Unit-3

Grammars for Natural Language

Grammars for Natural language, Movement Phenomenon in Language, Handling questions in Context Free Grammars, Hold Mechanisms in ATNs, Gap threading, Human Preferences in Parsing, Shift Reduce Parsers, Deterministic Parsers.

Augmented Context-free grammars provide a powerful formalism for capturing many generalities in NL.

Grammars for NL:-

English sentences typically contain a sequence of auxiliary verbs followed by a main verb.

eg: I can see the house.

~~the~~ I was watching the movie.

These may at first appear to be aribitary sequences of verbs, including have, be, do, can, will, and so on.

The auxiliary "have" must be followed by a past participle form, and the auxiliary "be" must be followed by a present participle form.

Auxiliaries such as can and must always be followed by a base form.

The principal idea is that auxiliary verbs have subcategorization features that restrict their verb phrase complements.

eg: I am not going.
You did not try it.

I ate the pizza. , Did I eat the pizza?
I have a pen. , Do I have a pen?

The primary auxiliaries are based on the root forms "be" and "have". The other auxiliaries are called modal auxiliaries and generally appear only in the finite tense forms (simple present and past).

do – did	shall – should
can – could	will – would
may – might	must, need, dare.

VP $\longrightarrow$ (AUX COMPFORM ?s) (VP VFORM ?s)

VP $\longrightarrow$ AUX [be] VP [ing, +main]

can : CAT AUX MODAL + VFORM pres ~~AGR~~.

$\downarrow$

do

AGR (1s 2s 3s 1p 2p 3p)

COMFORM base

$\downarrow$

lexicon entry for "can and do"

NL grammars are frameworks that describe the structure and rules of languages. Several types of grammars are used in the field of linguistics and computational linguistics to analyze and generate NL.

<u>CFG</u> : Context Free Grammar.

CFG is a type of formal grammar that is used to define the possible structures in a language. It consists of a set of production rules that describe how non-terminal symbols can be transformed into terminal symbols.

S $\rightarrow$ NP VP

NP $\rightarrow$ Det N

VP $\rightarrow$ V NP

Det $\rightarrow$ 'the' | 'a'

N $\rightarrow$ 'cat' | 'dog'

V $\rightarrow$ 'chased' | 'saw'

CFGs are used in parsing and syntax analysis

CSG: Context Sensitive Grammar

CSGs are more powerful than CFGs.

CSGs are used in situations where more context is required to define grammatical rules.

TAG: Tree Adjoining Grammar

TAG is a highly structured form of grammar where elementary trees represent basic syntactic structures.

eg: S → NP VP
  VP → V NP

TAGs are used for their ability to capture syntactic dependencies and hierarchical structures in sentences.

Probabilistic CFGs:

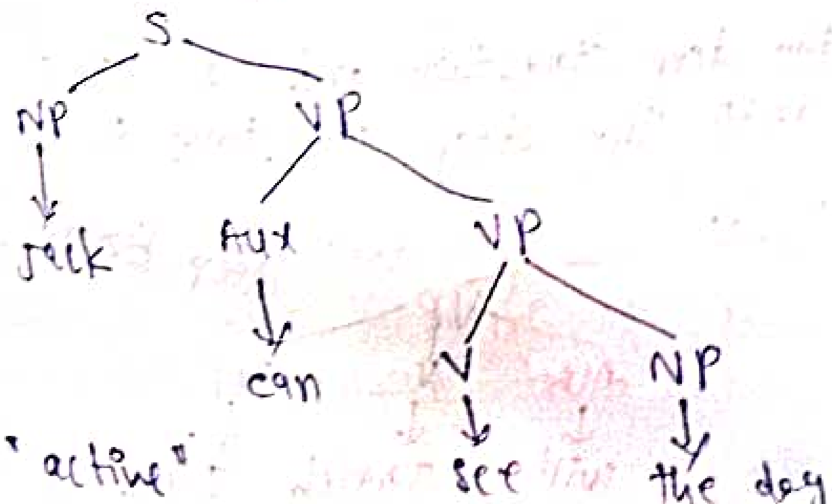
PCFGs extends CFG by adding probabilities to the production rules.

