Natural Language Processing

Regular Expression And Automata

Regex	Match,
/[Ww]oodchuck/	Woodchuck or woodchuck
/[0-9]/	a Single digit
/[a-z]/	a lower case letter
/[A-Z]/	a upper case letter
/[^A-Z]/	not an upper case letter
/[^Ss]/	neither 'S' nor 's'
/[^\.]/	not a period
/[e^]/	either 'e' or '^'
/a^b/	the pattern 'a^b'

Regex	Matches
colour?	Optional previous char
oo*h!	0 or more of previous char.
o+h!	1 or more of previous char
beg.n	any character in place of '.' E.g begun, begin
^[A-Z]	Check whether [A-Z] is at beginning of line
\.$	Check whether '.' is present at end of line
\bthe\b	Check whether 'the' is a word
guppy|ies	guppy and ies
guppy(y|ies)	guppy and guppies
/column[0-9]+ */	Here '*' denotes multiple spaces
/(column[0-9]+ *)*/	E.g column0 column1

Substitution regex

s/colour/color
find replace

s/[0-9]+/<\1> → Replace pattern with pattern surrounded by '<' and '>'.

Eg

E.g 40 is replaced by <40>.

Find Regex of

1) the Xer they were, the Xer they will be

E.g the Bigger they were, the bigger they will be

Find FSA of

1) Eg One cent / forty two cent
2) two dollar ten cent

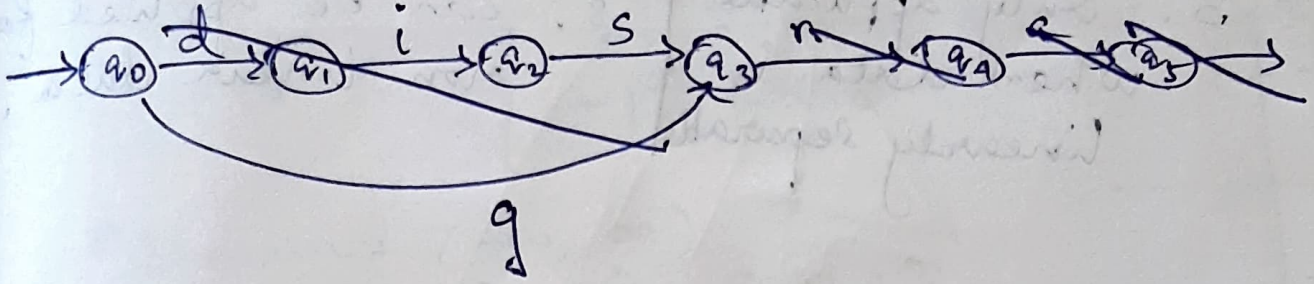
Word classes

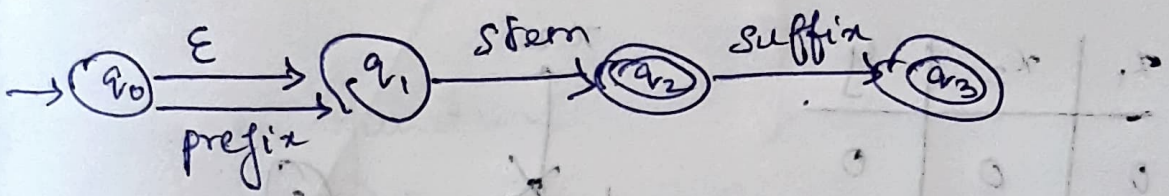
Content words

- Nouns
- Verbs
- Adjectives
- Adverbs

Function words

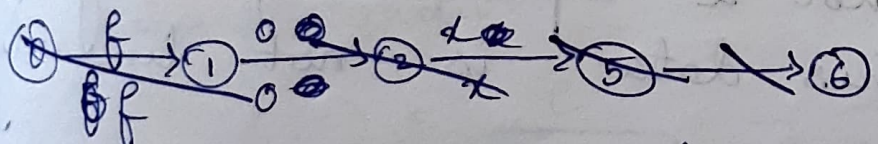
- Prepositions
- Conjunctions
- Determinants





The above automata accepts grace, graceful, disgrace, disgraceful.

