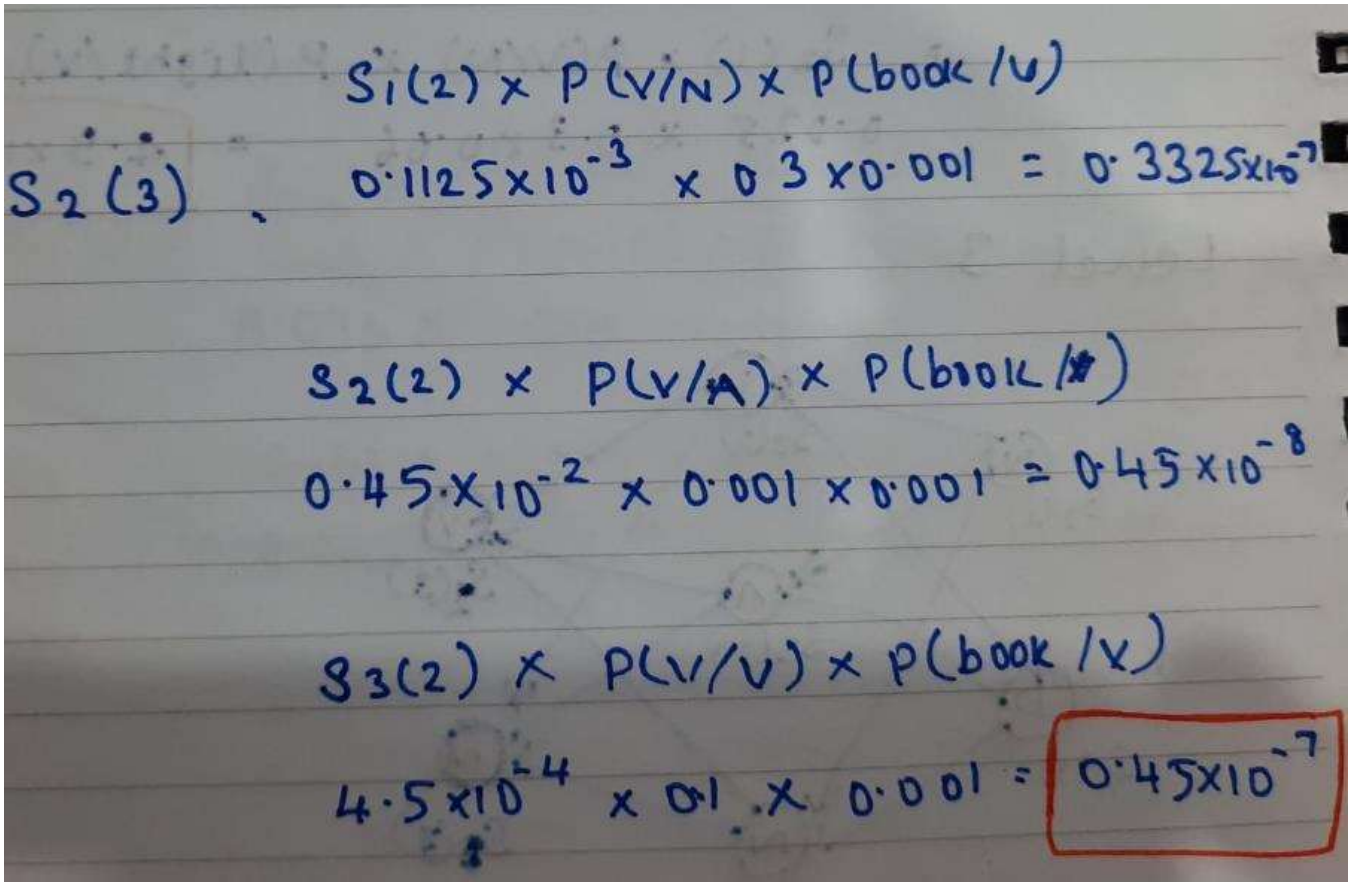
$0.075 \times 0.00001 \times 0.06 = 4.5 \times 10^{-8}$

