Natural Language Processing and Applications

Unit 2

Word Level and Syntactic Analysis

Word Level Analysis: word classes - part of speech tagging: HMM POS tagging; Named Entities (NE) - Conditional Random Field NE recognizers; syntactic analysis: constituency - context-free grammar - grammar rules - treebanks; parsing: top-down - bottom-up - ambiguity - CYK parsing - shallow parsing - dependency parsing

* Word Classes

→ Word classes are used in POS tagging. This gives a significant amount of information about the word and its neighbors.

→ There are 2 broad classes - (i) Closed class
(ii) Open class

A. Closed Class - have relatively fixed membership
- mostly grammatical words to deal with the formation of sentences
- show the structural relationships words have with one another.

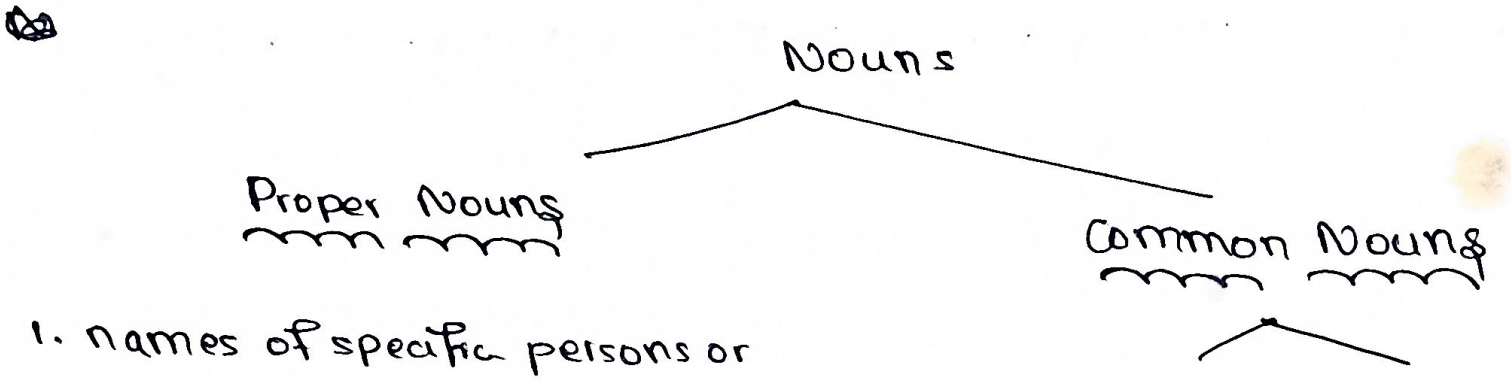
B. Open Class - lexical words that deal with content and vocabulary
- They have a concrete meaning that go beyond their function in a sentence.

- includes nouns, verbs, adjectives and adverbs

* Open Class

A. **Noun** - words for people, places and things

- nouns can be (i) concrete terms
  (ii) abstractions
- can occur with determiners (a goat)
- can take possessiveness (Pooja's cat)
- can take the plural form



1. names of specific persons or entities
2. not preceded by articles
3. usually capitalized

Can be enumerated (two goats)

a homogenous group (snow, salt)

B. **Verbs** → refers to actions & processes

→ can take a number of morphological forms

(i) non-3rd-person singular = eat

(ii) 3rd person singular = eats

(iii) Progressive = eating

(iv) Past participle = eaten

C. **Adjectives**

→ terms describing properties or qualities

→ can modify nouns (large elephant, yellow cab, pale face)

D. **Adverbs**

→ words that modify verbs. There are 4 kinds of adverbs:

(i) Directional : specify direction / location of some action
eg. downhill

(ii) Degree - specify the extent of some action
eg. extremely, very, somewhat

(iii) Manner - describe the manner of some action / process
eg. slowly, delicately

(iv) Temporal adverbs - describe the time that some action took place
eg. yesterday, now, daily

* Closed Classes

A. **Prepositions** → indicate spatial or temporal relations
eg. on, in, to, from, by

B. **Particles** → A particle is a word that resembles a preposition or an adverb.

→ It is often combined with a verb to form a larger unit called a phrasal verb. eg. " I went on ... "
" To throw off sleep "

C. **Articles** → used to begin a noun phrase

a, an ⇒ indefinite noun phrase

the ⇒ definite

D. **Conjunctions** → used to join two phrases, clauses or sentences

(i) Coordinating conjunctions : inaude and, or, but to join two elements of equal status.

(ii) Subordinating conjunctions : when one of the elements is of an embedded status. " I thought that you might like some milk "

E. **Pronouns** → act as a shorthand for referring to some noun phrase or entity or event

(i) Personal Pronouns : persons or entities (you, she, I, it, me)

(ii) Possessive Pronouns : a form of personal pronouns indicating actual possos or just an abstract between the person and some objects (my, your, his, her, its, one's, our)

(iii) con- pronouns - used in question forms ('what, who, whom, whoever)

F. **Auxiliary Verbs** → marks certain semantic features of a main verb, including (i) whether the action is in the past, present or future
(ii) whether it is completed
(iii) whether it is completed

eg. be, can, may, must, will, should

Other closed classes

1. Interjections - oh, an, hey, phew, alas
2. Negatives - no, not
3. politeness markers
4. greetings
5. ~~be. eee~~

* **POS Tagging** - The process of assigning a part-of-speech or other lexical class marker to each word in a corpus.

