19CSE453 – Natural Language Processing

MEMM & CRF

By

Ms. Kavitha C.R.

Dept. of Computer Science and Engineering

Amrita School of Computing, Bengaluru

Amrita Vishwa Vidyapeetham

Issues with Markov Model Tagging

Unknown Words

We do not have the required probabilities.

Possible solutions:

- Use morphological cues (capitalization, suffix) to assign a more calculated guess

Limited Context

- "is clearly **marked**" → verb, past participle
- "he clearly **marked**" → verb, past tense

Possible solution: Use higher order model, combine various n-gram models to avoid sparseness problem

Maximum Entropy Modeling: Discriminative Model

- We may identify a heterogeneous set of features which contribute in some way to the choice of POS tag of the current word.
  - Whether it is the first word in the article
  - Whether the next word is *to*
  - Whether one of the last 5 words is a preposition, etc.
- MaxEnt combines these features in a probabilistic model

The entropy **measures the "amount of information" present in a variable**.

The term "maximum entropy" refers to an optimization framework in which the goal is to find the probability model that maximizes entropy

Maximum Entropy Model(MEM)

$$p_\lambda(y|x) = \frac{1}{Z_\lambda(x)} exp\left(\sum_i \lambda_i f_i(x,y)\right)$$

where

- $Z_\lambda(x)$ is a normalizing constant given by

$$Z_\lambda(x) = \sum_y exp\left(\sum_i \lambda_i f_i(x,y)\right)$$

f_i is the feature function

- λ_i is a weight given to a feature f_i
- x denotes an observed datum and y denotes a class

Features in Maximum Entropy Model

- Features encode elements of the context x for predicting tag y
- Context x is taken around the word w, for which a tag y is to be predicted
- Features are binary values functions, e.g.,

$$f(x,y) = \left\{ \begin{array}{ll} 1 & \text{if } isCapitalized(w) \& y = NNP \\ 0 & otherwise \end{array} \right\}$$

Example: Named Entities

- LOCATION (in Arcadia)
- LOCATION (in Québec)
- DRUG (taking Zantac)
- PERSON (saw Sue)

NNP – Proper Noun Singular

Example Features

- $f_1(x,y) = [y = LOCATION \wedge w_{-1} = \text{“}in\text{”} \wedge isCapitalized(w)]$
- $f_2(x,y) = [y = LOCATION \wedge hasAccentedLatinChar(w)]$
- $f_3(x,y) = [y = DRUG \wedge ends(w, \text{“}c\text{”})]$

Tagging with Maximum Entropy Model

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

Tag sequence candidate $\{t_1, \ldots, t_n\}$ has conditional probability:

$$P(t_1, \ldots, t_n | w_1 \ldots, w_n) = \prod_{i=1}^{n} p(t_i | x_i)$$

- The context x_i also includes previously assigned tags for a fixed history.
- Beam search is used to find the most probable sequence

$$p_\lambda(y|x) = \frac{1}{Z_\lambda(x)} exp\left(\sum_i \lambda_i f_i(x,y)\right)$$

where

- $Z_\lambda(x)$ is a normalizing constant given by

$$Z_\lambda(x) = \sum_y exp\left(\sum_i \lambda_i f_i(x,y)\right)$$

- λ_i is a weight given to a feature f_i
- x denotes an observed datum and y denotes a class

MEM POS tagging problem

Consider the maximum entropy model for POS tagging, where you want to estimate $P(tag|word)$. In a hypothetical setting, assume that *tag* can take the values *D*, *N* and *V* (short forms for Determiner, Noun and Verb). The variable *word* could be any member of a set V of possible words, where V contains the words *a*, *man*, *sleeps*, as well as additional words. The distribution should give the following probabilities

- $P(D|a) = 0.9$
- $P(N|man) = 0.9$
- $P(V|sleeps) = 0.9$
- $P(D|word) = 0.6$ for any word other than *a*, *man* or *sleeps*
- $P(N|word) = 0.3$ for any word other than *a*, *man* or *sleeps*
- $P(V|word) = 0.1$ for any word other than *a*, *man* or *sleeps*

It is assumed that all other probabilities, not defined above could take any values such that $\sum_{tag} P(tag|word) = 1$ is satisfied for any word in V.

- Define the features of your maximum entropy model that can model this distribution. Mark your features as f_1, f_2 and so on. Each feature should have the same format as explained in the class. [**Hint:** 6 Features should make the analysis easier]
- For each feature f_i, assume a weight λ_i. Now, write expression for the following probabilities in terms of your model parameters
  - $P(D|cat)$
  - $P(N|laughs)$
  - $P(D|man)$
- What value do the parameters in your model take to give the distribution as described above. (i.e. $P(D|a) = 0.9$ and so on. *You may leave the final answer in terms of equations*)

Given:

$P(D/a) = 0.9$

$P(N/man) = 0.9$

$P(V/Sleeps) = 0.9$

$P(D/word) = 0.6$

$P(N/word) = 0.3$

$P(V/word) = 0.1$

word here is other than a, man, sleeps so denoting by 'V'

① Define the features of your Max Ent model that can model this distribution.

$f_i(x, y)$

① $f_1 : \begin{cases} 1 & \text{if word = 'a' \& tag = 'D'} \\ 0 & \text{otherwise} \end{cases}$

② $f_2 : \begin{cases} 1 & \text{if word = 'man' \& tag = 'N'} \\ 0 & \text{otherwise} \end{cases}$

③ $f_3 : \begin{cases} 1 & \text{if word = 'Sleeps' \& tag = 'V'} \\ 0 & \text{otherwise} \end{cases}$

④ $f_4 : \begin{cases} 1 & \text{if word} \neq \text{'V' \& tag = 'D'} \\ 0 & \text{otherwise} \end{cases}$

⑤ $f_5 : \begin{cases} 1 & \text{if word} \neq \text{'V' \& tag = 'N'} \\ 0 & \text{otherwise} \end{cases}$

⑥ $f_6 : \begin{cases} 1 & \text{if word} \neq \text{'V' \& tag = 'V'} \\ 0 & \text{otherwise} \end{cases}$

2) For each feature f_i, assume a weight λ_i. Now, write expression for the following probabilities in terms of your model parameters.

(i) $P(D/cat) = \frac{\exp \sum \lambda_i f_i}{Z}$

where $\sum \lambda_i f_i(x, y)$ — cat, D

$$= \frac{\exp(\lambda_1 \cdot 0 + \lambda_2 \cdot 0 + \lambda_3 \cdot 0 + \lambda_4 \cdot 1 + \lambda_5 \cdot 0 + \lambda_6 \cdot 0)}{Z}$$

where $Z = (P(D/cat) + P(N/cat) + P(V/cat))$

$$\frac{\exp \lambda_4}{Z} + \frac{\exp \lambda_5}{Z} + \frac{e^{\lambda_6}}{Z}$$

PACNET

$$P(D/cat) = \frac{\frac{e^{\lambda_4}}{Z}}{\frac{e^{\lambda_4}}{Z} + \frac{e^{\lambda_5}}{Z} + \frac{e^{\lambda_6}}{Z}}$$

$$= \frac{\frac{e^{\lambda_4}}{Z}}{\frac{e^{\lambda_4} + e^{\lambda_5} + e^{\lambda_6}}{Z}}$$

$$\therefore P(D/cat) = \frac{e^{\lambda_4}}{e^{\lambda_4} + e^{\lambda_5} + e^{\lambda_6}}$$

$$P(N/\text{laughs}) = \frac{\exp\{\sum \lambda_i f_i\}}{z}$$

$$= \frac{e^{(\lambda_1 \cdot 0 + \lambda_2 \cdot 0 + \lambda_3 \cdot 0 + \lambda_4 \cdot 0 + \lambda_5 \cdot 1 + \lambda_6 \cdot 0)}}{z}$$

$$= \frac{e^{\lambda_5}}{z}$$

where $z = P(D/\text{laughs}) + P(N/\text{laughs}) + P(V/\text{laughs})$

$$= \frac{e^{\lambda_4} + e^{\lambda_5} + e^{\lambda_6}}{z}$$

$$\boxed{P(N/\text{laughs}) = \frac{e^{\lambda_5}}{e^{\lambda_4} + e^{\lambda_5} + e^{\lambda_6}}}$$

$$P(D/\text{Man}) = \frac{\exp(\lambda_1 \cdot 0 + \lambda_2 \cdot 0 + \lambda_3 \cdot 0 + \lambda_4 \cdot 0 + \lambda_5 \cdot 0 + \lambda_6 \cdot 0)}{z}$$