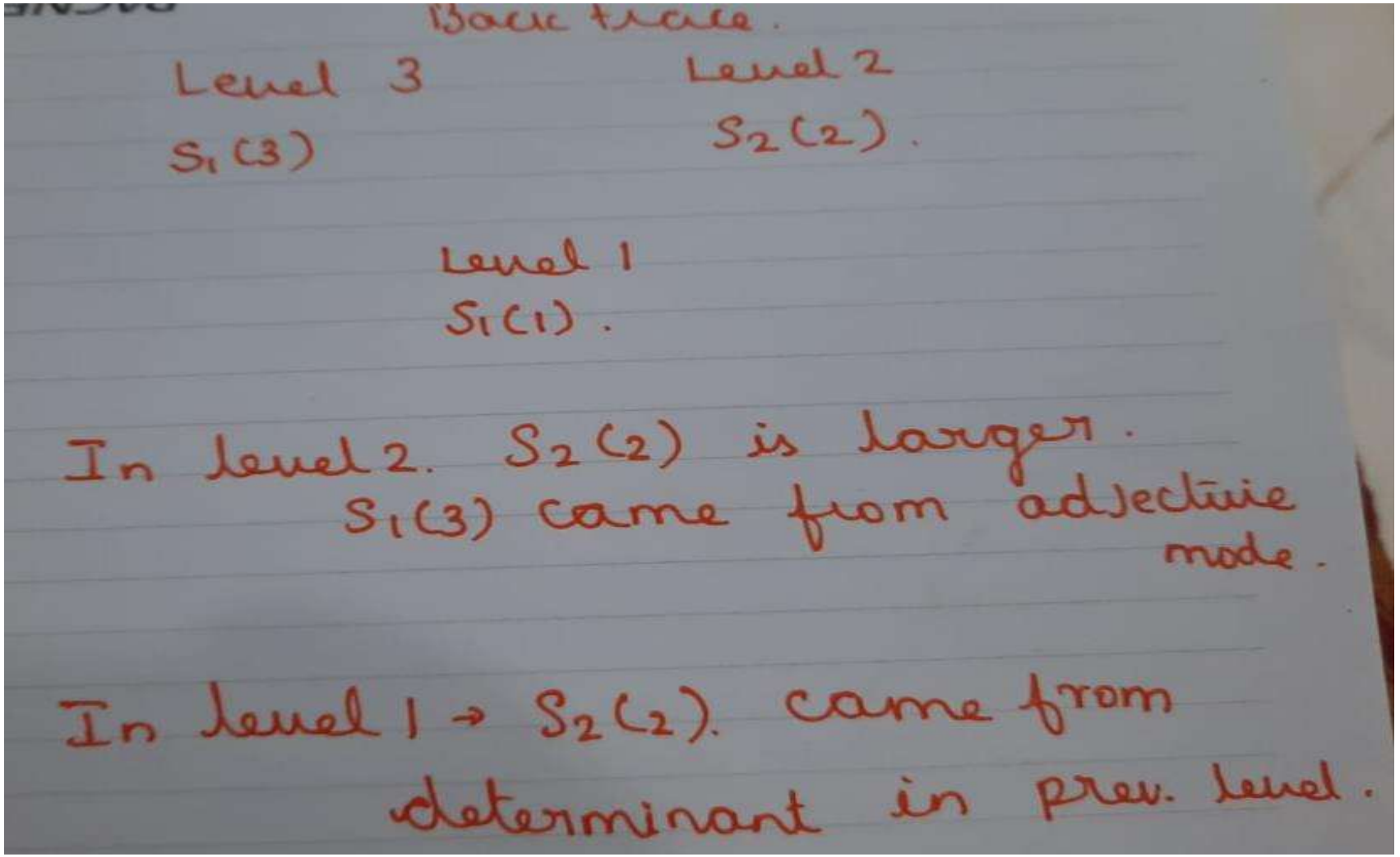
$= S_2(1) \times P(V/N) \times P(light/V)$

$0.025 \times 0.3 \times 0.06 = 4.5 \times 10^{-4}$


Level 3
N
$S_2(1)$
D
$S_1(1)$
$S_3(2)$
A
$S_2(2)$
N
$S_2(2)$
Y
$S_2(3)$
N
The Light Book.

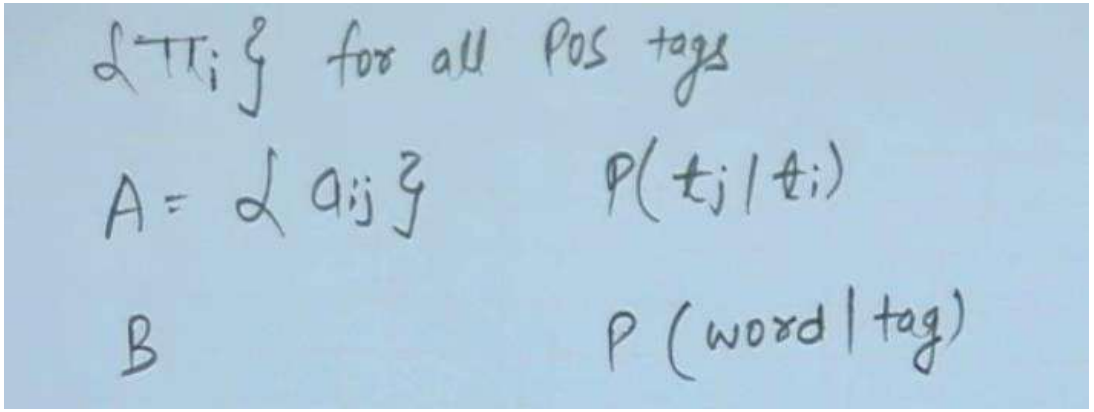

$S_1(2) \times P(N/N) \times P(book/N)$
$0.1125 \times 10^{-3} \times 0.2 \times 0.03 = 0.675 \times 10^{-6}$
$S_1(3) =$
$S_2(2) \times P(N/A) \times P(book/N)$
$0.45 \times 10^{-2} \times 0.002 \times 0.03 = 0.27 \times 10^{-6}$
$S_3(2) \times P(N/V) \times P(book/N)$
$4.5 \times 10^{-4} \times 0.3 \times 0.03 = 0.405 \times 10^{-5}$

