UNIT- I Introduction to Natural language

The Study of Language, Applications of NLP, Evaluating Language Understanding Systems, Different Levels of Language Analysis, Representations and Understanding, Organization of Natural language Under standing Systems, Linguistic Back ground: Anoutline of English Syn tax.

Defination of NLP : NLP typically stands for Natural Language Processing, a field of AI that enables computers to understand and process human language in text and speech. It's used in applications like voice assistants, translation apps, and spam filters.

History of NLP :

1. **1950s – Origins:** NLP began with early **machine translation** and the **Turing Test** (1950) to assess machine intelligence.
2. **1960s–1970s – Rule-Based Systems:** NLP used **handcrafted grammar rules** and early programs like **ELIZA** and **SHRDLU** for basic language understanding.
3. **1980s – Knowledge-Based Approaches:** Focus shifted to **syntactic and semantic parsing** using **grammar models** like CFGs and ATNs.
4. **1990s–2000s – Statistical & Machine Learning Methods:** The rise of **probabilistic models** (HMMs, n-grams) and **machine learning** improved accuracy in tasks like translation and tagging.
5. **2010s–Present – Deep Learning & Transformers: Neural networks** and **Transformer models** (e.g., BERT, GPT) revolutionized NLP, enabling **human-like understanding and generation** of language.

1.1 The Study of Language

Language is a very important part of human life. It is what makes us different from other living beings. We use language to share our thoughts, express our feelings, and communicate with others.

Language can be written or spoken.

- Written language helps us record and store information for a long time. Books, notes, and documents help people in the future to learn what we know today.

- Spoken language is used in our daily lives to talk to others. It helps us express ideas, emotions, and needs quickly and easily.

The goal of this research is to create computational models of language in enough detail that you could write computer programs to perform various tasks involving natural language. The ultimate goal is to be able to specify models that approach human performance in the linguistic tasks of reading, writing, hearing, and speaking.

Computational models are useful both for scientific purposes for exploring the nature of linguistic communication and for practical purposes for enabling effective human-machine communication.

The major disciplines studying language

Discipline	**Typical Problems**	**Tools**
Linguists	How do words form phrases and sentences? What constrains the possible meanings for a sentence?	Intuitions about well-formedness and meaning; mathematical models of structure (for example, formal language theory, model theoretic semantics)
Psycholinguists	How do people identify the structure of sentences? How are word meanings identified? When does understanding take place?	Experimental techniques based on measuring human performance; statistical analysis of observations
Philosophers	What is meaning, and how do words and sentences acquire it? How do words identify objects in the world?	Natural language argumentation using intuition about counter-examples; mathematical models (for example, logic and model theory)

Computational Linguists	How is the structure of sentences identified? How can knowledge and reasoning be modeled? How can language be used to accomplish specific tasks?	Algorithms, data structures; formal models of representation and reasoning; AI techniques (search and representation methods)

Purpose of Studying Language

In **Natural Language Processing (NLP)**, the purpose of studying language is to understand how humans **comprehend and generate language** so that computers can do the same. NLP researchers aim to model how people **interpret words, phrases, and sentences** and how they **produce meaningful responses**.

By studying language, scientists develop computational models and algorithms that enable machines to perform tasks such as text understanding, translation, speech recognition, human communication.

The focus in NLP is not on how words are entered (typing or speaking), but on how computers can **analyze, understand, and generate language** once the input is recognized.

1.2 Applications of NLP

Applications of natural language processing (NLP) include

1. **Chatbots:** Chatbots use NLP to understand user messages and give meaningful replies. They help in customer support and online services.

2. **Sentiment Analysis:** It helps identify the emotions or opinions in text, such as positive, negative, or neutral reviews.

3. **Search Autocorrect:** NLP corrects spelling mistakes and suggests the right search terms automatically when typing.

4. **Language Translator :** It converts text or speech from one language to another instantly, like Google Translate.

5. **Voice Assistants :** Voice assistants like Siri and Alexa use NLP to understand spoken commands and respond accurately.

6. **Survey Analysis :** NLP helps analyze open-ended survey responses to find common themes or opinions.

7. **Targeted Advertising :** It studies user behavior and language patterns to show personalized ads.

8. **Hiring and Recruitment**: NLP tools scan resumes and match candidates' skills with job descriptions automatically.

9. **Grammar Checkers:** They detect and correct grammar, spelling, and punctuation errors in written text.

10. **Email Filtering :** NLP is used in spam filters to classify emails as spam or important based on their content.

1.3Evaluating Language Understanding Systems

Evaluating a language understanding system means checking how well it performs the tasks it was designed for. However, what counts as "understanding" can vary depending on the application.

Types of Evaluation

A. Black Box Evaluation

- In this method, the system is tested based on its **overall performance** without looking at its internal workings.
- Example: Asking questions to a question-answering system or having a conversation with a chatbot.

- It is useful for **final system testing**, but **misleading in early stages** because short-term success may not show real understanding.
- When all systems perform poorly, comparing small performance differences (like 50% vs. 40% accuracy) gives little insight into long-term progress.

B. Glass Box Evaluation

- Here, the internal components of a system are **examined and tested separately**.
- Each subpart (like syntax analyzer, semantic interpreter, etc.) is evaluated using specific tests.
- This requires **agreement on what components a language system should have**, which researchers are still debating.
- It helps in identifying which part of the system needs improvement.

Importance of Evaluation

- Evaluation is crucial because looking at a few examples or transcripts can easily **mislead people** into overestimating a program's intelligence.
- Humans tend to attribute meaning and intelligence to systems that produce understandable responses, even when no real understanding exists.

Example – The ELIZA Program (1960s, MIT)

- ELIZA was one of the first computer programs that tried to "talk" to people using natural language.It acted like a psychotherapist, asking questions and giving short replies to what the user typed.
- **MIT** stands for **Massachusetts Institute of Technology**, a famous university in the **United States** known for its research in science and technology.

How ELIZA Worked

- The program had a list of keywords (important words like *mother, always, alike, you,* etc.).
- For each keyword, it stored:
    1. A pattern to match the user's sentence.

2. A response rule (what the program should reply).

When a user typed a sentence:

1. ELIZA searched for a keyword in it.
2. If it found one, it followed the rule connected to that keyword.
3. It used parts of the user's sentence to make its reply sound natural.
4. If no keyword was found, it replied with general phrases like:
    - "Tell me more."
    - "Why do you say that?"
    - "Go on."

Example Conversation

User: Why are you looking at me?
ELIZA: Would you prefer it if I weren't looking at you?

How it worked:

- ELIZA saw the keyword "are you" and matched it to a stored pattern:
  ?X are you ?Y
  (This means "some words (?X) + are you + some more words (?Y)")
- The output rule said:
  Would you prefer it if I weren't ?Y?
- ELIZA replaced ?Y with the last part of the user's sentence ("looking at me").
- So it replied: "Would you prefer it if I weren't looking at you?"

1.4 The Different Levels of Language Analysis

When a computer tries to understand human language, it must know many things about **how language works** what words mean, how they combine, and how people use them in real life.
Besides language rules, it must also understand **world knowledge** like facts, situations, and reasoning to make sense of what people say.

1. Phonetic and Phonological Knowledge

- This is about **sounds in language** how words are spoken and heard.
- Example: The words *"two"* and *"too"* sound the same, but have different meanings.
- Important for **speech-based systems** like **Alexa** or **Google Assistant**.

2. Morphological Knowledge

- This level deals with **how words are formed** from smaller meaning units called **morphemes**.
- A **morpheme** is the smallest unit of meaning.
    - Example: *"Friend"* + *"-ly"* → *"Friendly"*
- Morphological knowledge helps systems recognize word forms like *play*, *playing*, *played*, etc.

3. Syntactic Knowledge

- Syntax is about **sentence structure** how words are arranged to form correct sentences.
- It tells what role each word plays (subject, verb, object, etc.).
- Example:
    - ✓ "The cat chased the mouse." (Correct syntax)
    - ✗ "Chased the mouse cat the." (Wrong syntax)

4. Semantic Knowledge

- This level deals with **meaning of words and sentences**.
- It focuses on **context-independent meaning** meaning that stays the same regardless of situation.
- Example: *"The sky is blue"* is meaningful because the words combine logically.
- But *"Blue sky is sleep"* is not semantically correct.

5. Pragmatic Knowledge

- Pragmatics is about **how language is used in real-life situations** and how context changes meaning.
- Example:
  - "Can you open the door?" is not just a question — it's usually a **request**.
- Pragmatic knowledge helps computers understand **intended meaning**.

6. Discourse Knowledge

- This level deals with how **sentences connect together** in a conversation or text.
- It helps interpret **pronouns** and the **flow of ideas**.
- Example:
  - "John bought a book. He liked it."
  - Here, *"He"* refers to John and *"it"* refers to the book.
- Discourse knowledge helps in maintaining **coherence** across sentences.

7. World Knowledge

- This includes **general knowledge about the world** that helps in understanding sentences.
- Example: If someone says, "I put the ice cream in the oven," we know it's strange because world knowledge tells us ice cream melts in heat.
- It also includes knowledge about **other people's beliefs, goals, and situations**.

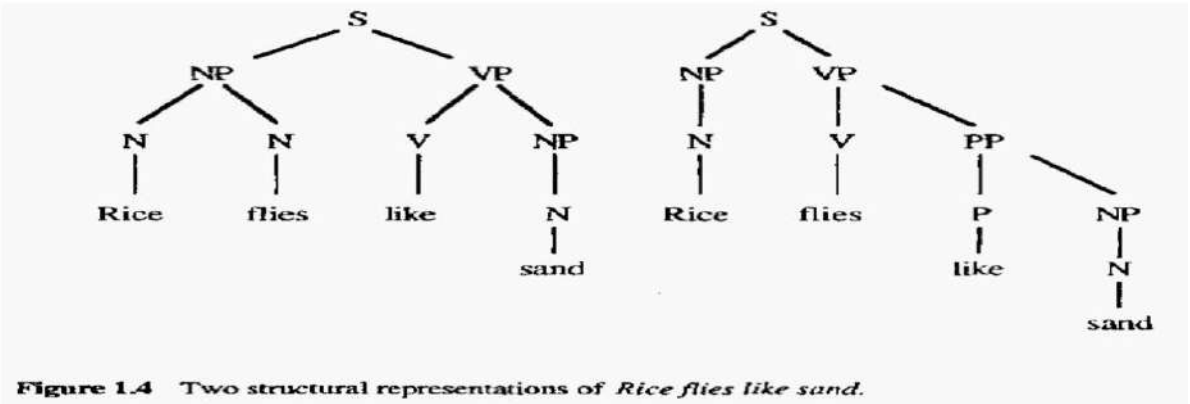


Figure 1.4 Two structural representations of *Rice flies like sand.*

Left Tree Interpretation: In this structure, "Rice flies" is parsed as a Noun Phrase (NP), where "Rice" modifies "flies" (likely referring to a type of fly). "Like" functions as a verb, and "sand" functions as a Noun Phrase (NP) that is the object of the verb. The sentence means that a specific type of fly enjoys or is fond of sand.

Right Tree Interpretation: In this structure, "Rice" is the subject Noun Phrase (NP), and "flies" is the verb. "Like sand" is a Prepositional Phrase (PP) that modifies the verb "flies", describing the manner in which the rice moves (similar to how sand might fly in the wind). The sentence means that rice moves in a way similar to sand.

1.5 Representations and Understanding

Understanding natural language involves creating precise **representations of meaning**. Using the sentence itself is insufficient because many words are **ambiguous** (e.g., "cook," "dish," "still") and a system must explicitly disambiguate word senses to model understanding.

Key Requirements for Meaning Representations:

1. **Precision and Unambiguity** – Each distinct reading of a sentence should correspond to a distinct formula.
2. **Structural Correspondence** – Representations should reflect the intuitive structure of sentences and preserve similarity for paraphrases.

Types of Representations:

- **Syntax (Sentence Structure)**:
    - Shows how words relate to each other in phrases.
    - Helps identify which words modify others and which are central.
    - Context-free grammars and tree structures are commonly used.
    - Example: "John sold the book to Mary" vs. "The book was sold to Mary by John" – same truth conditions but different syntactic structures.
    - Syntactic checks (like subject-verb agreement) help reduce ambiguity: e.g., "flying planes are/is dangerous."
- **Logical Form (Context-Independent Meaning)**:
    - Encodes **word senses** and **semantic roles** (AGENT, THEME, etc.).

- Helps resolve ambiguities based on verb-noun relationships.
- Example: In "Jack invited Mary to the Halloween ball," "ball" refers to a formal event because of the verb "invite."

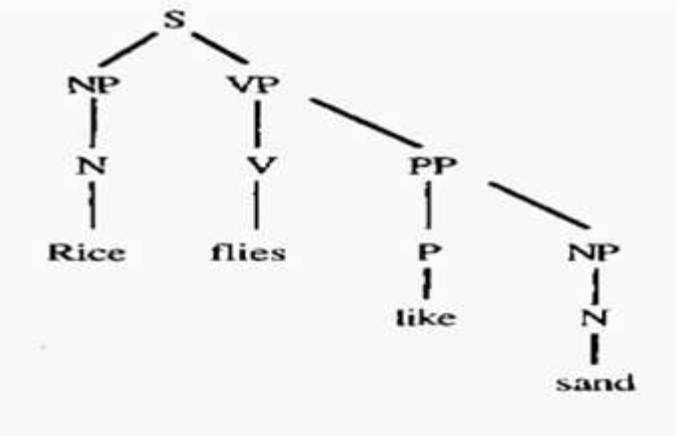
- **Knowledge Representation (Contextual/Domain Meaning)**:
  - Maps sentences and logical forms into a **general reasoning framework**.
  - Supports applications like **question-answering** or **story understanding**.
  - Typically uses **First-Order Predicate Calculus (FOPC)** for precise and well-studied representations.

Final Meaning Representation – Simplified

- The system needs a way to **understand and use sentences** in a real task.
- **Knowledge Representation (KR):**
  - Stores all important information for the system.
  - Turns sentence structure and meaning into something the system can act on.
  - Examples:
    - **Question-answering:** Turns a question into a database query.
    - **Story understanding:** Turns a sentence into a description of what's happening.
- **Language used:** Usually **First-Order Predicate Calculus (FOPC)** precise and well-studied.

  Example: " Rice Flies Like Sand"

  In this structure, "Rice" is the subject Noun Phrase (NP), and "flies" is the verb. "Like sand" is a Prepositional Phrase (PP) that modifies the verb "flies", describing the manner in which the rice moves (similar to how sand might fly in the wind). The sentence means that rice moves in a way similar to sand.



S →NP,VP

Noun Phrase (NP) → N→Rice

Verb Phrase (VP) → V, PP

VP → V→Flies

Prepositional Phrase (PP)→ P,NP

PP → P → Like

NP → N → Sand

1.6 Organization of Natural language Understanding Systems

Natural Language Understanding (NLU) is the branch of NLP that helps computers **interpret, analyze, and understand** human language.

1) What is NLU?

NLU is about **teaching computers to understand human language** the meaning behind words, sentences, and context.

To achieve this, an NLU system generally goes through **three levels of understanding**:

i. Syntactic Structure

- Understanding grammar, sentence structure, parts of speech.
- The system identifies how words are arranged.

ii. Logical Form

- Converting the sentence into a **meaningful, machine-interpretable structure**.
- Helps in understanding the logical relationships between words.

iii. Final Meaning Representation

- The system determines the **actual meaning** considering syntax, logic, and context.
- This is the final interpretation the system uses to answer questions or perform actions.

2) How NLU Works

NLU operates through a sequence of steps:

a) Interpretation Process

This is the first stage where the system transforms a raw sentence into structured information.

Main Steps:

i. Parsing

- The system breaks a sentence into its **grammatical components**.

Example:
Sentence: *"The boy ate an apple."*
Parse result:
NP → The boy
VP → ate an apple

ii. Lexicon

- A lexicon is like a digital **dictionary**.
- It contains:
  - Words
  - Meanings
  - Part-of-speech
  - Possible senses

Example:
Word: *bank*
Meanings:

- Financial institution
- River side

iii. Grammar Rules

Rules that determine **how words can be combined** to make valid sentences.

Example Rule:
S → NP + VP
NP → Det + N
VP → V + NP

b) Contextual Processing

After basic meaning is found, the system checks **context** to refine understanding.

Context helps in:

i. Identifying references

Who/what the sentence is talking about.

Example:
"John dropped the book. He picked it up again."
He → John
it → the book

ii. Understanding time

Determining when an action happened (past, present, future).

Example:
"I will meet you tomorrow."
→ future context

iii. Recognizing intentions

Understanding whether the sentence is:

- A question
- A command

- A request
- A statement

Example:
"Can you pass the pen?"
↳ Although phrased as a question → meaning is a **request**

3) Response Generation

If the NLU system needs to respond, it performs the following steps:

i. Plan the Response

- Decide the message to convey.
- Based on meaning, intention, and context.

Example:
User: "Where is the nearest ATM?"
System response planning → "Provide location"

ii. Generate Words

- Choose correct words and sentence structure.

Example:
Response plan → "Give location"
Generated sentence → "The nearest ATM is 200 meters away."

iii. Speech (Optional)

- Converting generated text into spoken output using speech synthesis.

Example:
Alexa/Siri saying the answer aloud.

    - validate user input
    - create system output
- Reduces system complexity

Advantages of NLU

Fewer errors : Integrated NLU systems combine syntax, semantics, and context, reducing chances of misinterpreting sentences.
Because all information is processed together, the system avoids mistakes that occur when components work separately.

Better handling of ambiguity: When words or sentences have multiple meanings, integrated systems use context and grammatical rules to choose the correct one.
This helps the model avoid confusion and interpret the meaning that best fits the situation.

Faster processing : Instead of performing each step one by one, modern systems process multiple layers (syntax + semantics + context) simultaneously.
This parallel processing reduces overall time and produces results more quickly.

More accurate interpretations: By considering grammar, context, and meaning together, the system can understand the sentence more precisely.
This leads to interpretations that are closer to what a human would understand.

Avoid unnecessary computations : Integrated systems filter out invalid or impossible interpretations early in the process.
This prevents the system from wasting time evaluating meanings that don't make sense.

1.7 LINGUISTIC BACK GROUND : Anoutline of English Syn tax

NLP is deeply rooted in linguistics, which provides theories and rules about how languages are structured and understood.

Linguistics is broadly divided into:

phonology → sound systems

morphology → stucture of woords

Syntax → stucture of sentences

Semantics → meaning of woords and sentences

pragmatics → meaning in context

WHAT IS SYNTAX

Syntax refers to the set of rules that govern how woords combine to form phrases, clauses, and sentences in a language.

For Example:

"the cat sat on the mat"

"Cat the mat on sat". (violates English Syntax)

In NLP, Syntax helps systems determine:

- word order
- Sentence structure
- Grammatical relations (subject, object, modifiers, etc.)