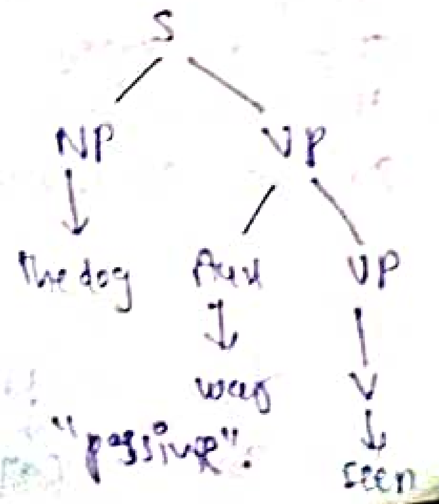
eg: S → NP VP [0.9]
  NP → Det N [0.8]

PCFGs are used in statistical parsing and language modeling.

Movement Phenomena in Language:-

Many sentence structures appear to be simple variants of other sentences' structures.





eg: John went to the store.
Did john go to the store?

eg: I found a bookcase.
Did I find a bookcase.

The place where a subconstituent is missing is called the "gap", and the constituent that is moved is called the "filler".

Many linguistic theories have been developed that are based on the intuition that a constituent can be moved from one location to another.

A new structure called the "hold list" was introduced that allowed a constituent to be saved and used later in the parse by a new arc called VIR - Virtual Arc.

Constituents are stored in a special feature called GAP and are passed from constituent to subconstituent to allow movement.

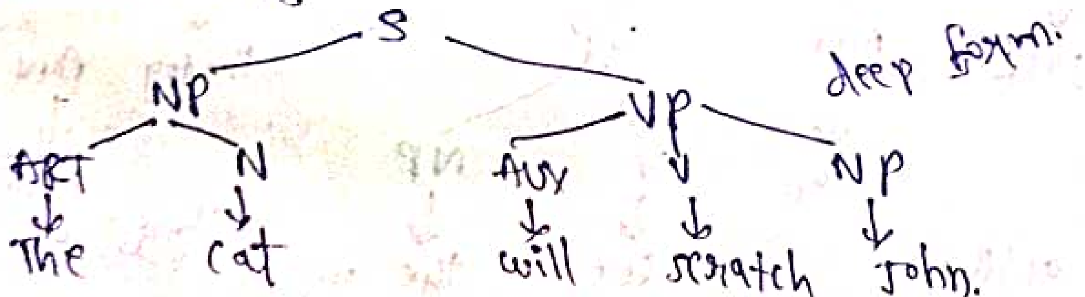
<u>Movement in Linguistics</u>:

The term movement arose in TG - Transformational Grammar. TG posited two distinct levels of structural representation:

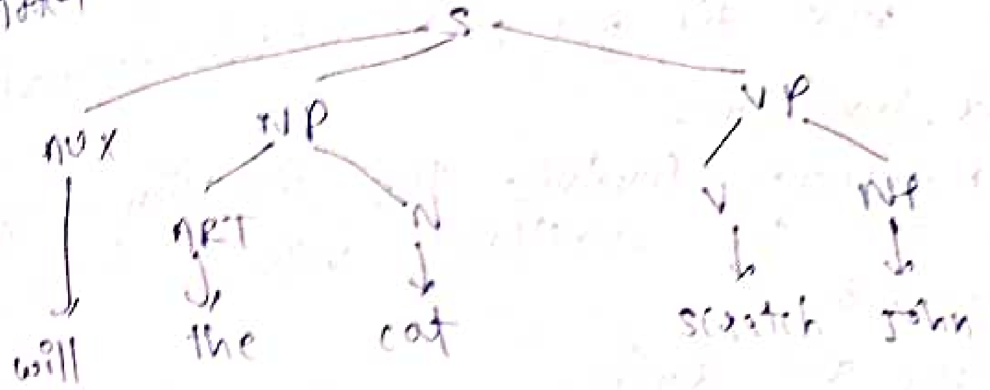
(i) surface structure → which corresponds to the actual sentence structure

(ii) deep structure.