Transducer



- It produces internal structure of a word

Natural Language Processing

Intrinsic evaluation of Language models – perplexity

If a model assigns high probability to a word, it is not perplexed (surprised) to see it

$$PP(W) = P(w_1, w_2 \ldots w_N)^{-1/N}$$

$$P(w_1, w_2, \ldots w_N) = \prod_{i=1}^{N} P(w_i \mid w_{i-1}, w_{i-2} \ldots$$

For bigrams

$$P(w_1, w_2, \ldots w_N) = \prod_{i=1}^{N} P(w_i \mid w_{i-1})$$

Test dataset

W = (<s>, This, is, the, first, sentence, ., </s>, <s>, This, is, the, second, one, ., </s>)

N = 16

Unigram

This = 0.04
is = 0.06
the = 0.07
fast = 0.0009
sentence = 0.0008
second = 0.001
one = 0.0012

Problem

Roll a die 10 times

$$PP(T) = \frac{1}{((1/6)^{10})^{1/10}} = 6$$

If we have an unfair die that rolls 6 with probability $7/12$ and others with $1/12$ each

$$PP(T) = \frac{1}{((7/12)^{7} \cdot (1/12)^{5})^{1/12}} = 3.9 \approx 4$$

Laplace smoothing
(also called Add-one estimation)

MLE estimate:

$$P_{MLE}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

Add-1 estimate:

$$P_{Add-1}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1}) + V}$$

Note:

B. After Laplace smoothing, Bigrams with respect to a certain word will have the same probability.

Good Turing Discounting

Calculate Frequency of frequencies for bigrams

$$c^* = (c+1) \cdot \frac{N_{c+1}}{N_c}$$

Backoff Interpolation

- use trigram if probability of all trigrams is available, otherwise move to bigrams and unigrams

Relation between perplexity and Cross Entropy:

$$PP(w) = 2^{H(w)}$$

Absolute discount

$$c^* = c - 0.75$$

Kneser-Ney Smoothing

$$P_{continuation}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

$$P_{continuation}(w) = \frac{|\{w_{i-1} : c(w_{i-1}, w) > 0\}|}{\sum_{w'} |\{w'_{i-1} : c(w'_{i-1}, w') > 0\}|}$$

POS Tagging

Distributional hypothesis

Words that appear in similar contexts have similar representation.

Due to ambiguity (and unknown words), we cannot rely on a dictionary to look up the correct POS tags.

Hidden Markov Model

$$P(y) = P(y_1, \dots y_n)$$

$$P(y_1, \dots, y_n) \approx \prod_{i=1}^{n+1} P(y_i \mid y_{i-1})$$

$$P(x \mid y) = P(x_1 \dots x_n \mid y_1 \dots y_n)$$

$$\approx \prod_{i=1}^{n} P(x_i \mid y_i).$$

$$P(x_t \mid y_t) = \frac{c(x, y)}{c(y)}$$

$$P(y_t \mid y_{t-1}) = \frac{c(y_{t-1}, y_t)}{c(y_{t-1})}$$

Natural Language Processing

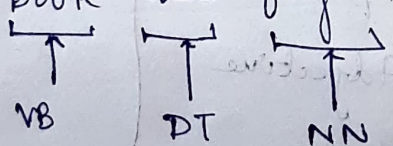
- 3 mostly used tagsets
  1) Penn Tree Bank Tagset (small size - 45 Tagsets)
  2) C5 tagset (medium size - 61 tagsets)
     to tag British National Corpus
  3) C7 target (Large size - 146 tagsets)

Penn Tree Bank POS Tags

Tag	Description	Example
CC	Conjunction	and, but
DT	Determiner	a, the, that
IN	Proposition	of, in, by
JJ	Adjective	
JJR	Adjective comparative	
JJS	Adjective superlative	
NN	Noun - Singular	
NNS	Noun - Plural	
NNP	Proper noun - Singular	
NNPS	Proper noun - Plural	