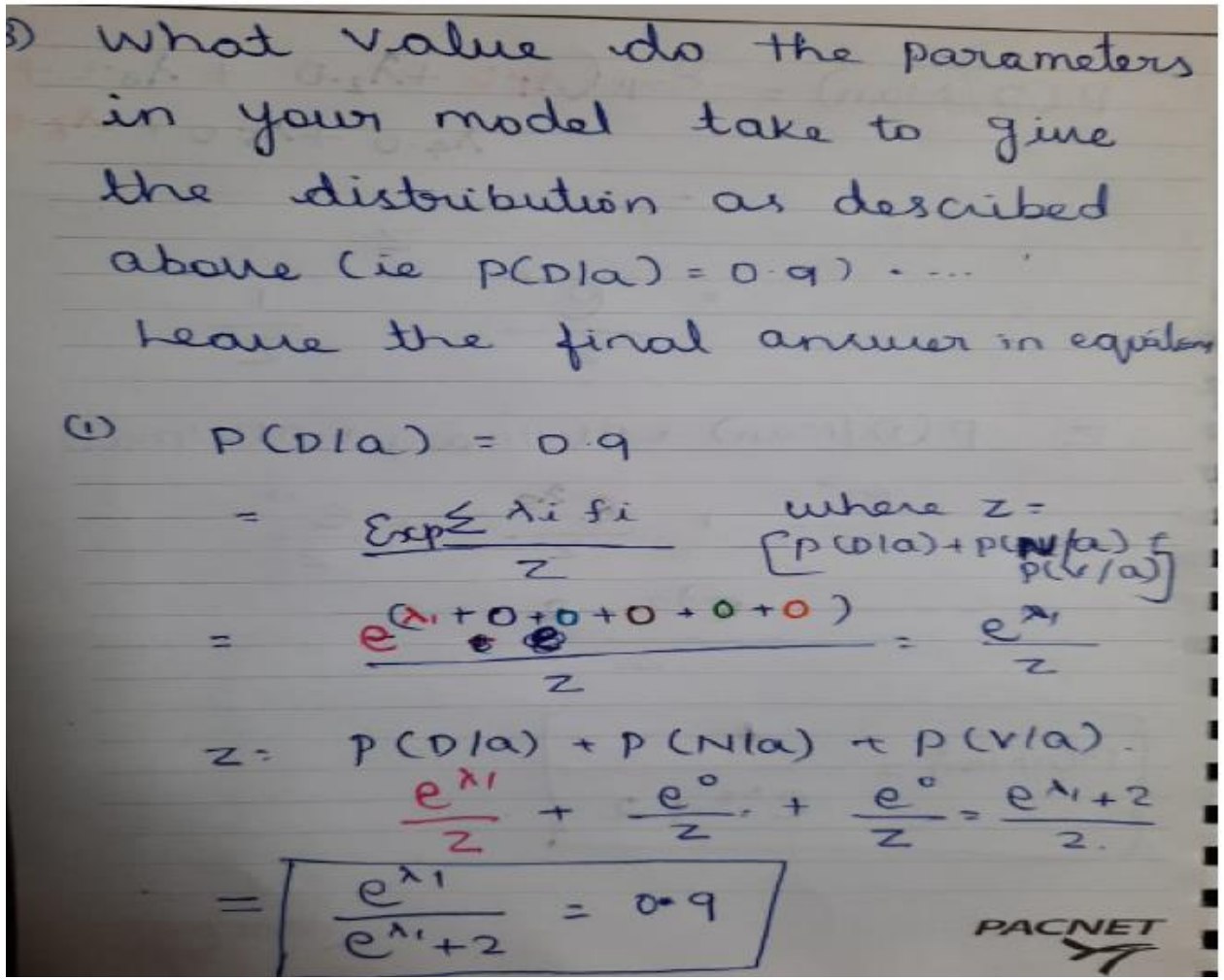
$$= \frac{e^0}{z} = \frac{1}{z}$$

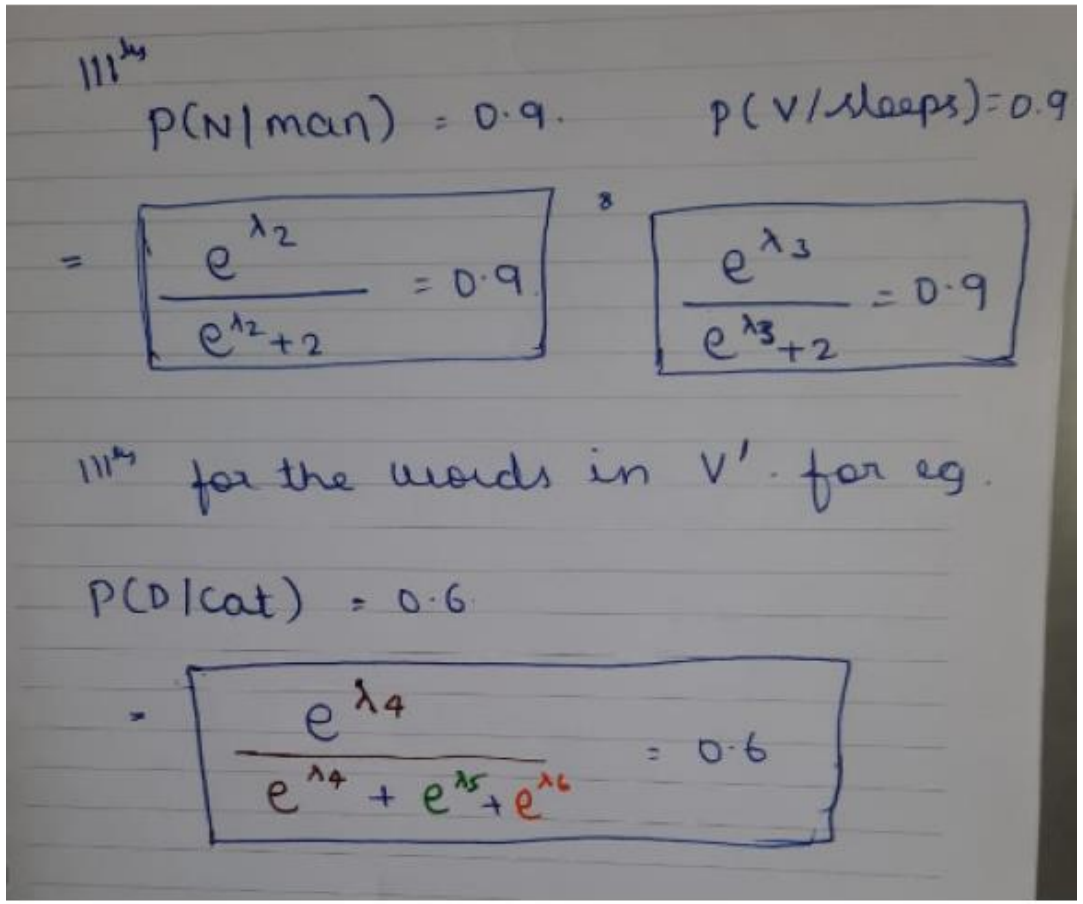
$$z = P(D/\text{Man}) + P(N/\text{man}) + P(V/\text{man})$$

$$= \frac{e^0}{z} + \frac{e^{\lambda_2}}{z} + \frac{e^0}{z}$$

$$= \frac{e^{\lambda_2} + 2}{z}$$

$$\boxed{P(D/\text{Man}) = \frac{1}{e^{\lambda_2} + 2}}$$


3) What value do the parameters in your model take to give the distribution as described above (ie P(D|a) = 0.9) ....
Leave the final answer in equation
(i) P(D|a) = 0.9
= Exp Σ λi fi / z where z = [P(D|a) + P(N|a) + P(V|a)]
= e^(λ1 + 0 + 0 + 0 + 0 + 0) / z = e^λ1 / z
z = P(D|a) + P(N|a) + P(V|a)
e^λ1/z + e^0/z + e^0/z = e^λ1 + 2 / z
= e^λ1 / e^λ1 + 2 = 0.9
PACNET


III^ly
P(N|man) = 0.9. P(V|sleeps) = 0.9
= e^λ2 / e^λ2 + 2 = 0.9 & e^λ3 / e^λ3 + 2 = 0.9
III^ly for the words in V'. for eg.
P(D|cat) = 0.6
= e^λ4 / e^λ4 + e^λ5 + e^λ6 = 0.6

MAXIMUM ENTROPY MODELS

- Machine learning framework called Maximum Entropy modeling.

- Used for Classification
  - The task of classification is to take a single observation, extract some useful features describing the observation, and then based on these features, to classify the observation into one of a set of discrete classes.

- Probabilistic classifier: gives the probability of the observation being in that class

- Non-sequential classification
  - in text classification we might need to decide whether a particular email should be classified as spam or not
  - In sentiment analysis we have to determine whether a particular sentence or document expresses a positive or negative opinion.
  - we'll need to classify a period character ('.') as either a sentence boundary or not

LINEAR REGRESSION

- Given a set of real-valued observations, each observation associated with a set of features, linear regression formulates a linear expression which predicts the value of an outcome given its associated features.
- MULTIPLE LINEAR REGRESSION
  - The true power of linear regression is visible when we have multiple features
- LOGISTIC REGRESSION
  - Linear regression is what we want when we are predicting a real-valued outcome.
  - But somewhat more commonly in speech and language processing we are doing classification, in which the output *y we are trying to predict takes on one from a small set of* discrete values.

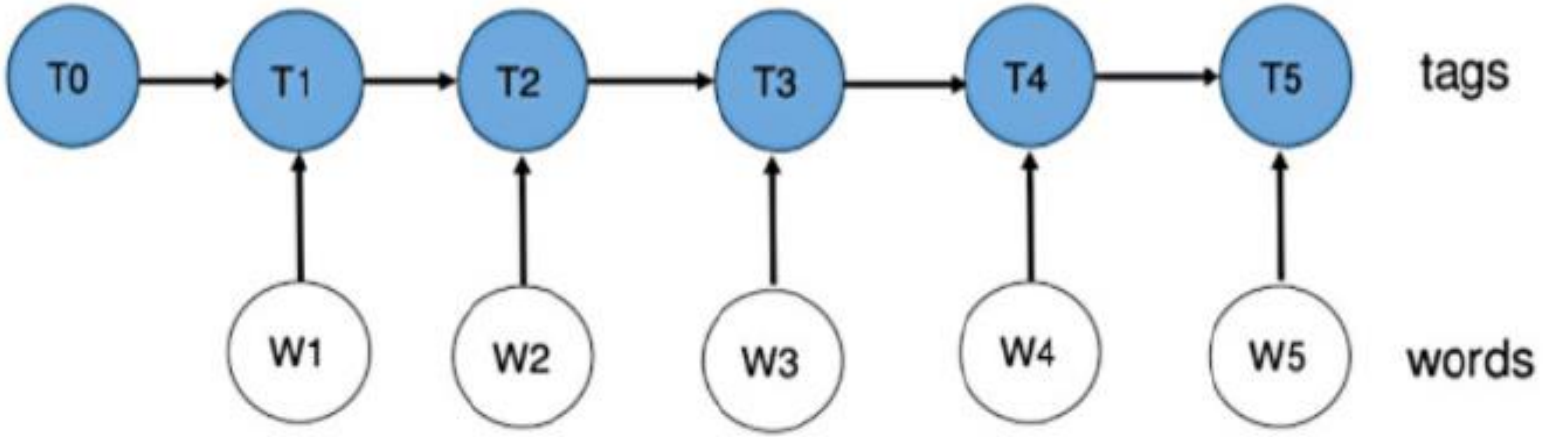
LOGISTIC REGRESSION

- Furthermore, instead of just returning the 0 or 1 value, we'd like a model that can give us the probability that a particular observation is in class 0 or 1.
- This is important because in most real-world tasks we're *passing the results of this classifier* onto some further classifier to accomplish some task.
- Since we are rarely completely certain about which class an observation falls in, we'd prefer not to make a hard decision at this stage, ruling out all other classes

Building blocks of Maximum-Entropy Markov Model

- Let's consider the task of email spam detection.
- Using **logistic regression**, we can obtain the probability that an email is spam. It's essentially binary classification.
- We can extend this to a multiclass problem.
- For example, in image analysis, we're required to classify the object into one of many classes.
- We estimate the probability for each class.
- Rather than take a hard decision on one of the outcomes, it's better to output probabilities, which will benefit downstream tasks.

- Multinomial logistic regression is also called softmax regression or Maximum Entropy (MaxEnt) classifier.
- Entropy's related to "disorder". Higher the disorder, less predictable the outcomes, and hence more information.
- For example, an unbiased coin has more information (and entropy) than a coin that mostly lands up heads.
- MaxEnt is therefore about picking a probability distribution that maximizes entropy.
- Then, there's Markov Chain. It models a system as a set of states with probabilities assigned to state transitions.
- While MaxEnt computes probabilities for each input independently, Markov chain recognizes that there's dependency from one state to the next.
- Thus, MEMM is about maximizing entropy plus using state dependencies (Markov Model).



Discriminative model, model conditional probability Pr(T | W) directly.

$$\Pr(\boldsymbol{T}|\boldsymbol{W}) = \prod_{i=1}^{L} \Pr(t_i|t_{i-1}, w_i) = \prod_{i=1}^{L} \frac{\exp(\sum_j \beta_j f_j(t_{i-1}, w_i))}{Z(t_{i-1}, w_i)}$$

t_0 is a dummy start s

Example of POS tagging with MEMM. Sour

$$p_\lambda(y|x) = \frac{1}{Z_\lambda(x)} exp\left(\sum_i \lambda_i f_i(x, y)\right)$$

$$\Pr(\boldsymbol{T}|\boldsymbol{W}) = \prod_{i=1}^{L} \Pr(t_i|t_{i-1}, w_i) = \prod_{i=1}^{L} \frac{\exp(\sum_j \beta_j f_j(t_{i-1}, w_i))}{Z(t_{i-1}, w_i)}$$

t_0 is a dummy start state.