CFG generates the deep structure, and a set of transformations map the deep structure to the surface structure.



After transformation,



surface form.

The GB - Government Binding theory, a single transformation rule, allows any constituent to be moved anywhere.

<u>Different types of movement</u>:

wh-movement ⇒ move a wh-term to the front of the sentence.

topicalization ⇒ move a constituent to the beginning of the sentence.

eg: I never killed this picture.
This picture, I never killed.

adverb preposing ⇒ move an adverb to the beginning of the sentence.

eg: I will see you tomorrow.
Tomorrow, I will see you.

extraposition ⇒ move certain NP complements to the sentence final position.

Movement phenomenon in language refers to the syntactic process by which elements in a sentence are repositioned from their canonical or original position to a new position within the sentence.

wh ⇒ "You saw what" ← canonical
"what did you see?" ← wh-movement.

Sub-Aux Inversion:
This movement involves the inversion of the subject and the auxiliary verb to form questions.

eg: You are coming.
Are you coming?

Raising: It occurs when a noun phrase moves out of an embedded clause to the main clause.

eg: It seems that John is happy.
John seems to be happy.

Clefting: Clefting involves splitting a sentence into two clauses to put focus on a particula

eg: She bought a car.
It was a car that she bought.

Pied-Piping: This occurs when a wh-word pulls along with it a preposition or other elements to the front of the sentence.
eg: You talked about what.
About what did you talk.

Handling Questions in CFG:-
The goal is to extend a CFG minimally so that it can handle questions.
This enforces sub-verb agreement b/w the AUX and the sub NP.
A special feature GAP is introduced to handle wh-questions.

The wh feature is signaled by words such as who, what, when, where, why and how.
These words fall into several different grammatical categories,

who ate the pizza?
what book did ~~he~~ she steal?
where did you put the book?
How quickly did he run?
whose book did you find?

<u>parsing with Gaps:</u>

A grammar that allows gaps creates some new complications for parsing algms.

In linguistics the principles that govern where gaps may occur are called island constraints.

The A over A Constraint : No constituent of category A can be moved out of a constituent of type A.

Complex - NP Constraint : No constituent may be moved out of a relative clause in noun complement.

Sentential Subject Constraint : No constituent can be moved out of a constituent serving as the sub of a sentence.

wh - Island Constraint : No constituent can be moved from an embedded sentence with a wh- complementizer.

Coordinate Structure Constraint : A constituent can not be moved out of a coordinate structure.

Note that these constraints apply to all forms of movement, not just wh-questions.

Handling questions in a CFG involves defining production rules that account for the syntactic structure of questions.
English questions typically involve movements like sub-aux inversion and wh-movement.

Yes/No-Questions:
These usually involve the inversion of the sub and aux verb.
eg: She can swim.
Can she swim?

Wh-Questions:
These involve moving a wh-word to the beginning of the sentence.
eg: she can swim.
what can she do?

Terminals: words in the language eg: she, can,...
Non-terminals: syntactic categories eg: S, NP, VP, AUX,...

Hold Mechanism in ATNs:-

Another technique for handling movement was first developed with the ATN framework.
A data structure called the hold list maintains the constituents that are to be moved.
Unlike GAP features, more than one constituent may be on the hold list at a single time.
Constituents are added to the hold list by a new action on arcs, the hold action, which takes a constituent and places it on the hold list.

The hold action can store a constituent currently in a register.

$$\begin{tikzcd}
 & \text{VP} \arrow[d, shift left] \arrow[r] \arrow[rrr, bend left] & & & \\
\text{S} \arrow[ur] \arrow[r] \arrow[dr] & \text{WH-S} \arrow[u, shift left] \arrow[d] & \text{VP-AUX} \arrow[r] & \text{VComp} \arrow[r, bend left] \arrow[d] & \text{OBJ} \arrow[l, bend left] \arrow[d] \\
 & \text{S-Inv} \arrow[ur] & & \text{pop} & \text{pop}
\end{tikzcd}$$

An S n/w for questions and relative clauses.