Tag	Description
PP	Personal pronoun
PP$	Possessive pronoun
RB	Adverb
RBR	Adverb - comparative
RBS	Adverb - superlative
VB	Verb - base form
VBD	Verb - past tense
VBG	Verb gerand
VBN	Verb past participle
VBZ	Verb 3 sg Pres
WDT	wh - determinier
WP	wh - pronoun
WRB	wh - adverb
(	Left Paranthesis [, (, {,

E.g



Assign POS Tags for the following sentence based on Penn Tree Bank Tagset

1) Does that flight serve dinner?
2) This crap game is over a garage in fifty second street
3) I need a flight from India
4) I have a friend daiving in Australia

5) What flight do you have from India to America?

6) Can you list the non stop afternoon flights to England?

Answers

Does that flight serve dinner

PoS tagging algorithms

- 2 classes
  1) Rule based PoS tagging
  2) Stochastic tagging

Rule based
- suitable for large databases
- handwritten unambiguous rules for each word
E.g. specify a word as a noun rather than an adverb if it follows a determiner

Stochastic
- Probability is assigned to each word with a given tag defined in the given context
- Hidden Markov Model (HMM) tagger / Maximum Likelihood tagger

Natural Language Processing

Hidden Markov Model

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \; P(t_1^n \mid w_1^n)$$

t_1^n ↓ tag sequence; w_1^n → word sequence

$$= \underset{t_1^n}{\operatorname{argmax}} \; P(w_1^n \mid t_1^n) \cdot P(t_1^n)$$

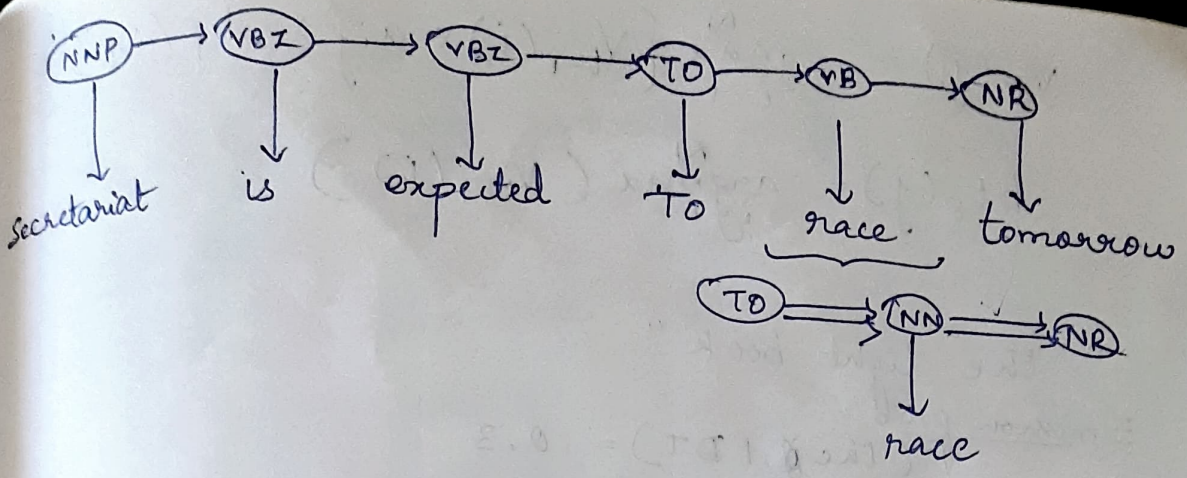
$$P(t_1^n) = \prod_{i=2}^{n} P(t_i \mid t_{i-1})$$ → Prior probability / transition probability

