SAP ID: 60009200027
Name: Aditya Potdar
Div./Batch: K/K1

CL Assignment 2

Chapter 3: Syntax Analysis

1. Explain Constituency Parsing with help of example.

Constituency parsing is a technique in NLP that involves breaking down a sentence into its constituent parts or phrases. It's a process of analyzing the syntactic structure of a sentence and identifying its various grammatical units or constituents, such as noun phrases, verb phrases and prepositional phrases.

For eg:- "The cat sat on the mat"

A constituency parser would analyze this sentence and break it down into its various constituents, which would look something like this:-

The cat: noun phrase
sat: verb phrase
on the mat: prepositional phrase

2. Explain Types of Ambiguity in Constituency parsing with the help of example.

4 types of ambiguity are possible :-

① POS Ambiguity :- It refers to cases where a word can be assigned multiple possible parts of speech.
For eg :- Word "can",

I can swim , I kicked the can.
↑ ↑
Modal verb Noun

② Structural Ambiguity :- It occurs when a sentence can be parsed in different ways due to the arrangement of its constituent parts. For eg :- the sentence "I saw her duck". Here, "duck" can be a noun or a verb.

③ Attachment Ambiguity :- It occurs when a phrase can be attached to different parts of the sentence. For eg :- "He saw the man with the binoculars". This sentence can be parsed in 2 difference ways whether the observer is carrying the binoculars or the man is carrying the binoculars.

④ Coordination Ambiguity :- It occurs when a sentence has coordinated phrases or clauses that can be parsed in different ways. For eg :- "She likes swimming and running". Here "swimming and running" can be coordinated verbs or coordinated nouns.

3- Explain CKY Parsing Algorithm for parsing a given string with the help of example.

Cocke-Kasami-Younger (CKY) algorithm, the most widely used dynamic-programming based approach to parsing.
CFG should be in Chomsky Normal Form (CNF) before applying CKY algorithm.

Restricting a grammar to CNF does not lead to any loss in expressiveness, since any context-free grammar can be converted into a corresponding CNF grammar that accepts exactly the same set of strings as the original grammar.

CKY Algorithm :-

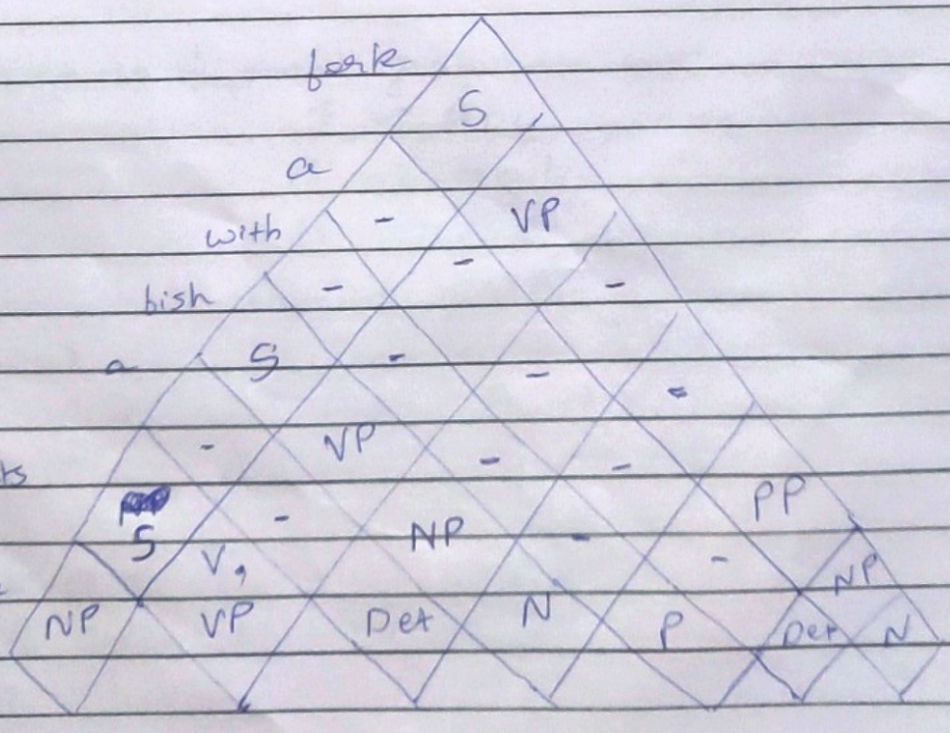
① Let n be the number of words in the input. Thinking about n+1 lines seperating them, numbered from 0 to n
② x_{ij} will denote the words between line i and j.
③ We build a table so that x_{ij} contains all the possible non-terminal spanning of words between line i and j.
④ We build the table bottom-up.