We have seen two approaches to handling questions in grammars: the use of the GAP feature and the use of the hold list in ATNs.

The hold list mechanism in the ATN framework must be extended to capture constraints.

while the technique of GAP feature propagation was introduced with CFGs, and the hold list mechanism was introduced with ATNs, ~~[illegible]~~

*

ATNs - Augmented Transition Networks are a type of finite state machine used in computational linguistics for parsing natural language.

They extend simple transition n/ws by incorporating additional feature like registers, tests and procedures, allowing for more sophisticated language processing.

One important mechanism in ATNs is the "hold" mechanism, which is used to temporarily store parts of the i/p for later processing.

Basic Concept :

The hold mechanism allows an ATN to remember certain positions of the i/p. string while it processes other parts.

This is useful in handling various syntactic phenomena, such as:

→ Nested structures → Complex sentences

→ Long-distance dependencies

States and Arcs:

State-0 : start state
State-1 : Processing main clause.
State-2 : Encountering an embedded clause
State-3 : Returning from embedded clause
State-4 : final state.

Benefits:

→ Handling Complexity
→ Maintaining State
→ Flexibility.

Gap Threading:-

For handling gaps combines aspects of both the GAP feature approach and the hold list approach. This technique is usually called gap threading. It is often used in logic grammars, where two extra argument positions are added to each predicate - one argument for a list of fillers and one for the resulting list of fillers.

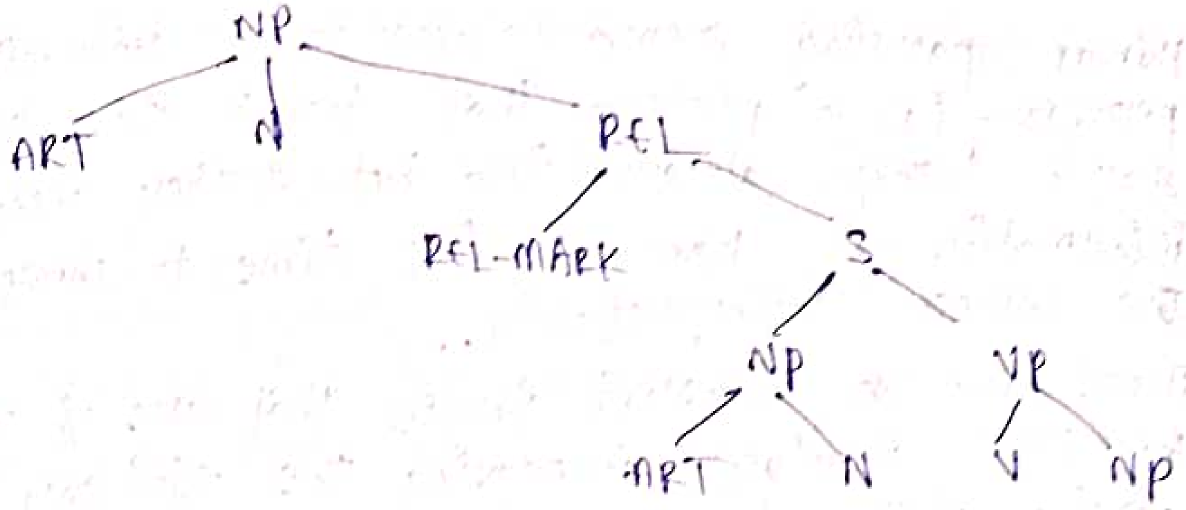
S(position-in, position-out, fillers-in, fillers-out)

It is true only if there is a legal 'S' constituent b/w position-in and position-out of the input.

In case where there are no gaps in a constituent, the fillers-in and fillers-out will be identical.

A prolog program, is a notation has been designed to facilitate the specification of grammars that handle gaps.

There is a formalism called extraposition grammar, which besides allowing normal content free rules.



A parse tree in extraposition grammar.

Gap threading is a technique used in syntactic parsing to handle long-distance dependencies in natural language. such as relative clauses, wh-questions and topicalizations.

Long - Distance Dependencies:

It occur when a constituent (NP) is displaced from its canonical position, often due to syntactic transformations like wh-movement (or) relativization.

wh-question: " what did you see __ ? ".
gap.