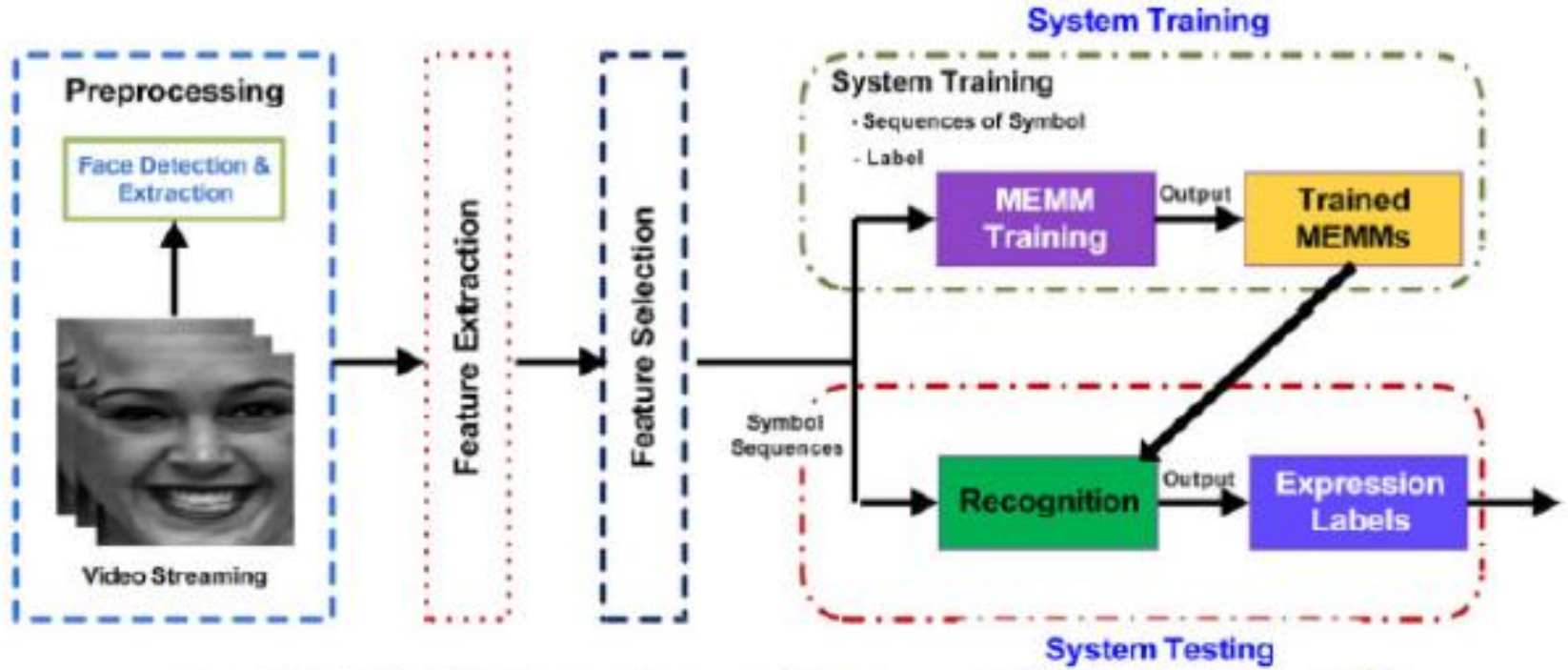
or

$$P(t_1, \ldots, t_n|w_1 \ldots, w_n) = \prod_{i=1}^{n} p(t_i|x_i) = \prod_{i=1}^{n} \frac{1}{Z_\lambda(x)} exp\left(\sum_i \lambda_i f_i(x, y)\right)$$

1. Consider the problem of POS tagging.
2. Given a sequence of words, we wish to find the most probable sequence of tags.
3. MEMM predicts the tag sequence by modelling tags as states of the Markov chain.
4. To predict a tag, MEMM uses the current word and the tag assigned to the previous word. In reality, MEMM doesn't take a hard decision on what tag to select.
5. Using MaxEnt, it calculates the probability of each tag. Probabilities of past tags are used to take a soft decision so that the word sequence as a whole get the best possible tag sequence.

Applications of MEMM

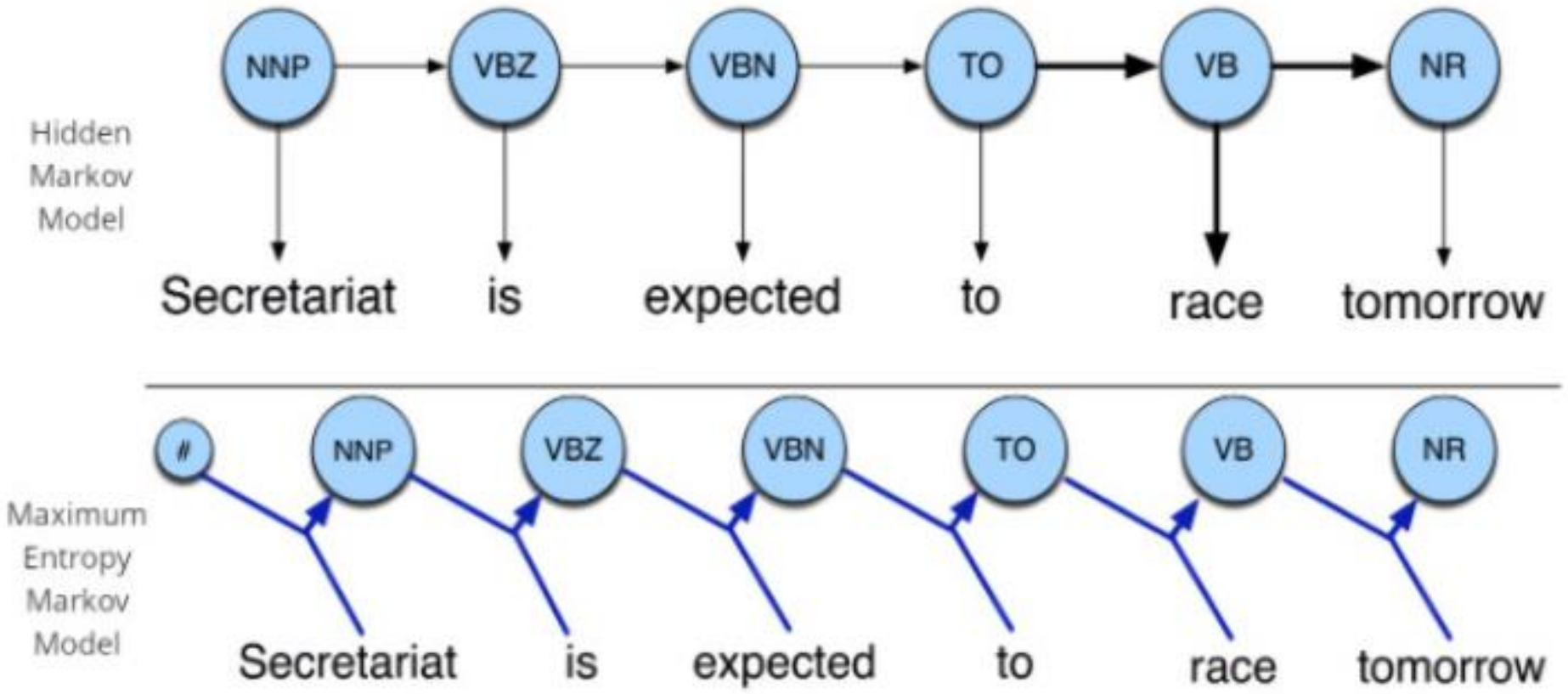
MEMM has been used in many NLP tasks including POS tagging and semantic role modelling.
It's been used in computational biology for protein secondary structure prediction.
One study used MEMM for recognizing human facial expressions.
Human expressions are modelled as states. Observations are from video sensors



Use of MEMM for facial expression recognition. Source: Siddiqi et al. 2016, fig. 1.*

MEMM Vs HMM

How is MEMM different from Hidden Markov Model (HMM)



HMM vs. MEMM for POS tagging. Source: Adapted from Jurafsky and Martin 2009, fig. 6.20.

MEMM Vs HMM

HMM is a **generative model** because words as modelled as observations generated from hidden states.
Even the formulation of probabilities uses likelihood P(W|T) and prior P(T).

On the other hand, MEMM is a **discriminative model**. This is because it directly uses posterior probability P(T|W);
that is, probability of a tag sequence given a word sequence. Thus, it discriminates among the possible tag sequences.

It can also be said that HMM uses **joint probability** for maximizing the probability of the word sequence.
MEMM uses **conditional probability**, conditioned on previous tag and current word.

In HMM, for the tag sequence decoding problem, probabilities are obtained by training on a text corpus.
In MEMM, we build a distribution by adding features, which can be hand crafted or picked out by training.

The idea is to select the maximum entropy distribution given the constraints specified by the features.
MEMM is more flexible because we can add features such as capitalization, hyphens or word endings, which are hard to consider in HMM. MEMM allows for diverse non-independent features.

MEMM Vs HMM

MEMM

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

$$= \underset{T}{\operatorname{argmax}} \prod_i P(tag_i|word_i, tag_{i-1})$$

MEMM gives one probability estimate per hidden state, which is the probability of the next tag given the previous tag and the observation.

$$= \underset{T}{\operatorname{argmax}} \prod_i P(tag_i|word_i, tag_{i-1})$$

Posterior

HMM

$$= \underset{T}{\operatorname{argmax}} \prod_i P(word_i|tag_i) \prod_i P(tag_i|tag_{i-1})$$

HMM model includes distinct probability estimates for each transition and observation

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

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

$$= \underset{T}{\operatorname{argmax}} \prod_i P(word_i|tag_i) \prod_i P(tag_i|tag_{i-1})$$

Likelihood

Prior

Features for POS Tagging

The specific word and tag context available to a feature is

$$h_i = \{w_i, w_{i+1}, w_{i+2}, w_{i-1}, w_{i-2}, t_{i-1}, t_{i-2}\}$$

Example: $f_j(h_i, t_i) = 1$ if $suffix(w_i) = \text{“}ing\text{”} \& t_i = VBG$

Example Features

Condition	Features	
w_i is not rare	$w_i = X$	& $t_i = T$
w_i is rare	X is prefix of w_i, $\|X\| \leq 4$	& $t_i = T$
	X is suffix of w_i, $\|X\| \leq 4$	& $t_i = T$
	w_i contains number	& $t_i = T$
	w_i contains uppercase character	& $t_i = T$
	w_i contains hyphen	& $t_i = T$
$\forall\ w_i$	$t_{i-1} = X$	& $t_i = T$
	$t_{i-2}t_{i-1} = XY$	& $t_i = T$
	$w_{i-1} = X$	& $t_i = T$
	$w_{i-2} = X$	& $t_i = T$
	$w_{i+1} = X$	& $t_i = T$
	$w_{i+2} = X$	& $t_i = T$

Example Features

Word:	the	stories	about	well-heeled	communities	and	developers
Tag:	DT	NNS	IN	JJ	NNS	CC	NNS
Position:	1	2	3	4	5	6	7

w_i = about & t_i = IN
w_{i-1} = stories & t_i = IN
w_{i-2} = the & t_i = IN
w_{i+1} = well-heeled & t_i = IN
w_{i+2} = communities & t_i = IN
t_{i-1} = NNS & t_i = IN
$t_{i-2}t_{i-1}$ = DT NNS & t_i = IN

w_{i-1} = about & t_i = JJ
w_{i-2} = stories & t_i = JJ
w_{i+1} = communities & t_i = JJ
w_{i+2} = and & t_i = JJ
t_{i-1} = IN & t_i = JJ
$t_{i-2}t_{i-1}$ = NNS IN & t_i = JJ
prefix(w_i)=w & t_i = JJ
prefix(w_i)=we & t_i = JJ
prefix(w_i)=wel & t_i = JJ
prefix(w_i)=well & t_i = JJ
suffix(w_i)=d & t_i = JJ
suffix(w_i)=ed & t_i = JJ
suffix(w_i)=ied & t_i = JJ
suffix(w_i)=eled & t_i = JJ
w_i contains hyphen & t_i = JJ