Page No.
Date

for eg :- Consider given grammar

S → NP VP
VP → VP PP
VP → V NP
VP → eats
PP → P NP
NP → Det N
NP → she
V → eats
P → with
N → fish
N → fork
Det → a

Consider given sentence : she eats a fish with a fork



Page No.
Date

4. How to convert CFG to CNF?

Assuming we are dealing with an ε free grammar, there are 3 situations we need to address in any generic grammar :-

- Rules that mix terminals with non-terminals on the right-hand side
- Rules in which the length of the right hand side is greater than 2
- Rules that have a single non-terminal on the right-hand side ($A \rightarrow B$)

We can eliminate unit productions by rewriting the right hand side of all the non-unit production rules that they ultimately lead to.

For eg :- if given grammar is :-

$S \rightarrow ABa$
$A \rightarrow aab$
$B \rightarrow Ac$

After converting to CNF

$X \rightarrow a$
$Y \rightarrow b$
$Z \rightarrow c$

$\longrightarrow$

$S \rightarrow ABX$
$A \rightarrow XXY$
$B \rightarrow AC$

$AB \rightarrow P$
$XX \rightarrow Q$

$\longrightarrow$

$S \rightarrow PX$
$A \rightarrow QY$
$B \rightarrow AC$

Page No.
Date

6. Explain Dependency Parsing with the help of example

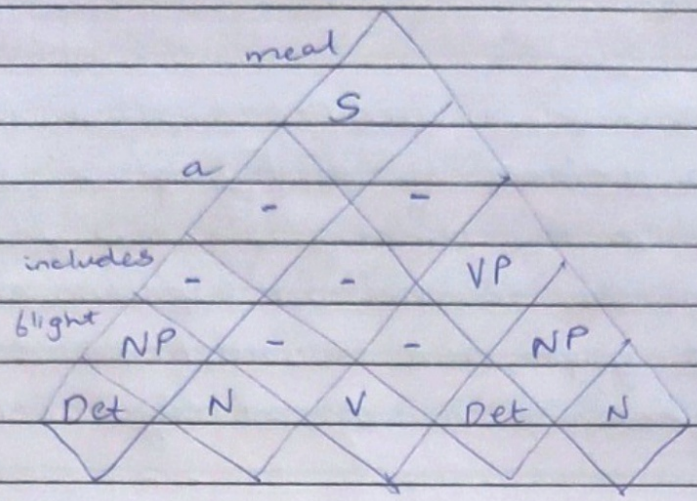
Q.5. Given Grammar in CFG :-

S → NP VP
NP → Det N
VP → V NP
V → includes
Det → the
Det → a
N → meal
N → Flight