Gap Threading in ATNs:

ATNs can use gap threading to parse sentences with long distance dependencies.

Benefits:

→ Handling long-distance dependencies

→ Flexibility and robustness.

→ Clear syntax - semantics interface

Human Preferences in Parsing:-

All the parsing frameworks have depended on complete search techniques to find possible interpretations of a sentence..

Human parsing seems closer to a deterministic process - i.e, a process that doesn't extensively search through alternatives but rather uses the information it has at the time to choose the correct interpretation.

There are 2 different issues that are of concern.

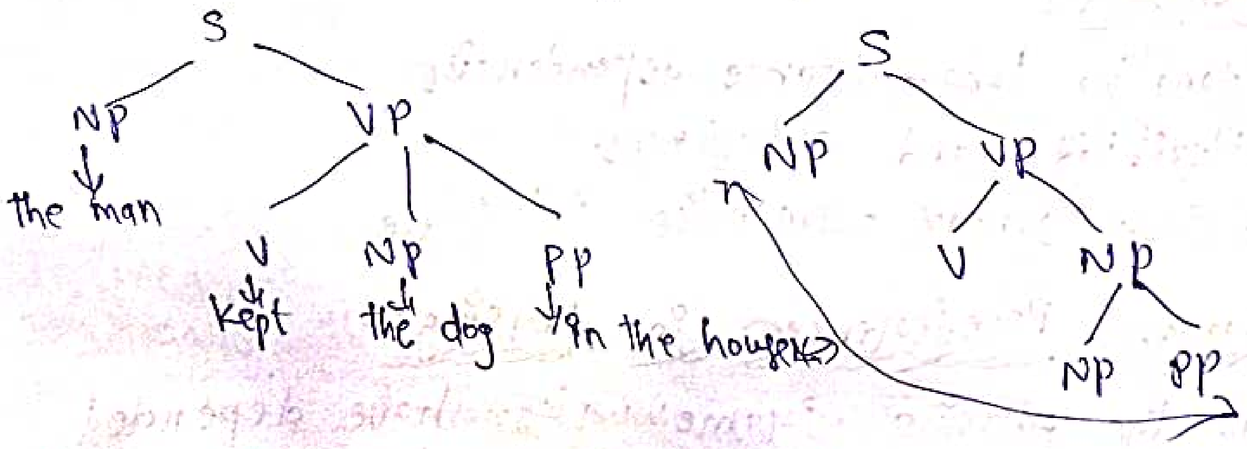
→ The first involves improving the efficiency of parsing algms by reducing the search but not changing the final outcome.

→ The second involves techniques for choosing b/w different interpretations that the parser might be able to find.

Psycholinguists have conducted many investigations into this issue using a variety of techniques from intuitions about preferred interpretations to detailed experiments monitoring moment-by-moment processing as subjects read and listen to language.

S → NP VP	NP → ART N
VP → V NP VP	NP → NP PP
VP → V NP	PP → P NP

A simple CFG



In the literature, some sentences are often called "garden-path" sentences.

Minimal Attachment:

The most general principle is called the minimal attachment principle, which states that there is a preference for the syntactic analysis that creates the least number of nodes in the parse tree.

eg: We painted all the walls with cracks
The man saw the boy with the telescope.

Right Association:

The second principle is called right association or late closure. This principle states that all other things being equal, new constituents tend to be interpreted as being part of the current constituent under construction.

eg: George said that Henry left in his car.

Garden Path Sentences:

These are sentences that lead the reader towards an incorrect interpretation initially, forcing a reanalysis.

eg: The horse raced past the barn fell.

Implementing Human Parsing Preferences in NLP sys:-

PCFG: Probabilistic CFGs.

PCFGs extend CFGs by assigning probabilities to production rules, enabling the parser to choose the most likely structure based on learned preferences from a corpus.

WFSTs: Weighted Finite State Transducers.

WFSTs can be used to model human parsing preferences by assigning wts to transitions making preferred transitions.

NN Based Parsers:

modern NLP systems use neural n/ws, especially RNNs and transformers, to learn from large datasets.