$$P(w_1^n \mid t_1^n) = \prod_{i=1}^{n} P(w_i \mid t_i)$$

also called Observation likelihood / Emission probability

Problem

Secretariat / NNP is / BEZ expected / VBN to / TO race / VB tomorrow / NR



~~Given 'Today is hot' 'tomorrow will be is~~

Given → today = hot

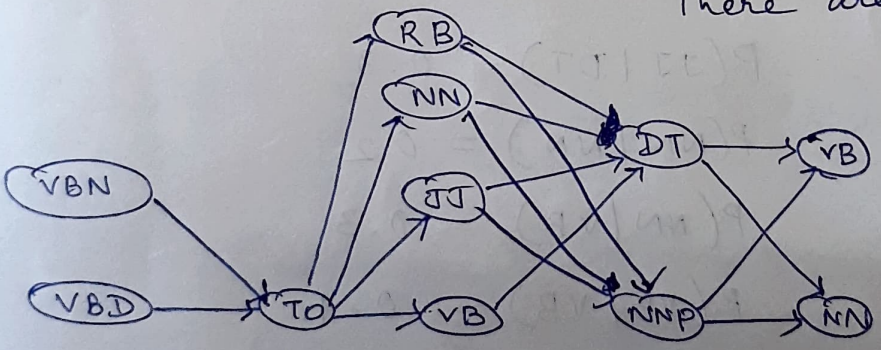
tommorrow = cold

day after = warm

$P(\overset{q_{n+2}}{warm}, \overset{q_{n+1}}{cold} \mid \overset{q_n}{hot}) = P(q_{n+1} \mid q_n) * P(q_{n+2} \mid q_{n+1}, q_n)$

Viterbi Algo

There are 32 paths



promised to back the bill

Step/Time 1 2 3 4 5

$V_{t+1}(K) = max\{ V_t(j) * P(\blacksquare^{K} \mid \blacksquare^{j}) * P(W_{t+1} \mid \blacksquare^{K}) \}$

$bt(j) = \underset{\blacksquare}{argmax}\{ V_t(j) \}$

$$V_t(j) = \max_{i=1}^{N} \{ V_{t-1}(i) * a_{ij} * b_j(O_t) \}$$

$$bt_t(j) = \arg\max_{i=1}^{N} (V_{t-1}(i))$$

" the light book "

Emission prob :

$P(\text{the} \mid DT) = 0.3$

$P(\text{light} \mid JJ) = 0.002$

$P(\text{book} \mid \text{verb}) = 0.01$

$P(\text{the} \mid NN) = 0.1$

$P(\text{light} \mid VB) = 0.06$

$P(\text{book} \mid NN) = 0.003$

$P(\text{light} \mid NN) = 0.003$

Transition prob

$P(VB \mid DT) = 0.00001$

$P(NN \mid DT) = 0.5$

$P(JJ \mid DT) = 0.3$

$P(NN \mid NN) = 0.2$

$P(NN \mid VB) = 0.3$

$P(VB \mid VB) = 0.1$

$P(JJ \mid NN) = 0.002$

$P(VB \mid NN) = 0.3$

$P(NN \mid JJ) = 0.2$

$P(VB \mid JJ) = 0.001$

Note: transition
a_{ij} - ~~emission~~ probability

$b_j(O_t)$ emission probability

Baum-Welch Algo

$X_t \rightarrow$ hidden state $X = (X_1 = x_1, X_2 = x_2, \ldots)$

$Y_t \rightarrow$ observation $Y = (Y_1 = y_1, Y_2 = y_2, \ldots)$

$\theta = (\pi, A, B)$

Forward

$Y_1 = y_1, \; Y_2 = y_2, \ldots \; Y_t = y_t$

$\downarrow$

$X_t = i$

Backward

$Y_{T-1} = y_{T-1}, \; Y_T = y_T$

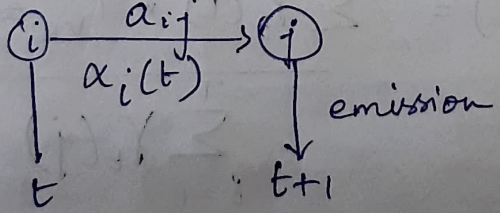
$$\alpha_i(t) = p(y_1, y_2, \ldots y_t, x_t = i \mid \theta)$$