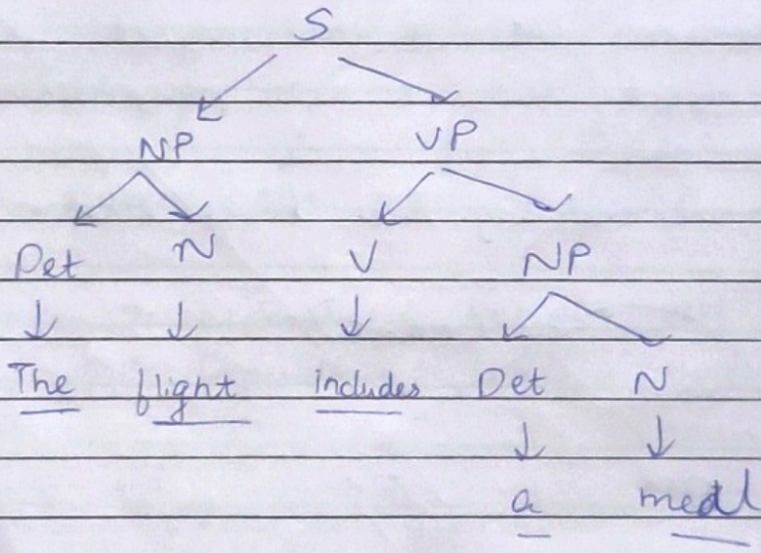
Converting to CNF :-

S → NP VP
NP → Det N
VP → V NP
V → includes
Det → the | a
N → meal | flight

For the sentence, "The flight includes a meal".



Parse Tree :-



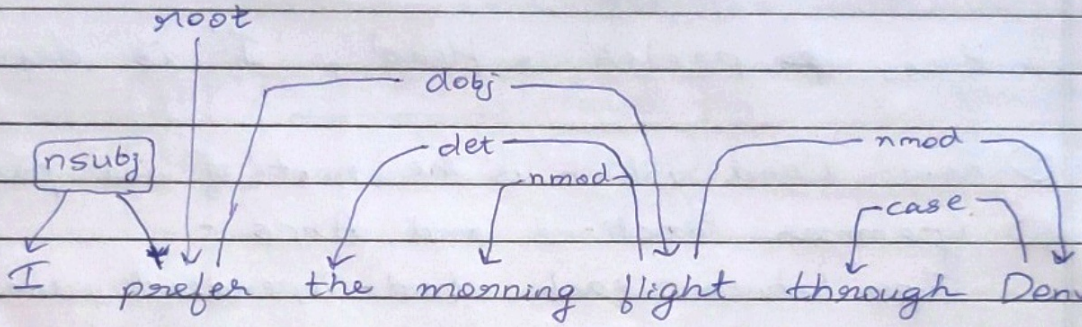
Page No.
Date

Q.6. Explain Dependency parsing with the help of example.

Dependency parsing is used to analyze the grammatical structure of a sentence by identifying the relationships between words in the sentence. In this parsing technique, the words in the sentence are represented as nodes in a directed graph, and the relationships between the words are represented as direct edges between the nodes.

For example :- 'I prefer the morning flight through Denver'

Dependency tree looks like :-

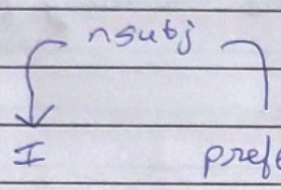


nsubj → Dependency between verb and subject
dobj → Dependency between verb and direct object
det → " " noun and determiner
nmod → " " noun and modifier in prepositional phrase
case → " " preposition and its object or a noun and its grammatical phrase

Page No.
Date

Q.7. Explain terminologies as well used for Dependency parsing.

① Head - Dependent:
In the arrows representing relationship, the origin word is the head & the destination word is Dependent.



Here, ~~so~~ 'prefer' is Head, 'I' is dependent

② Root: Word which is the root of our parse tree.

③ Grammar Functions and Arcs:-
Tags between each Head-Dependent pair is a grammar function determining the relation between the Head & Dependent.

④ The arrowhead carrying the tag is called an arc.