Shift Reduce Parsers :-

One way to improve the efficiency of parsers is to use techniques that encode uncertainty, so that the parser need not make an arbitory choice and later backtrack.

These techniques were developed for use with unambiguous CFGs - grammars for which there is at most one interpretation for any given sentence.

A parser that maintains 2 stacks:

→ The parse stack, which contains parse states. and ~~the~~ grammar symbols.

→ the i/p stack, which contains the input and some grammar symbols.

shift reduce parsers are a type of deterministic parsing algm used for syntax analysis in NLP and compiler design.

The process involves two primary operations:

→ shift

→ Reduce

Stack:

The parser uses a stack to hold symbols and states during the parsing process.

Input Buffer: The input buffer contains the remaining input string to be parsed.

Actions:
shift: Move the next symbol from the input buffer to the stack.

Reduce: Replace a sequence of symbols on the stack with a single non-terminal symbol.

Working Mechanism:
The shift-reduce parser operates in the following steps:
Initialization: Start with an empty stack and the entire input string in the buffer.

shift (or) Reduce Decision: At each step, decide whether to shift the next input symbol on to the stack (or) to reduce the top elements of the stack according to a production rule.
Continue Until Complete: Repeat the process until the stack contains the start symbol and the i/p buffer is empty, indicating successful parsing (or) until no valid actions remain, indicating a parsing error.

Example:
"the cat chases the rat"

S → NP VP

NP → Det N

VP → V NP

Det → the

N → cat, rat

V → chases

Step-by-step process using a shift-reduce parser:

⇒ Stack : [ ]
I/p Buffer : [the, cat, chases, the, rat]

⇒ move "the" to stack

Stack : [the]
I/p Buffer : [cat, chases, the, rat]

⇒ Stack	⇒ I/p Buffer
[the cat]	[chases the rat]
[NP]	↓ NP → Det N [chases the rat]
[NP chases]	[the rat]
[NP chases the]	[rat]
[NP chases the rat]	[ ]
[NP chas NP]	↓ NP → Det N
[NP V NP]	
[NP VP]	↓ VP → V NP
[S]	↓ S → NP VP

The stack contain the start symbol 'S' and the input buffer is empty, indicating that the sentence has been successfully parsed according to the grammar.

Deterministic Parser :-

A deterministic parser can be built that depends entirely on matching parse states to direct its operation.

Instead of allowing only shift and reduce actions, a richer set of actions is allowed that operates on an i/p stack called the buffer.

This parser has the following operations:
→ Create a new node on the parse stack to push the symbol onto the stack.
→ Attach an i/p constituent to the top node on the parse stack.
→ Drop the top node in the parse stack into the buffer

Three other operations prove very useful in capturing generalizations in NL.
→ Switch the nodes in the first two buffer positions.
→ Insert a specific lexical item into a specified buffer slot.
→ Insert an empty NP into the first buffer slot.

Rules are organized into packets, which may be activated (or) deactivated during the parse.

→ Activate a packet
→ Deactivate a packet.

Deterministic parsers are a type of parser used in NLP and compiler design that make parsing decisions based on a fixed set of rules without backtracking.
They follow a predetermined path through the i/p string, making parsing efficient and predicatable.

<u>Key Features</u>:

Efficiency : Deterministic parsers operate in linear time O(n).

Predictability : They always produce the same output for the same i/p, without ambiguity.

No backtracking : These parsers do not revisit earlier decisions.

Types of Deterministic Parsers:

(i) LL Parsers

(ii) LR Parsers

LL Parsers:

Left to right, leftmost derivation.
LL parsers read the i/p from left to right and construct a leftmost derivation of the sentence. They are often implemented as recursive descent parsers.

LR Parsers:

Left-to-right, Rightmost derivation in reverse.
LR parsers also read the i/p from left to right but construct a rightmost derivation in reverse.
They are generally more powerful than LL parsers and can handle a wider range of grammars.

Advantages:
→ Speed
→ Simplicity.
→ Predictability.

Limitations:
→ Grammar restrictions
→ Ambiguity