NNS-Noun plural
IN – preposition
JJ- adjective
CC-coordin or conjunction

Search Algorithm

Conditional Probability

Given a sentence $\{w_1, \ldots, w_n\}$, a tag sequence candidate $\{t_1, \ldots, t_n\}$ has conditional probability:

$$P(t_1, \ldots, t_n | w_1 \ldots, w_n) = \prod_{i=1}^{n} p(t_i | x_i)$$

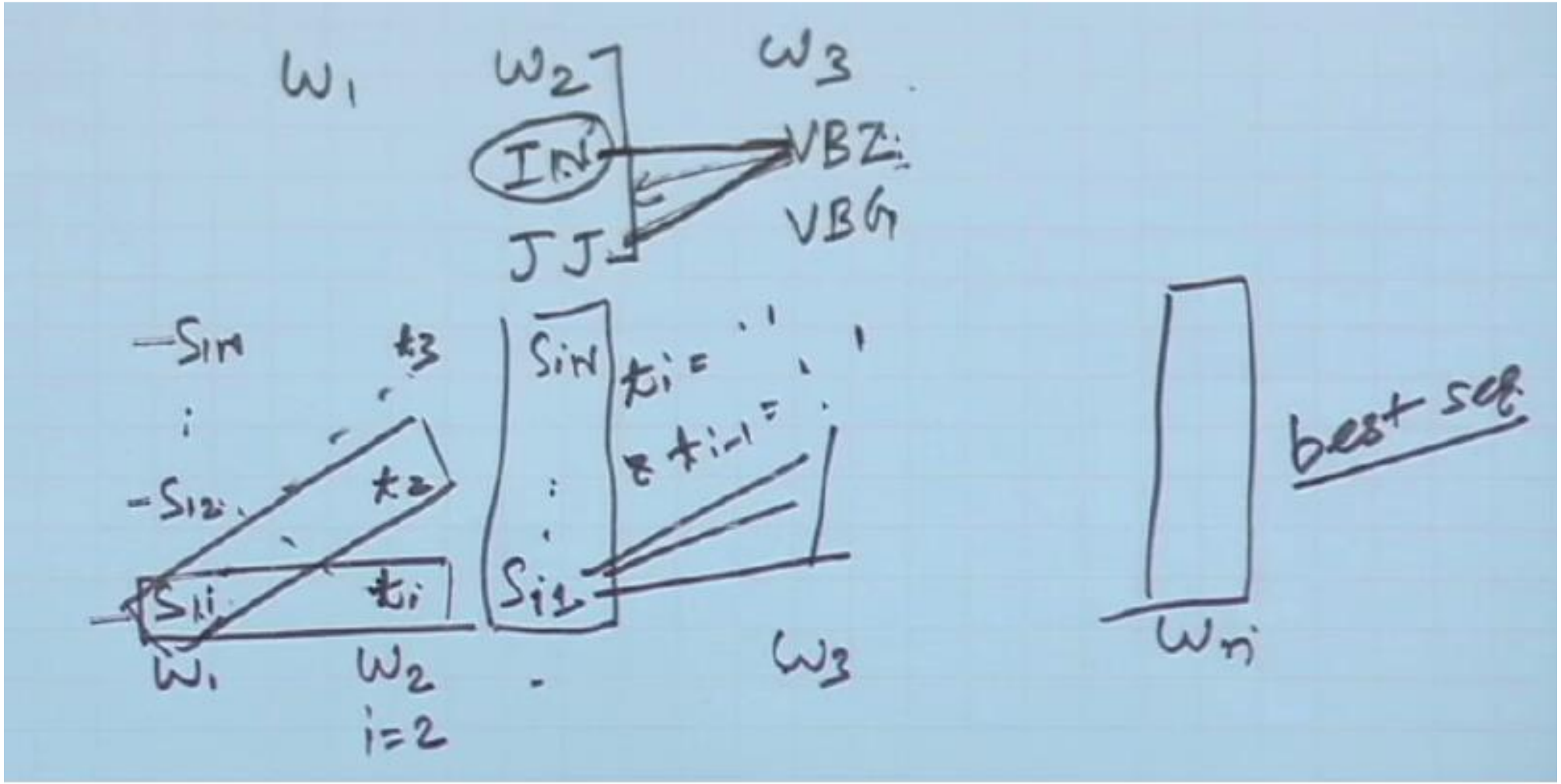
A *Tag Dictionary* is used, which, for each known word, lists the tags that it has appeared with in the training set.

Search Algorithm

Let $W = \{w_1, \ldots, w_n\}$ be a test sentence, s_{ij} be the jth highest probability ta sequence up to and including word w_i.

Search description

- Generate tags for w_1, find top N, set s_{1j}, $1 \leq j \leq N$, accordingly.
- Initialize $i = 2$
  - Initialize $j = 1$
  - Generate tags for w_i, given $s_{(i-1)j}$ as previous tag context, and append each tag to $s_{(i-1)j}$ to make a new sequence
  - $j = j + 1$, repeat if $j \leq N$
- Find N highest probability sequences generated by above loop, set s_{ij} accordingly
- $i = i + 1$, repeat if $i \leq n$
- Return highest probability sequence s_{n1}


W1
W2
W3
IN
VBZ
VBG
JJ
Si1
Si2
SiN
ti
best seq
Wn
i=2

Problem with Maximum Entropy Model

Per-state normalization

All the mass that arrives at a state must be distributed among the possible successor states

This gives a 'label bias' problem

- Per-state normalization of transition scores implies "conservation of score mass"
- Bias towards states with fewer outgoing transitions
- State with single outgoing transition effectively ignores observation

Conditional Random Fields(CRF)

Conditional random fields (CRFs) are **a class of statistical modeling methods often applied in pattern recognition and machine learning and used for structured prediction**.

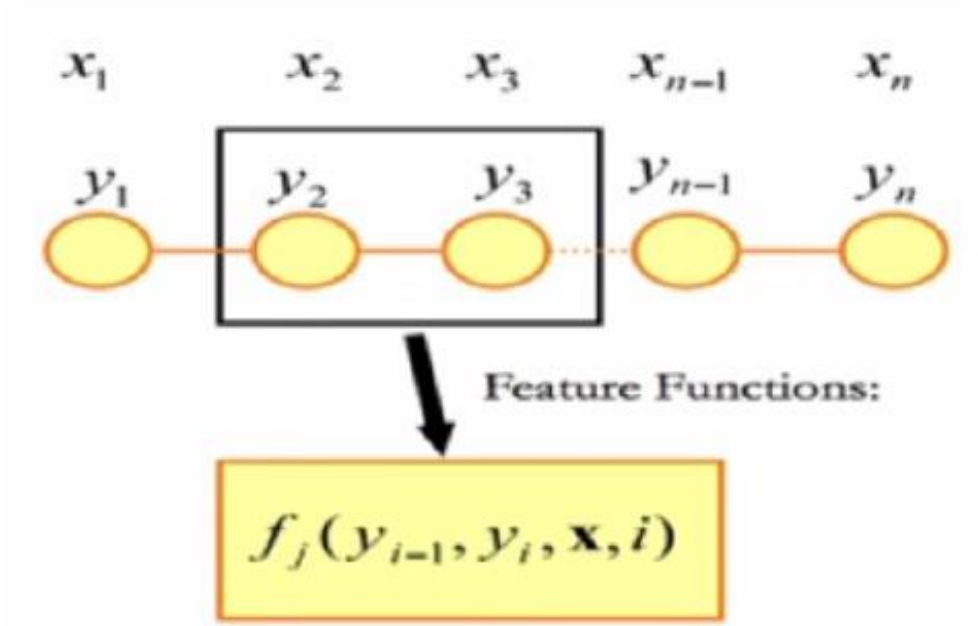
Whereas a classifier predicts a label for a single sample without considering "neighbouring" samples, a CRF can take context into account.

- Undirected graph (random field)
- Construct conditional model p(Y|X)
- Does not explicitly model marginal p(X)
- Assumption: graph is fixed
  - concerns itself with chain graphs and sequences

Condition Random Fields

- CRFs are conditionally trained, undirected graphical models.
- Let's look at the linear chain structure

Condition Random Fields – Feature Function



Feature Functions

Express some characteristic of the empirical distribution that we wish to hold in the model distribution

$f_j(y_{i-1}, y_i, \mathbf{x}, i)$

1 *if* $y_{i-1} = IN$ *and*
$y_i = NNP$ *and*
$x_i = September$

0 *otherwise*

Conditional Random Fields: Distribution

Label sequence modelled as a normalized product of feature functions:

$$P(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{Z(\mathbf{x})} \exp \sum_{i=1}^{n} \sum_{j} \lambda_j f_j(y_{i-1}, y_i, \mathbf{x}, i)$$

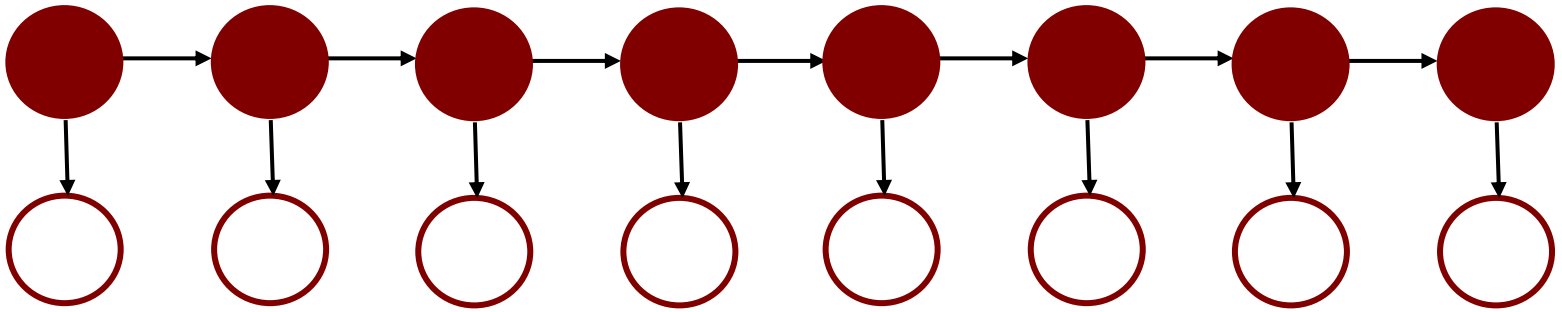
$$Z(\mathbf{x}) = \sum_{y \in Y} \sum_{i=1}^{n} \sum_{j} \lambda_j f_j(y_{i-1}, y_i, \mathbf{x}, i)$$

CRFs

- Have the advantages of MEMM but avoid the label bias problem
- CRFs are globally normalized, whereas MEMMs are locally normalized.
- Widely used and applied. CRFs have been (close to) state-of-the-art in many sequence labeling tasks.