$$B_i(t) = p(y_{t+1}, y_{t+2}, \ldots y_T \mid x_t = i, \theta)$$

Forward probabilities

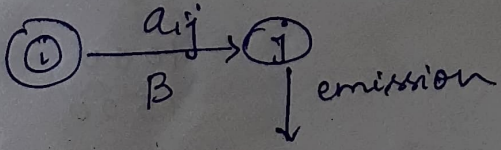
$$\alpha_i(1) = \pi_i * b_i(y_1)$$

$$\alpha_j(t+1) = \sum_{i=1}^{N} \alpha_i(t) \, a_{ij} \, b_j(y_{t+1})$$



Backward probabilities

$$B_i(t) = \sum_{j=1}^{N} B_j(t+1) \, a_{ij} \, b_j(y_{t+1})$$



Path probabilities

$$\gamma_i(t) = \frac{\alpha_i(t) \cdot B_i(t)}{\sum_{j=1}^{N} \alpha_j(t) B_j(t)}$$

If there are 2 states, $i \longrightarrow j$

$y_1 \; y_2 \; y_3 \ldots \; y_t \; y_{t+1} \ldots y_T$

$$S_{ij}(t) = \frac{\alpha_i(t) B_i(t+1) a_{ij} B_j(t+1)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i(t) a_{ij} B_j(t+1) b_j(y_{t+1})}$$

$$\pi = \gamma_i(1)$$

$$\hat{a}_{ij} = \frac{\sum S_{ij}(t)}{\sum_{i=1}^{T}\left(\sum_{j=1}^{N} S_{ij}(t)\right)}$$

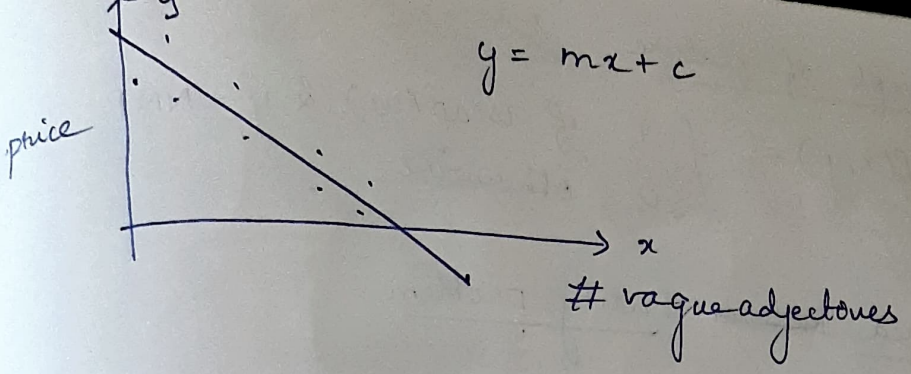
(updated transition probability)

$$b_j(v_k) = \frac{\sum_{t=1}^{T} \gamma_t(1) * 1_{v_k = y_t}}{\sum_{t=1}^{T} \gamma_t(j)} = \begin{cases} 1 & \text{if } v_k = y_t \\ 0, & \text{otherwise} \end{cases}$$

Linear Regression

* Predicting house prices using description

# Vague adjectives	Amount
4	0
3	10000
2	15000
2	60000
1	140000
0	180000



Instead of m and c, we use weights,

$$price = w_0 + w_1 (\#\ vague\ adjectives)$$

In general

$$\hat{y} = \sum_{i=1}^{n} w_i f_i \quad [f_i - i\text{th feature}]$$

We have to minimize cost function.

$$cost(w) = \sum_{j=0}^{m} (\hat{y}_{pred} - y_{actual})^2$$

We can obtain weights using gradient descent or directly using the bellow formula

$$W = (X^T X)^{-1} X^T \vec{y}$$

Issues with HMM

(i) Unknown
Solution: Morphological cues

(ii) Limited context
Solution: increase the window size upto which probability is calculated

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

Normalizing — weight for ith feature — ith feature — Observation label

Example of feature

$$f(x,y) = \begin{cases} 1 & \text{if } \text{uscap}(w) \ \& \ y = \text{NNP} \\ 0 & \text{otherwise} \end{cases}$$