BASIC SYNTACTIC CONCEPTS IN ENGLISH

(A)parts of speech

English Syntax is built around categories like:

Part of Speech	Function	Example
Noun (N)	Names a person, place, thing, or idea	cat, school
Verb (V)	Shows an action or state	run, eat
Adjective (Adj)	Describes a noun	red, tall
Adverb (Adv)	Describes a verb or adjective	quickly, very
Preposition (P)	Shows relationship or position	on, under
Determiner (Det)	Introduces a noun	a, the, some
Pronoun (Pro)	Replaces a noun	he, she, it

Example : the (det)cat (n)sleep(v)peacefully(adv)

(B) phrase structure :

A phrase is a group of words that work together as a single unit.

Words group together into phrases:

Noun phrase (NP) → the big cat

Verb phrase (VP) → is sleeping

Prepositional phrase (PP) → on the mat

(C) Sentence Structure

The general English sentence structure is:

Sentence (S) → Noun Phrase (NP) + Verb Phrase (VP)

Example:

The dog barked loudly.

NP → The dog

VP → barked loudly

Types of Sentences:

Simple: Contains one clause.

Example: She runs fast.

Compound: Two independent clauses joined by and, or, or but.

Example: He cooked dinner and washed the dishes.

Complex: Main clause + subordinating clause.

Example: She smiled when she saw her friend.)

(D) Syntactic Representations in NLP

Understanding sentence structure is essential in Natural Language Processing (NLP). Two common ways to represent syntax are **Parse Trees (Constituency Grammar)** and **Dependency Grammar**.

(i) Parse Tress (Constituency Grammar)

Parse trees show the **hierarchical structure** of a sentence according to grammar rules.
A parse tree divides a sentence into **phrases** such as NP (Noun Phrase), VP (Verb Phrase), PP (Prepositional Phrase), etc.

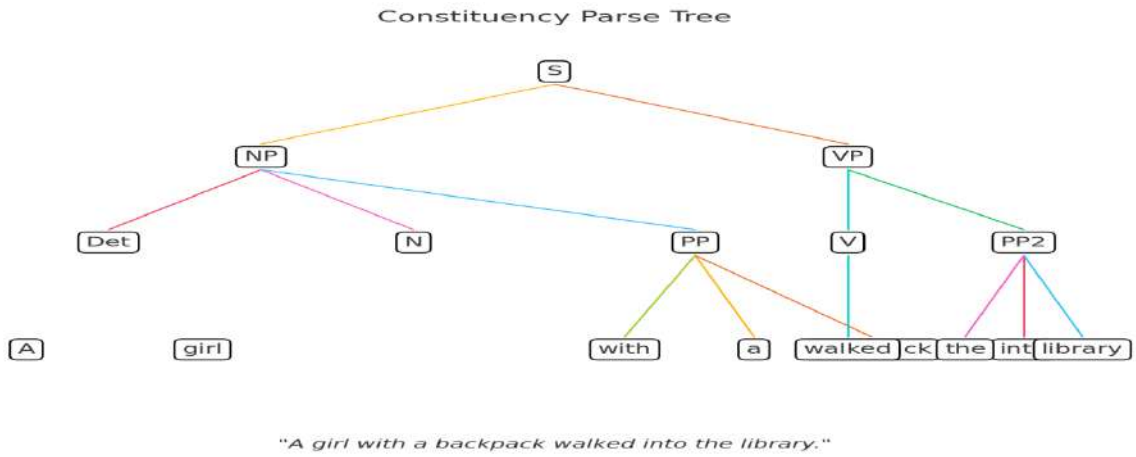
Example:

Sentence: *"A girl with a backpack walked into the library."*

Constituency Structure:

- **S → NP + VP**
- **NP → A girl with a backpack**
- **VP → walked into the library**
- **PP → with a backpack**
- **PP → into the library**

This structure shows how words group together into meaningful chunks.



(ii) Dependency grammar

Dependency Grammar focuses on **word-to-word relationships**, not phrases.
Each word depends on another, forming a tree with a **root** (usually the main verb).

Example:

Sentence: *"The scientist discovered a new planet yesterday."*

Dependencies:

- **discovered → root**
- **scientist → subject of "discovered"**
- **planet → object of "discovered"**
- **a, new → modifiers of "planet"**
- **yesterday → time modifier of "discovered"**

Why Dependency Grammar Is Useful:

- Captures **direct grammatical relations** (subject, object, modifier).
- Easier to use in machine learning models.
- Works well for **information extraction**, **semantic role labeling**, and **relation detection**

UNIT-II Grammars and Parsing

Grammars and Parsing – Top – Down and Bottom-Up Parsers, Transition Network Grammars, Feature Systems and Augmented Grammars, Morphological Analysis and the Lexicon, Parsing with Features, Augmented Transition Networks, Bayees Rule, Shannongame, Entropy and Cross Entropy.

2.1 Grammars and Parsing

In NLP, a grammar is a set of rules that define how to form well-structured sentences, while parsing is the process of using those rules to analyze an input sentence to determine its grammatical structure and syntactic relationships between words. This process generates a structured representation like a parse tree to reveal the sentence's underlying meaning

Grammars and Structure :

- **Goal**: To describe the **structure of sentences** and the rules that determine legal sentence formations in a language.
- **Representation**: Sentences are most commonly represented as **trees**, showing how a sentence is broken into subparts.

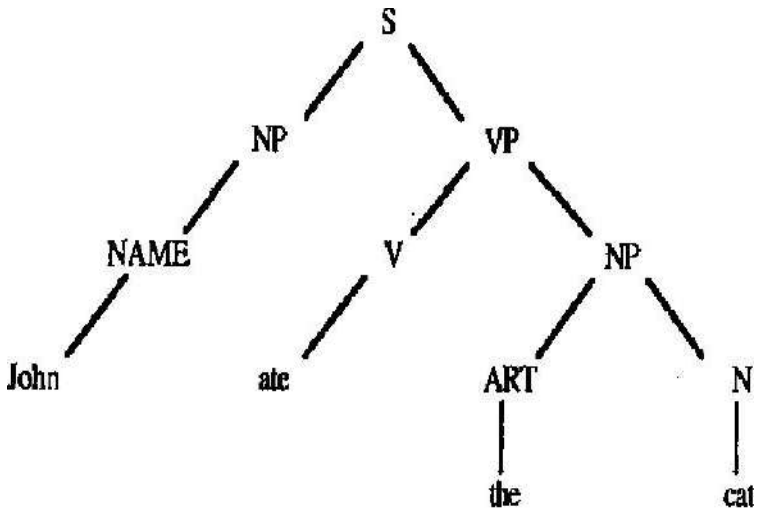


Figure 3.1 A tree representation of *John ate the cat*

1.	S → NP VP	5.	NAME → John
2.	VP → V NP	6.	V → ate
3.	NP → NAME	7.	ART → the
4.	NP → ART N	8.	N → cat

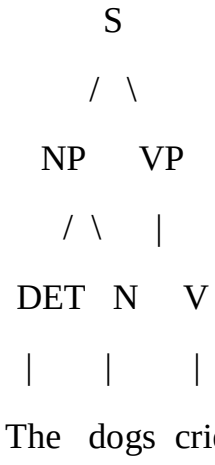
Grammar 3.2 A simple grammar

Parsing: Parsing is the process of analyzing a sentence to find its structure using grammatical rules. There are two main types of parsing: Top-Down Parsing and Bottom-Up Parsing.

1. Top-Down Parsing

- Starts with the **start symbol (S)** and tries to rewrite it into a sequence of words in the sentence.
- Works like **prediction**: it predicts what structure the sentence should have.
- Uses a **symbol list** and a **position counter** to track progress.
- If there are multiple ways to expand a non-terminal, it uses **backtracking** to try all possibilities.
- Can be seen as a **search problem**: using **depth-first** (stack) or **breadth-first** (queue) strategies.

Example:
Sentence: *The dogs cried*



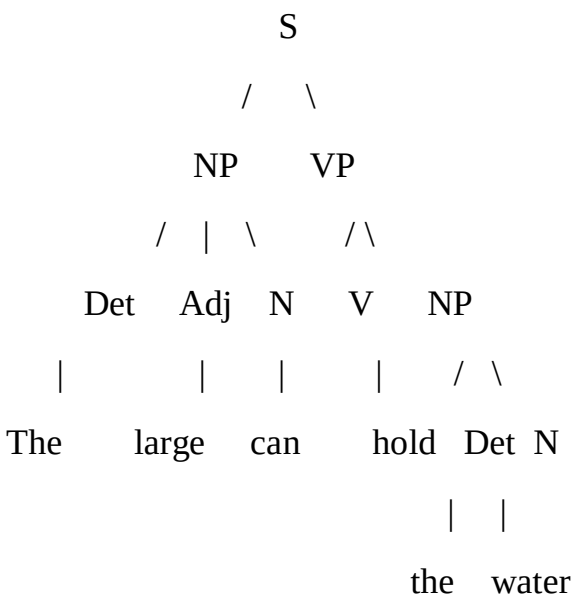
1. Start with S → rewrite to NP VP
2. NP → ART N → match "The" as ART and "dogs" as N
3. VP → V → match "cried" as V
4. Sentence successfully parsed because all words are matched.

Key Point: Top-down parses **predicts structure first and checks words**, may fail early if prediction is wrong.

2. Bottom-Up Parsing (Chart Parser)

- Starts with the **words in the sentence** and tries to combine them into larger constituents until reaching S.
- Works like **building from pieces**: matches sequences of words to the right-hand side of rules to create non-terminals.
- Uses a **chart** to store partially built constituents (active arcs and completed constituents) to avoid repeating work.
- More efficient than naive search because **each constituent is built only once**.

Example:
Sentence: The large can hold the water



1. Start with words: "The" → ART, "large" → ADJ, "can" → N/AUX/V, "hold" → V/N, "the" → ART, "water" → N
2. Build NP from ART + ADJ + N → "The large can" as NP

3. Build VP from V + NP → "hold the water" as VP
4. Combine NP + VP → S covering the whole sentence

Key Point: Bottom-up parses **starts with words and builds structure**, uses a chart to avoid repeated work, and can find multiple interpretations efficiently.

2.3 Transition Network Grammar (TNG)

Transition Network Grammar (TNG) is a way of representing grammar rules in **graph form**, where **states** represent stages in parsing and **transitions** represent grammar elements such as words or phrases.
It is widely used in **early NLP parsers**, especially **chart parsers**, **speech understanding**, and **syntactic analysis**.

What is a Transition Network Grammar?

Transition Network Grammar represents a grammar using:

- **Nodes (states)** → points in the parsing process
- **Arcs (transitions)** → represent words, categories, or sub-networks
- **Start state** → where parsing begins
- **End state** → where successful parsing ends

A TNG behaves like a **FINITE STATE MACHINE (FSM)** but with extra power:
• It can call other networks (like a function call).
• It can return to the calling network after finishing a subnetwork.

Recursive Transition Network (RTN)

When a network can call itself or others recursively.

RTNs are more powerful than regular FSAs because they can handle:

- Nested structures
- Recursion
- Infinite-length languages
- Natural language syntax

Structure of a Transition Network Grammar

A TNG consists of:

1. Start State : The point where parsing begins.

2. End/Accept State : Parsing is successful if the parser reaches this state after consuming all words.

3. Intermediate States : Used to represent different points in grammar rules.

4. Arcs (Transitions) : Each arc can contain:

a) Terminal symbols : Actual words

Example: “the”, “cat”, “runs”

b) Non-terminal symbols : Phrase categories

Example: NP (Noun Phrase), VP (Verb Phrase), PP (Prepositional Phrase)

c**) Push arcs Call/subroutine :** jumps to another network (like calling NP from S).

d) Pop arcs : Returns to the calling network when the subnetwork is completed.

RTN Parsing Algorithm

Step 1: Start

Begin at the **Start node** of the Sentence (S) network.
The input pointer starts at the first word.
The stack is empty.

Step 2: Read the next arc from the current node

There are four types of arcs.
Depending on the type of arc, do one of the following actions:

1. Start at S with empty stack
2. Match word arcs → move pointer
3. Push arcs → go to subnetwork
4. Pop arcs → return to previous network

5. Accept if:

- all words read
- stack empty
- at final node

Example

Sentence: THE CAT RUNS

Grammar Networks:

- **S → NP VP**
- **NP → Det N**
- **VP → V**

Parsing Steps

1. Start in **S**
2. Push **NP**
3. In NP, match **Det the**
4. Match **N cat**
5. Pop back to S
6. Push **VP**
7. In VP, match **V runs**
8. Pop back to S
9. No words left and stack empty → **Accept**

2.3 Feature Systems and Augmented Grammars

(A) Understanding the Problem

1. Singular vs. Plural Forms

In natural language, words must agree in number (singular or plural).

- ✅ ***a man*** **→ Correct**
- ❌ ***a men*** **→ Incorrect**

Because CFGs only care about structure, not agreement, they cannot detect mismatches between singular/plural forms.

(2) Subject–Verb Agreement

Subject–verb agreement ensures that the **subject** and **verb** in a sentence match in both **number** and **person**.
For example, we say *"He runs"* (not *"He run"*) because the singular subject "He" requires a singular verb form.
In plural forms, both the subject and verb must match, as in *"They run."*

Examples:

- ✅ *He runs* → subject (singular), verb (singular)
- ❌ *He run* → subject (singular), verb (plural)

This rule can be written as:

S AGR ?a → (NP AGR ?a) (VP AGR ?a)

Here:

- AGR = Agreement (includes person + number)
- ?a = Shared variable for agreement

So, the system checks that both NP and VP share the same features.

Sentence	Subject AGR	Verb AGR	Result
He runs	3rd Singular	3rd Singular	✅
He run	3rd Singular	Plural	❌
They run	3rd Plural	3rd Plural	✅

Gender Agreement

Many languages, such as **French**, **Spanish**, and **German**, require **gender agreement** between words in a phrase.
In these languages, nouns, adjectives, and articles have genders (masculine, feminine, or neuter).
The grammar must ensure that all the words in a phrase share the same gender feature.

Examples (French):

Sentence	Meaning	Result
✅ une fille intelligente	an intelligent girl (feminine singular)	Correct
❌ un fille intelligent	gender mismatch (masculine + feminine)	Incorrect

How do we represent agreement in grammars?

Context-Free Grammar (CFG)

- A CFG is ok, but struggles with agreement rules
- Simple Grammar Rule Example:
    - NP ART N
    - (a men, the man) both are correct as per the simple rule
    - Does not check if ART and N match in sing/plu form

(b) How Augmented Grammar Fixes the Problem

In normal **Context-Free Grammars (CFGs)**, the rules only define sentence structure, not grammatical correctness.
To handle grammatical agreement (like number, gender, and person), we use **Augmented Grammars**.
They extend CFGs by adding **features** small labels or attributes that describe grammatical properties of words.
These features make it possible for the grammar to check if the words **fit together correctly**, just like humans do when forming sentences.
For example, in CFG:

NP → ART N

This rule allows both
✅ *a man* and ❌ *a men*,
because CFG does **not know** that "a" is **singular** and "men" is **plural**.

To fix this, **Augmented Grammar** adds **features** small grammatical labels such as

number (singular/plural), **gender (masculine/feminine)**, or **person (1st/2nd/3rd)**.

These features make the grammar **more powerful and linguistically accurate**.

We rewrite the rule like this:

NP number ?n → (ART number ?n) (N number ?n)

Explanation:

- number is a **feature** that describes whether a word is singular or plural.
- ?n is a **variable** that must take the **same value** for both ART and N.
- This means both the **article (ART)** and the **noun (N)** inside the NP must share the **same number** (either singular or plural).

Thus, the grammar will **automatically reject mismatched pairs** like *a men* and only accept pairs like *a man* or *the dogs*.

How the Rule Works

Let's see how this rule checks agreement step by step:

Sentence	**ART Feature**	**N Feature**	**Result**
a man	Singular	Singular	✅ Matches (Correct)
a men	Singular	Plural	❌ Mismatch (Rejected)
the dog	Singular	Singular	✅ Matches (Correct)
the dogs	Plural	Plural	✅ Matches (Correct)
the men	Plural	Plural	✅ Matches (Correct)

Step-by-Step Working:

1. The parser reads the NP rule:
   NP number ?n → (ART number ?n) (N number ?n)
2. It assigns a **feature value** (?n) to both parts of the NP.
3. If both parts share the **same value** (e.g., both singular),
   ✅ the NP is **grammatically correct**.
4. If they differ (one singular, one plural),
   ❌ the NP is **rejected** as ungrammatical.

2. 4 Morphological Analysis and the Lexicon

Morphological Analysis:

Morphological Analysis is the study and computational processing of the internal structure and formation of words. It involves breaking down words into their constituent parts called morphemesthe smallest units of meaningsuch as roots, prefixes, suffixes, and inflectional endings.

It plays a crucial role in understanding how words convey meaning and grammatical functions in language. Morphological analysis helps identify the base or root forms of words, their derivations, and variations based on tense, number, gender, case, etc.

Importance of Morphological Analysis

1. **Understanding Word Formation:** It helps in identifying the basic building blocks of words, which is crucial for language comprehension**.**
2. **Improving Text Analysis:** By breaking down words into their roots and affixes, it enhances the accuracy of text analysis tasks like sentiment analysis and topic modeling.
4. **Facilitating Multilingual Processing:** It aids in handling the morphological diversity of different languages, making NLP systems more robust and versatile.

Key Techniques used

Stemming: A fast rule based process thatchops off words ending (suffixes / prefixes)& make it easier to analyze & process text.

ex: retrieval becomes retrieve
used in text summarization
likes → like

2) Lemmatization: Text normalization technique used to reduces words to their base (or)It uses chops off endings, But dose not change the meaning.

Ex: better → good

3) Neural Networks: It is a ML model used to solve complex functions, it work like a human brain.It converts words into numerical vectors called embeddings, which capture semantic meaning & processed to understand context.

The Lexicon

The lexicon is a comprehensive dictionary or database that lists words with their associated morphological, syntactic, and semantic information. It serves as a reference for morphological analyzers to look up word stems, affixes, and irregular forms.

Examples

Verb Forms and Changes:

- "want (base form)"
- "wants (3rd person singular)"
- "wanted (past tense)"
- "wanting (present participle)"

These illustrate regular verb formation by attaching appropriate suffixes for tense and person.

Examples of Multiple Meanings:

- "saw" (as a noun 'The saw') and "saw" (as a verb 'I saw him eat pizza', 'Jack wanted me to saw the board')

How lexicons are used

Sentiment analysis: A lexicon can assign a polarity score (e.g., +1 for positive, -1 for negative) to each word, and the overall sentiment of a text is determined by aggregating these scores.

Text classification and topic modeling: Lexicons can help identify the main subjects or categories within a document.