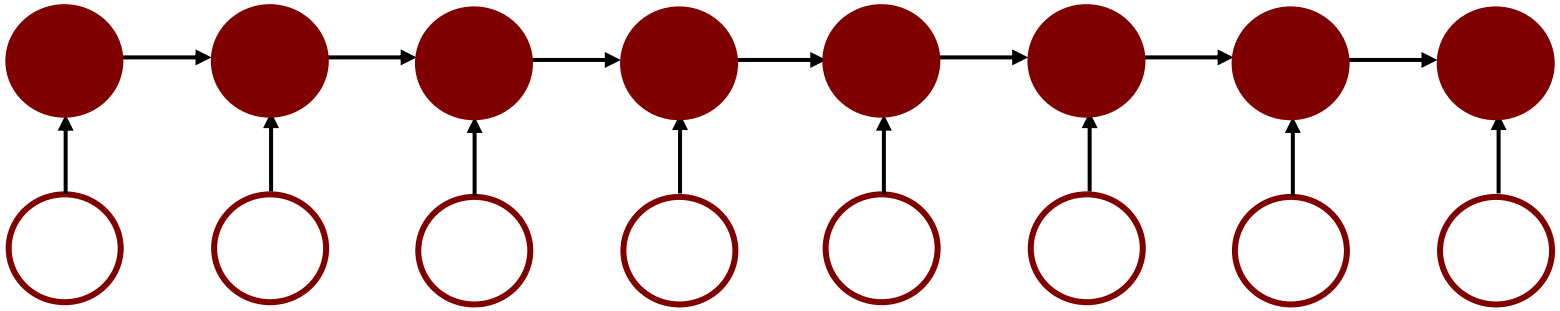
- CRFs do not suffer from the label bias problem!
- Parameter estimation guaranteed to find the global optimum
- Limitation: Slow convergence of the training algorithm

Hidden Markov Models (HMMs)



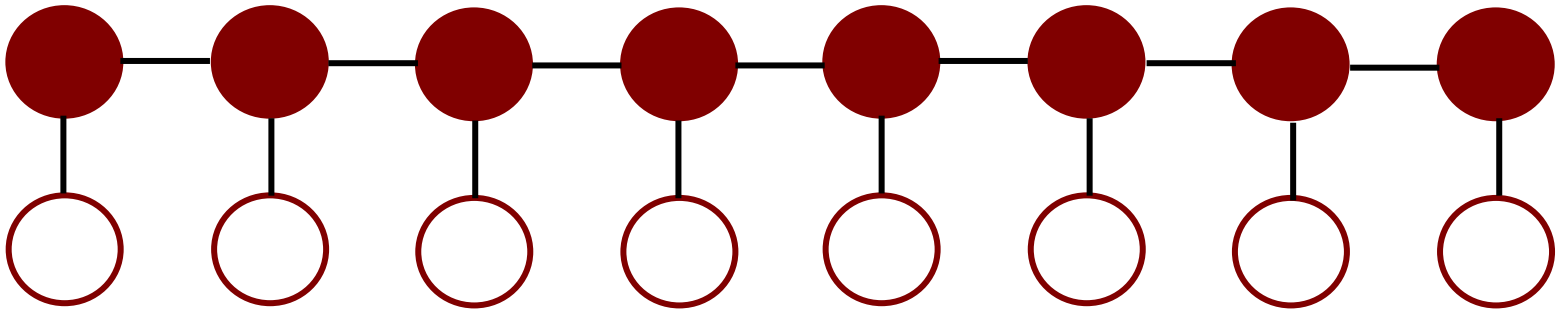
- Generative
  - Find parameters to maximize P(X,Y)
- Assumes features are independent
- When labeling X_i future observations are taken into account (Baum-Welch or forward-backward algorithm while training)

MaxEnt Markov Models (MEMMs)



- Discriminative
  - Find parameters to maximize P(Y|X)
- No longer assume that features are independent
- Do not take future observations into account (forward-backward algorithm for training)
- A **Maximum Entropy Markov Model** or **MEMM** is a sequence model augmentation of MaxEnt which makes use of the Viterbi decoding algorithm

Conditional Random Fields (CRFs)



- Discriminative
- Doesn’t assume that features are independent
- When labeling Y_i future observations are taken into account

➔ The best of both worlds!

Model Trade-offs

	Speed	Discrim vs. Generative	Normalization
HMM	very fast	generative	local
MEMM	mid-range	discriminative	local
CRF	slow	discriminative	global

Practice Question on Beam Search algorithm

Suppose you want to use a MaxEnt tagger to tag the sentence, "the light book". We know that the top 2 POS tags for the words *the*, *light* and *book* are {*Det*, *Noun*}, {*Verb*, *Adj*} and {*Verb*, *Noun*}, respectively. Assume that the MaxEnt model uses the following history h_i (context) for a word w_i:

$$h_i = \{w_i, w_{i-1}, w_{i+1}, t_{i-1}\}$$

where w_{i-1} and w_{i+1} correspond to the previous and next words and t_{i-1} corresponds to the tag of the previous word. Accordingly, the following features are being used by the MaxEnt model:

- $f_1: t_{i-1} = Det$ and $t_i = Adj$
- $f_2: t_{i-1} = Noun$ and $t_i = Verb$
- $f_3: t_{i-1} = Adj$ and $t_i = Noun$
- $f_4: w_{i-1} = the$ and $t_i = Adj$
- $f_5: w_{i-1} = the \& w_{i+1} = book$ and $t_i = Adj$
- $f_6: w_{i-1} = light$ and $t_i = Noun$
- $f_7: w_{i+1} = light$ and $t_i = Det$
- $f_8: w_{i-1} = NULL$ and $t_i = Noun$

Assume that each feature has a uniform weight of 1.0.
Use Beam search algorithm with a beam-size of 2 to identify the highest probability tag sequence for the sentence.

Given that …

- $f_1: t_{i-1} = Det$ and $t_i = Adj$
- $f_2: t_{i-1} = Noun$ and $t_i = Verb$
- $f_3: t_{i-1} = Adj$ and $t_i = Noun$
- $f_4: w_{i-1} = the$ and $t_i = Adj$
- $f_5: w_{i-1} = the \& w_{i+1} = book$ and $t_i = Adj$
- $f_6: w_{i-1} = light$ and $t_i = Noun$
- $f_7: w_{i+1} = light$ and $t_i = Det$
- $f_8: w_{i-1} = NULL$ and $t_i = Noun$

$h_i = \{w_i, w_{i-1}, w_{i+1}, t_{i-1}\}$

Each feature has uniform weight of 1.0
Beam size is 2 to find the highest probability tag sequence

Words and tags

Noun	Adj	Noun
Det	Verb	Verb
the	light	book

$$P(t_i|w_i) = P(t_i \mid (x_i| h_i))$$
$$= P(t_i \mid x_i \mid \{w_i, w_{i-1}, w_{i+1}, t_{i-1}\}$$
$$= \frac{e^{(\sum \lambda_i f_i)}}{Z}$$

MEMM

$$p_\lambda(y|x) = \frac{1}{Z_\lambda(x)} exp\left(\sum_i \lambda_i f_i(x,y)\right)$$

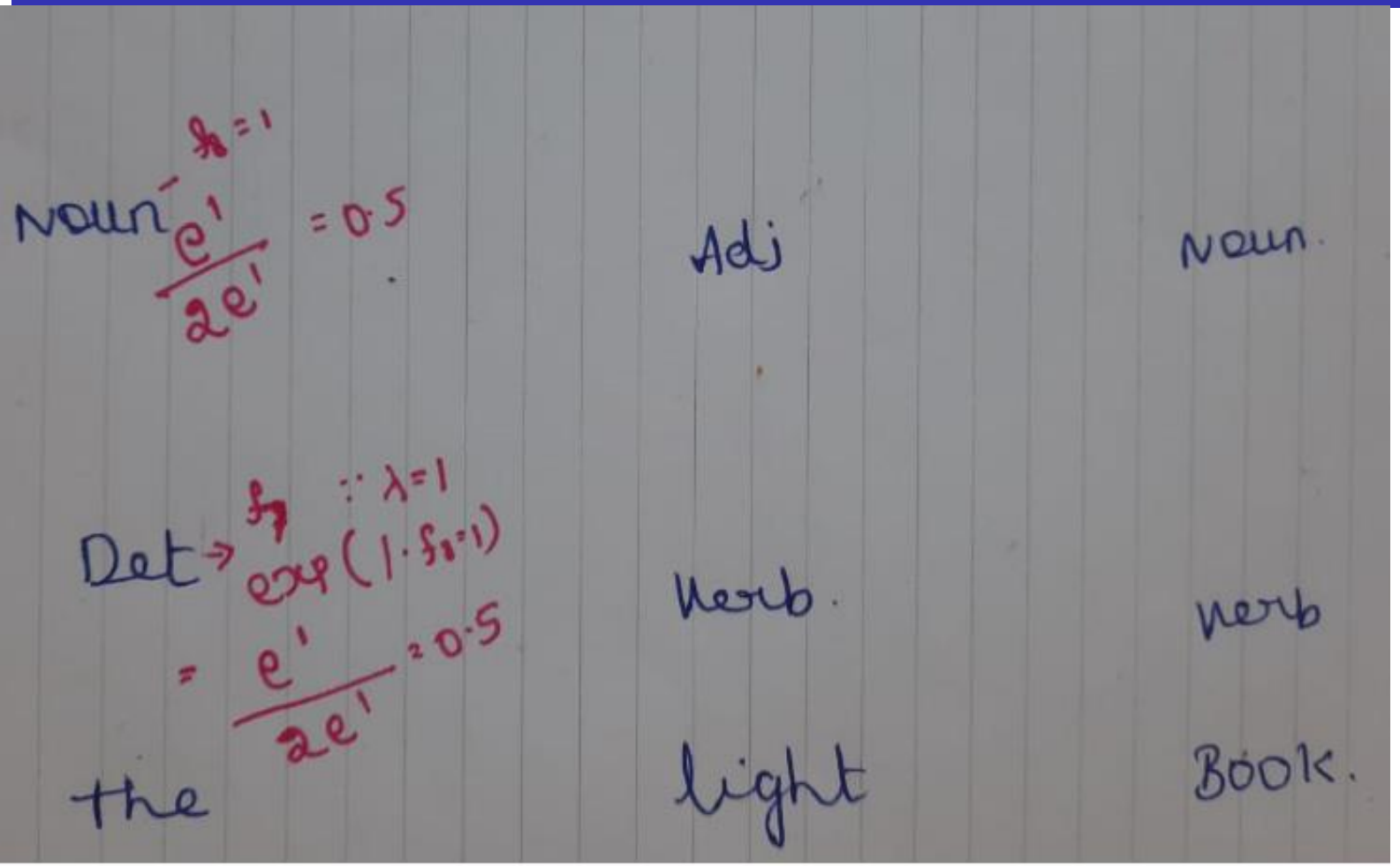
where

- $Z_\lambda(x)$ is a normalizing constant given by

$$Z_\lambda(x) = \sum_y exp\left(\sum_i \lambda_i f_i(x,y)\right)$$

- λ_i is a weight given to a feature f_i
- x denotes an observed datum and y denotes a class