Named ~~Intent~~ Entity problem

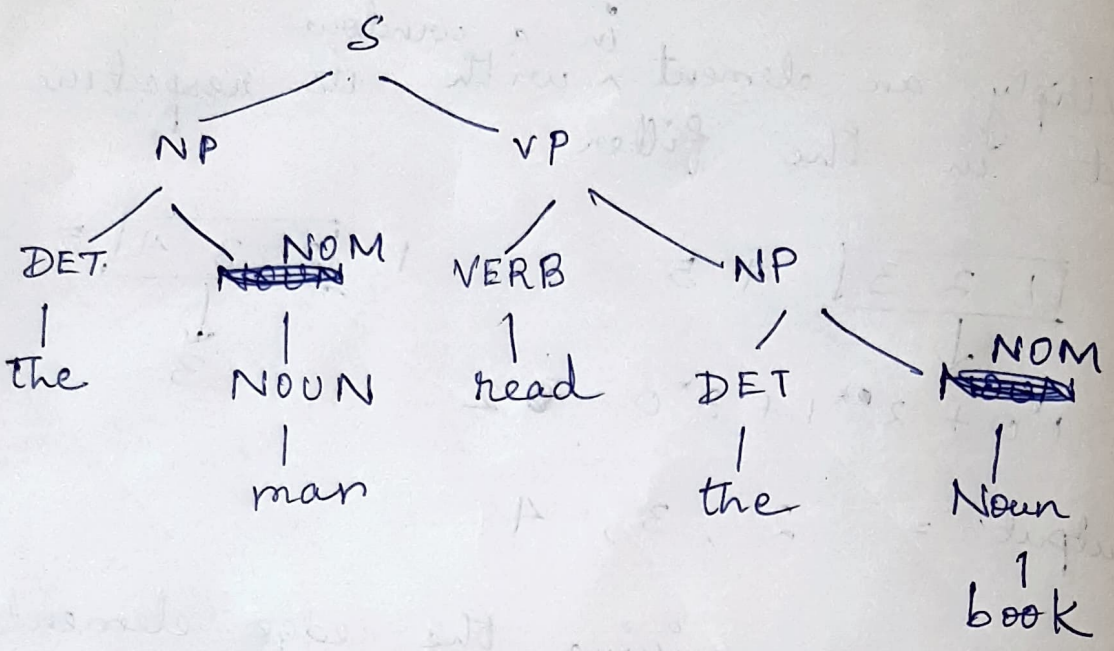
	Location,	Drug,	Person
	f_1	f_2	f_3
1) in Arcadia	1	0	0
2) in Quebec	1	1	0
3) taking Zantac			
4) saw Sue			

We construct 3 features

$$f_1(x,y) = [y = \text{location} \wedge w_{-1} = \text{"in"} \wedge \text{uscap}(w)]$$

$$f_2(x,y) = [y = \text{location} \wedge \text{has\_central\_latin\_char}(w)]$$

$$f_3(x,y) = [y = \text{Drug} \wedge \text{end}(w, c)]$$



NP (Noun Phrases)

{Kermit the frog} comes on stage
{They} come to Massachusetts every summer
{December twenty sixth} comes after christmas

PP = On dec twenty-sixth

{On dec 26th} I 'd like to fly to Florida
I'd like to fly to Florida {on dec 26th}

The group of words ~~which~~ that ~~occur commonly enough that they~~ can be interchanged with one another is called Constituent. They always form meaningful sentences.

NP → proper noun | Det NOM

NOM → noun | noun NOM

Using a Context Free Grammar we can construct a syntax tree from a given sentence.