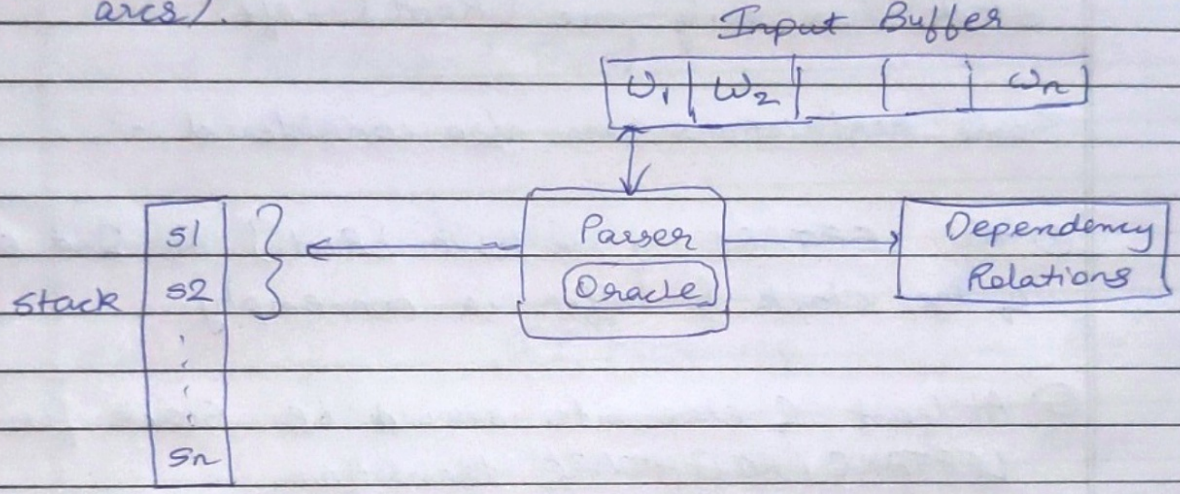
Examples of Grammar Functions:-
nsubj (Nominal Subject): Dependency b/w verb & subject
dobj (Direct Object): Dependency b/w verb & direct object
det (Determiner): Dependency b/w noun & determiner

Page No.
Date

Q-8. How to create a Dependency parse tree - Shift Reduce Parsing (Arc standard) algorithm?

Setup for this algorithm :-

① A Stack
② Buffer of input words
③ Oracle function/parser & set of dependency relations (like obj, iobj, etc tags that are used to represent arcs).



① Initially the Stack has [Root], a dummy variable
② All the words of the sentence to be parsed are in buffer.
③ A word is popped from buffer & pushed in Stack.
④ As a word is pushed in Stack, the Oracle/parser takes the top 2 elements of the stack & outputs/predicts any of the below 3 transitions with relationship type:

LEFTARC_FUNCTION
RIGHTARC_FUNCTION
SHIFT

⑤

Page No.
Date

⑤ Hence, the actual prediction has 2 parts:
Determining Head & Dependent using LEFTARC/RIGHTARC
Determining their relation type i.e functions like conj, iobj.

⑥ Hence, the actual predictions done by Oracle are something like this LEFTARC-IOBJ, RIGHTARC-CONJ, etc.

⑦ We will stop as & when the buffer is empty & stack has only one [Root] left.

Some restrictions are also considered :-

① No LEFTARC, if we have [Root] as 2nd element of the stack & buffer is non-empty.

② At least 2 elements should be Stack for LEFTARC/RIGHTARC transition.

③ [Root] can never be dependent (as it is a dummy variable).

Page No.
Date

Q.9. For the CFGs given: S → NP VP
VP → V NP
NP → Det N

Draw the Shift-Reduce Parser in processing the sentence " the policeman saw the thief ".