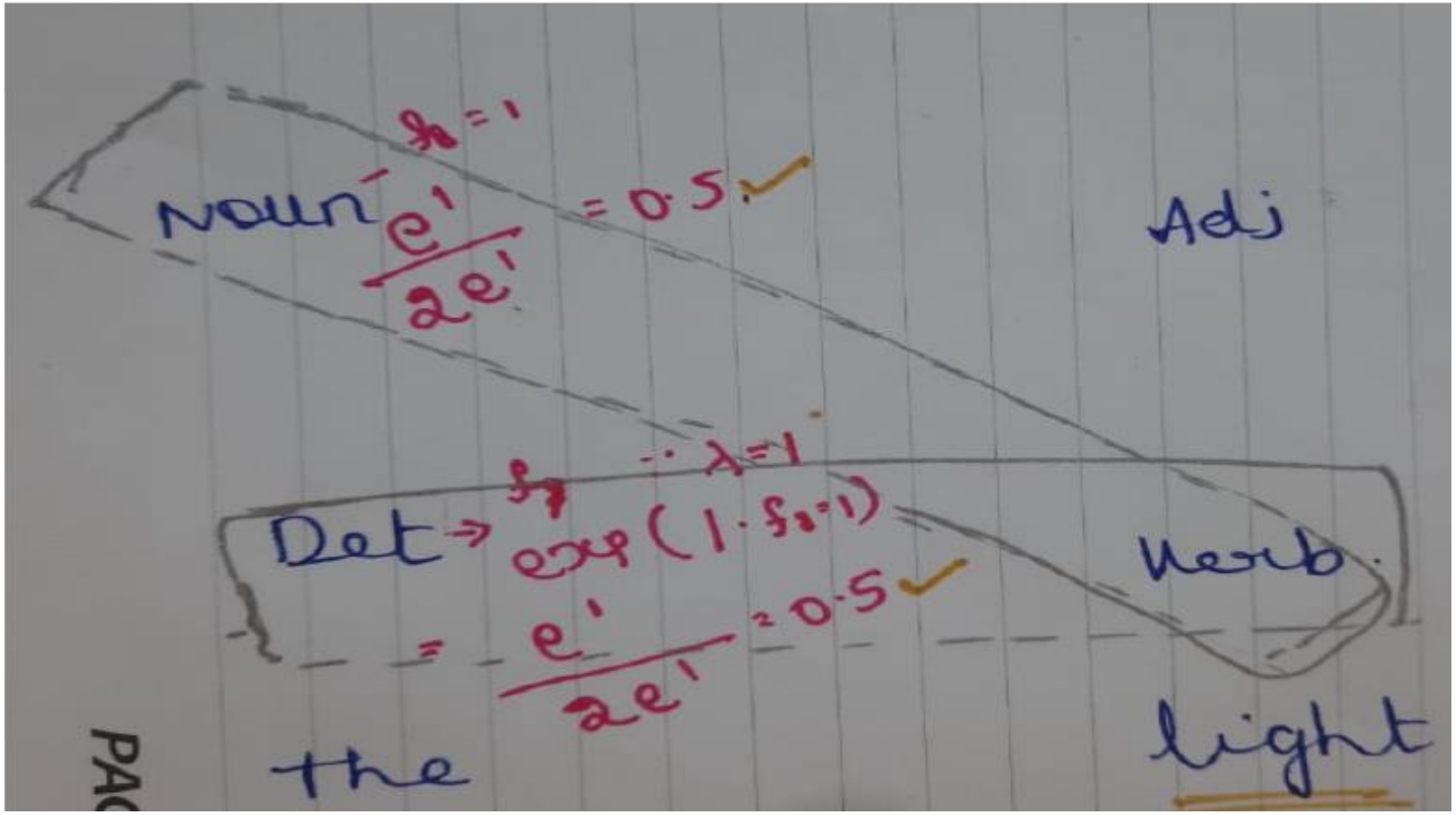
word “the” with tag D and N



- f_1: $t_{i-1} = Det$ and $t_i = Adj$
- f_2: $t_{i-1} = Noun$ and $t_i = Verb$
- f_3: $t_{i-1} = Adj$ and $t_i = Noun$
- f_4: $w_{i-1} = the$ and $t_i = Adj$
- f_5: $w_{i-1} = the \& w_{i+1} = book$ and $t_i = Adj$
- f_6: $w_{i-1} = light$ and $t_i = Noun$
- f_7: $w_{i+1} = light$ and $t_i = Det$
- f_8: $w_{i-1} = NULL$ and $t_i = Noun$

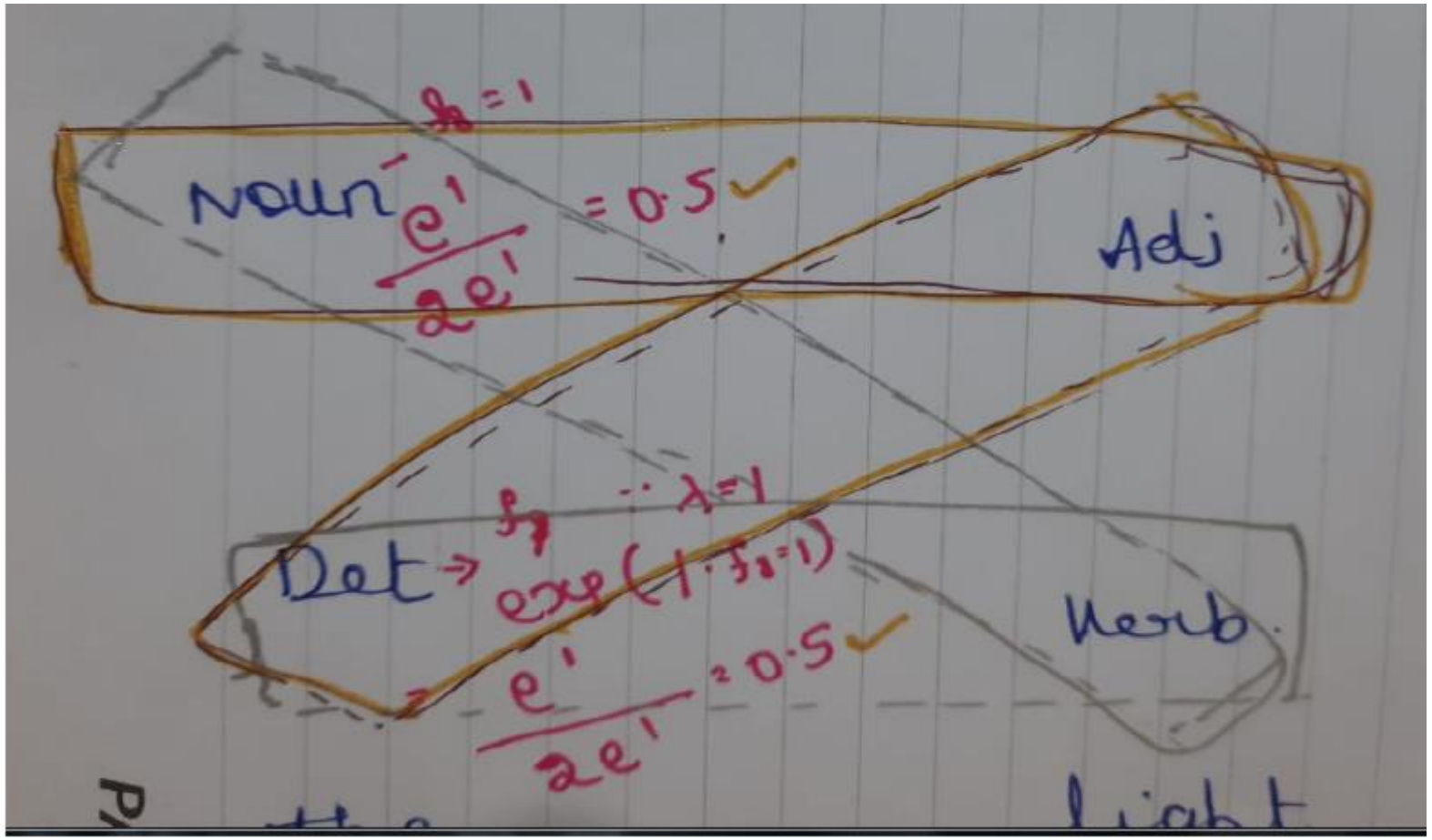
word “light” being V with previous being D and N

Given Beam size as 2



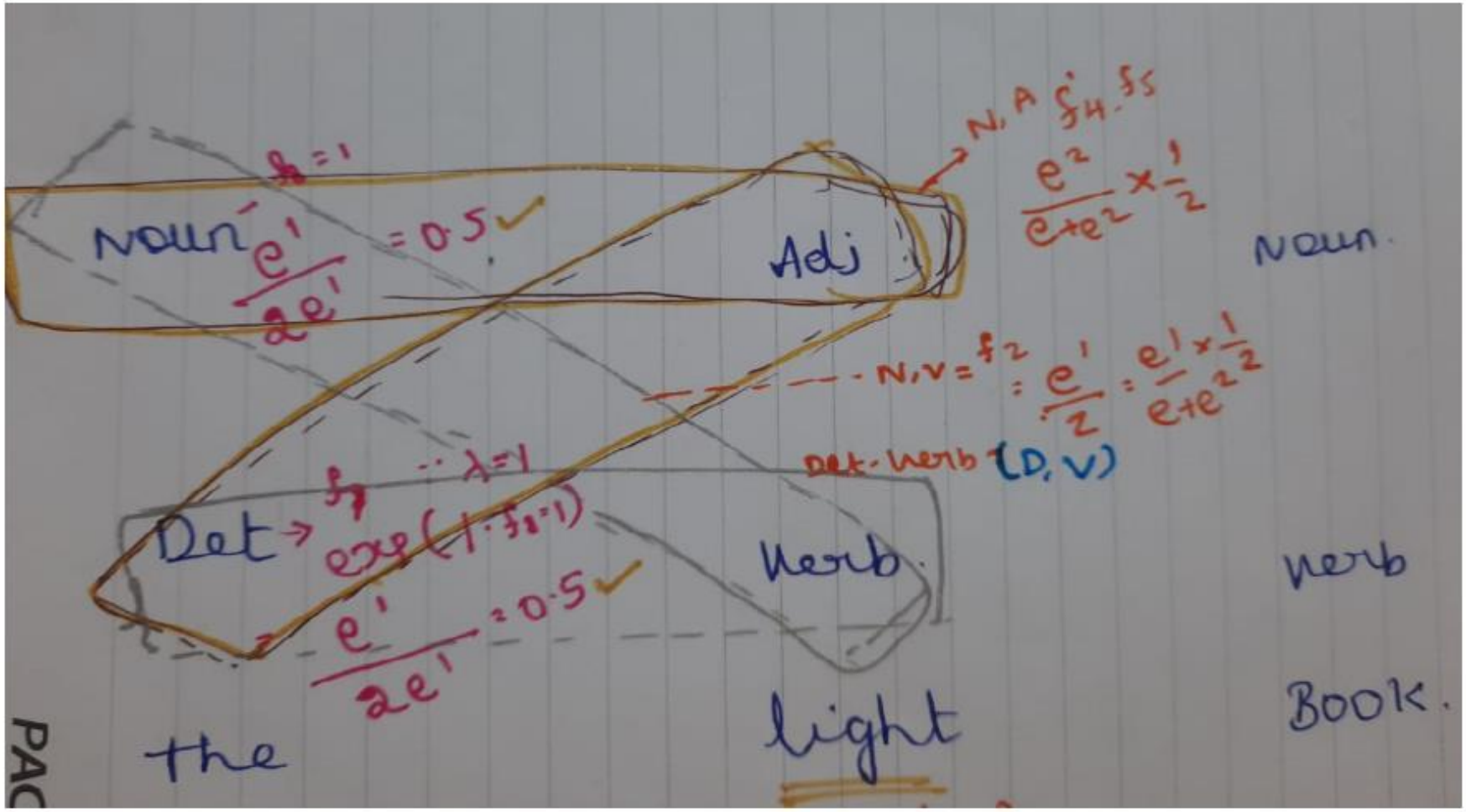
- $f_1: t_{i-1} = Det$ and $t_i = Adj$
- $f_2: t_{i-1} = Noun$ and $t_i = Verb$
- $f_3: t_{i-1} = Adj$ and $t_i = Noun$
- $f_4: w_{i-1} = the$ and $t_i = Adj$
- $f_5: w_{i-1} = the \& w_{i+1} = book$ and $t_i = Adj$
- $f_6: w_{i-1} = light$ and $t_i = Noun$
- $f_7: w_{i+1} = light$ and $t_i = Det$
- $f_8: w_{i-1} = NULL$ and $t_i = Noun$