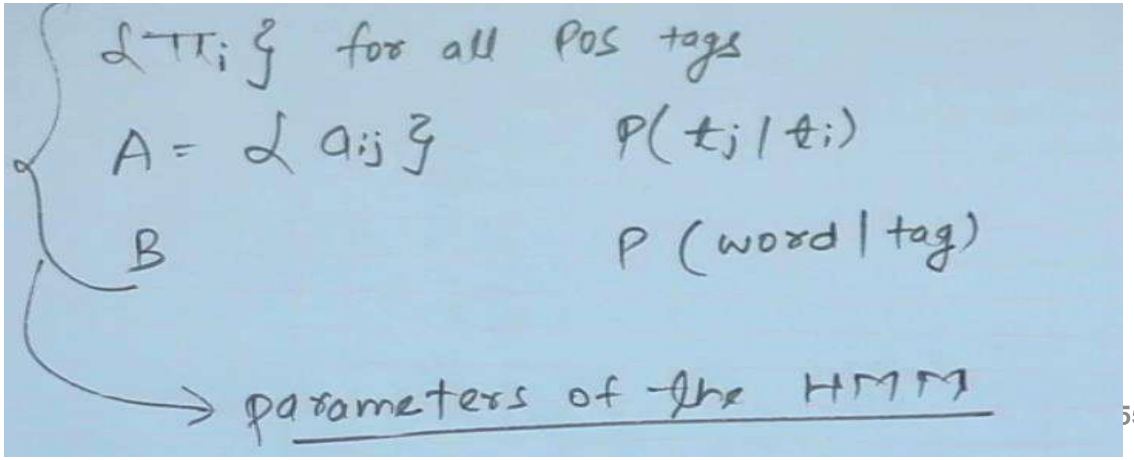

$S_1(2) \times P(V/N) \times P(book/V)$
$S_2(3)$. $0.1125 \times 10^{-3} \times 0.3 \times 0.001 = 0.3325 \times 10^{-7}$
$S_2(2) \times P(V/A) \times P(book/*)$
$0.45 \times 10^{-2} \times 0.001 \times 0.001 = 0.45 \times 10^{-8}$
$S_3(2) \times P(V/V) \times P(book/V)$
$4.5 \times 10^{-4} \times 0.1 \times 0.001 = 0.45 \times 10^{-7}$


Back trace.
Level 3
$S_1(3)$
Level 2
$S_2(2)$.
Level 1
$S_1(1)$.
In level 2. $S_2(2)$ is larger.
$S_1(3)$ came from adjective mode.
In level 1 → $S_2(2)$. came from determinant in prev. level.

- The most suitable POS tagging for the given phrase is

Determinant, Adjective and Noun respectively.





Two Scenarios

- A labeled dataset is available, with the POS category of individual words in a corpus
- Only the corpus is available, but not labeled with the POS categories

- $P(t_j/t_i) = C(t_i, t_j) \ / \ C(t_i) \ = \ MLE$

Two Scenarios

- A labeled dataset is available, with the POS category of individual words in a corpus
- Only the corpus is available, but not labeled with the POS categories

Methods for these scenarios

- For the first scenario, parameters can be directly estimated using maximum likelihood estimate from the labeled dataset
- For the second scenario, *Baum-Welch Algorithm* is used to estimate the parameters of the hidden markov model.

Exercise

- Suppose you want to use a HMM tagger to tag the phrase**, "the beautiful lady"**, where we have the following probabilities:
- P(the|Det) = 0.3, P(the|Noun) = 0.1, P(beautiful|Noun) = 0.01, P(beautiful|Adj) =0.07, P(beautiful|Verb) = 0.001, P(lady|Noun) = 0.08, P(lady|Verb) = 0.01 ,
- P(Verb|Det) = 0.00001, P(Noun|Det) = 0.5, P(Adj|Det) = 0.4,
- P(Noun|Noun) =0.2, P(Adj|Noun) = 0.002, P(Noun|Adj) = 0.2,
- P(Noun|Verb) = 0.3, P(Verb|Noun) = 0.3, P(Verb|Adj) = 0.001,
- P(Verb|Verb) = 0.1

Work out in detail the steps of the Viterbi algorithm. Assume all other conditional probabilities, not mentioned to be zero. Also, assume that all tags have the same probabilities to appear in the beginning of a sentence.

Thank You