CFG (T, N, S, R)

- T → Terminals (Words in Language)
- N → Pre-terminal (POS tags)
- R → Rules

Each rule will be of the form

$X \rightarrow Y$

$Y \in NUT$

E.g

NP → Det NOM

NP → Proper Noun

NOM → Noun | Noun Nominal

Det → a | the

Noun → flight

Problem

Derive "a flight"

NP → Det NOM

Det → a

NOM → Noun

Noun → flight

Example 2

S → NP VP

S → AUX NP VP

S → VP

NP → Det NOM

NOM → Noun

NOM → Noun NOM

VP → verb

VP → verb NP

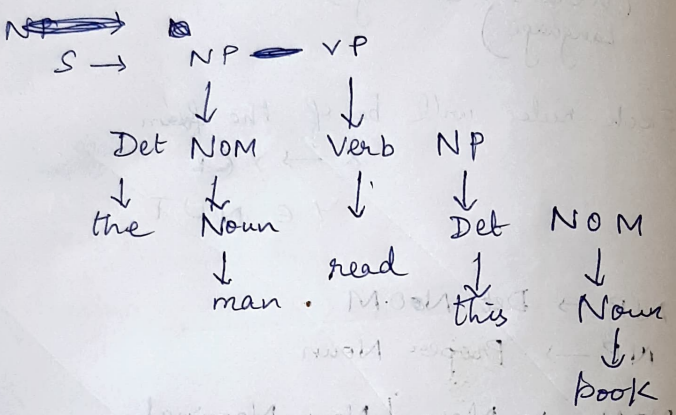
Det → that | this | a | the

Noun → book | flight | meal | man

Verb → book | read

AUX → does

Derive "The man read this book"

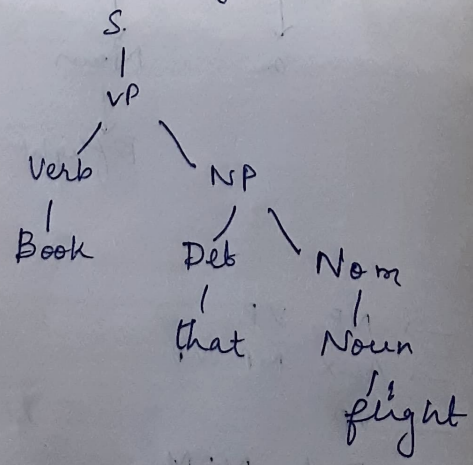


Adding new rules

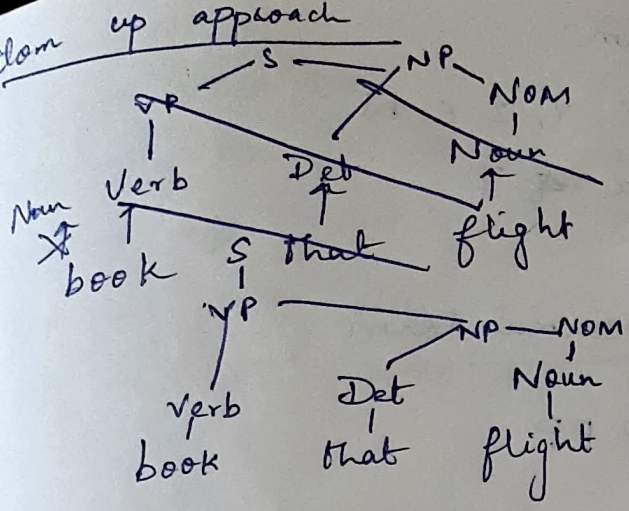
VP → VP PP

PP → prep NP

Derive "Book that flight"