- Some tagsets in English for POS tagging are : (i) Brown Corpus
  (ii) Penn Treebank
  (iii) C5 Tagset
  (iv) C7 Tagset

* **Tagsets for English**

IN → preposition
EX → Existential there
JJ → adjective
JJR → comparative adjective
JJS → superlative

NN → noun
NNS → noun, plural

NNP → proper noun
NNPS → proper noun, plural

PP → personal pronoun
PP$ → possessive pronoun
DT → determiner → that, the articles (a, an)
RB → adverb
RBR → adverb - comparative
RBS → adverb - superlative

RP → particle (up, off)
MD → can, could, should, modal
VB → base verb
VBD → verb, past tense
VBG → gerund
VBN → past participle
VBP → verb - non3sg
VBZ - verb - 3sg

Example - Tag the following sentence using the Penn Treebank

1. Book that flight

Book | VB that | DT flight | NN

2. The grand jury commented on a number of other topics.

The | DT grand | JJ jury | NN commented | VBD on | IN

a | DT number | NN of | IN other | JJ topics | NNS

3. There are 70 children there

There | EX are | VBP 70 | CD children | NNS there | RB

cardinal no — adverb

* Tagging

→ refers to the process of assigning a part-of-speech or other lexical marker to each word in a corpus

→ It is also applied to punctuation marks.

→ Tagging for NL is the same process as tokenization for computer language, though the tags for NL are much more ambiguous

→ Taggers are used in speech recognition, NL parsing and information retrieval.

→ Automatically assigning a tag to a word is not trivial.

eg. book = verb/noun

that = determiner / complementizer

→ The problem of POS tagging is to resolve these ambiguities, choosing the proper tag for the context.

→ Many of the ambiguous tokens are easy to disambiguate since the various tags associated w/ words are not equally likely.

* POS Tagging Algorithms

→ Tagging algorithms fall into 2 classes:

A. **Rule-based Taggers** - involves a large database of hand-written disambiguation rule specifying. eg. ENGTWOL Tagger

B. **Stochastic Tagger** - resolve tagging ambiguities by using a training corpus to count the probability of a given word having a given tag in a given context.

* **Brill Tagger** - has features of both tagging architectures

Example: Find one tagging error in each of the following sentences that are tagged with the Penn Treebank tagset

1. I / PRP need / VBP a / DT flight / NN from / IN Atlanta / NN

PRP → Pronoun (personal)

VBP → verb non-3rd sg

DT → determiner

NN → noun

IN → prep

NN → error — should be NNP — proper noun

2. Does / VBZ this / DT flight / NN serve / VB dinner / NNS

VBZ → verb

DT → determiner

NN → noun

VB → verb

NNS → error — should be NN noun singular.

3. I / PRP have / VB a / DT friend / NN living / VBG in / IN Denver / NNP

PRP → personal pronoun

VB → verb

DT → article

NN → common noun

VBG → verb gerund

IN → preposition

NNP → noun - proper

VB → error - should be VBP verb (plural)

4. Can / VBP you / PRP list / VB the / DT nonstop / JJ afternoon / NN flights / NNS ?

PRP → personal pro

VB → verb

DT → determiner

JJ → adjective

NN → noun

NNS → plural noun

VBP → error : can, could, should = modal

* Hidden Markov Model (HMM) Tagging

→ HMM is a special case of Bayesian inference.

→ Given a sequence of words $w_1 \ldots w_n$, in a sentence, the objective is to find the best sequence of tags which corresponds to that sequence.

→ With the probabilistic/Bayesian view: consider all possible sequences of tags. Out of this sequence choose the most probable one.

→ Let $T = t_1, t_2 \ldots t_n$

$$W = w_1 w_2 \ldots w_n$$

Out of all the sequence of tags $t_1 \ldots t_n$

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

By Baye's rule:

$$P(x|y) = \frac{P(y|x)\,P(x)}{P(y)}$$

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

the dr. is not needed, as it does not change across tag sequences

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}}\, \overbrace{P(w_1^n \mid t_1^n)}^{\text{likelihood}}\, \overbrace{P(t_1^n)}^{\text{prior}}$$

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

The probability of a word appearing depends only on its own POS tag.

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

Biaram assumption

The probability of a tag appearing depends only on the previous tag

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i \mid t_{i-1})$$

* Two Kinds of Probabilities

① Tag Transition Probabilities - $P(t_i \mid t_{i-1})$

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

② Word Likelihood Probability = $P(w_i \mid t_i)$

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

* Viterbi Algorithm

(8) Using the HMM model for POS tagging on the following corpus.

1. Mary Jane can see will.

2. Spot will see Mary

3. Will Jane spot Mary?

4. Mary will pat Spot.

Step 1 : Assign POs to each word in all the sentences

1. Mary Jane can see will
   N N Modal V N

2. Spot will see Mary.
   N Modal V N

3. Will Jane spot Mary
   M N V N

4. Mary will pat Spot
   N Modal V N

Step 2 : Compute the emission probabilities.

	N	M	V
Mary	4	0	0
Jane	2	0	0
will	1	3	0
spot	2	0	1
can	0	1	0
see	0	0	2
Pat	0	0	1

⇓.

(-X) see vertically

	N	m	V.
Mary	4/9	0	0
Jane	2/9	0	0
will	1/9	3/4	0
spot	0	0	1/4
can	0	1/4	0
see	0	0	1/2
Pat	0	0	1/4