Machine translation: They provide a knowledge base for translating words and phrases between languages.

How lexicons are created

Hand-crafted: Experts can manually create and curate lexicons, which is a time-consuming process.

Automatically generated: Machine learning models can automatically generate lexicons by analyzing large corpora of text and identifying patterns and relationships between words.

2.5 PARSING WITH FEATURES

It means analysing a sentence using syntactic rules + extra linguistic information such as :

Number (singular / plural) : Indicates whether a word refers to one entity or more than one. It helps the parser enforce agreement between nouns and verbs.

Gender (masculine / feminine): Classifies nouns into gender categories. It is useful for agreement with pronouns, adjectives, and verbs in some languages.

Person (1st / 2nd / 3rd): Specifies who is involved in the action: speaker, listener, or others. It supports correct verb conjugation and agreement.

Tense / Aspect / Mood: Tense shows time of action, aspect shows its progression or completion, and mood expresses attitude or intention. Together, they clarify the meaning of the verb.

Agreement (subject–verb agreement): Ensures the subject and verb match in number and person. This helps maintain grammatical correctness in sentence structure.

Morphological features (root, prefix, suffix): Describe the internal structure of words. They help identify word forms, meanings, and grammatical functions during parsing.

These features help the parser build more accurate parse trees.

Ex:-

(i) The boy runs ✓

boy → Noun singular
runs → verb, singular, 3rd person

(ii) The boys runs ✗

boys → Noun Plural
runs → verb, singular, 3rd person

Parsing with features works

1. **Feature–Based grammars**
   context free grammar with feature constraints

Ex:–
S → NP [NUM = ?n] VP [NUM = ?n]

2. **Feature structures**
   It looks like a dictionary

   Ex: [Cat = V
   NUM = plur,
   Tense = PRES]

3. **Unification–Based Parsing**

   It is used in NLP models such as:

- HPSG (Head–Driven Structure Grammar) : It is the latest link in the chain of context free grammar ,which means that each phrase is assigned with a HEAD which controls the grammar behavior of that phrase.

  Ex: The Cat in the mat , the sentence is mainly talking about the Cat

- LFG (Lexical–functional Grammar) : It is the model used for analyzing of language in which different types of linguistic information are represented in separate dimentions are mapped by using principles.

4. **Modern NLP Parsers (Dependency + features)**

   Modern libraries like spaCy, stanZa, CoreNLP use universal features.

   Ex:–

word	POS	Dep	features
dogs	NOUN	nsubj	Number = plur
runs	VERB	root	Number = plur

Why features are important in NLP

- Improves accuracy for languages with rich morphology(Hindi, Telugu, Tamil)
- Helps detect grammar errors
- Helps in machine translation, speech recognition,text generation
- Helps parsers resolve ambiguity

2.6 Augmented Transition Networks

Augmented Transition Networks (ATNs) in NLP are a type of top-down parser used to analyze the sentence structure . they extends finite state machines with extra power by adding some components like :

Key Components & Concepts

States & Arcs (Transitions) : Like a state machine, ATNs have states and arcs connecting them, but arcs can trigger actions or require tests.

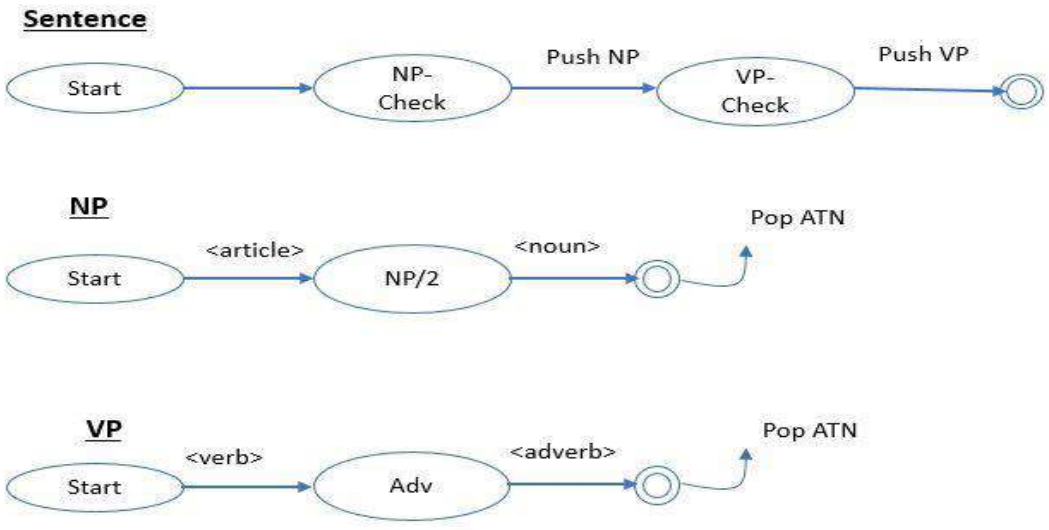
Registers (Memory) : Special variables (e.g., SUBJECT, VERB) that store features (like number, tense) of parsed words or phrases as they are processed.

Tests (Conditions) : Logic on arcs that must pass for traversal (e.g., checking if subject number matches verb number).

Actions : Operations performed when traversing an arc, often modifying registers or storing parsed parts.

Recursion : ATNs use "push" arcs to call other networks (e.g., an NP network) and "pop" back, allowing complex structures (like nested phrases) to be handled like function calls.

Working



Initialization: Start at the initial state of the main network (e.g., the Sentence S network).

Traversal: Match input words to arc labels (lexical or category, like Noun).

Action & Test: On an arc, perform an action (e.g., store the word in a Noun-Phrase register) and then check a test (e.g., does the subject agree with the verb?).

Recursion: If an arc pushes to another network (e.g., NP), parse that sub-network, store its results in registers, and then pop back.

Completion: Successfully parse the sentence by reaching a final state in the network, with relevant structure stored in registers.

Applications

a) **Syntactic Parsing:** The primary use is to parse sentences, identifying their grammatical structure (noun phrases, verb phrases) by traversing networks, much like a flowchart.
b) **Sentence Generation:** Generalized ATNs can work in reverse, taking a meaning representation and generating a surface sentence.
c) **Question Answering Systems :** ATNs help parse natural language questions to identify intent and expected answers.The parsed structure can be mapped to database or knowledge-base queries.
d) **Speech Understanding Systems :** In early speech systems, ATNs were used to guide parsing of spoken input.They helped resolve ambiguity by combining syntactic and semantic expectations.

2.7 Bayees Rule

Bayes' Theorem is a mathematical formula used to determine the conditional probability of an event based on prior knowledge and new evidence.
It adjusts probabilities when new information comes in and helps make better decisions in uncertain situations.

Formula

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

- **P(A)** and **P(B)** are the probabilities of events A and B; also,
- P**(B)** is never equal to zero.
- **P(A|B)** is the probability of event A when event B happens,
- **P(B|A)** is the probability of event B when A happens.

Example : 1

Disease is rare

Only 1% of people have the disease.
$P(D) = 0.01$

So 99% of people do not have the disease.
$P(\neg D) = 0.99$

Test Accuracy :

If a person has the disease, the test is positive 90% of the time.
$P(+ve \mid D) = 0.9$

If a person does not have the disease, the test still gives a false positive 5% of the time.
$P(+ve \mid \neg D) = 0.05$

Required :

We need to find the probability that a person really has the disease after getting a positive test.

Using Bayes Rule:

$$P(D \mid +ve) = \frac{[P(+ve \mid D)\, P(D)]}{[P(+ve \mid D)\, P(D) + P(+ve \mid \neg D)\, P(\neg D)]}$$

Substituting values:

$P(D \mid +ve) = (0.9 \times 0.01) / (0.9 \times 0.01 + 0.05 \times 0.99)$

$= 0.009 / (0.009 + 0.0495)$

$= 0.009 / 0.0585$

Final Answer:

$P(D \mid +ve) \approx 0.1538$ (about 15%)

Because the disease is very rare, the probability that a person truly has the disease after testing positive is only **about 15%**, showing the importance of considering prior probabilities using Bayes' rule.

Example: 2
An email contains the word "free". What is the chance that the email is spam?

20% of emails are spam, so P(spam) = 0.20.
80% of emails are not spam, so P(not spam) = 0.80.

Spam emails contain the word "free" 70% of the time, so P(free | spam) = 0.70.
Normal (not spam) emails contain the word "free" 10% of the time,

so P(free | not spam) = 0.10.

After seeing that the word "free" appears in the email, we want to find the probability that the email is spam, that is P(spam | free).

Using Bayes' theorem,

$$P(\text{spam} \mid \text{free}) = \frac{[P(\text{free} \mid \text{spam}) \times P(\text{spam})]}{[P(\text{free} \mid \text{spam}) \times P(\text{spam}) + P(\text{free} \mid \text{not spam}) \times P(\text{not spam})]}$$

Substituting the values,

$$P(\text{spam} \mid \text{free}) = (0.70 \times 0.20) / (0.70 \times 0.20 + 0.10 \times 0.80)$$
$$= 0.14 / (0.14 + 0.08)$$
$$= 0.14 / 0.22.$$

Final Answer:
$P(\text{spam} \mid \text{free}) \approx 0.64$, which means there is about a **64% chance** that the email is spam.

2.8 Shannongame

The Shannon Game in NLP is a classic experiment from information theory where a player guesses the next letter or word in a sequence, revealing language's predictability and measuring its entropy (uncertainty).

It is a core concept showing how language models (like a phone's predictive text) work by learning patterns to reduce the number of guesses needed. It links human language understanding to statistical compression and serves as a benchmark for model performance.

How it works

The Task:
Given a partial text (for example, “The cat sat on the …”), the player guesses the next character or word (such as “mat”, “rug”, or “floor”).

Predictability:
If the next element is highly predictable (for example, “q” is almost always followed by “u”), the game is easy. If it is less predictable, the uncertainty is higher.

Entropy & Compression:
The average number of guesses needed directly relates to the entropy of the language. Fewer guesses mean lower entropy and higher predictability, which is the goal of compression.

Role in NLP

Measuring Language Models:
It evaluates how well a language model predicts text. A good model minimizes the number of guesses, similar to how a human player performs.

Foundation for Modern Tech:
Modern predictive text, auto-completion, and large language models like GPT are conceptually based on this idea. They predict the next token using context to reduce uncertainty.

Benchmarking:
It helps estimate the theoretical limits of language compression. It also assesses whether models are approaching human-level understanding of linguistic structure and common sense.

2.9 Entropy and Cross Entropy

Entropy & Cross Entropy

Entropy measures the **uncertainty** or **average information content** in a probability distribution.Entropy means how **unsure or uncertain** we are about an outcome.Entropy was introduced by **Claude Shannon**.

If we are very confused, **entropy is high**.
If we are almost sure, **entropy is low**.

Entropy represents the **uncertainty of true data**.

Examples

1. **Fair Coin**
   Head = 0.5
   Tail = 0.5
   Uncertain outcome, so **entropy = 1 bit**.
2. **Biased Coin**
   Head = 0.9
   Tail = 0.1
   Less uncertainty, so **entropy < 1 bit**.
3. **Certain Event**
   Head = 1
   Tail = 0
   We already know the result, so **entropy = 0** (least entropy).

Entropy

$$\textbf{Entropy} = -\Sigma\, p_i \log_2(p_i)$$

$p_i \rightarrow$ probability of an event
$\log(p_i) \rightarrow$ information gained when the event happens

Example

Once upon a time there was a coin flip game.

En**tropy (uncertainty):**

Imagine flipping a **fair coin** (50% head, 50% tail).
You are most uncertain about what will happen, so entropy is **high**.

If the coin is **biased** (for example, 90% heads), there is less uncertainty.
So entropy is **low**.

Entropy measures **uncertainty**.

Cross Entropy

The **cross-entropy formula** measures the difference between two probability distributions: the **true distribution (P)** and the **predicted distribution (Q)**. It is defined as

$$\textbf{H(P,Q)}=-\sum \textbf{P(x)log(Q(x))H(P,Q)}$$

Here,

$P(x)$ is the true probability of outcome x.

$Q(x)$ is the predicted probability of outcome x.

$\log$ denotes the natural logarithm.

Cross Entropy (Guessing Game Story)

Now you are trying to **guess the coin's bias** (fair or biased).

True distribution (P):
The coin is actually fair (50% H, 50% T).

Your guess (Q):
You think the coin is 70% H and 30% T.

Cross entropy tells **how bad your guess (Q) is compared to the true distribution (P)**.Cross entropy indicates **how wrong your prediction is**.It represents the **difference between true data and predicted data**.

Example

Entropy

Entropy = $-\Sigma P(x) \log_2 P(x)$

For a fair coin:
H = 0.5, T = 0.5

Entropy calculation:
$H = -[0.5 \log_2(0.5) + 0.5 \log_2(0.5)]$
$= -[0.5(-1) + 0.5(-1)]$
= 1 bit

This means each coin flip has **1 bit of uncertainty**.
When uncertainty is more, entropy is higher.

Low Entropy (Biased Coin)

H = 90%, T = 10%

Entropy calculation:
$H = -[0.9 \log_2(0.9) + 0.1 \log_2(0.1)]$
≈ 0.47 bits

This shows **less uncertainty**, so entropy is low.

ross Entropy

Example:
True distribution (P): Fair coin (50% Head, 50% Tail)

P(H) = 0.5
P(T) = 0.5

Your guess (Q): Biased coin (70% Head, 30% Tail)

Q(H) = 0.7
Q(T) = 0.3

Cross Entropy Formula

$$H(P, Q) = -[\, P(H) \log(Q(H)) + P(T) \log(Q(T)) \,]$$

Substituting values:

$$H(P, Q) = -[\, 0.5 \log(0.7) + 0.5 \log(0.3) \,]$$
$$= -[\, 0.5\,(-0.357) + 0.5\,(-1.737) \,]$$
$$= -[\, -0.1785 - 0.8685 \,]$$
$$= 1.047 \text{ bits}$$

Since 1.047 bits > 1 bit, the guess is **worse than the true distribution**.

UNIT–III: Grammars for Natural Language

Grammar for Natural Language, Movement Phenomenon in Language, Gap Threading, Human Preferences in Parsing, Shift Reduce Parsers, Deterministic Parsers.

Grammar for Natural Language refers to the **set of rules and structures** used to describe how words are combined to form meaningful sentences in human languages. It helps in understanding the **syntax and structure** of sentences.

In Natural Language Processing (NLP), grammar is used to **analyze, interpret, and generate language** so that computers can understand and process human communication effectively.

3.1 Movement Phenomenon in Language

The **Movement Phenomenon** refers to cases where a word or phrase appears in a position **different from where it is interpreted** semantically. It Shows how sentences are rearranged for focus or grammatical function, This phenomenon allows for sentence variation, like forming questions or focusing on specific parts, revealing underlying grammatical structures.

Jack is giving Sue a back rub.
He will run in the marathon next year.
Is Jack giving Sue a back rub?
Will he run in the marathon next year?
As you can readily see, yes/no questions appear identical in structure to their assertional counterparts except that the subject NPs and first auxiliaries have swapped positions. If there is no auxiliary in the assertional sentence, an auxiliary of root "do" in the appropriate tense, is used:

John went to the store. Henry goes to school every day.
Did John go to the store? Does Henry go to school every day?
Taking a term from linguistics, this rearranging of the subject and the auxiliary is called **subject-aux inversion**. On the other hand, if you are interested in how it is done, you might ask one of the following questions:

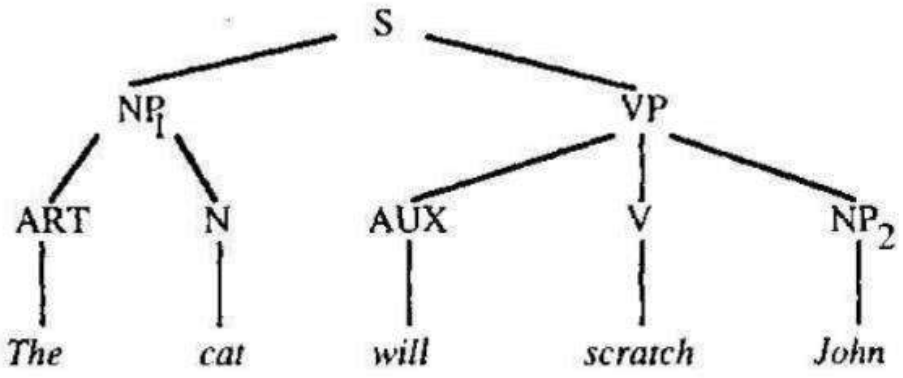
How will the fat man put the book in the corner?
In what way will the fat man put the book in the corner?
If you are interested in other aspects, you might ask one of these questions:
What will the fat man angrily put in the corner?
Where will the fat man angrily put the book?
In what corner will the fat man angrily put the book?
What will the fat man angrily put the book in?

Each question has the same form as the original assertion, except that the part being questioned is removed and replaced by a wh-phrase at the beginning of the sentence. In addition, except when the part being queried is the subject NP, the subject and the auxiliary are apparently inverted, as in yes/no questions. This similarity with yes/no questions even holds for sentences without auxiliaries. In both cases, a "do" auxiliary is inserted:

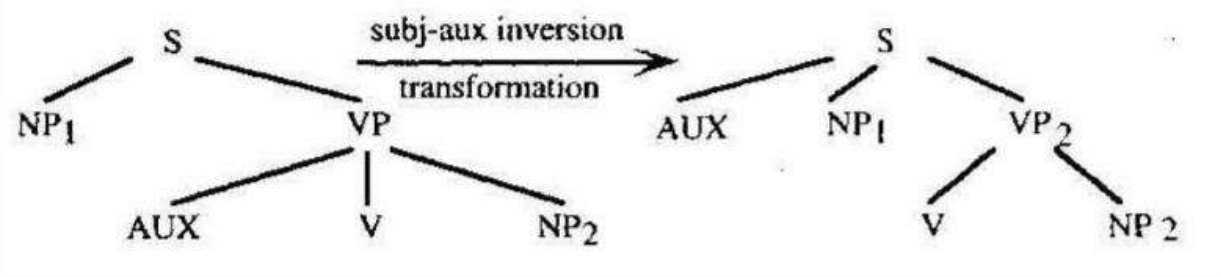
I found a bookcase.
Did I find a bookcase?
What did I find?