Use the following lexical entries to create the chart parser :-
Det → The | a
N → Policeman | Thief
V → saw

Shift-Reduce Parser :-

Stack	I/P remaining	Action
()	The policeman saw the thief $	Shift
(The)	policeman saw the thief $	Reduce, Det → The
(Det)	policeman saw the thief $	Shift
(Det, policeman)	saw the thief $	Reduce, N → Policeman
(Det, N)	saw the thief $	Reduce NP → Det N
(NP)	saw the thief $	Shift
(NP, saw)	the thief $	Reduce V → saw
(NP, V)	the thief $	Shift
(NP, V, the)	thief $	Reduce Det → The
(NP, V, Det)	thief $	Shift
(NP, V, Det, thief)	$	Reduce N → Thief
(NP, V, Det, N)	$	Reduce NP → Det N
(NP, V, NP)	$	Reduce VP → V NP
(NP, VP)	$	Reduce S → NP VP
(S)	$	-

Page No.
Date

Q.10. What is Need of Parsing?

In POS tagging, we assign POS tags for individual words in the given sentence.

For eg:- "the blue umbrella".

Here, the → Determine
blue → Adjective
Umbrella → Noun

But here nowhere is mentioned 'blue' is used for umbrella.

Hence, we need some sort of grouping so as to retrieve the relationship between the words of a sentence.

The meaning of any sentence can only be derived when we can know how much these words combine together in a sentence.

Page No.
Date

Q.11. Explain Probabilites parsing with the help of suitable example.

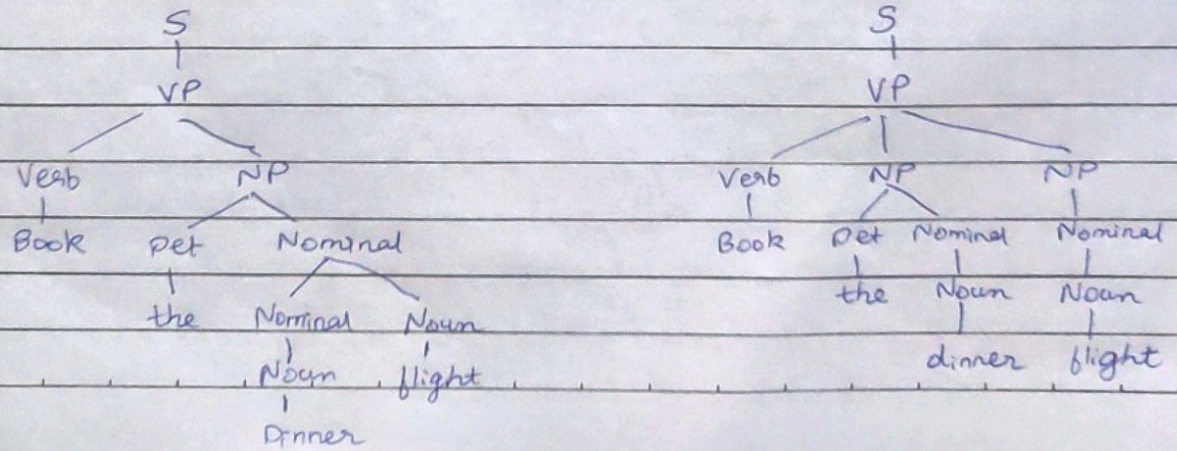
Probabilistic parsing is a type of parsing that assigns probabilities to each potential phrase of a sentence, based on a statistical model of the language.
It also solves the problem of disambiguation.
It computes the probability of each interpretation and choose the most probable interpretation.

For example :-

Grammar		Lexicon
S → VP	[0.05]	Verb → "Book" [0.30]
VP → Verb NP	[0.20]	Det → the [0.60]
VP → Verb NP NP	[0.05]	Noun → flight [0.3] | dinner [0.1]
NP → Det Nominal	[0.20]	
NP → Nominal	[0.15]	
Nominal → Nominal Noun	[0.20]	
Nominal → Noun	[0.75]	

Now consider sentence "Book the dinner flight".

Parse Tree :-



$P(T_{left}) = 2.2 \times 10^{-6} = \{0.05 \times 0.2 \times 0.2 \times 0.2 \times \ldots\}$

$P(T_{right}) = 6.1 \times 10^{-7}$

Therefore, the most probable of the two parses is T_{left}.