Step 3: Compute the transition probability

(i) <S> N N M V N <E>
Mary Jane can see will

(ii) <S> N M V N <E>
Spot will soe Mary

(iii) <S> M N V N <E>
Will Jane spot Mary

(iv) <S> N M V N <E>
Mary will pat Spot

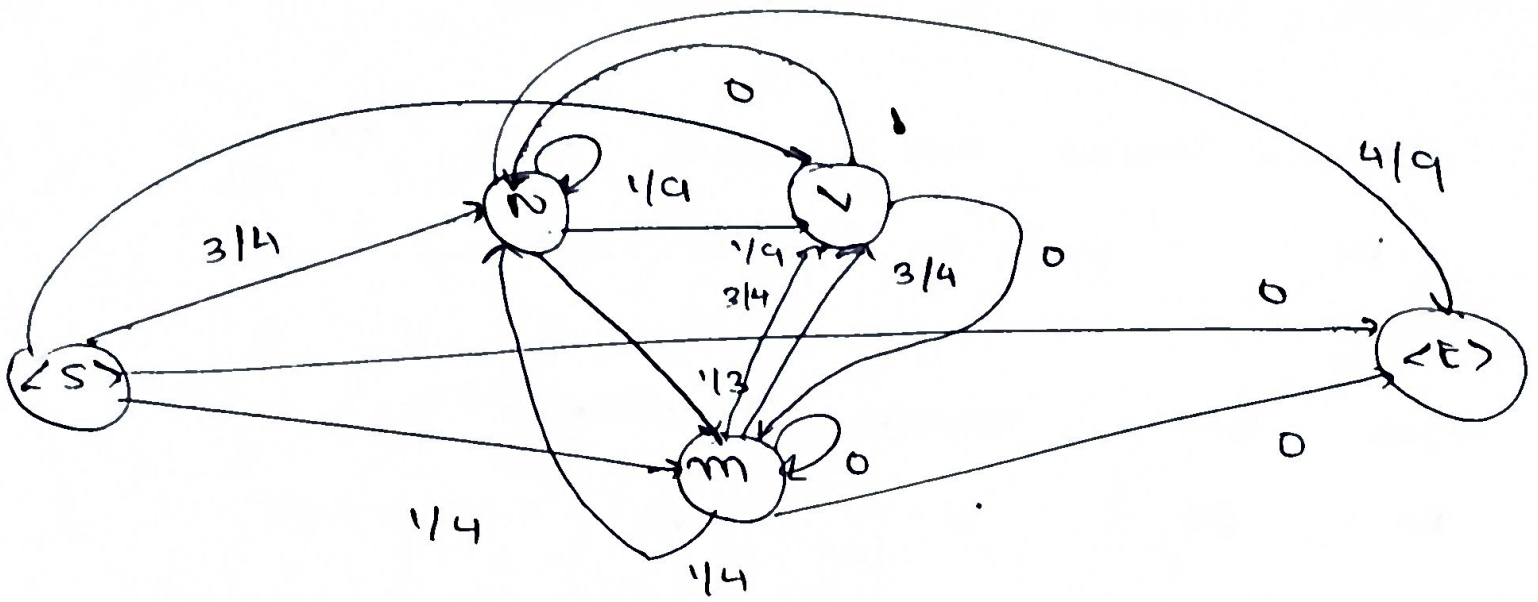
	N	M	V	<E>
<S>	3	1	0	0
N	1	3	1	4
M	1	0	3	0
V	34	0	0	0

⇓

	N	M	V	<E>
<S>	3/4	1/4	0	0
N	1/9	1/3	1/9	4/9
M	1/4	0	3/4	0
V	1	0	0	0

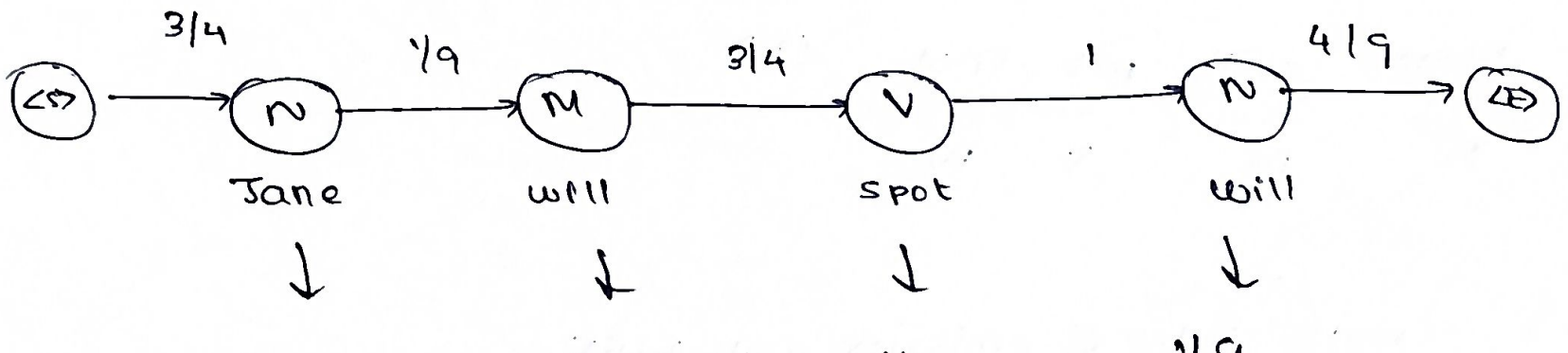
(*) see horizontally

Step 4 : Draw the transition diagram



Step 5 : Consider a testing sentence

Jane will spot Will



multiply = 0.0003858

Example 2 : Tag the following sentence using the Viterbi algorithm. with the following training corpus.

1. Martin Jushin can watch Will
2. spot will watch Martin
3. Will Jushin spot Martin
4. Martin will pat Spot

Test: Jushin will spot will.

Step1 Training corpus with correct POS Tag.