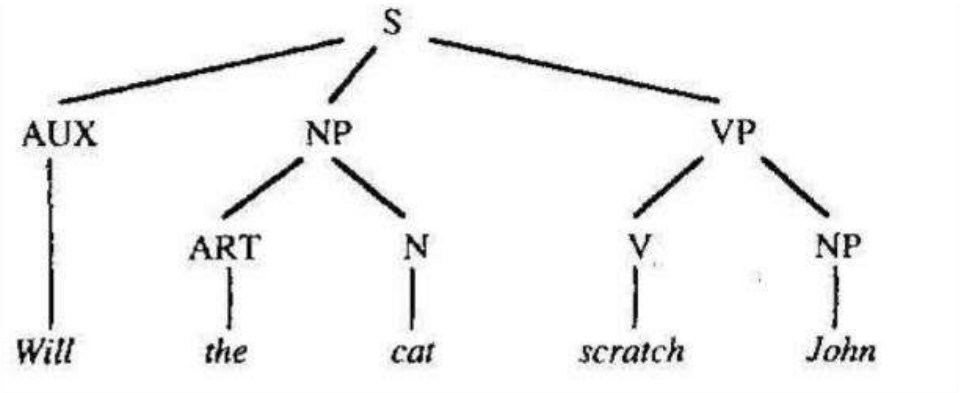
The term movement arose in transformational grammar (TG). TG posited two distinct levels of structural representation: surface structure, which corresponds to the actual sentence structure, and deep structure. A CFG generates the deep structure, and a set of transformations map the deep structure to the surface structure.

For example, the deep structure of "Will the cat scratch John?" would be:



The yes/no question is then generated from this deep structure by a transformation expressed schematically as follows





3.2 GAP THREADING

Gap threading is a parsing technique used in natural language processing, especially in logic and feature-based grammars, in which information about missing constituents (gaps) created by syntactic movement is passed through grammar rules so that fronted elements (fillers) can be correctly linked to their original grammatical positions, thereby handling long-distance dependencies in sentences.

Notation

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

This means:

- **position-in**: where parsing of the sentence starts
- **position-out**: where parsing ends
- **fillers-in**: list of gaps that are waiting to be filled
- **fillers-out**: list of gaps that remain unfilled after parsing

So an **S (sentence)** is valid only if all these constraints are satisfied.

How it works:

1. **Encountering a Filler**: When a fronted element (like a Wh-word, e.g., "What") appears, the parser adds its type (e.g., gap(np)) to the "in" list of the gap feature.
2. **Passing the Thread**: This gap list is passed down through subsequent grammar rules (like verb phrases).
3. **Finding the Gap**: When the parser reaches a point where a constituent (like an NP) is grammatically expected but missing (the "gap"), it checks the gap list.
4. **Connecting**: If the missing constituent matches the gap type on the list, the parser "fills" the gap, removes it from the list, and continues.

Example Sentence

Who did John see?

Who _____ did John see

NP ----------> gap (object of see)

This sentence has:

- **Filler**: *Who*
- **Gap**: missing **NP object** of *see*

Step 1: Filler Introduce

When the parser reads **"Who"**, it introduces an **NP gap**.

So the sentence starts with:

S(0, ?, [NP], ?)

Meaning:

- Start parsing at position 0
- One **NP gap** is expected
- It has not been filled yet

Step 2: Parsing Continues (Threading the Gap)

The parser processes:

did → John → see

No gap is filled yet, so the NP gap is **passed unchanged** through grammar rules.

Still:

fillers-in = [NP]
fillers-out = ?

This is the **threading** part.

Step 3: Gap Filled at Verb Object Position

The verb **see** normally requires an **NP object**, but it is missing.

The parser:

- Checks fillers-in
- Finds NP
- Uses it to fill the missing object position

The NP gap is **consumed**.

So now:

fillers-out = nil

Step 4: Final Representation

After successful parsing:

S(0, N, [NP], nil)

This means:

- A valid sentence exists from position 0 to N
- One NP gap was introduced

- That NP gap was successfully filled
- No unfilled gaps remain

Parsing problem:
How does the parser remember that *who* must be connected to the missing object position **later**, possibly many rules deeper?

That is exactly what **gap threading** solves.

3.3 Human Preferences in Parsing

→ Psycholinguists have studied this behaviour and discovered several guiding principles that explain why people prefer specific interpretations.

Key Principles

1) Garden-path sentences : why some sentences feel confusing.

A garden-path sentence is one where the reader initially interprets the sentence incorrectly, leading to confusion when they reach the end.

Ex: *The raft floated down the river sank.*

→ Confusion happens because humans try to build the simplest possible structure when reading.

The sentence has **two verbs**:

- **floated**
- **sank**

Normally, we expect **only one main verb**.

So when we see:

The raft floated…

our brain thinks:

"Okay, the main verb is *floated*."

Missing word "that"

In the correct structure, the sentence actually has a **hidden word**:

*The raft **that** floated down the river sank.*

But the word **"that"** is missing.

Without **"that"**, the brain cannot immediately see that:

- *floated down the river* is **describing the raft**, not the main action.

2. The minimal attachment principle:

This principle states that when we build a sentence structure in our minds, we prefer the interpretation that creates fewer branches (nodes) in a tree diagram.

Ex:
"The man kept the dog in the house."

- The man did the action
- The action is **kept**
- The dog is the object
- **"in the house"** tells us **where the action happened**

Minimal Attachment

- Attaching the PP to the VP creates **fewer branches**
- This is the **simplest syntactic structure**

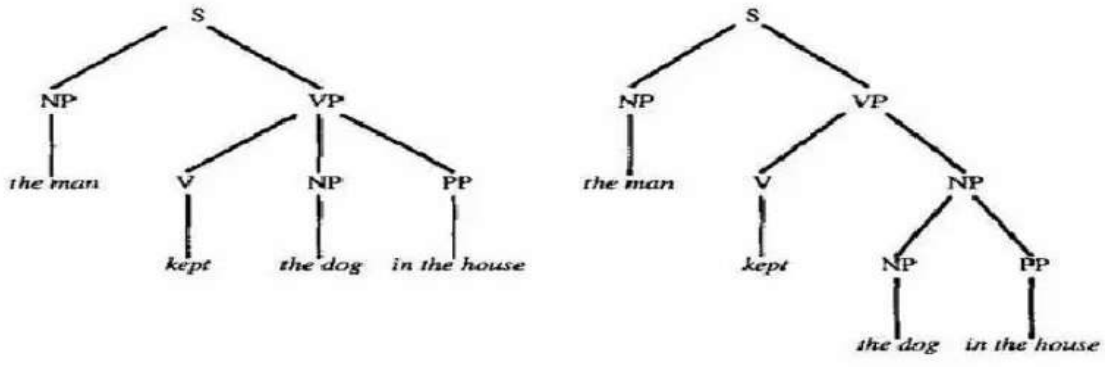


Fig: The interpretation on the left is preferred by the minimal attachment principle

3. Right Association (Late Closure): New words or phrases are attached to the most recent (rightmost) constituent possible.

- Example: "The girl in the chair with the spindly legs" prefers "with the spindly legs" modifying "chair".

Example

“The girl in the chair with the spindly legs”

The ambiguous part is:

“with the spindly legs”

It can describe:

1. **the girl**, or
2. **the chair**

Right Association

The brain prefers:

the chair with the spindly legs

So the meaning is:

- The girl is sitting **in the chair**
- The **chair** has spindly legs

Why this happens

- **“chair”** is the **most recent noun**
- Attaching the phrase to **chair** is faster
- This follows **Late Closure**

3. Lexical preference means:

The meaning and behavior of **individual words (especially verbs)** strongly influence how a sentence is understood.

So:

- Not all verbs behave the same

- Different verbs **expect different sentence patterns**
- These expectations can **override structural rules**

Example :

1. **I wanted the dog in the house.**
2. **I kept the dog in the house.**
3. **I put the dog in the house.**

All three look similar, but **they are understood differently** because of **lexical preferences of the verb**.

Sentence 1

1. **"I wanted the dog in the house."**

Preferred meaning:

I wanted **the dog that was in the house**.

Here:

- **"in the house"** describes **the dog**
- It tells us *which dog* I wanted

Why?

- **want** is a **mental-state verb**
- It does not describe an action happening in a place
- So the PP naturally modifies the **noun (dog)**

➢ **PP → NP (dog)**

(Other meaning is possible: *I wanted the dog to be in the house*, but it is weaker.)

Sentence 2

2. **"I kept the dog in the house."**

Preferred meaning:

I kept the dog **inside the house**.

Here:

- **"in the house"** tells us **where the keeping happened**

Why?

- **keep** is an **action verb**
- It involves control, location, or containment
- So the PP naturally modifies the **verb**

➢ **PP → VP (kept)**

(Other meaning is possible: *the dog that was in the house*, but it is less preferred.)

Sentence 3

3. "I put the dog in the house."

Only possible meaning:

I put the dog **into the house**.

Why?

- **put** *requires* a location
- Without a PP, the sentence feels incomplete
- The PP must describe **where the action happened**

➢ **PP → VP only**

NP attachment is **not possible** here.

3.4 Shift Reduce Parsers

A **shift-reduce parser** is a type of **bottom-up parser** used in Natural Language Processing (NLP) and compiler design to determine the grammatical structure of a sentence. It works by processing input words from left to right, building a parse tree from the leaves (words) up to the root (the start symbol of the grammar).

How it Works

The parser uses two main data structures:

Input buffer (or queue) : To hold the remaining unprocessed words.

Stack : to store and manage the symbols (words and partial phrases) that have been processed so far.

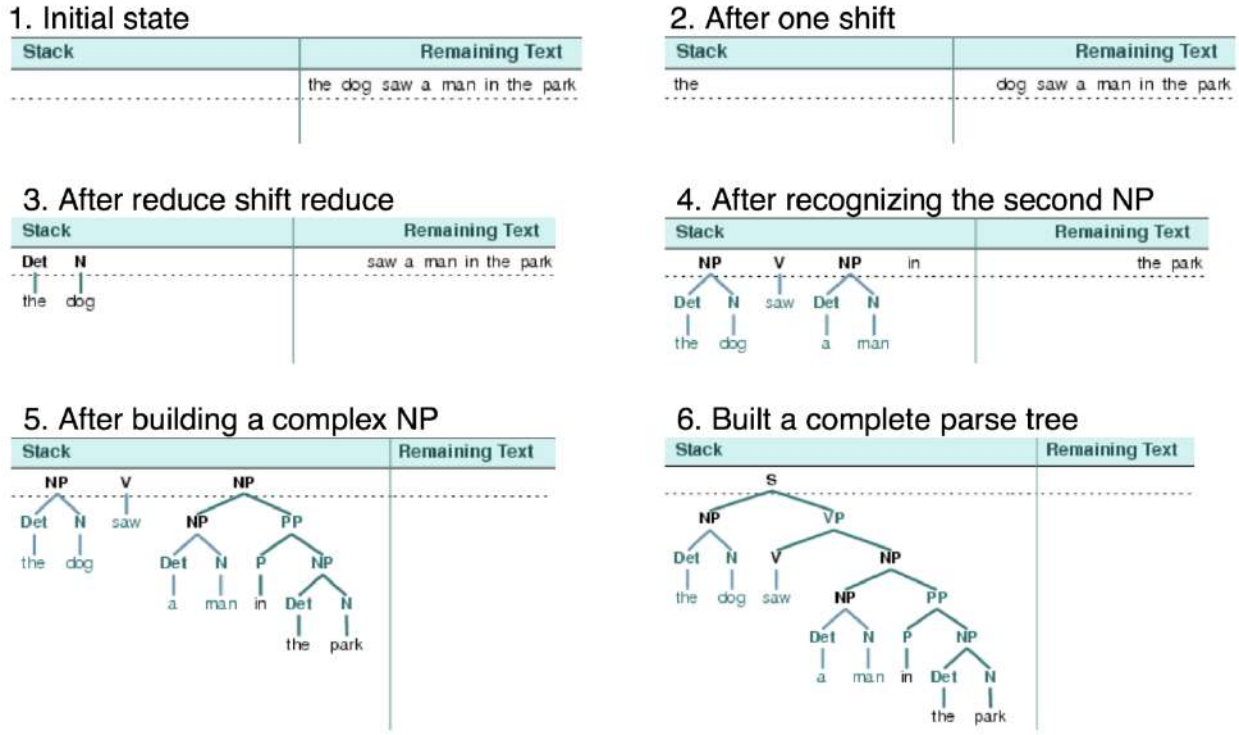
At each step, the parser performs one of four fundamental actions based on the current stack content and the next input symbol (lookahead):

Shift: The next symbol from the input buffer is moved onto the top of the stack. This operation effectively "reads" the next word.

Reduce: When a sequence of symbols at the top of the stack matches the right-hand side (RHS) of a grammar production rule (known as a "handle"), that sequence is replaced by the non-terminal symbol on the left-hand side (LHS) of the rule. This is akin to forming a larger grammatical phrase from smaller components.

Accept: The parsing is successfully completed when the entire input has been processed (input buffer is empty) and the stack contains only the grammar's start symbol.

Error: If the parser cannot perform a valid shift, reduce, or accept action, an error is reported, indicating an ungrammatical input.



- **Initial state:** The stack is empty, and the full sentence is in the remaining text.
- **After one shift:** The first word, "the", is shifted onto the stack.
- **After reduce shift reduce:** The parser applies grammar rules to form a Noun Phrase (NP) "the dog" on the stack.
- **After recognizing the second NP:** The verb "saw" and the next Noun Phrase "a man" are processed, showing the structure NP V NP.
- **After building a complex NP:** A Prepositional Phrase (PP) "in the park" is built and attached to "a man", forming a more complex NP structure.
- **Built a complete parse tree:** The parser successfully combines all elements into a complete Sentence (S) structure, indicating a successful parse of the entire sentence.

Advantages:

1. **Efficient and fast :** Shift–reduce parsing processes input in a single pass.It usually works in linear time.This makes it suitable for large programs.

2. **Deterministic parsing :** Parsing decisions are clear and rule-based.There is no need for backtracking.This improves accuracy and performance.

3. **Builds correct parse trees** : Each reduce action creates part of the parse tree.The tree is built systematically from bottom to top.This ensures correct syntactic structure.

4. **Uses simple data structures** : It mainly uses a stack and input buffer.These are easy to implement and manage.Memory usage remains efficient.

3.5 Deterministic Parsers

- A deterministic parser is a type of parser that follows fixed rules and cannot guess or backtrack.
- A deterministic parser reads a sentence and makes one decision at a time, following a fixed path without re-evaluating previous choices.

Ex:
“the boy eats an apple”

A deterministic parser will identify
the → as determiner
boy → as noun
eats → as verb

an → as determiner
apple → as noun

It follows one path based on a pre-defined grammar, without backtracking or making multiple guesses.

working

A deterministic parser works **step by step** by following **fixed grammar rules**.
At each step, it makes **only one decision** and never goes back to change earlier choices.
This ensures fast and predictable parsing without guessing.

Implementation methods

i. Finite State Automata (FSA)
Finite State Automata work like a **decision-making machine**.
Each state represents a grammar condition, and transitions happen based on input symbols.
The parser moves forward state by state until it reaches an accept or reject state.

ii. Shift–Reduce Parsing
Shift–reduce parsing uses a **stack** to store symbols while reading input.
It shifts input symbols onto the stack and reduces them using grammar rules.
This process builds the sentence structure from words to a complete parse tree.

Types of Deterministic Parsers

A **deterministic parser** follows fixed grammar rules and makes only **one decision at a time** without backtracking or re-evaluating earlier steps.

1. Shift–Reduce Parser (Bottom-Up)

- Starts parsing from **input symbols (words)** and builds up to the sentence structure.
- Uses a **stack** to shift input symbols and reduce them using grammar rules.
- Commonly used in **LR parsers** and compiler design.

2. Recursive Descent Parser (Top-Down)

- Starts parsing from the **start symbol (S)** and breaks it into smaller parts.

- Builds the parse tree from **top to bottom** like a tree structure.
- Uses **pre-defined grammar rules** and does not backtrack.

Example

Sentence: **"she runs fast"**

Grammar rules applied:

S → NP + VP
NP → she
VP → V + Adverb
V → runs
Ad → fast

This forms a correct sentence structure without changing earlier decisions.

Uses of Deterministic Parsers

Chatbots : Deterministic parsers help chatbots understand sentence structure correctly.They enable fast and accurate responses without ambiguity.

Grammar Checking Tools : They analyze sentence syntax to detect grammatical errors.This helps in providing correct and reliable language suggestions.

Speech Recognition Systems : Deterministic parsers assist in understanding spoken sentences.They improve accuracy by mapping speech to correct grammatical structures.

Advantages

Fast and Efficient Parsing : Parsing is done quickly by following fixed rules.This reduces processing time and improves performance.

Uses Less Memory : They rely on simple data structures like stacks.This keeps memory usage low and efficient.

Suitable for Real-Time Applications : They produce results without delay or backtracking.This makes them ideal for real-time systems and applications.

UNIT-IV

Semantic Interpretation :Semantic &Logical form, Word senses &ambiguity, The basic logical form language, Encoding ambiguity in the logical Form, Verbs &States in logical form, The maticroles, Speech acts & embedded sentences, Defining semantics structure model theory.

Language Modelling: Introduction, n- Gram Models, Language model Evaluation, Parameter Estimation, Language Model Adaption, Types of Language Models, Language Specific Modelling Problems, Multilingual and Cross lingual Language Modelling.

4.1 Semantic &Logical form

Semantic:

Semantic interpretation in NLP is the process of mapping natural language inputs into a structured, machine-understandable representation, often called a logical form. It bridges the gap between syntax (structure) and meaning (semantics) by resolving ambiguities and translating phrases into unambiguous representations like predicate logic.

In simple terms:

> **Syntax** tells us how words are arranged.
> **Semantics** tells us what those words mean together.

Example

Sentence:
"John eats an apple."

Syntax: Subject–Verb–Object structure

Semantics: Who performs the action and what is affected

Machine representation:

eats(John, apple)

Why Semantic Interpretation Is Needed

Natural language is often **ambiguous** and **context-dependent**, while computers need **clear and exact meaning**.

Example of Ambiguity

"I saw the man with the telescope."

This sentence can mean:

1. *I used a telescope to see the man.*
2. *The man I saw was holding a telescope.*

Key Concepts in Semantic Interpretation

1. Compositionality

Compositionality means that the meaning of a whole sentence is built from the meanings of its parts.

Example

"The cat sleeps."

- "cat" → an animal
- "sleeps" → an action
- Combined meaning → *the cat is sleeping*

The computer builds the meaning step by step using grammar rules.

2. Ambiguity Resolution

Some sentences have **more than one possible meaning**.
Semantic interpretation works to **select the correct meaning** based on structure or context.

Example

"She saw the boy with a camera."

• Did **she** use the camera to see the boy?
• Or did the **boy** have the camera?

Semantic interpretation tries to remove such confusion.

3.Syntax-Driven Semantic Analysis: Syntax-driven semantic analysis links semantic rules to grammar rules, allowing NLP systems to build meaning directly while parsing sentence structure**.** Using a **rule-to-rule approach**, such as

Logical Form

A logical form **is a** formal and exact representation of the meaning of a sentence**.**
It removes ambiguity and converts natural language into a format that a computer can understand, analyze, and reason with**.**

Logical forms are usually written using **predicate logic** or **lambda calculus**, which allow precise representation of meaning.