word “light” being A with previous being D and N

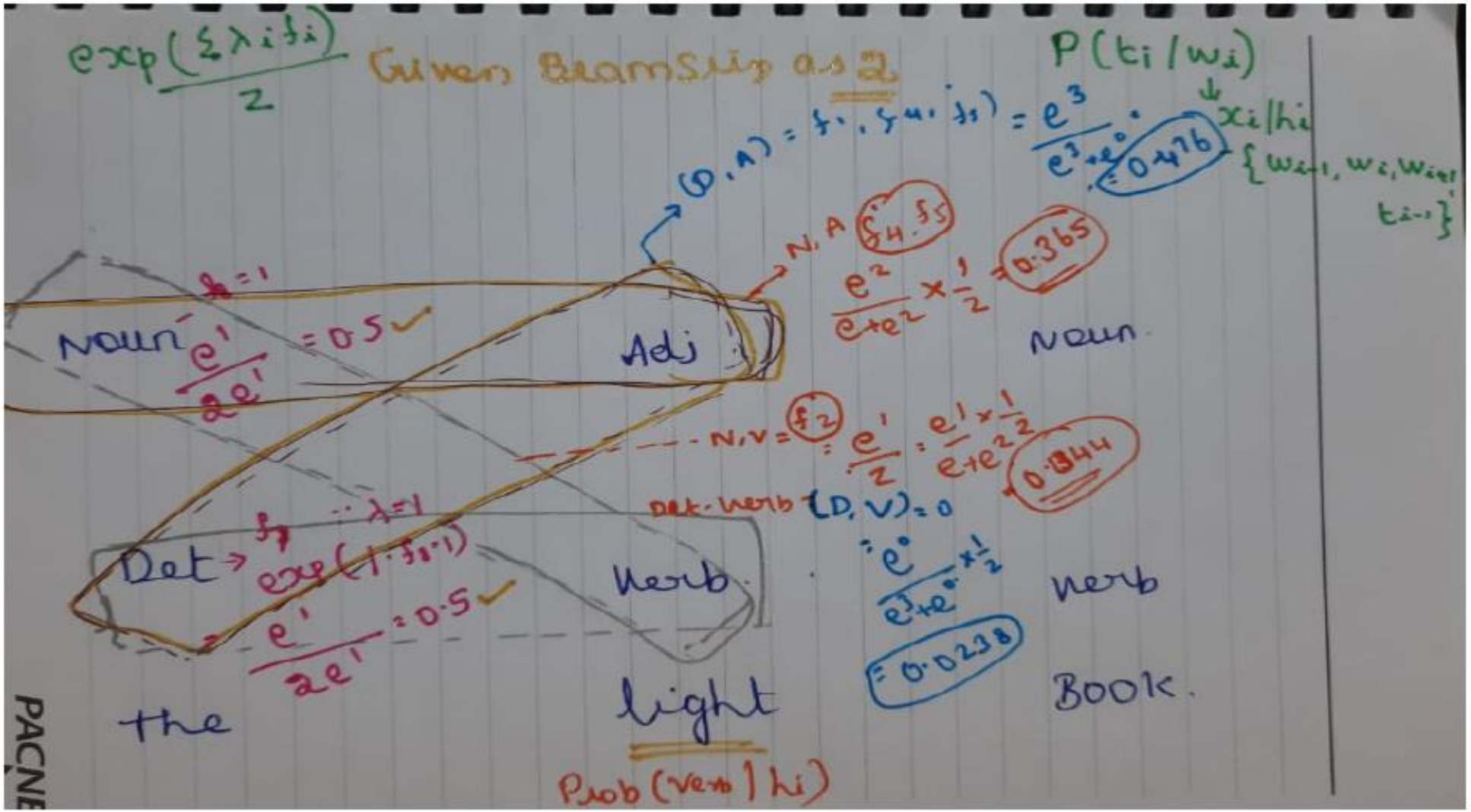


- $f_1: t_{i-1} = Det$ and $t_i = Adj$
- $f_2: t_{i-1} = Noun$ and $t_i = Verb$
- $f_3: t_{i-1} = Adj$ and $t_i = Noun$
- $f_4: w_{i-1} = the$ and $t_i = Adj$
- $f_5: w_{i-1} = the \& w_{i+1} = book$ and $t_i = Adj$
- $f_6: w_{i-1} = light$ and $t_i = Noun$
- $f_7: w_{i+1} = light$ and $t_i = Det$
- $f_8: w_{i-1} = NULL$ and $t_i = Noun$

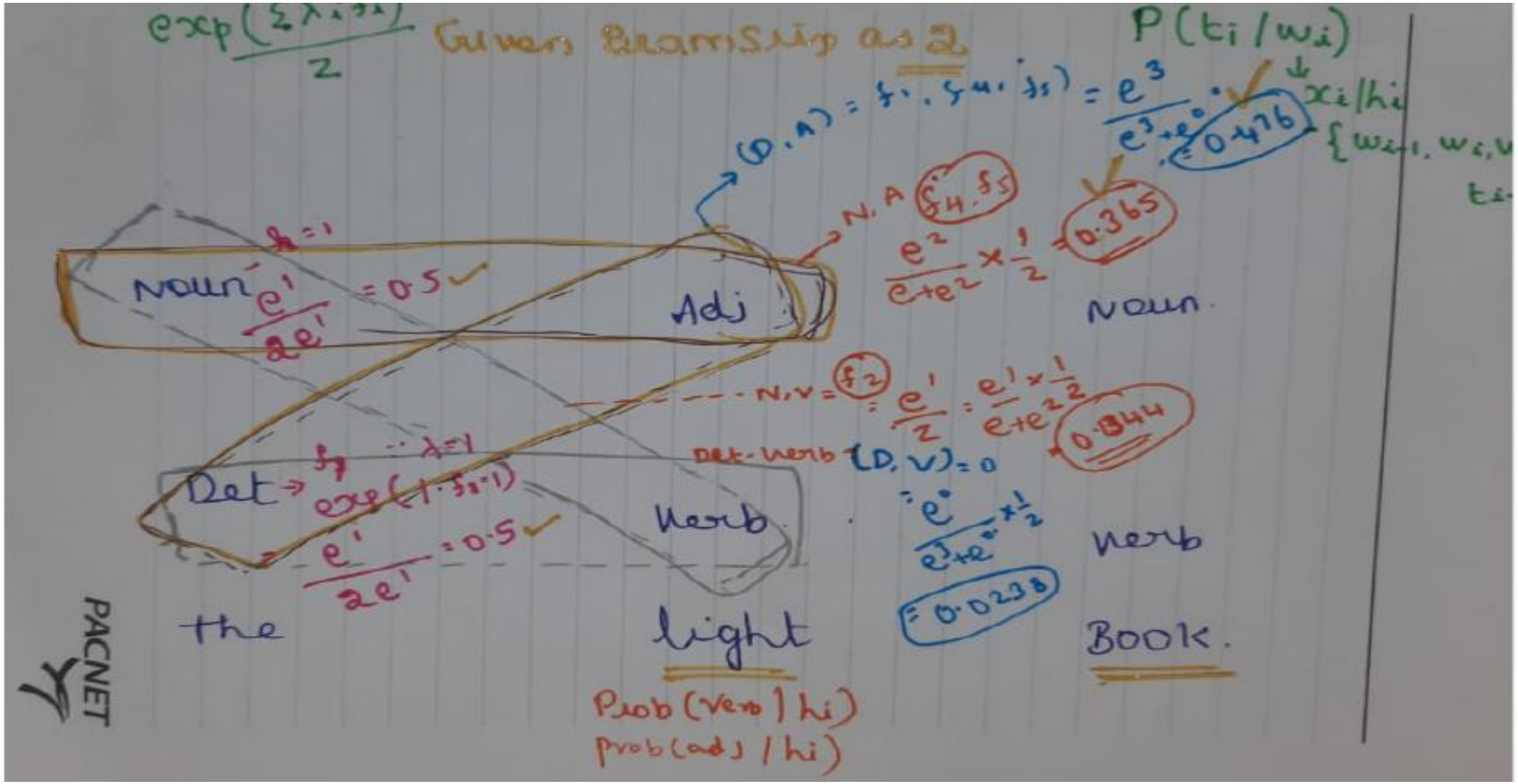
(D, V), (N, V), (D, A) and (N, A)



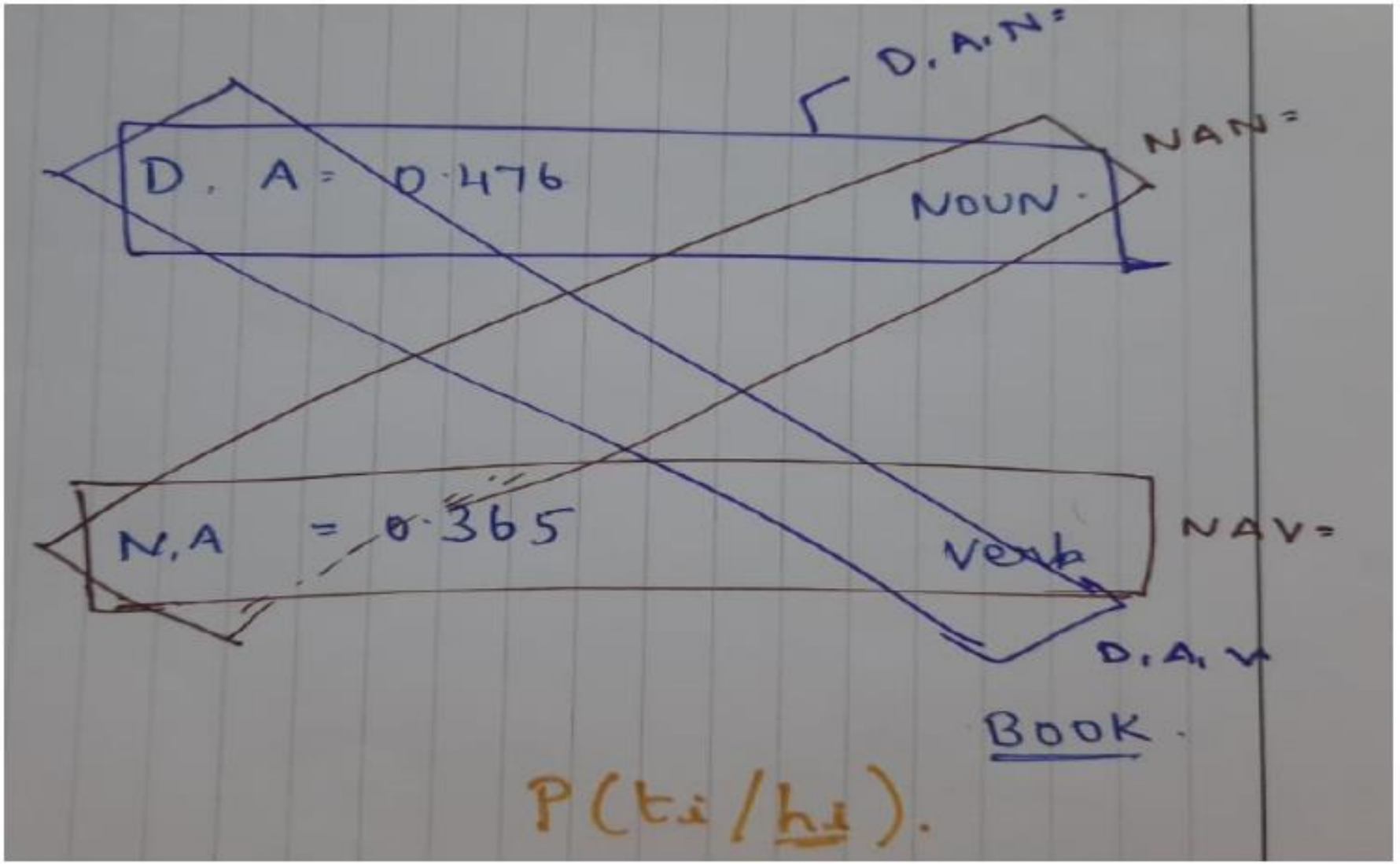
(D, V), (N, V), (D, A) and (N, A)