<S> Martin Justin can . watch will <E>

N N M . V N

<S> Spot will watch Martin <E>

N M V N

<S> Will Justin spot Martin <E>

M N V N

<S> Martin will pat Spot <E>

N M V N

Step2 : Create table of emission probabilities

	N	M	V
Martin	4/9	0	0
Justin	2/9	0	0
will	4/9	3/4	0
Spot	2/9	0	1/4
can	0	1/4	0
watch	0	0	2/4
Pat	0	0	1/4

Step3 Create table of state transition probability

<S>	N	M	V	<E>
	3/4	1/4	0	0
N	1/9	1/3	1/9	4/9
M	1/4	0	3/4	0
V	1	0	0	0

Testing Justin will spot Will

N M V N

N N M

(i) Justin $P(N \mid Justin, <S>)$

$= P(Justin \mid N) \times P(N \mid <S>)$ ← see vertically

$= \frac{2}{9} \times \frac{3}{4} = \frac{1}{6}$

(N)

(ii) Will $P(N \mid will, \overset{N}{\text{~~Justin~~}})$

$= P(will \mid N) \times P(N \mid \overset{N}{\text{~~Justin~~}})$

$= \frac{1}{9} \times \frac{1}{9} \times \frac{1}{6} = \frac{1}{486}$

$P(M \mid will, \overset{N}{\text{~~Justin~~}})$

$= P(will \mid M) \times P(M \mid \overset{N}{\text{~~Justin~~}})$

$= \frac{3}{4} \times \frac{1}{3} \times \frac{1}{6} = \frac{1}{24}$

$\frac{1}{24} > \frac{1}{486} \Rightarrow$ (M)

Spot $P(N \mid Spot, M)$

$= P(Spot \mid N) \times P(N \mid M)$

$= 2/9 \times 1/4 \times 1/24 = 1/432$

$P(V \mid Spot, M)$

$= P(Spot \mid V) \times P(V \mid M)$

$= 2/9 \times 3/4 \times 1/24 = 1/108$

$\frac{1}{108} > \frac{1}{432} \Rightarrow$ (V)

Will $P(N \mid will, V)$

$= P(will \mid N) \times P(N \mid V)$

$= \frac{1}{9} \times 1 \times \frac{1}{108} = \frac{1}{1152}$

$P(M \mid will, V)$

$= P(will \mid M) \times P(M \mid V)$

$= \frac{3}{4} \times$

*Named Entity Recognition with CRF

Named Entity - anything that can be referred to with a proper name.

The most common tags are:

PER
LOC
ORG
GPE (Geo-political entity)

→ NER aims to find spans of text that constitute proper names, and to tag the type of entity.

→ NER can be used for sentiment analysis, question answering & information extraction.

Difficulties in NER

(i) segmentation - entities, not words have to be segmented

(ii) Type ambiguity - words with multiple possible entity types

*Bio Tagging

→ aim to assign one label to each word.

B: token that begins a span

I: tokens inside a span

O: tokens outside of any span

eg. Jane Villanueva of United Airlines Holding discussed the
B-PER I-PER O B-ORG I-ORG I-ORG O O

Chicago route
B-LOC O

* Standard Algorithms for NER

1. Hidden Markov Models
2. Conditional Random Field (CRF) / Maximum Entropy Markov Models (MEMM)
3. Neural Sequence Models (RNNs or Transformers)
4. LLMs like BERT

* Linear Chain Conditional Random Fields

→ In a linear chain CRF, tag assignment depends only on the tag of one previous word.

→ The features for CRF are:

1. w_i → ith word of a sentence
2. embeddings – vector representation of word
3. gazetteer – list of places' names with their geographical & political information
4. word shape – notation in which letters of a word are denoted as

   (i) small letter = x

   (ii) capital letter = X

   (iii) digits = d

   eg. Delhi %.123 %.DD = Xxxxx%.ddd%.XX

5. Short word shape – similar to word shape, but with a slight change – remove consecutive similar letter types. The eg. would be:

   Xx%.d%.X

→ CRF for a word sequence is:

$$P_\theta(y|x) = \frac{\exp\left(\sum_j w_j F_j(x,y)\right)}{\sum_{y'} \exp\left(\sum_j w_j F_j(x,y')\right)}$$

* Syntactic Analysis

Syntax - the way words are arranged together

The main ideas of syntax are:

(i) constituency
(ii) grammatical relations
(iii) subcategories and dependencies

A. **Constituency** - group of words may behave as a single unit or a phrase called a constituent.

eg. Rama Krishnan

B. **Grammatical Relations** - A formalization of ideas from traditional grammar

C. **Subcategorization and Dependencies** - referring to certain kind of relations between words and phrases

eg. the verb want can be followed by infinitive / noun phrase

* Context-Free Grammar

→ a mathematical system for modelling constituent structure in English

→ A CFG has: (i) a set of rules / productions
(ii) a lexicon of words and symbols

→ The symbols used in a CFG are divided into two classes:

(i) Terminal Symbols - correspond to words

(ii) Non-Terminals - symbols that represent clusters / generalizations

→ A set of strings derived is derivable from the start symbol S

→ The formal definition of a CFG is a 4-tuple

- a set of non-terminal symbols N
- a set of terminal symbols Σ
- a set of production rules P, of the form A → α where A is a non-terminal
- a start symbol S

Example - Construct a parse tree for the following sentence using the given grammar & lexicon.

Sentence: I prefer a morning flight

Lexicon:

noun → flight | breeze | trip | morning

verb → is | prefer | like | need

adjective → cheapest | non-stop | first | latest

Pronoun → me | I | you | it

proper-noun → Alaska | Baltimore

Determinor → the | a | an | this | these

preposition → from | to | on | near

conjunction → and | or | but

Grammar - S → NP VP

NP → pronoun | proper-noun | det | nominal

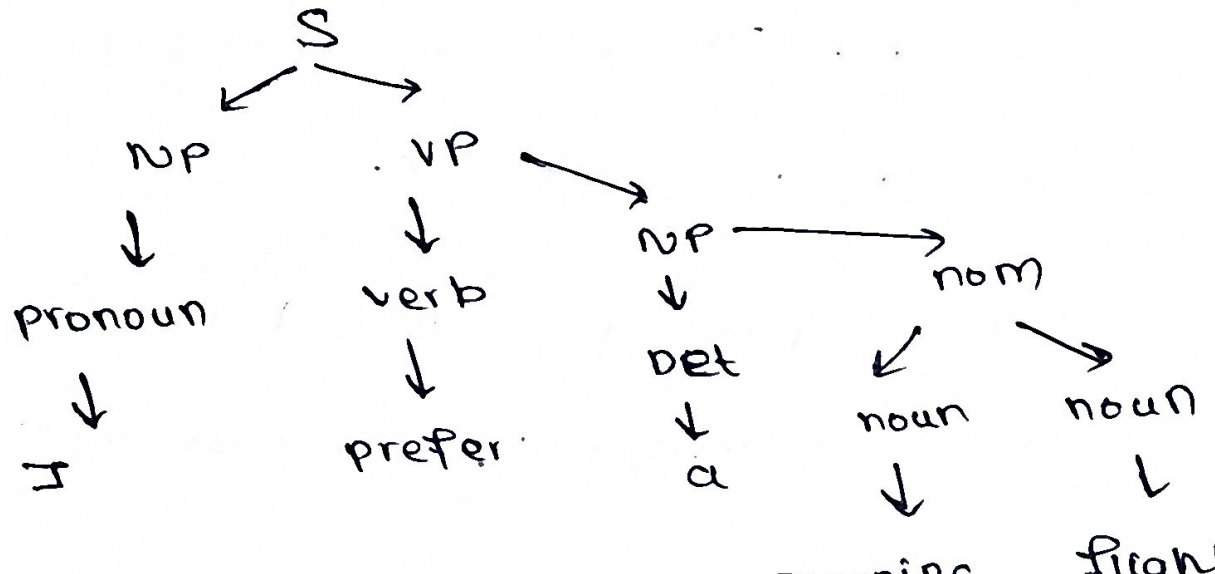
nominal → noun nominal | noun

VP → verb | verb NP | verb NP PP | Verb PP

PP → preposition NP

Ans

I prefer a morning flight



Example - Draw the parse tree for the sentence " The moan read this book " using the grammar given below.

S → NP VP

S → Aux NP VP

S → VP

NP → Det Nom

Nom → Noun

Nom → Noun Nom

VP → Verb

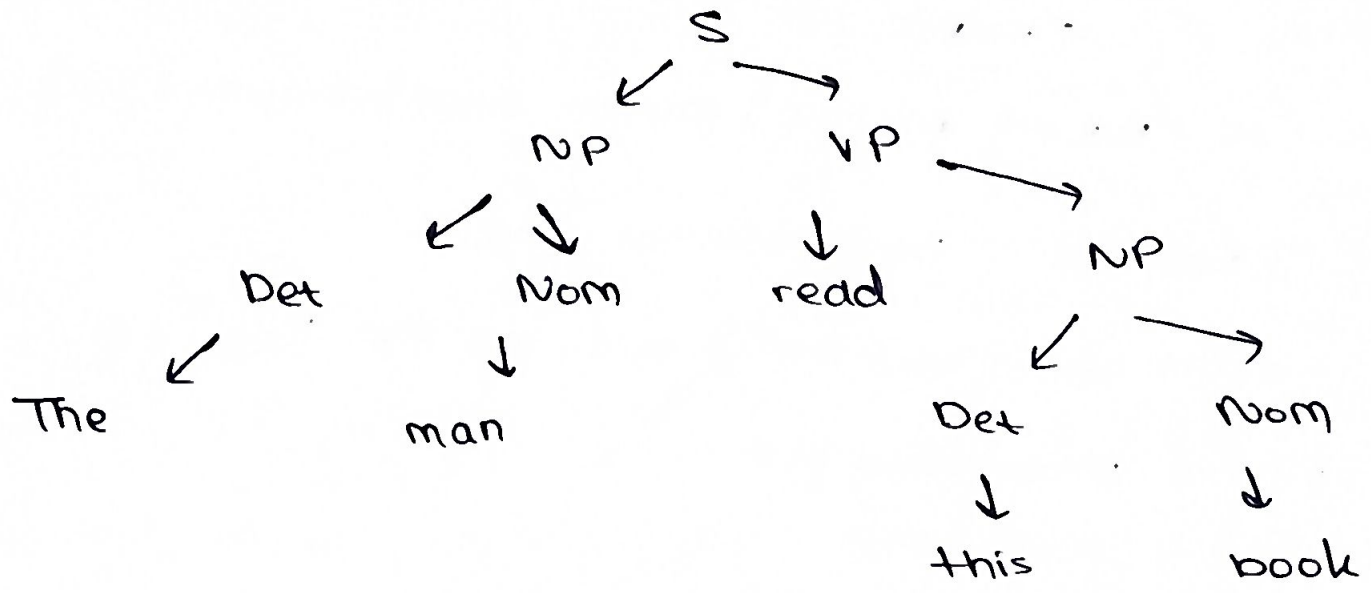
VP → Verb NP

Det → that | this | a | the

Noun → book | flight | meal | man

Verb → book | include | read

Aux → does



Suntactic Parsing – The tasu of recoanizing a sentence and assianing a syntachic structure to it.

→ The aoal of a parsing search is to find all trees whose root is the stait symbol S, which cover exactly the coords in input.

→ There are 2 strateqies :

(i) Top-dowon or qoal-directed search

(ii) bottom-up or data-directed search

* Top-Dowon Parsing

→ Uses a top-dowon, depth-first, left-to-riaht approach – expand the left-most unexpanded node in the tree

Example : Create a parse-tree using top-dowon parsina for "~~Book this flight~~ at" using the followina lexicon and arammar.

Does this flight include a meal?