Bottom up approach

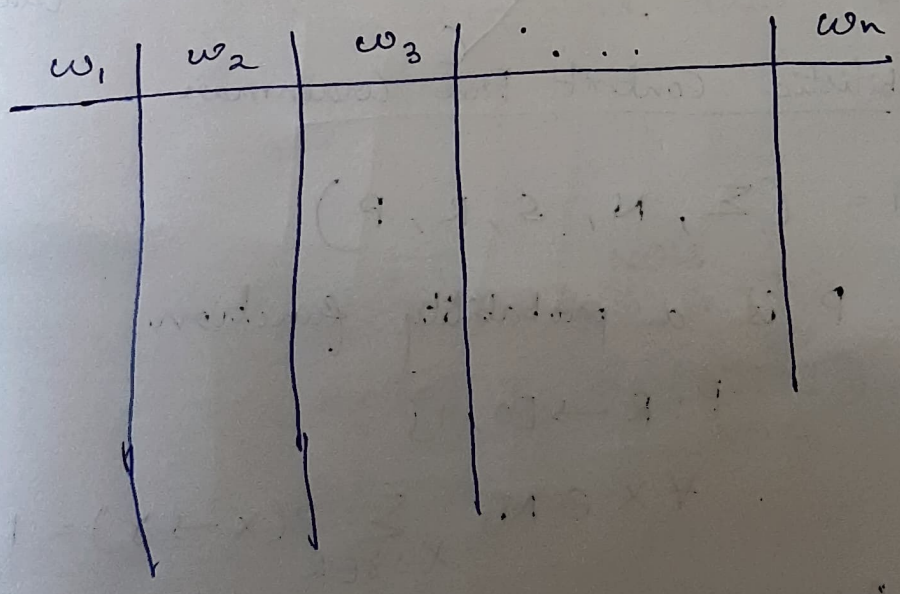


Since lot of searches are made to find which rule is correct, we make use of Dynamic Programming.

CKY algorithm

- uses dynamic programming
- requires grammar to be present in Chomsky Normal form

n = # words



Note:

Group of words which behave as a single unit.

Chomsky Normal Form

$X \rightarrow YZ$, $X, Y, Z \in N$

$X \rightarrow \alpha$, $\alpha \in T$

Convert to Chomsky Normal Form

VP → V NP PP

NP → N

NP → e

VP → V PP | V N PP

VP → V N PP

or VP → V N P NP

or VP → ~~V (people | fish | tanks~~

V N P | V N P N

VP → V N P

or VP → (people | fish | tanks) (people | fish | tanks | rods)

Probabilistic Context Free Grammar

$G = (\Sigma, N, S, R, P)$

P is a probability function

$P : R \rightarrow [0, 1]$

$\forall X \in N, \sum_{X \rightarrow \gamma \in R} P(X \rightarrow \gamma) = 1$

A grammar G generates a language model L.

$\sum_{\gamma \in T^*} P(\gamma) = 1$

$S \rightarrow NP\ VP$
$VP \rightarrow VBG\ NNS$
$VP \rightarrow VBZ\ VP$
$VP \rightarrow VBZ\ NP$
$NP \rightarrow DT\ NN$
$NP \rightarrow JJ\ NNS$

$DT \rightarrow a$
$NN \rightarrow pilot$
$VBZ \rightarrow likes$
$VBG \rightarrow flying$
$JJ \rightarrow flying$
$NNS \rightarrow planes$

a	pilot	likes	flying	planes
DT	NP	—	—	$S_1 \rightarrow NP\ VP_1$ $S_2 \rightarrow NP\ VP_2$
	NN	—	—	—
		VBZ	—	VP_1 VP_2
			VBG JJ	VP NP
				NNS

Note:

$dp[i][j]$ – Non terminals that derive words from ith to jth place

$dp[0][2] = (dp[0][0], dp[1][2])$
$(dp[0][1], dp[2][2])$
DT NULL
NP VBZ

$dp[2][4] = (dp[2][2], dp[3][4])$
$dp[2][3], dp[4][4])$
VBZ

Problem

$S \rightarrow NP\ VP$	0.9	
$S \rightarrow VP$	0.1	
$VP \rightarrow V\ NP$	0.5	
$VP \rightarrow V$	0.1	
$VP \rightarrow V\ @VP\_V'$	0.3	
$VP \rightarrow V\ PP$	0.1	
$VP\_V' \rightarrow NP\ PP$	1.0	
$NP \rightarrow NP\ NP$	0.1	
$NP \rightarrow NP\ PP$	0.2	
$NP \rightarrow N$	0.7	
$PP \rightarrow P\ NP$	1.0	

$N \rightarrow$ people	0.5
$N \rightarrow$ fish	0.2
$N \rightarrow$ tanks	0.2
$N \rightarrow$ rods	0.1
$V \rightarrow$ people	0.1
$V \rightarrow$ fish	0.6
$V \rightarrow$ tanks	0.3
$P \rightarrow$ with	1.0