Why Logical Form Is Important

Natural language is often:

- Ambiguous
- Context-dependent
- Informal

Logical form makes meaning:

- **Precise**
- **Unambiguous**
- **Machine-processable**

Components of Logical Form

Logical form breaks a sentence into **basic, computable parts**.

1. Entities (Constants)

Entities represent **objects, people, or things** in the real world.

Examples:

- John
- Mary
- Dog
- Spot

These are treated as **fixed symbols** in logical expressions.

2. Predicates

Predicates describe:

- Actions
- Relationships
- Properties

They usually **take entities as arguments**.

Examples:

- eats(John, Apple) → John eats an apple
- tall(John) → John is tall

Predicates help describe **what is happening** or **what qualities an entity has**.

3.Variables &Quantifiers

Variables : In logic**, a** variable **(like x, y, z) represents an object or element in a group.**

Quantifiers: specify **how many** or **which entities** are involved in a statement.

a) Universal Quantifier (∀— "for all")

Used when a statement applies to **every member of a group**.

A) Universal Quantifier (∀) — "For All"

Used when something is true for **every member** of a group.

Example 1

Sentence:
All birds can fly.

Logical form:
∀x (bird(x) → can_fly(x))

Meaning:
For all x, if x is a bird, then x can fly.

Example 2

Sentence:
Every student studies.

Logical form:
$\forall x$ (student(x) $\rightarrow$ studies(x))

Example 3

Sentence:
All humans are mortal.

Logical form:
$\forall x$ (human(x) $\rightarrow$ mortal(x))

Example 4

Sentence:
Every dog barks.

Logical form:
$\forall x$ (dog(x) $\rightarrow$ barks(x))

b) Existential Quantifier ($\exists$ — "there exists")

Used when a statement says **at least one entity exists**.

Example 1

Sentence:
A girl is tall.

Logical form:
$\exists x$ (girl(x) $\wedge$ tall(x))

Meaning:
There exists at least one girl who is tall.

Example 2

Sentence:
Some students are intelligent.

Logical form:
$\exists x$ (student(x) $\wedge$ intelligent(x))

Example 3

Sentence:
There is a cat on the roof.

Logical form:
$\exists x$ (cat(x) $\wedge$ on_roof(x))

Example 4

Sentence:
Some flowers are red.

Logical form:
$\exists x$ (flower(x) $\wedge$ red(x))

4.2 Word senses &ambiguity

Word Sense in NLP

A **word sense** is the **specific meaning of a word in a particular context**. Many words in natural language are **polysemous**, meaning they have multiple meanings. However, in any sentence, only **one meaning is correct**, depending on the context.

Example:

- *"bank"*
  - financial institution
  - river edge

In the sentence: *"She deposited money in the bank"*, the word **bank** refers to a **financial institution**.
In: *"He sat on the bank of the river"*, it refers to a **river edge**.

Thus, understanding **word sense** is crucial for computers to process and interpret human language accurately.

Ambiguity in Natural Language Processing (NLP)

Ambiguity arises when a **word, phrase, or sentence can be interpreted in more than one way**. Ambiguity is one of the **major challenges in NLP**, because computers cannot easily determine the intended meaning without additional context.

Types of Ambiguity in NLP

1. Lexical Ambiguity

Occurs when a **single word has multiple meanings**.

Examples:

- *"bank"* → financial institution / river side
- *"bat"* → animal / sports equipment
- *"light"* → illumination / not heavy

Lexical ambiguity is solved by analyzing **context words** around the ambiguous word.

2. Syntactic (Structural) Ambiguity

Occurs when a **sentence can be structured in multiple ways**, resulting in different interpretations.

Example:

- *"I saw the man with the telescope."*
  - I used a telescope to see him
  - The man was holding a telescope

This ambiguity arises due to **sentence grammar and structure**.

3. Semantic Ambiguity

Occurs when a **sentence can have multiple meanings even though the grammar is clear**.

Example:

- *"Every student read a book."*
  - All students read the same book
  - Each student read a different book

Semantic ambiguity requires **understanding the meaning of the words** in context.

4. Referential Ambiguity

Occurs when it is **unclear what a pronoun or phrase refers to**.

Example:

- *"Ravi told Sita that she won."*
  → Who won, Ravi or Sita?

This type of ambiguity is common with **pronouns like he, she, it, they**.

5. Pragmatic Ambiguity (Additional)

Occurs when the meaning **depends on the situation or speaker's intention**.

Example:

- *"Can you pass the salt?"*
  - Literal question about ability
  - Polite request to hand the salt

Pragmatic ambiguity is solved by **understanding context beyond grammar and word meaning**.

4.3 The basic logical form language

The basic logical form language in Natural Language Processing (NLP) is a structured, symbolic representation used to convert ambiguous natural language sentences into a precise, computer-interpretable format. It serves as an intermediate, context-independent representation of meaning.

Logical operators **connect ideas** and form complex expressions.

(i) AND (∧)

Used when **both conditions are true**.

Example:

John is tall and smart.

Logical form:

tall(John) ∧ smart(John)

(ii)OR (∨)

Used when **at least one condition is true**.

Example:

Ritika is a doctor or an engineer.

Logical form:

doctor(Ritika) ∨ engineer(Ritika)

(iii)NOT (¬)

Used to **negate** a statement.

Example:

He is not hungry.

Logical form:

¬hungry(x)

Implication (→)

Used to express **if–then relationships**.

Example:

If something is a dog, then it is an animal.

Logical form:

dog(x) → animal(x)

4.4 ENCODING AMBIGUITY IN THE LOGICAL FORM

In **NLP**, a single sentence can have **more than one meaning**. This is called **ambiguity**.
Encoding ambiguity in Logical Form (LF) means representing **all possible meanings of a sentence together**, instead of choosing one meaning immediately.

Encoding Techniques

In Natural Language Processing, ambiguity is encoded in Logical Forms by using techniques that **represent multiple possible meanings at the same time**, without selecting one interpretation too early.

1. Quasi-Logical Forms (QLF)

QLF is an **underspecified semantic representation**.It captures the core meaning of a sentence while leaving ambiguous parts (like word meaning or quantifier scope) unresolved.

Example:
"Every student read a book"
QLF represents both:

- One common book
- Different books for different students

Both meanings are stored until more context is available

2. Underspecified Quantifier Scoping

Instead of fixing the order of quantifiers (every, some, a), LF **leaves their scope open**.

Example:
"Every boy loves a dog."
LF does not decide whether:

- Each boy loves the same one dog.
- Or Each boy loves his own dog.

Instead of choosing one, LF **keeps both interpretations**.

3. Disjunction "OR" Representation

Multiple interpretations are encoded using **logical OR (∨)**.

Example:
"bank" =

- bank(financial) **OR** bank(river)

Both meanings are stored until context resolves them.

4. Lexical Sense Variables

Ambiguous words are linked to **multiple semantic senses** using variables.

Example:
"bat" →

- bat(animal)
- bat(sports equipment)

LF associates the word with all possible senses.

5 .Syntactic Ambiguity

Sentence structure can also cause ambiguity when a sentence can be **parsed in more than one way**.

Example:

Sentence:
"She saw the girl with a knife."

Possible meanings:

1. **She used a knife** to see the girl
2. **The girl had a knife**

the **Logical Form (LF)** represents **both meanings** without choosing one.
Later, context is used to decide the correct interpretation.

4.5 Verbs & States in Logical Form

The verbs have mapped to appropriate senses acting as predicates in the logical form. This treatment can handle all the different forms but loses some generalities that could be captured. It also has some annoying properties. Consider the following sentences, all using the verb "break":

John broke the window with the hammer.
The hammer broke the window.
The window broke.

The verb "break" mapped to the same sense in each case.
The first seems to be a ternary relation between John, the window, and the hammer, the second a binary relation between the hammer and the window, and the third a unary relation involving the window. It seems you would need three different senses of break, BREAK1, BREAK2, and BREAK3, that differ in their arity and produce logical forms such as

1. (<PAST BREAK1> (NAME j1 "John") <THE w1 WINDOW> <THE h1 HAMMER>),
2. (<PAST BREAK2> <THE h1 HAMMER> <THE w1 WINDOW>), and
3. (<PAST BREAK3> <THE w1 WINDOW>)

Furthermore, to guarantee that each predicate is interpreted appropriately, the representation would need axioms so that whenever 1 is true, 2 is true, and whenever 2 is true, 3 is true. These axioms are often called meaning postulates.

In particular, the quasi-logical form for the sentence "John broke the window" using this abbreviation is

(<PAST BREAK!> e1 [AGENT (NAME j1 "John")]
[THEME <THE w1 WINDOW>])

It turns out that similar arguments can be made for verbs other than event verbs. Consider the sentence "Mary was unhappy". If it is represented using a unary predicate as

(<PAST UNHAPPY> (NAME j1 "Mary"))

In many situations using explicit event and state variables in formulas is cumbersome and interferes with the development of other ideas. As a result, we will use different representations depending on what is best for the presentation. For example, the logical form of "Mary sees John" will sometimes be written as

(PRES (SEES1 **l1** [AGENT (NAME **j1** "Mary")]
[THEME (NAME **m1** "John")]))
which of course is equivalent to
(PRES (j **l1** (& (SEES1 **l1**) (AGENT **l1** (NAME **j1** "Mary"))
(THEME **l1** (NAME **m1** "John")))))

4.6 The matic Roles(OR) Semantic Roles in NLP

Thematic roles, also called **semantic roles**, describe the **function played by each entity** (noun phrase) in an event or state expressed by a sentence. They explain **how participants are involved** in the action, beyond their grammatical position (subject or object).

In NLP, thematic roles help systems understand **who did what to whom, how, where, and why**, enabling accurate semantic interpretation.

Example:

The ball was kicked by John

Although *John* is not the grammatical subject, he is still the **Agent**.

Major Thematic Roles

1. Agent : The **Agent** is the entity that **intentionally performs** an action.

Key Characteristics:

- Usually **animate**
- Has **control** over the action
- Often appears as the **subject**, but not always

Examples:

John kicks the ball

Kick(John,ball)

- **Agent**: John
- **Theme**: ball

The ball was kicked by John
Agent remains **John**, despite passive voice.

2. Theme / Patient

Def: The **Theme** (or **Patient**) is the entity that is **affected, moved, or changed** by the action.

Difference:

- **Patient**: undergoes a change of state
- **Theme**: entity involved but not necessarily changed

(In practice, NLP often merges them.)

Examples:

John kicks the ball

- Theme/Patient: ball

Mary read the book

- Theme: book (not physically changed)

Logical Form:

Kick(e) ∧ Agent(e,John) ∧ Theme(e,ball)

e = Event Instance

3. Experiencer

Def: The **Experiencer** is the entity that **perceives, feels, or experiences** a psychological or sensory state.

Key Characteristics:

- Does **not control** the event
- Typically associated with **mental or emotional verbs**

Common Verbs:

fear, love, hate, see, hear, believe

Examples:

Mary fears dogs

- Experiencer: Mary
- Theme: dogs

John saw the accident

- Experiencer: John

Logical Form:

Fear(Mary,dogs)Fear(Mary, dogs)Fear(Mary,dogs)

4. Instrument

Def: The **Instrument** is the object **used to perform** an action.

Key Characteristics:

- Not the agent
- Enables or assists the agent

Examples:

John opened the door with a key

- Agent: John
- Instrument: key
- Theme: door

Logical Form:

$\exists e(Open(e) \wedge Agent(e,John) \wedge Instrument(e,key) \wedge Theme(e,door))$

5. Location

Def: The **Location** specifies **where** an event or state occurs.

Key Characteristics:

- Answers the question **"Where?"**
- Can be static or dynamic

Examples:

Children are playing in the park

- Location: park

The book is on the table

- Location: table

Logical Form:

$Play(e) \wedge Location(e,park)$

4.7 Speech Acts and Embedded Sentences in NLP

In NLP, understanding a sentence involves not only **what it says** (propositional content) but also **what the speaker intends to do** with it. This is captured by **speech act theory**. Additionally, natural language often contains **embedded (nested) sentences**, which require special semantic treatment.

What is a Speech Act?

A **speech act** is an action performed **by saying something**. When a speaker utters a sentence, they are not just producing words they are performing an act such as asserting, requesting, promising, or expressing emotion.

Embedded Sentences

An **embedded sentence** is a sentence that occurs **inside another sentence** and functions as part of it.

Example

John believes that Mary is happy.

- Main clause: *John believes*
- Embedded sentence: *Mary is happy*

Logical Form

Believe(John,Happy(Mary))

Types of Speech Acts

A **speech act** is what a speaker is **doing** when they speak.
When we say something, we may be **stating a fact, giving an order, making a promise, showing feelings, or changing a situation**.
NLP systems must understand this **intention**, not just the sentence form.

1. Assertives : Assertives are used to **tell something**, **state facts**, or **describe the world** as the speaker believes it is.

- Speaker **believes** the statement is true
- Can be **true or false**
- Used for information sharing

Examples

- *The sky is blue.*
- *John is eating an apple.*
- *Water boils at 100°C.*

Logical Form

Blue(sky)

2. Directives : Directives are used to make someone do something**.**

Includes

- Commands
- Requests
- Questions (used as requests)

Examples

- *Close the door.* (command)
- *Please open the window.* (request)
- *Can you close the door?* (question form, request meaning)

Important Idea

The **sentence form** and the **intended meaning** may be different.

Aspect	**Example**
Surface form	Question
Real intention	Request

3. Commissives : Commissives are used when the speaker **promises or commits** to doing something in the future.

Key Points

- Speaker takes **responsibility**
- Refers to **future actions**

Examples

- *I promise to help you.*
- *I will finish the work.*
- *I swear to tell the truth.*

4. Expressives : Expressives are used to **show feelings, emotions, or attitudes** of the speaker**.**

- Do **not describe facts**
- No true or false value
- Show **internal emotional state**

Examples

- *I am happy.*
- *I am sorry.*
- *Thank you!*
- *Congratulations!*

5. Declaratives : Declaratives **change the situation or reality** just by being spoken**.**

- Speaker must have **authority**
- Words themselves perform the action

Examples

- *I pronounce you husband and wife.*
- *You are fired.*
- *The meeting is now closed.*

4.8 Defining semantics structure model theory

- It is a **formal method** to understand the meaning of sentences.
- Instead of guessing meanings from mental ideas, it uses **mathematical models** to define when a sentence is **true or false**.
- The meaning of a sentence is explained by the **conditions that make it true** in a specific "model" or situation.

Core Components of a Semantic Model

A **model** represents a world or situation and has three important parts:

1. Domain (D)

- This is the **set of all objects/entities** we are talking about.
- It can include people, things, places, animals, etc.

Example:

D={John,Mary,apple,book}

This means our world contains these entities.

2. Interpretation Function (I)

- A function that tells us what words mean by linking them to things in the domain.
- It assigns:
    - **Constants** (names) to specific objects.
      For example, John → the person John in the domain.
    - **Predicates** (properties or relations) to sets or relationships between objects.

Predicates explained:

- A **predicate** is like a property or an action involving objects.
- Examples:
    - Apple(x)Apple(x)Apple(x) means "x is an apple." This applies to all apples in the domain.
    - Eat(x,y)Eat(x, y) means "x eats y."
    - This is a relation between two entities.

3. Truth Conditions

- These are **rules** to decide whether a sentence is **true or false** in the model.
- A sentence is **true** if what it says matches the objects and relationships in the domain.
- If not, it's **false**.

Example: "John eats an apple."

- **Logical form:**
  There is some thing xxx that is an apple, and John eats that thing.
  Written as: $\exists x(\text{Apple}(x) \wedge \text{Eat}(\text{John},x))$

  Model-theoretic meaning:
  We check if in our world (model) there really is an apple that John eats.

    - If yes, the sentence is **true**.
    - If no, the sentence is **false**.

Language Modelling

4.9 Introduction of Language Modelling

Language Modelling (LM) is a technique in Natural Language Processing (NLP) that helps computers understand and predict human language. In simple words, a language model predicts the next word in a sentence.

Example

Example Sentence: 'I am going to the ____.'
Possible predictions: school, market, park, temple.

The language model calculates which word is most probable based on previous words.

Correct Sentence: 'I like mango.'
Incorrect Sentence: 'Mango like I.'

The model gives higher probability to the correct sentence because it follows proper grammar.

Working of Language Model

A language model calculates the probability of a sequence of words.
Instead of predicting the whole sentence at once, it predicts word by word.

Example:
The cat is drinking milk.
It calculates:
P(The)
P(cat | The)
P(is | The cat)
P(drinking | The cat is)
P(milk | The cat is drinking)

Advantages of Language Modelling

1. **Helps in Machine Translation:**Improves fluency and grammatical correctness of translated text.Selects the most contextually appropriate word among multiple translation candidates.

2. **Improves Speech Recognition systems.**Reduces word error rate by predicting the most probable word sequence.Helps disambiguate similar-sounding words using context (e.g., “their” vs. “there”).
3. **Used in Chatbots and Virtual Assistants:**Enables natural, human-like responses.Supports context-aware conversations and intent understanding.
4. **Provides Text Prediction (mobile keyboards):**Suggests next words or phrases based on typing history and context.Speeds up typing and reduces spelling errors.
5. **Enhances Text Generation and Content Creation:**Assists in writing emails, reports, and summaries.Generates coherent paragraphs, stories, and code snippets.

4.10 n- Gram Models

An **N-gram model** is a statistical language model used in Natural Language Processing (NLP).
It predicts the next word in a sentence based on the previous **N–1 words**.
The model assumes that the probability of a word depends only on a limited history of previous words (Markov assumption).

Meaning of N

- **Unigram (N = 1):** Considers only one word at a time.
- **Bigram (N = 2):** Considers two consecutive words.
- **Trigram (N = 3):** Considers three consecutive words.
- **4-gram (N = 4):** Considers four consecutive words.

As N increases, the model uses more context but also requires more data.

Example

Example Sentence:
"I love natural language processing"

Unigrams:

I, love, natural, language, processing

Bigrams:

I love
love natural
natural language
language processing

Trigrams:

I love natural
love natural language
natural language processing

Each higher-order N-gram captures more contextual information.

Working of n-gram Model

Step 1: Training Data

1. I am going to school
2. I am going to school
3. I am going to market

Here, two sentences are the same and one is different.

Step 2: Create Bigrams

From Sentence 1:

- I am
- am going
- going to
- to school

From Sentence 2:

- I am
- am going
- going to
- to school

From Sentence 3:

- I am
- am going

- going to
- to market

Step 3: Count Bigram Frequencies

Bigram	Count
I am	= 3
am going	=3
going to	=3
to school	=2
to market	=1

Step 4: Calculate Probability

We want to predict the word after **"to"**.