S → NP VP ✓

S → Aux NP VP

S → VP

NP → Det Nominal

Nominal → Noun

Nominal → Noun Nominal

NP → Proper-Noun

VP → Verb

VP → Verb NP

Det → that | this | a

Noun → book | flight | meal | money

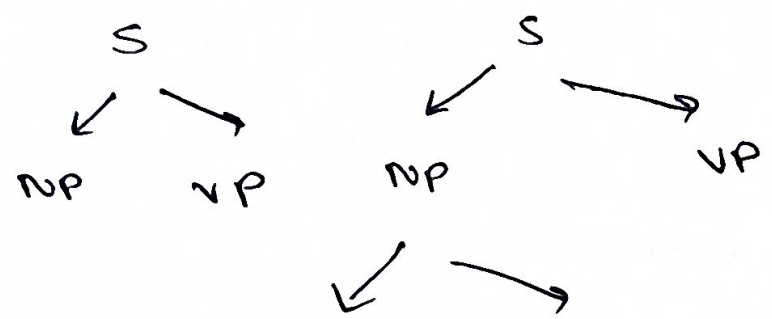
Verb → book | include | prefer

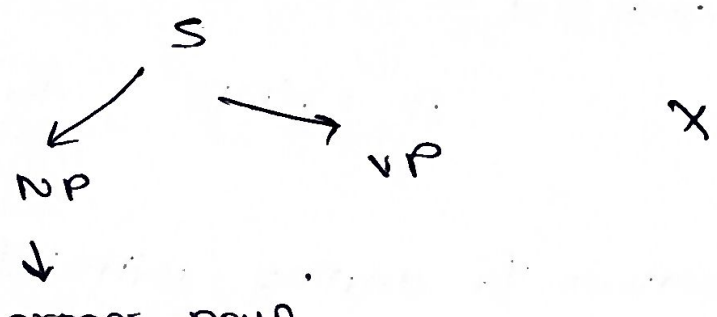
Aux → does

Prep → from | to | on

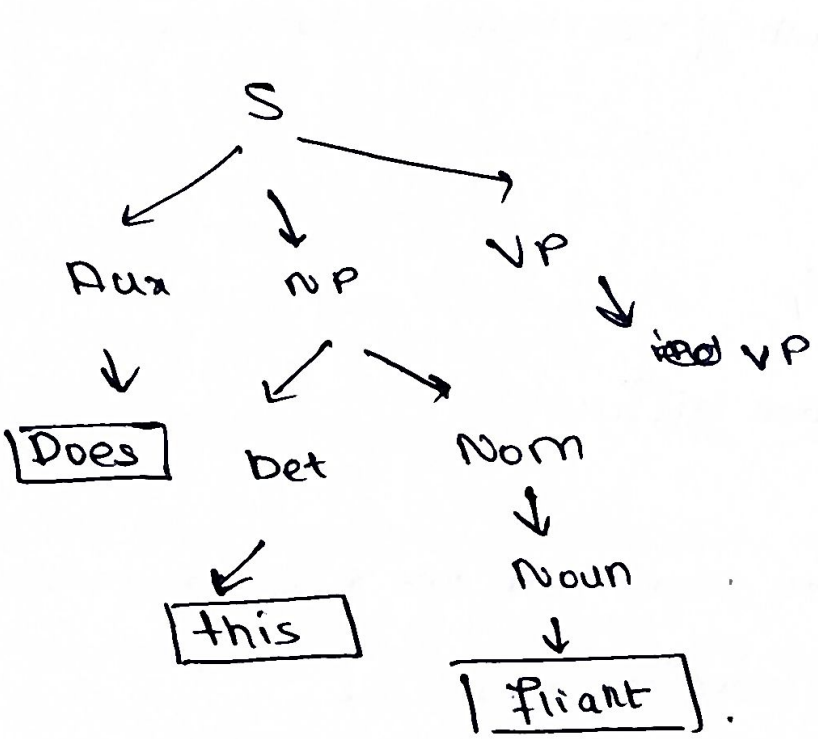
Proper-Noun → Houston | TWA

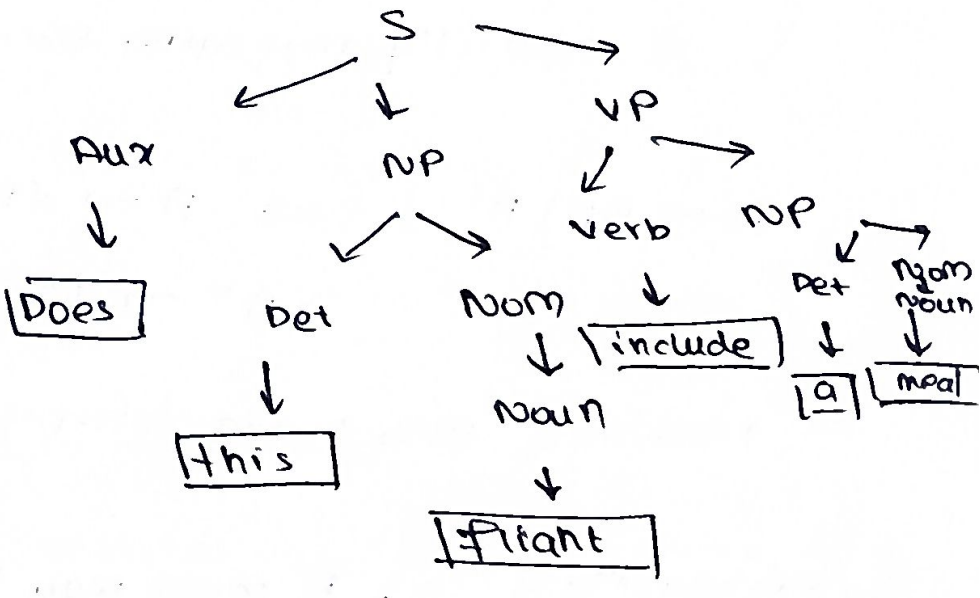
Nominal → Nominal PP





X





(X) Problems with Top-Down Parsing

1. Left Recursion
2. Ambiguity
3. Inefficient reparsing of subtrees

A. Left Recursion

→ leads to infinite exploration of search spaces, when left-recursive grammars are used.