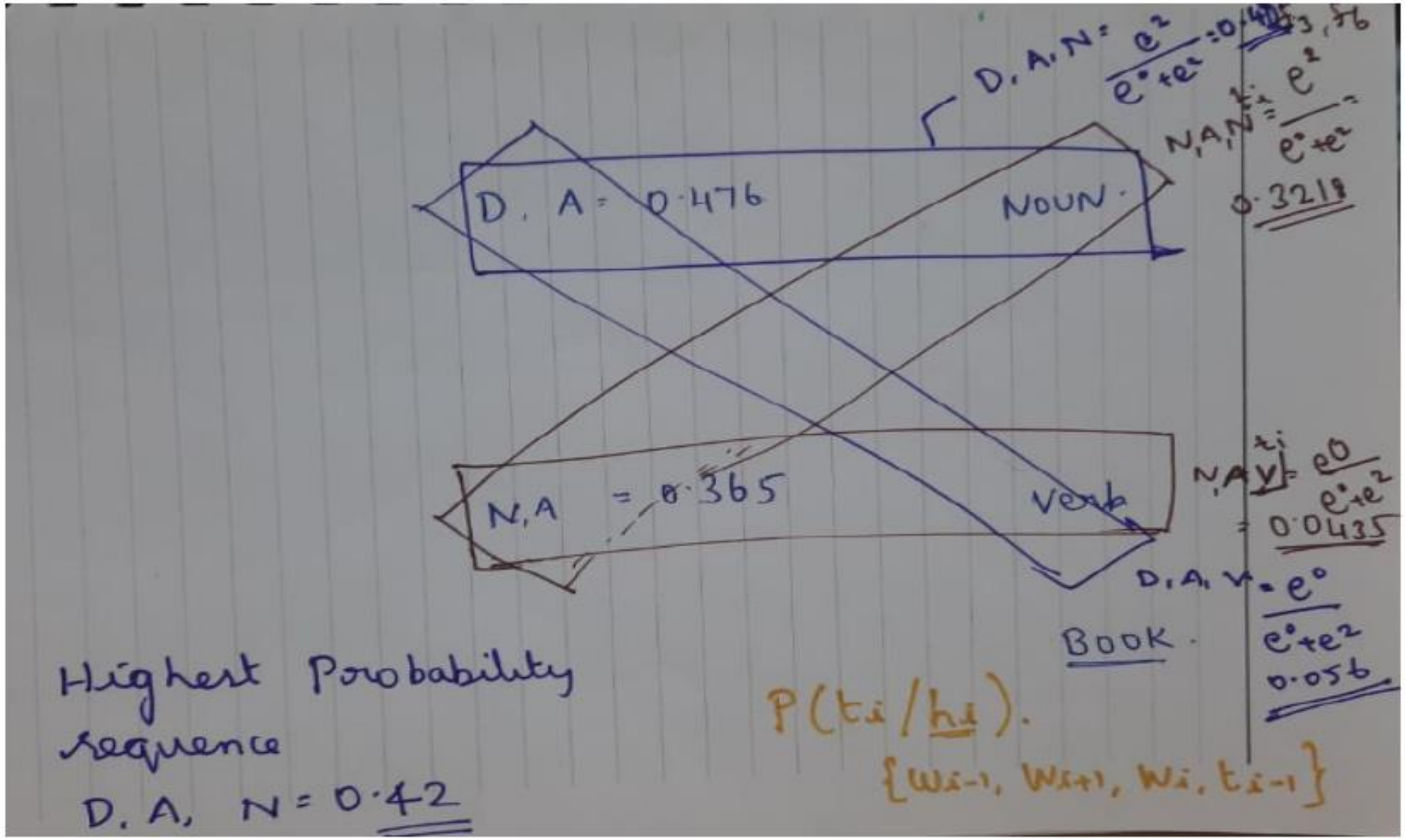
(D, V), (N, V), (D, A) and (N, A)



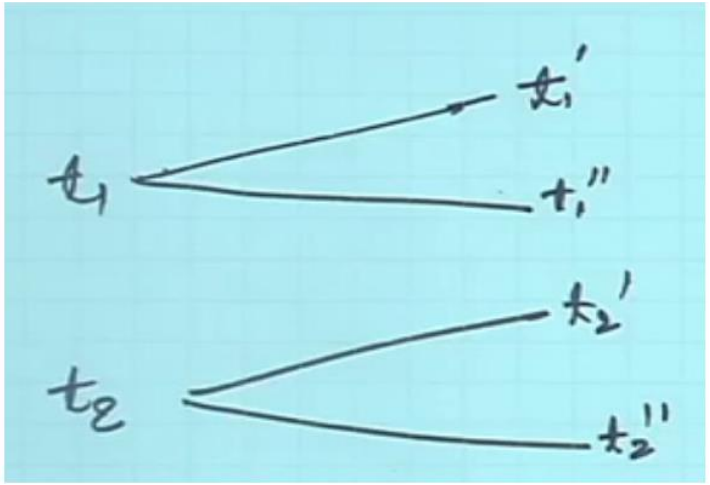
(D, A, N), (N,A,N), (D,A,V) and (N,A,V)

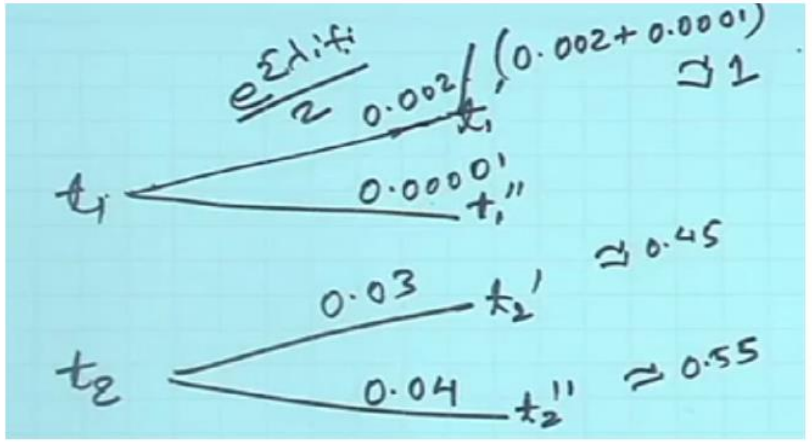


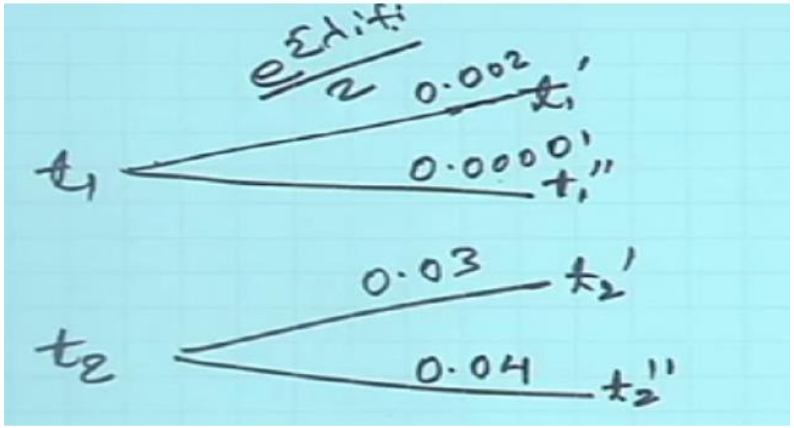
(D, A, N), (N,A,N), (D,A,V) and (N,A,V)

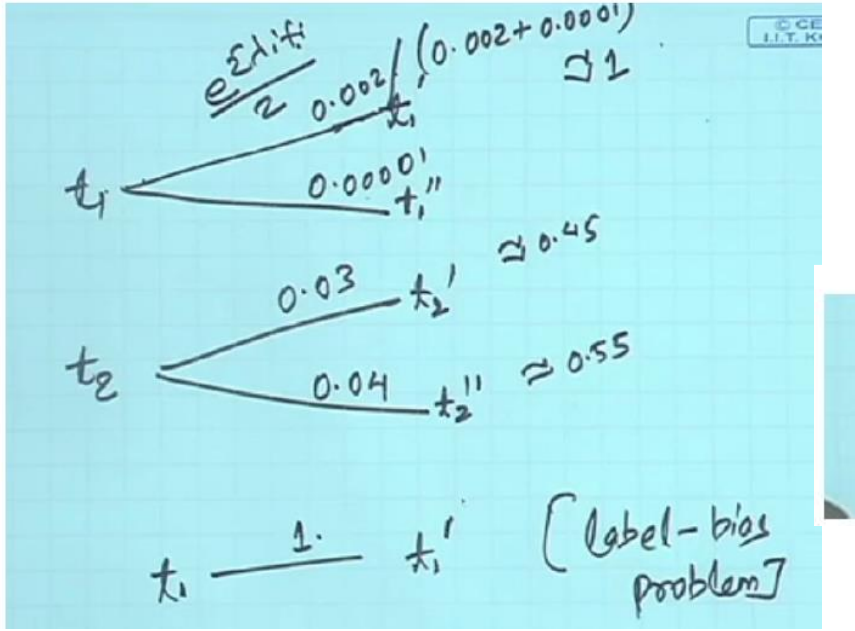


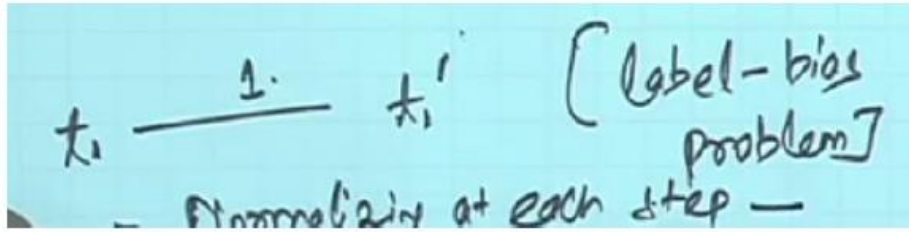
Example of Label-bias problem











Thank You