Count of "to" = 3
(because "going to" appears 3 times)

Now calculate probabilities:

$$P(\text{school} \mid \text{to}) = \frac{2}{3} \approx 0.67$$

$$P(\text{market} \mid \text{to}) = \frac{1}{3} \approx 0.33$$

Step 5: Final Prediction

Since **0.67 > 0.33**, the model predicts:

"school"

4.11 Language model Evaluation

Language Model Evaluation means measuring how well a language model predicts the next word in a sentence.
It helps us know whether the model is performing well or not.

Why Do We Evaluate?

Evaluation is important because:

1. To compare different language models.
2. To improve model performance.
3. To check prediction accuracy.
4. To reduce errors in real-world applications.

Types of Evaluation

(A) Intrinsic Evaluation

- It directly measures how well the model predicts words.
- It uses mathematical measures.
- The most common measure is **Perplexity**.

Perplexity (Simple Meaning)

- Perplexity tells how confused the model is.
- **Low Perplexity = Good Model**
- **High Perplexity = Bad Model**

Example:
Correct sentence: *She is eating food.*
Wrong sentence: *Eating she food is.*

A good model gives higher probability to the correct sentence and lower probability to the wrong sentence.

(B) Extrinsic Evaluation

- It checks how the language model performs in real applications.
- If the application works better, the model is good.

Examples of Applications:

1. Machine Translation
2. Speech Recognition
3. Chatbots
4. Text Prediction

If these systems improve after using the model, then the model is considered effective.

(c) Human Evaluation

Human evaluation is when people manually check the output of a language model to judge its quality.

What Humans Evaluate:

1. **Fluency:** Does the sentence sound natural?

   Example: "She is eating food" ✓ vs "Eating she food is" ✗

2. **Grammar:**Correct sentence structure and word usage.
3. **Meaning / Relevance:**Does the sentence make sense in context?

   Example: Response of a chatbot should be relevant to the question.

4. **Coherence:**Does the text flow logically from one sentence to the next?
5. **Overall Quality / Preference:**Sometimes humans are asked to rank multiple outputs to select the best one.

Simple Comparison Example

Model A Perplexity = 40
Model B Perplexity = 90

Model A is better because it has lower perplexity.

4.12 Language Model Adaptation

Language Model Adaptation means modifying or adjusting a general language model so that it works better for a specific domain, topic, or user.

Example

Before Adaptation:
Sentence: The doctor performed ____
Prediction: homework (incorrect)

After Adaptation (trained with medical data):
Prediction: surgery (correct)

Types of Language Model Adaptation

1. Domain Adaptation

2. Topic Adaptation

3. User Adaptation

1.Domain Adaptation

This focuses on adjusting a model to a broad professional field with its own "language." It often involves **Continual Pre-training** on large, unlabeled datasets like medical journals or legal case files.

Explanation: The model learns technical vocabulary and relationships between specialized terms that don't appear in everyday conversation.

Example (Medical): A general model might see the word "culture" and think of art or society. An adapted medical model to understands "culture" as a laboratory test for bacteria.

2.Topic Adaptation

This narrows the model's focus to a specific subject matter, like sports or politics, often using **Fine-Tuning** or **Prompt Engineering**.

Explanation: The model is trained to prioritize certain themes and "entities" (like team names or policy terms) over others.

Example (Sports): In a general context, "he hit a home run" is a metaphor for success. In a sports-adapted model used for live commentary, it is recognized as a specific scoring event in a baseball game.

3.User Adaptation (Personalization)

This is the most granular level, where the model adapts to an individual's unique writing style, "quirks," or history.

Explanation: The model uses a person's past data (like emails or tweets) to predict what they might say next or to match their tone.

Example (Predictive Text): If you frequently use "Cheers!" as a sign-off in emails, your smartphone's predictive text will begin suggesting it as the next word after your name, even if it isn't a standard suggestion for other users.

4.13 Types of Language Models

A Language Model is a system that predicts the next word in a sentence. There are three main types of language models: Rule-Based, Statistical, and Neural Language Models.

1.Rule-Based Language Model:

A Rule-Based Language Model uses grammar rules written by humans. It checks whether a sentence follows correct grammatical structure.

Instead of learning from data, it follows fixed grammatical rules such as:

- Subject-verb agreement
- Tense rules
- Sentence structure rules

Example:
Correct Sentence: She eats rice.
Wrong Sentence: She eat rice.

The model checks grammar rules and identifies the correct sentence.

2. Statistical Language Model:

A Statistical Language Model uses probability and counts word frequency from training data.

Example Training Data:
I am going to school.
I am going to market.

After 'I am going to', the model predicts 'school' or 'market' based on which word appears more frequently.

3. Neural Language Model :A Neural Language Model uses deep learning and neural networks to learn patterns in language.

Instead of counting words manually, it learns automatically from large datasets using artificial neural networks.

Modern neural models use architectures like:

- Recurrent Neural Networks (RNN)
- Long Short-Term Memory (LSTM)
- Transformers

Neural models understand meaning, context, and relationships between words better than earlier models.

Example:
Sentence: The doctor performed ____
Prediction: surgery

4.14 Language-Specific Modelling Problems

Introduction

Language-Specific Modelling Problems refer to the challenges faced in **Natural Language Processing (NLP)** because each language has its own unique characteristics.:

Major Problems

1.Word Order Differences

Different languages follow different word order structures.

Example:

English: I eat mango. (Subject + Verb + Object)
Hindi: मैं आम खाता हूँ। (Subject + Object + Verb)

English generally follows **SVO (Subject–Verb–Object)** order, while Hindi follows **SOV (Subject–Object–Verb)** order.

2. Rich Morphology

Morphology refers to the structure of words and how they change form.

Some languages have many word forms created from a single root word. This creates a large number of variations.

Example in English: play, plays, playing, played.

3. Free Word Order

In some languages like Hindi, the order of words can change without changing the meaning.

Example:

राम ने सीता को फल दिया।
सीता को राम ने फल दिया।

Both sentences mean the same.

However, a language model may treat them as completely different patterns because the word positions change.

4. Compound Words

Some languages create long compound words by combining multiple words together.

Example (German): Long words formed by combining smaller words.

5. Lexical Ambiguity

Lexical ambiguity occurs when a single word has multiple meanings depending on context.

Example: The word "bank" can mean:

- A financial institution
- The side of a river

6. Low Resource Problem

Some languages do not have enough digital text data available for training.

Example: Tribal or regional languages may have very limited online content.

7. Script and Spelling Variations

Some languages use multiple scripts or informal spellings.

Example: Hindi written in English letters (Hinglish):
mai ghar ja raha hu

4.15 Multilingual and Cross-Lingual Language Modelling

Traditional language models are usually trained for **one specific language**, such as English or Hindi. However, in real-world applications like global search engines, translation systems, voice assistants, and chatbots, systems must understand and generate text in **multiple languages**.

To solve this problem, researchers developed two important approaches:

- **Multilingual Language Modelling**
- **Cross-Lingual Language Modelling**

Both approaches aim to support multiple languages, but they work in different ways.

Multilingual Language Modelling

A **Multilingual Language Model (MLLM)** is a single model trained on text data from multiple languages at the same time. Instead of building separate models for English, Hindi, Telugu, etc., one model learns all languages together.

The model shares internal representations across languages and learns similarities between them.

Simple Example

English: I am going to school.
Hindi: Main school ja raha hoon.
Telugu: Nenu paatashaala ku velthunnanu.

The same model understands and predicts words in all these languages.

This means:

- It recognizes patterns in English.
- It learns grammar in Hindi.
- It understands sentence formation in Telugu.
- It stores this knowledge in shared neural network parameters.

How Multilingual Models Work

1. A shared vocabulary is created (often using subword tokenization).
2. Training data from multiple languages is combined.
3. The model learns common linguistic patterns.
4. Similar words across languages may get similar internal representations.

Advantages of Multilingual Models

1. **One model handles many languages** – reduces the need for separate models.
2. **Saves memory and storage** – efficient for deployment.
3. **Helps low-resource languages** – knowledge from high-resource languages (like English) helps smaller languages.
4. **Easier maintenance** – updating one model is simpler than maintaining many.
5. **Better generalization** – shared learning improves robustness.

Cross-Lingual Language Modelling

Cross-Lingual Language Modelling focuses on **transferring knowledge from one language to another**.

Instead of just learning multiple languages together, cross-lingual models connect meanings across languages so that learning in one language improves performance in another language.

The goal is **knowledge transfer**.

Simple Example

English: The cat is sleeping.
Hindi: Billi so rahi hai.

The model learns that:

- “cat” corresponds to “billi”
- “sleeping” corresponds to “so rahi hai”

This alignment allows the model to:

- Translate text
- Understand similar meanings across languages
- Perform tasks even if trained in only one language

How Cross-Lingual Models Work

Cross-lingual models create a **shared semantic space**, where words with similar meanings across languages are placed close together.

For example:

- dog (English)
- perro (Spanish)
- kutta (Hindi)

All may have similar vector representations internally.

This enables:

- Cross-lingual transfer learning
- Zero-shot learning (performing tasks in a language without direct training data)
- Multilingual embeddings

Advantages of Cross-Lingual Models

1. Useful for machine translation.
2. Reduces need for large data in every language.
3. Helps low-resource languages using high-resource language knowledge.
4. Improves cross-language tasks like information retrieval.
5. Enables zero-shot or few-shot learning across languages.

UNIT-V

Machine Translation Survey: Introduction, Problems of Machine Translation, Is Machine Translation Possible, Brief History, Possible Approaches, Current Status. **Anusaraka or Language Accessor:** Background, Cutting the Gordian Knot, The Problem, Structure of Anusaraka System, User Interface, Linguistic Area, Giving up Agreement in Anusarsaka Output, Language Bridges.

Multilingual Information Retrieval Introduction, Document Pre-processing, Monolingual Information Retrieval, CLIR, MLIR, Evaluation in Information Retrieval, Tools, Software and Resources.

Multilingual Automatic Summarization Introduction, Approach esto Summarization, Evaluation, How to Build a Summarizer, Competitions and Datasets.

Machine Translation Survey

Introduction

Machine Translation (MT) refers to the use of computers to automatically translate text or speech from one language to another.
Its main goal is to help people communicate across different languages quickly and efficiently.

Today, MT is used in:

- Google Translate
- Websites that translate pages automatically
- Chat apps
- Customer service chatbots
- Travel applications

MT has grown from simple word substitution to advanced systems that try to understand full meaning and context.

Problems of Machine Translation

Machine translation (MT) systems face several persistent challenges that hinder their ability to achieve human-like accuracy and fluency. These problems stem from the inherent complexities and nuances of human language.

1. **Ambiguity in Language** Many words and phrases have multiple meanings (polysemy), and without a deep understanding of the surrounding context, machines struggle to select the correct interpretation. A classic example is the word "bank," which could mean the edge of a river or a financial institution.
2. **Idioms and Figurative Expressions** Idiomatic expressions and slang often cannot be translated literally. A phrase like "It's raining cats and dogs" would be nonsensical if translated word-for-word into another language; the machine needs to recognize the entire phrase as a single semantic unit meaning "it is raining very heavily."
3. **Syntax and Grammatical Differences** Different languages have distinct grammatical rules and sentence structures (e.g., word order). MT systems must accurately transform the syntax from the source language to match the target language's structure, which can be particularly complex for long or involved sentences.
4. **Lack of Context and Cultural Understanding** Machines often miss the subtle context, tone, emotional undertones, or cultural references embedded in language. This limitation can result in translations that sound unnatural, overly formal, or completely miss the speaker's true intent.
5. **Limited Resources for Less-Common Languages** For languages with fewer digital resources such as limited datasets, online texts, and dictionaries there is insufficient data for MT systems to train on effectively. This data scarcity leads to significantly lower translation quality compared to resource-rich languages like English or Spanish.

Is Machine Translation Possible?

Machine Translation (MT) refers to the automatic translation of text or speech from one language to another using computers. The question of whether MT is truly possible has been debated since the 1950s. Although modern systems like Google Translate and DeepL look advanced, MT still faces many challenges due to the complexity of human language. Below is a detailed explanation.

1. Complexity of Human Language

Human languages are rich, flexible, and full of hidden meanings.
Words can have multiple meanings (ambiguity), sentences may carry cultural references, and many expressions cannot be understood literally. Because of this, getting a perfect translation is extremely difficult. A machine needs to understand

context, culture, tone, and grammar to translate accurately—tasks that even humans sometimes struggle with.

2. Early Belief: MT is Impossible

In the early years (1950s–1960s), researchers believed MT was almost impossible. The first attempts used simple rule-based systems that relied on grammar rules and dictionaries. These systems could translate very basic, direct sentences but failed with real-life language. Due to rigid rules, they could not handle ambiguity, idioms, long sentences, or natural conversation. This led many experts to believe that accurate MT could never be achieved.

3. Statistical and Neural Advances Made MT Possible

In the 1990s, Statistical Machine Translation (SMT) proved that translation could improve through large amounts of bilingual data. This shifted the belief from "impossible" to "possible with limitations." Later, in the 2010s, Neural Machine Translation (NMT) made a huge breakthrough. Using deep learning, NMT systems began producing natural, fluent, and context-aware translations. They are now widely used by Google, Microsoft, and DeepL.

4. MT is Possible, But Not Perfect

Modern Machine Translation can translate:

- Short sentences accurately
- Common, everyday language
- Popular and well-documented languages
- General topics with good fluency

However, MT still has limitations. It struggles with:

- Idioms, jokes, expressions
- Cultural meanings
- Long, complex sentences
- Technical, legal, or medical language
- Low-resource languages with fewer datasets

So, MT is *possible* but not perfect. It can support humans but cannot fully replace human translators in specialized contexts.

5. Future Possibilities

With continued advances in artificial intelligence, MT systems are expected to:

- Understand deeper context
- Capture tone, emotion, and cultural meaning
- Handle low-resource languages better
- Produce professional-level translations
- Learn continuously from new data

Researchers believe MT may reach near-human accuracy in the future, but achieving 100% perfection is still a challenge because human language is dynamic and continuously evolving.

Brief History

Machine Translation (MT) has evolved through several distinct phases:

- **1950s – Early Ideas:** MT research started around 1954, using basic **Rule-Based MT (RBMT)**, which could only translate simple sentences.
- **1960s to 1980s – Rule-Based Systems:** Linguists manually encoded thousands of grammar and vocabulary rules. These systems struggled with the complexities of natural language.
- **1990s – Statistical MT (SMT):** A major breakthrough occurred when systems began using probability and large datasets to select the best translation, exemplified by early IBM statistical models.
- **2010s – Neural Machine Translation (NMT):** The current dominant paradigm uses deep learning and neural networks to automatically learn language patterns, resulting in much more natural-sounding translations.

Possible Approaches

Machine translation can be achieved through several approaches, each with distinct methodologies and characteristics:

- **Rule-Based Machine Translation (RBMT)**: This approach relies on extensive bilingual dictionaries and predefined linguistic rules. It operates by breaking down sentences into components like subject, verb, and object. While effective for formal, structured documents, it struggles with the nuances and variations of natural, conversational language.

- **Statistical Machine Translation (SMT)**: SMT learns translation patterns from vast amounts of bilingual text data. It employs statistical models and probabilities to choose the most probable translation for a given phrase. This method generally offers improved accuracy over RBMT, but it can sometimes result in translations that feel choppy or unnatural.
- **Neural Machine Translation (NMT)**: NMT utilizes deep learning and artificial neural networks to understand context more effectively than previous methods. It produces fluent, human-like translations and is the technology powering popular services such as Google Translate, Microsoft Translator, and DeepL. Its primary limitations are the significant need for large training datasets and considerable computing power.

Current Status

Machine translation (MT) has reached an advanced state, primarily driven by **Neural Machine Translation (NMT)** and the integration of **Large Language Models (LLMs) and Generative AI**. This technology provides fast, high-volume, and increasingly nuanced translations, fundamentally changing global communication and the translation industry itself.

Current Key Aspects

- **Dominance of Neural MT and Generative AI:** Rule-based and statistical methods have been largely superseded by NMT models, particularly those based on the Transformer architecture and advanced AI. These models can capture broader context, idiomatic expressions, and produce more natural-sounding output than previous technologies.
- **Significant Quality Improvements:** The perceived quality of machine translation has improved substantially in recent years, with a notable reduction in user dissatisfaction regarding its ability to handle cultural nuances and context. Research suggests some systems can approach human parity in specific, limited domains.
- **Widespread Integration and Applications:** MT is integrated into numerous applications, including:
    - **Real-time communication:** Chatbots, video conferencing tools, and mobile apps (e.g., Google Translate, DeepL) offer instant text, speech, and even image translation for everyday use and travel.

    - **Business and Localization:** Companies leverage MT for high-volume content, such as e-commerce product descriptions, legal documents, and internal communications, to expand globally faster and reduce costs.
    - **Regulated Industries:** Healthcare and life sciences use MT for patient documents and communications, though human review remains critical due to the high stakes of errors.
- **Human-in-the-Loop Workflow:** Despite advancements, the "fully automatic high-quality translation" goal for all content types remains elusive. **Machine Translation Post-Editing (MTPE)** is a common hybrid approach where human linguists refine machine-generated content, combining the speed of AI with human accuracy and cultural understanding, particularly for high-value or creative content.
- **Market Growth:** The machine translation market is experiencing significant growth, projected to reach over $3 billion by 2027, driven by the increasing need for content localization and cost-efficient solutions.

Challenges and Future Directions

Current challenges include data privacy concerns, the risk of bias from training data, difficulty with highly creative or nuanced content (humor, metaphors), and less accurate performance for low-resource or complex language pairs.

Future directions involve:

- **Enhanced Contextual Understanding:** Further improvements in models that can consider document-level or even visual context (multimodal translation) beyond single sentences.
- **Better Quality Assurance:** Development of robust automated quality estimation tools to reliably determine which translations require human post-editing and which can be published without review.
- **Regulatory Frameworks:** Emerging regulations focused on the ethical use of AI and data privacy will shape the industry, particularly in sensitive sectors.

Anusaraka or Language Accessor

Introduction:

What is an accessor?

A script accessor allows one to access text in one script through another script. GIST terminals available in India, is an example of script accessor. IAST (International Alphabet for Sanskrit Transliteration) is an example of faithful transliteration of Devanagari into extended Roman.

What is a Language accessor?

Language Accessor tries to generalise and apply this philosophy to the problem of language conversion which is several order more complex than that of script conversion.