→ A grammar is left-recursive if it is of the form:

$A \Rightarrow A\alpha\beta$ for some α and β, and $\alpha \rightarrow \varepsilon$

→ Two methods for dealing with left recursion include:

1. rewriting the grammar
2. explicitly managing the depth of the searches during parsing

$A \rightarrow A\beta \mid A' \quad \Rightarrow \quad A \rightarrow \alpha A'$

$A' \rightarrow \beta A' \mid \varepsilon$

rewriting may make interpretation difficult

B. Ambiguity

→ A word may have more than one POS, requires disambiguation to choose the correct pos for a word.

→ Structural ambiguity refers to how a grammar assigns more than one possible parse to a sentence.

→ There are 3 kinds of structural ambiguity: (i) attachment ambiguity
(ii) coordination ambiguity
(iii) non-phrase bracketing ambiguity

(i) Attachment ambiguity - a particular constituent can be attached to the parse tree at more than one place.

(ii) Coordination ambiguity - different sets can be conjoined by conjunctions

old men & women

old [men & women]

[old men] & women

→ Choosing the correct parse requires disambiguation, and it requires both statistical and semantic knowledge.

c. **Inefficient Reparsing of Subtrees**

→ The parser often builds valid trees for portions of the input, then discards them during backtracking, only to find that it has to be rebuilt again.

* Bottom-Up Parsing

→ The initial goal list is the string to be parsed.

→ If the sequence in the goal list matches the RHS of a rule, then this sequence may be replaced by the LHS of the rule.

→ Parsing is finished when the goal list has just the start symbol.

→ The standard method is shift-reduce parsing.

Shift → push the next input symbol from the buffer onto the stack

Reduce pop the Rule's RHS off the stack and replace it w/ the terminal on the LHS

When either shift or reduce is possible, choose arbitrarily

Example3 Use bottom-up parsing to parse "Book that flight" using the following lexicon & grammar.

*CYK Parsing

CYK ⟶ Kasami
↙ ↓
Cocke Younger

→ The structure of rules is in a Chomsky Normal Form grammar

→ A dynamic programming approach can be used.

→ A CFG is in CNF if each rule is of the following form:

(i) $A \to BC$

(ii) $A \to a$ (terminal)

(iii) $S \to \lambda$ (null string)

Eg1 Apply the CYK algorithm for the string baaba w/ the following grammar.

$S \rightarrow AB \mid Bc$

$A \rightarrow BA \mid a$

$B \rightarrow cc \mid b$

$c \rightarrow AB \mid a$

S,A,C X15				
– X14	S,C,A X25			
– X13	B X24	B X35		
S,A X12	B X23	S,C X34	S,A X45	
B X11	A,C X22	A,C X33	B X44	A,C X55
① b	a	a	b	a

X_{11} : b $B \rightarrow cc \mid b$

X_{22}, X_{33} : a $A \rightarrow BA \mid a$, $c \rightarrow AB \mid a$

X_{34} : b $B \rightarrow cc \mid b$

X_{55} : a $A \rightarrow BA \mid a$ $c \rightarrow AB \mid a$

② X_{12} : ba = B and A,C

BA BC

$A \rightarrow BA \mid a$ $S \rightarrow AB \mid BC$

$\Rightarrow S, A$

$X_{23} = aa = A,C$ and A,C

$\Downarrow$

AA AC CA CC

ϕ ϕ ϕ $B \to cc|b$

$\Rightarrow \underline{B}$

$X_{34} = ab = A,C$ and B

$\Downarrow$

AB and CB

$S \to AB|BC$ ϕ

$C \to AB|a$

$\Rightarrow \underline{S,C}$

$X_{43} = ba = B$ and A,C

$= BA$ and BC.

$A \to BA|a$ $S \to AB|BC$

$\Downarrow$

$\underline{S,A}$

(3) $X_{13} = baa$

possible combinations are:

ba and a — and — b and aa

ba → S,A; a → A,C

$\Downarrow$

SA, SC, AA, AC

ϕ ϕ $\downarrow$ $\downarrow$

ϕ ϕ

b → B; aa → B

$\Downarrow$

BB

$\downarrow$

ϕ

X_{24} = aab

possible combinations are:

a → A, C

ab → S, C

and

aa → B

b → B

BB → ϕ

AS, CS, AC, CC → B

⇒ B

X_{35} = aba

possible combinations are

a → A, C

ba → S, A

and

aba → S, C

a → A, C

SA, SC, CA, CC → B

AS, AA, CS, CA → ϕ

⇒ B

(4) X_{14} = baab

possible combinations are:

b → B

aab → B

BB → ϕ

ba → S, A

ab → S, C

SS, SC, AS, AC → ϕ

⇒ ϕ

baa → ϕ

b → B

ϕ

⇒ ϕ

x_{25} : a a b a

possible combinations are :

a → A, C

aba → B

A, C → AB, CB

AB, CB → S, C

aa → B

ba → S, A

B → BS, BA

BS, BA → A

aab → B

a → A, C

B → BA, BC

BA, BC → A, S

⇒ S, C, A

(5)

x_{15} = b a a b a

possible combinations are :

b → B

aaba → S, C, A

B → BS, BC, BA

BS, BC, BA → A

ba → S, A

aba → B

S, A → SB, AB

SB, AB → S, C

bad → ϕ

ba → S, A

ϕ → ϕ

baab → ϕ

a → a

ϕ → ϕ

⇒ S, A C

Ex8 Apply CYK grammar for the sentence:

"I shot an elephant in my pajamas"

with the grammar rules:

S → NP VP

PP → IN NP

NP → DET NP

NP → NP PP

VP → VBD NP

VP → VP PP

NP → PRP$ NP

DET → an

VBD → shot

NP → pajamas

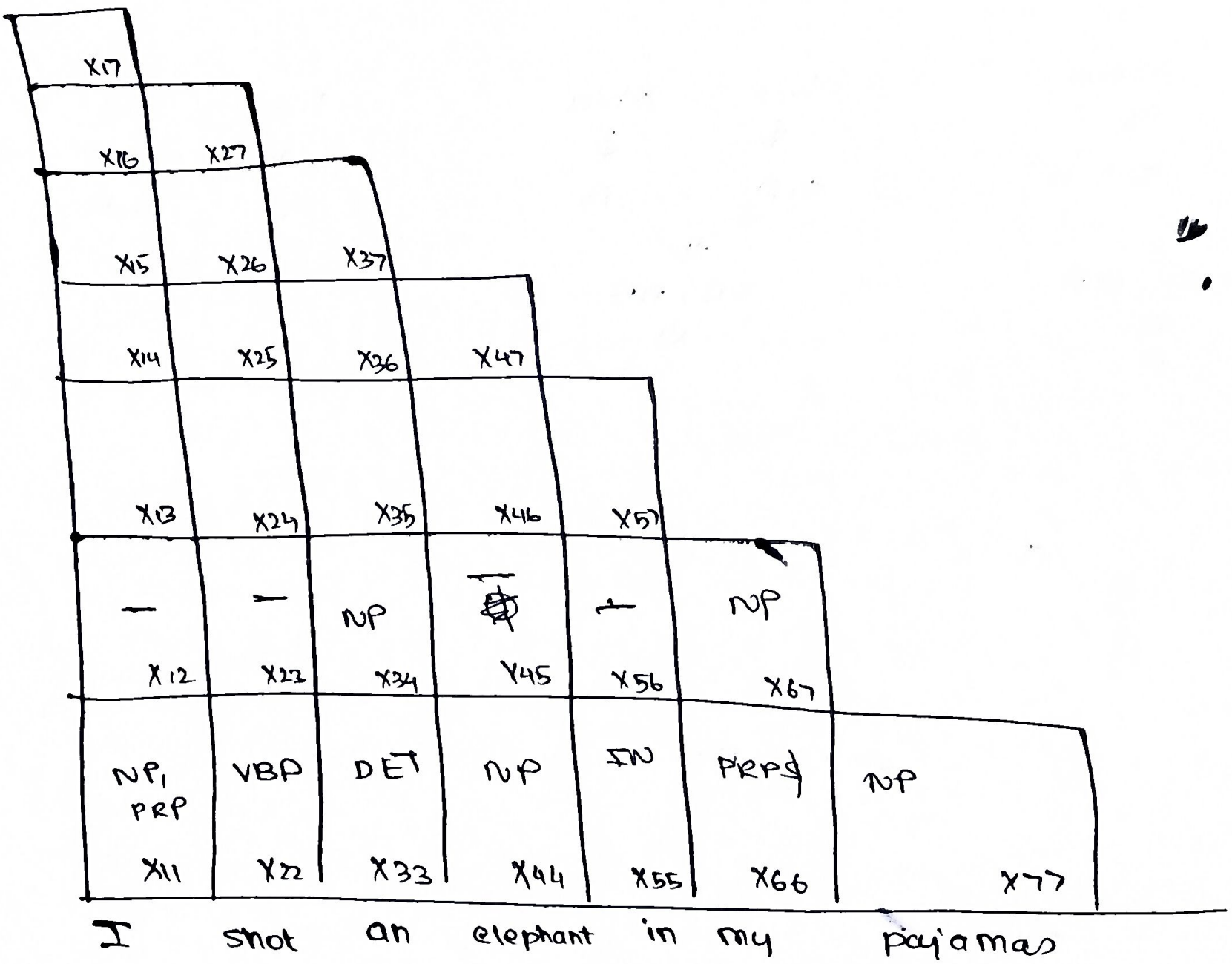
NP → elephant

NP → I

PRP → I

IN → in

PRP$ → my



① X_{11} → I
⇓
NP, PRP

X_{44} → elephant
⇓
NP

X_{29} → shot
⇓
VBD

X_{55} → in
⇓
IN

X_{33} → an
⇓ DET

X_{66} → my
⇓
PRP$

X_{77} → pajamas
⇓
NP

② X_{12} → I shot
↓ ↓
NP, PRP VBD
⇓
NP. VBD PRP. VBD
⇓ ⇓
ϕ ϕ

X_{23} → shot an
↓ ↓
VBD DET
⇓
VBD. DET
⇓
ϕ

X_{34} → an elephant
↓ ↓
DET NP
⇓
DET. NP
⇓
NP

X_{45} → elephant in
↓ ↓
NP IN
⇓
NP. IN
⇓
ϕ

X_{56} → in my
↓ ↓
IN PRP$
⇓
IN. PRP$ ⇒ ϕ

X_{67} my pajamas
↓ ↓
PRP$ NP
⇓
PRP$. NP
⇓
NP

* Shallow Parsing

→ a technique that aims to identify and extract meaninaful phrases or chunks from a sentence.

→ Unlike full parsing, which involves andlyzing the arammatical structure of a sentence, shallow parsing focuses on identifying individual phrases like noun phrases, verb phrases etc.

Benefits

→ efficient, full parsing is computationally intencive

→ beneficial for processing large volumes of text like web crawling, document indexing, and machine translation

Steps

1. divide sentence into words or tkens

2. Pos tagging

3. identify and extract relevant phrases / constituents. This is done using pattern matching / ML algorithms that have been trained to recognize particular hypes of phrases

* Dependency Parsing