Background

- **Origin and Development:** The Anusaaraka system was developed by researchers at the Indian Institute of Technology, Kanpur (IIT Kanpur), the International Institute of Information Technology, Hyderabad (IIIT-H), and the University of Hyderabad Department of Sanskrit Studies. The project has been ongoing for over two decades.
- **Core Philosophy:** The central premise is to divide the workload between human and machine. The machine handles the language-based analysis and dictionary lookups (tasks difficult for a human to memorize), while the user (reader) provides the common-sense, background knowledge needed for interpretation (tasks difficult for a machine).
- **Linguistic Basis:** The system is based on the principles of Paninian Grammar (an ancient Indian linguistic framework) and the concept of *karaka* (thematic roles). This framework is particularly suited to Indian languages, which are generally free word order languages and share significant vocabulary and grammatical structures.

Key features and background

- **Language Access, not Direct Translation:** The core idea is to present an "image" of the source text in the target language, so users can access the

information faithfully even if the output contains some source-language constructions.

- **Human-Computer Partnership:** The system divides the work between the computer and the user. The machine handles the linguistic processing, while the human user interprets the output, making it a collaborative process.
- **Layered Output:** It provides a layered output that shows the stages of translation, making the process transparent and allowing the user to access all levels of information.
- **Grammatical Framework:** Anusaaraka is built on the principles of Indian grammatical tradition, specifically Pāṇini's *Ashtadhyayi* grammar, to analyze and process languages.
- **Robust and Graceful Degradation:** The system is designed to be robust, aiming for a faithful representation of the source text and "graceful degradation" in case of translation failures.
- **User Participation:** A unique feature is that it is designed to allow users to contribute to and improve the system's quality over time.
- **Application:** It was initially developed as an English-to-Hindi translation tool to help speakers of Hindi and other Indian languages access a large volume of information that is available in English.
- **Technical Details:** Anusaaraka uses an eclectic combination of the Apertium architecture and an expert system, utilizing both deep and shallow parsing for analysis.

The "Gordian Knot" analogy in the context of Anusaraka or Language Accessor refers to a system that simplifies the complex, often seemingly unsolvable, problem of human-assisted machine translation between languages. The Anusaraka system "cuts the knot" by strategically dividing the translation task between the user and the computer, using a "layered" approach to create a readable, though imperfect, text that can be easily interpreted by a human. It presents an "image" of the source text in a close-to-target language, with the user's background knowledge handling the complex interpretation rather than the machine.

Cutting the Gordian Knot

Anusaraka: The "Gordian Knot" approach to translation

- **Problem:** Direct machine translation between languages is difficult due to complexities like unspoken context, differing grammar, and vocabulary. The challenge is to create a system that is accurate enough to be useful but simple enough to be built in stages.
- **Solution:** The Anusaraka system "cuts the knot" by simplifying the process through a unique approach that divides the labor between the machine and the human.
    - **Machine's role:** The machine performs the initial, simpler tasks like analyzing words, identifying their roots and grammatical features, and mapping them to a language that is close to the target language.
    - **Human's role:** The machine presents an "image" of the source text with the machine-translated elements, but also includes some original or specially notated "spill-overs" from the source language. The user then uses their world knowledge and common sense to interpret these elements and fully understand the text.

Key advantages of the Anusaraka system

- **Early availability:** Building the initial, simpler modules of the Anusaraka system is much easier than building a fully automated one, which makes the system available for use much earlier.
- **Transparency:** The process is transparent to even a layman, as the user is given all the information needed to interpret the text.
- **Reliability:** The system separates reliable resources from those that are inherently unreliable, and it can produce a "rough" translation in case of failures.

The Problem

The main problem with the Anusaraka system, or Language Accessor, is the significant cognitive load placed on the user, especially when the source and target languages are significantly different. The system presents a "layered" or "semi-translated" output that requires the user to have some training to interpret, and this load increases with greater language divergence.

User-centric problems

- **Requires user training:** The system's output isn't perfect translation, and users must be trained to read and understand the modified text, which includes special notations and constructions from the source language.
- **High cognitive load:** The user must fill in the gaps and interpret the output, which is especially demanding for languages that are not close to each other, like English and Hindi.
- **Not a fully automatic solution:** Unlike true machine translation, Anusaraka is not intended to provide a final, polished translation. It requires the user to do the final interpretation, making it unsuitable for users who want immediate, error-free translation.

System-centric problems

- **Tension between brevity and precision:** Like many machine translation systems, Anusaraka faces the challenge of balancing conciseness with accuracy, leading to inherent ambiguities in the output.
- **Inability to handle ambiguity:** The system struggles with ambiguity because it lacks the "world knowledge" or common sense that humans use to understand context and cultural nuances in a text.
- **Dependence on language distance:** The effectiveness and user-friendliness of the system are heavily dependent on how similar the source and target languages are. The "journey is not comfortable" for users when the languages are very different, say the developers.

Structure of Anusaraka System

The Anusaraka system is a hybrid approach to machine translation that divides the workload between the machine and the user, rather than aiming for fully automated translation. Its structure involves a layered output that preserves information from the source language, and a user with some training can "access" the text by understanding the output through special notation. The system is transparent, allowing users to see the translation stages, and is designed for user participation to improve accuracy.

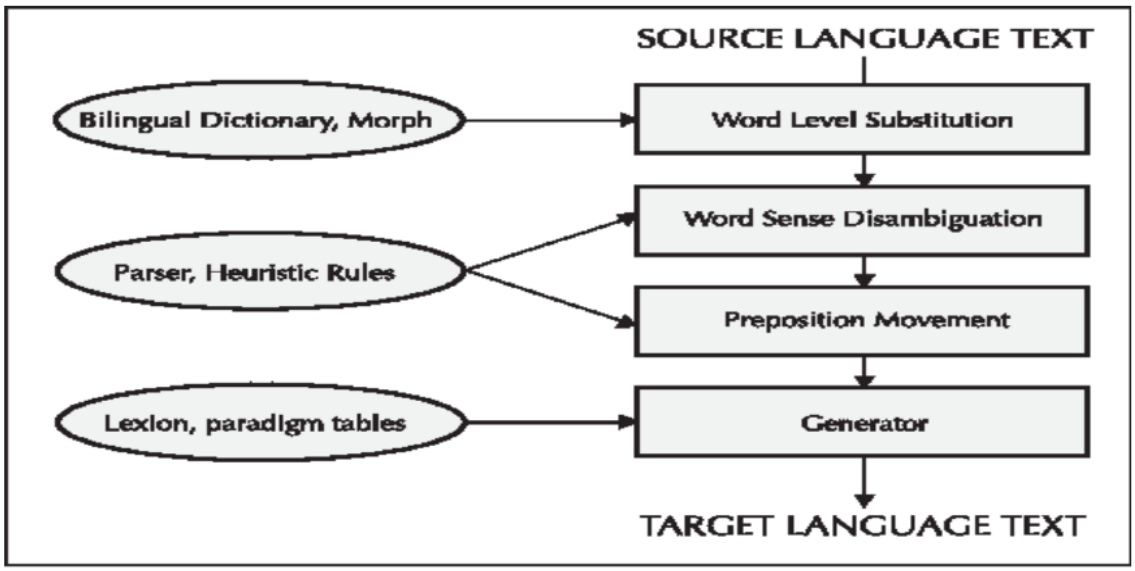


Figure 1 –The architecture of "core" anusaaraka

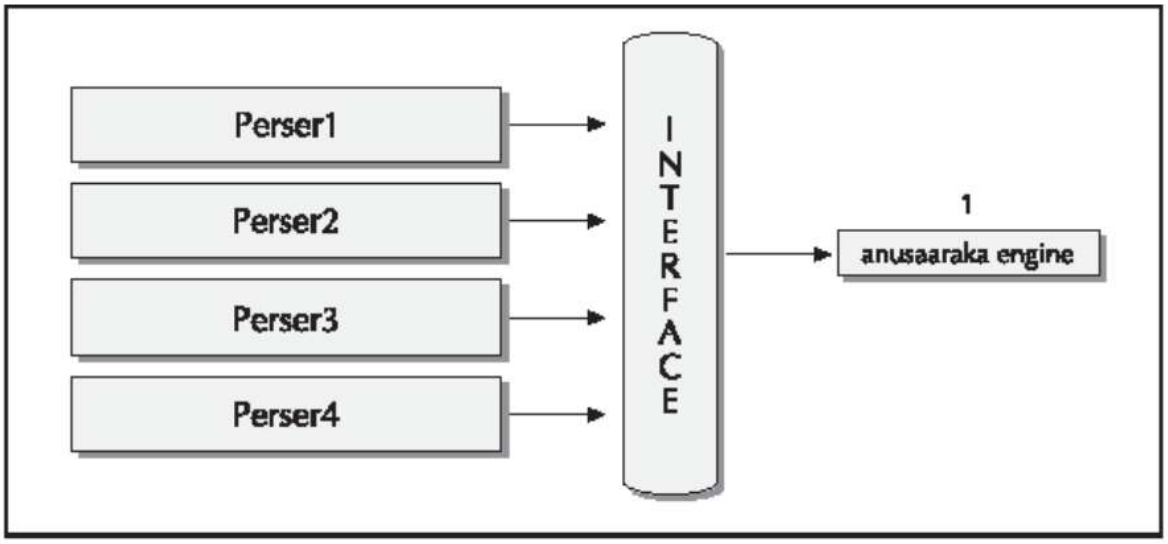


Figure 2 –Interfaces that map output of parsers to an intermediate form

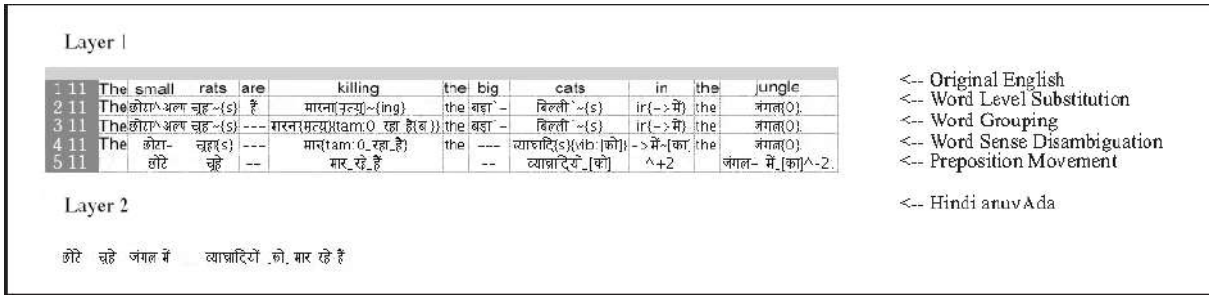


Figure 3 – Snapshot of a sample Engilsh-Hindi Anusaaraka output

How it Works

1. The source text goes through word-level substitution for maximum faithfulness.
2. Rules and context are applied for word sense disambiguation.
3. Grammar adaptation, such as preposition placement, ensures the output matches the target language's syntax.
4. The generator constructs the final text, keeping the output transparent, layered, and user-accessible.
5. Throughout each stage, the system logs its changes, so users can inspect, adjust, or access deeper information when required.

User Interface

The process involves the following stages:

- Source Language Text: The process begins with the original text in the source language.
- Pre-editing: The source text undergoes a pre-editing phase, where a "Pre-editor" prepares the text for the machine translation system.
- Anusaaraka: The pre-edited text is then processed by the "Anusaaraka" system, which performs the initial machine translation.
- Anusarit Hindi: The output from the Anusaaraka system is the "Anusarit Hindi" (translated Hindi) text.
- Reading Interface: A "Trained user" interacts with the "Anusarit Hindi" text via a "Reading Interface," suggesting the system might be designed for comprehension or interactive review.
- Post-editing: The translated text also goes through a "Post-editing" phase, where a "Post-editor" refines and corrects the output to produce the final, high-quality translation.
- Hindi: The final result is the polished text in Hindi.

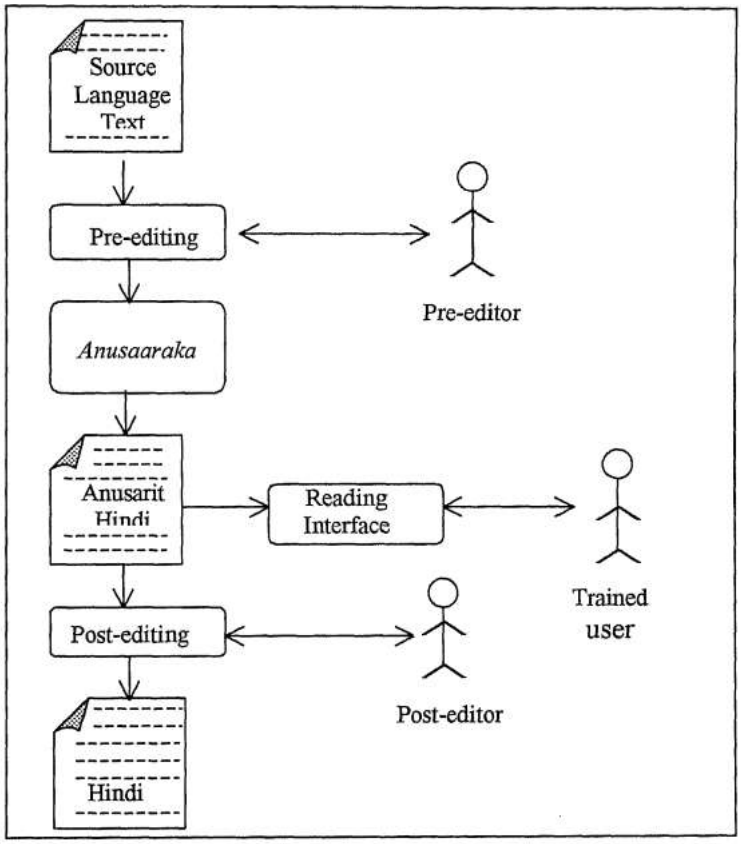


Fig: user interface

Linguistic Area

The concept of an "Anusaaraka" (Language Accessor) system is based on the idea that **India constitutes a "linguistic area"**, where languages share significant structural similarities in vocabulary, grammar, and pragmatics despite belonging to different families. This shared structure makes it easier to build a machine translation system focused on meaning preservation rather than perfect grammatical fluency in the target language.

Anusaaraka and the "Linguistic Area" Concept

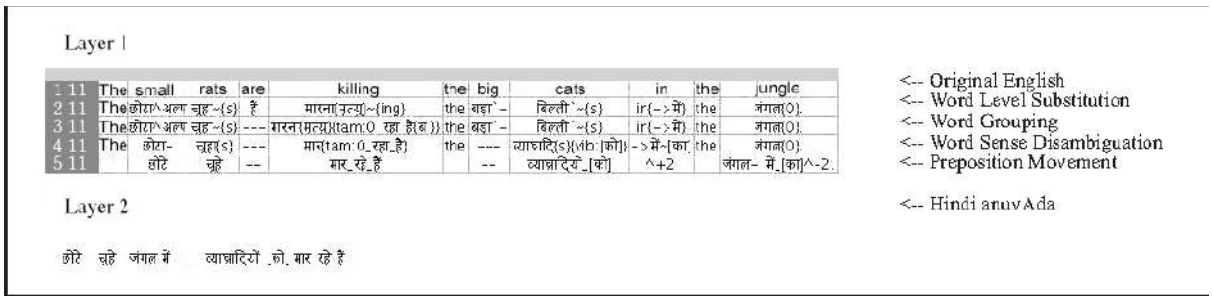
The Anusaaraka project, developed by researchers at the International Institute of Information Technology, Hyderabad (IIIT-H) and the University of Hyderabad, leverages these shared linguistic features.

- **Shared Features:** Indian languages, even those from different major families like Indo-Aryan and Dravidian, exhibit commonalities due to centuries of shared cultural history. For example, many words have a shared Sanskrit origin, and most Indian languages are highly inflectional, with a relatively free word order.

- **System Design:** The Anusaaraka system exploits these similarities. Instead of performing complex, deep linguistic analysis (which is error-prone), the system primarily maps word groups and constructions from the source language to the target language. This approach simplifies the machine's task and shifts some of the interpretative load to the human user.
- **User Training:** The output is an "image" of the source text in the target language, sometimes including special notation for source language constructions that lack a direct equivalent. Users need a short training (expected to be a few weeks) to learn this specific output "dialect" and interpret the text faithfully.
- **Faithfulness over Fluency:** The primary goal is to ensure all information from the source text is preserved, even if the resulting sentence in the target language is not perfectly grammatical or stylistically natural. This approach makes the translation process transparent, allowing bilingual users to trace back to the source information.

Giving up Agreement in Anusarsaka Output

1. The system takes "Source Language Text" as input and produces "Target Language Text" as output.
2. The process involves a sequence of operations: Word Level Substitution, Word Sense Disambiguation, Preposition Movement, and Generation.
3. External resources such as bilingual dictionaries, parsers, heuristic rules, and lexical databases are used to inform the various stages.
4. The Anusaaraka approach divides the task between machine processing of language structure and human interpretation of the final output.



1. **Word-Level Substitution**
   - Each English word is replaced with its closest Hindi equivalent.
   - At this stage, the word order still follows English syntax.
   - Example: “rats → चूहे”, “cats → बिल्लियाँ”.

2. **Word Grouping (Chunking)**

- Words that belong together in meaning are grouped.
- For example, “small rats” becomes a group, and “big cats” becomes another group.

3. **Word Sense Disambiguation**

- Some words have multiple meanings (example: “cat” as an animal or as a category).
- Here the system selects the correct meaning based on context → “cats” = “व्याघ्रादि” (wild/big cats), not “घरकीबिल्ली”.

4. **Preposition Movement**

- English uses **prepositions before nouns** (e.g., “in the jungle”).
- Hindi uses **postpositions after nouns** (e.g., “जंगलमें”).
- The system moves “in” to the correct Hindi position.

5. **Intermediate Hindi Output**

- After applying all rules, the output is Hindi-like but not yet perfectly grammatical.
- This layer is still a rough stage before final refinement.

Anusaraka is a system for language bridging that acts as a machine translator, while a "language accessor" can be a broader term for a tool or person that facilitates language access. Anusaraka is a specific type of language accessor that works by presenting a source text in a target language, but using layered output to show the translation process, making it transparent to the user who can read it with minimal training.

Language Bridges

Anusaraka: A layered translation system

- **How it works:** The user provides an English text, and Anusaraka produces an output in an Indian language. The output appears in "layers," where each layer represents a successive step in the translation process, closely following the structure and meaning of the original text.

- **User experience:** Instead of a full, polished translation, the system shows "layers" of the translation. The user, after some training, can understand this output because it retains elements from the source language.
- **Transparency:** This layered, step-by-step approach makes the translation process transparent and "reversible," allowing the user to see how the output was constructed.
- **Goal:** To bridge the language barrier, especially between English and Indian languages, by allowing a user to read a text in their native language.

Language Accessor: A broader term

- A **language accessor** is a more general term that can refer to any method, tool, or person that provides access to information across different languages.
- This can include systems like Anusaraka, but also other machine translators or human interpreters.
- In the context of software, an "accessor" can refer to a function that retrieves data without performing an action, such as getting information about a web element in the case of Selenium testing, which is different from Anusaraka's function.

Multilingual Information Retrieval

Introduction

What is Information Retrieval (IR)?

Information retrieval refers to the process of finding relevant documents or data from a large collection (like the internet or a database) based on a user's query. A classic example is using Google or a search engine: you type in a question or keywords, and the system returns a list of web pages or documents that are most relevant to your query.

Multilingual Information Retrieval (MLIR)
Multilingual Information Retrieval (MLIR) refers to the ability of a system to retrieve information in **multiple languages**. This becomes crucial in today's globalized world, where people speak and write in many languages, and information is scattered across the internet in those different languages.

In simpler terms, imagine you are looking for information about "global warming" in English, but the most relevant documents might be in French, Spanish, or

Chinese. MLIR helps you retrieve and understand relevant documents, no matter the language they are written in.

Why is MLIR Important?

- **Global Reach:** People speak different languages around the world, and not all information is available in one language.
- **Cross-Language Access:** Users who speak one language (e.g., English) can search for information in another language (e.g., Spanish or Arabic) and still get meaningful results.

Example:

- You type "global warming" in English, and you want to see results not just in English, but also documents in French, German, or Chinese.

Document Preprocessing

Before the system can perform multilingual information retrieval, it needs to prepare the documents and queries to make the process efficient. This is called **document preprocessing**.

Steps in Document Preprocessing:

1. Text Cleaning:

- This step involves removing unnecessary elements from the text such as special characters, HTML tags, or irrelevant punctuation.
- **Example:** If you have a web page, you would remove the tags like `<div>`, `<p>`, and other HTML elements to focus only on the text.

2. Tokenization:

- Tokenization breaks the text into smaller pieces (tokens), such as words or phrases. In multilingual IR, tokenization must consider the language structure.
- **Example:** The sentence "Global warming is serious" would be split into tokens: ["Global", "warming", "is", "serious"].

3. Stop Word Removal:

- Stop words are common words like “the,” “is,” “and,” etc., that don’t add much meaning and can be safely ignored.

4. Stemming or Lemmatization:

- This is the process of reducing words to their root forms. For example, "running" becomes "run." This helps in treating similar words the same way.
- In multilingual IR, stemming needs to consider the specific rules of each language. For example, in French, "marcher" (to walk) can be reduced to its root form "march."

5. Language Detection:

- The system must first identify the language of the document (and the query) to process it correctly.
- **Example:** If the user searches in French, the system must understand the query in French and match it to French documents.

Monolingual Information Retrieval

Monolingual Information Retrieval (IR) refers to the process of searching for and retrieving information in **only one language**. In this system, both the user's query and the documents in the database must be in the same language. For example, if a user types a query in English, the search engine will only look through English documents to find relevant results. It relies on language-specific indexing and ranking, meaning the system understands and processes text based on the rules and vocabulary of a single language.

Key Features of Monolingual Information Retrieval

1. Single-Language Query Processing

The system accepts queries in only one language and retrieves information exclusively from documents written in the same language.

2. Language-Specific Indexing

Documents are indexed based on the vocabulary, grammar, and linguistic rules of a single language, making search results more accurate within that language.

3. High Search Accuracy Within One Language

Since the system works with only one language, it can analyze and rank documents more effectively without dealing with translation challenges.

4. Faster Processing and Simpler Architecture

The system does not need translation tools or multilingual models, allowing quicker search and simpler implementation.

5. Limited Cross-Language Access

A major feature (and limitation) is that users cannot retrieve documents written in other languages, even if they are highly relevant.

Example of Monolingual IR:

- If you are searching for documents in English about "climate change," the system will search only English content. It will not check or retrieve documents written in French, Spanish, or any other language, even if those documents are more detailed or relevant.

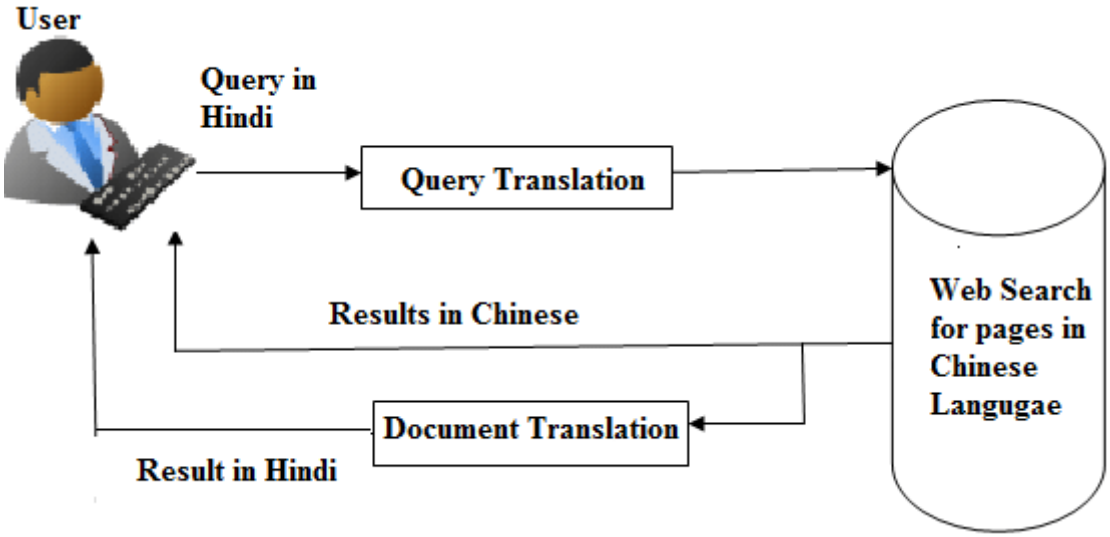
Limitations of Monolingual IR:

- The biggest limitation is that it cannot access information written in other languages. If the most important or updated information is available in a different language, the user will not be able to retrieve it. This creates a significant knowledge gap, especially in global topics like medicine, climate change, or international research.
- As a result, monolingual IR becomes insufficient in today's multilingual and interconnected world. This challenge is what makes **multilingual information retrieval** essential, as it allows users to access documents across different languages without being restricted to just one.

Cross-Language Information Retrieval (CLIR)

What is Cross-Language Information Retrieval (CLIR)?

Cross-Language Information Retrieval (CLIR) is a subset of multilingual IR where a user can input a query in **one language**, but retrieve documents in another language. The idea is that the system translates the query into different languages and searches for documents in those languages.



How CLIR Works:

1. **Query Translation:** The user's query is translated into the target language(s) using machine translation tools (like Google Translate).
2. **Document Retrieval:** Once the query is translated, the system searches documents in the target language for relevant results.
3. **Ranking:** The system ranks documents based on their relevance to the translated query.

Example of CLIR:

- You are a **Spanish-speaking user** searching for "global warming" in **English**. CLIR will:
    1. Translate your Spanish query into English.
    2. Search for documents in English that match the query.
    3. Rank the documents and show you the most relevant ones in English.

Challenges in CLIR:

- **Translation Accuracy:** The quality of translation can affect the search results. Bad translations may lead to irrelevant documents being retrieved.
- **Cultural and Contextual Nuances:** Some phrases or idioms might not translate well between languages, making it hard to find exactly what you're looking for.

Multilingual Information Retrieval (MLIR)

What is Multilingual Information Retrieval (MLIR)?

Multilingual Information Retrieval (MLIR) takes a broader approach than CLIR. Instead of translating a query into one or more target languages, the system can handle multiple languages simultaneously. In MLIR, a user can search for information in one language, and the system retrieves documents in several languages at once, without needing to translate the query.

How MLIR Works:

1. **Multilingual Document Indexing:** The system indexes documents in multiple languages. This means that documents in French, Spanish, German, and others are all processed and stored in a way that can be retrieved in response to a multilingual query.
2. **Query Understanding:** The system understands the query in the language it is written, and the documents in different languages are ranked and returned based on relevance.

3. **Document Ranking and Relevance:** The system needs to ensure that documents in different languages are ranked correctly, even though they are in different languages. This often involves sophisticated language models and algorithms that understand semantic meaning across languages.

Example of MLIR:

- If you search for "climate change" in **English**, MLIR will retrieve relevant documents in English, French, Spanish, and other languages, showing the most relevant ones first, regardless of the language they are written in.

Advantages of MLIR:

- **Broader Results:** MLIR allows you to access a wide range of documents in various languages, helping you find information that might not be available in your language.
- **No Need for Translation:** The system doesn't rely on machine translation, so it avoids potential translation errors and can process documents directly in their original languages.

Challenges of MLIR:

- **Language Diversity:** Different languages have different structures, and the system needs to handle this variety effectively.
- **Data Availability:** Some languages have more online content than others, and for less common languages, it can be challenging to find enough relevant documents.

Conclusion

To summarize:

1. **Multilingual Information Retrieval (MLIR)** allows users to find information in multiple languages without necessarily translating the query.
2. **Document Preprocessing** is the preparation stage where text is cleaned, tokenized, and processed before searching or indexing.
3. **Monolingual Information Retrieval** is when search happens within one language, but it has limitations in multilingual contexts.
4. **Cross-Language Information Retrieval (CLIR)** lets you search in one language but retrieve results in another by translating the query.
5. **Multilingual Information Retrieval (MLIR)** handles multilingual queries and returns documents in multiple languages simultaneously, without translating the query.

These technologies make it possible for us to access and understand information in different languages, breaking down barriers in today's globalized world.

Common Evaluation Methods

1. **Evaluation of Multilingual Summarisation**

   a) ROUGE Score (Recall-oriented Understudy for Gisting Evaluation)

- Most commonly used metric.
- Compares generated summary to a "gold-standard" (a human-written summary).
- ROUGE checks if key words, phrases, and sentences are present in the automatic summary.

Example:
Gold summary → *The dog is brown.*
Automatic summary → *"dog", "brown"* must be included.

b) BLEU Score (Bilingual Evaluation Understudy)

- Often used for translation systems.
- Checks if the n-grams (groups of words) in the system's summary match those in the reference summary.
- BLEU works well when comparing summaries in multiple languages, but less flexible than ROUGE when dealing with different sentence structures.

c) Human Evaluation

- Human judges score the summaries based on the following criteria like: *relevance, coherence, fluency,* and *grammatical correctness* in the language.
- Since NLP models may struggle with nuances, human evaluation ensures that specific details are correctly handled.

Challenges in Evaluation

- **Language Diversity:**
  Different structures, word order, expressions, etc., can't fit under one umbrella.
- **Cultural Differences:**
  Cultural context varies between languages, making evaluation difficult.
- **Resources:**
  Some languages have more summarisation data available, while others have less, which makes multilingual evaluation tricky.

Tools Used in Information Retrieval (IR)

Information Retrieval (IR) means finding useful information from large collections of data, like search engines (Google). Many tools help in building and testing IR systems.

1. Search and Indexing Tools

These tools help store and search text quickly.

- **Apache Lucene** – A free tool that helps store text data and search it fast.
- **Elasticsearch** – A powerful search engine built using Lucene. It is used in websites and apps for real-time search.

2. Text Processing Tools

These tools prepare text before searching.

- **NLTK (Python)** – Helps in:
  - Breaking sentences into words (tokenization)
  - Removing common words like "is", "the" (stop words)
  - Reducing words to base form (stemming)
- **spaCy** – A fast tool used for:
  - Cleaning text
  - Finding names of people and places
  - Understanding sentence structure

3. Machine Learning & IR Frameworks

These tools help build smart search systems using AI.

- **TensorFlow / PyTorch** – Used to build deep learning models for better search results.
- **Gensim** – Used to find topics in documents and understand word meaning using word vectors.

4. Evaluation Tools

These tools check how good a search system is.

- **trec_eval** – Used to measure search accuracy.
- **NLTK ROUGE / BLEU** – Used to check quality in text summary and translation tasks.

5. Visualization & Analysis Tools

These tools help see and analyze search results.

- **Kibana** – Shows search data in charts and graphs.
- **Jupyter Notebook** – Used for writing and testing IR code step by step.

Software and Resources for Information Retrieval

IR systems also need software and data to work properly.

1. IR Software Platforms

- **Elasticsearch** – Used for large search systems.
- **Solr** – Another search server built on Lucene.

- **Whoosh** – Simple Python search library for small projects.

2. NLP Software

- **Stanford CoreNLP** – Helps in:
  - Grammar checking
  - Finding parts of speech
  - Detecting emotions in text
- **OpenNLP** – Tool for:
  - Breaking text into parts
  - Finding names and important words

3. IR Datasets

These are collections of data used for testing search systems.

- **TREC Collections** – Standard datasets used to test search engines.
- **CLEF** – Data for multilingual search testing.
- **Wikipedia Dumps** – Large collection of Wikipedia articles used for training models.

4. Linguistic Resources

These help understand word meaning.

- **WordNet** – A word database that shows word meanings and relationships.
- **FastText / Word2Vec** – Pre-trained word meaning models used in search systems.

5. Multilingual Resources

Used when working with multiple languages.

- **OPUS Corpus** – Text data in many languages.
- **Europarl Dataset** – Text data from European Parliament in many languages.

Multilingual Automatic Summarization (MAS)

Multilingual Automatic Summarization (MAS) involves condensing text from multiple languages into a shorter version while retaining key information. It helps manage information overload across diverse linguistic sources and enhances information accessibility.

Introduction

The exponential growth of digital content in various languages has made automatic text summarization a crucial field in Natural Language Processing (NLP). MAS systems aim to produce summaries that are concise, accurate, and coherent, regardless of the input language. This process is challenging due to the linguistic variations (syntactic and morphological patterns) across languages, especially for low-resource languages that lack extensive data or tools.

Approaches to Summarization

Approaches generally fall into two categories:

- **Extractive Summarization:** This method identifies and selects key sentences or phrases directly from the original document to form the summary. Techniques often rely on statistical and graph-based methods like **TF-IDF**, **LexRank**, and **TextRank**, which score sentences based on importance features (e.g., word frequency, position, and centrality). Extractive summaries are typically grammatically correct but may lack overall coherence.
- **Abstractive Summarization:** This more complex approach generates new sentences that might not be in the original text, similar to how humans summarize. It requires advanced NLP and deep learning models, such as sequence-to-sequence (Seq2Seq) models and transformer architectures (e.g., mT5, mBART, PEGASUS), to build an internal semantic representation and generate a fluent, human-like summary.
- **Hybrid Methods:** These combine both extractive and abstractive techniques, often using an extractive step to select key information, followed by an abstractive step to refine and rewrite the content.

Evaluation

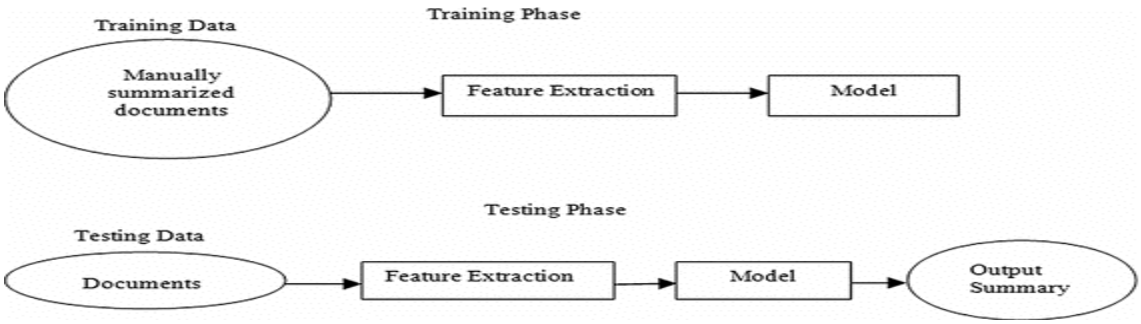
Evaluating MAS systems is crucial and involves both automated and human-based methods.

- **Automated Metrics:**
  - **ROUGE (Recall-Oriented Understudy for Gisting Evaluation):** The most widely used metric; it measures the n-gram overlap (unigrams, bigrams, etc.) between the generated summary and human-written reference summaries.
  - **BLEU (Bilingual Evaluation Understudy):** Originally for machine translation, it is adapted for summarization evaluation by measuring n-gram similarity.
  - **METEOR &BERTScore:** These metrics go beyond exact word matches, considering synonyms and contextual embeddings to assess semantic similarity, which is important for abstractive summaries.

- **Human Evaluation:** Human judges assess summaries for quality aspects like informativeness, coherence, fluency, and factuality. This remains a critical, albeit expensive, component of evaluation.

How to Build a Summarizer

Building a multilingual summarizer typically involves several steps:



1. **Data Acquisition and Preprocessing:** Collect and clean multilingual data, which involves language detection, tokenization, stemming, and removing noise/stopwords.
2. **Feature Extraction/Representation:** Convert text into an intermediate representation using methods like TF-IDF, graph-based models, or deep learning embeddings.
3. **Modeling:** Apply a summarization technique (extractive, abstractive, or hybrid). Modern approaches leverage pre-trained multilingual language models (e.g., XLM-RoBERTa) and fine-tune them on specific summarization tasks and languages.
4. **Generation and Evaluation:** Generate the summary and evaluate its quality using metrics like ROUGE and human judgment.

Competitions and Datasets

Now, let's look at **competitions and datasets** that help advance multilingual automatic summarisation.

Competitions in Multilingual Summarisation:

Competitions are organized to encourage the development of new and better summarisation tools. Some notable ones include:

1. Text Summarization Challenge (TSC):

- A competition that evaluates summarisation models across different languages and domains. It encourages the creation of robust and diverse models.

2. SemEval:

- SemEval (Semantic Evaluation) includes several tasks related to summarisation, such as sentence-level or document-level summarisation in multiple languages. Researchers worldwide participate to push the boundaries of summarisation technology.

3. DUC (Document Understanding Conference):

- DUC is another prominent competition for summarisation. Although it originally focused on English, it has expanded to consider multilingual summarisation tasks.

4. MS MARCO:

- MS MARCO is a dataset designed for various information retrieval and summarisation tasks, which includes multiple languages. It's often used to benchmark different summarisation systems.

Datasets for Multilingual Summarisation:

Datasets are critical in training and evaluating multilingual summarisation systems. Here are a few well-known datasets:

1. MultiLingual TED Talks (MTED):

- This dataset consists of TED Talks in several languages, accompanied by human-written summaries. It’s a great resource for building multilingual summarisation systems.

2.XSum (Extreme Summarization):

- XSum is a dataset that includes short, concise summaries of news articles. While primarily in English, it has been extended to other languages like German and French for multilingual experiments.

3. Newsroom:

- Newsroom is a large-scale news dataset that includes over a million documents, offering summaries in several languages. It’s often used for training extractive summarisation models.

4. SAMSum:

- SAMSum is a conversational dataset with summaries, useful for dialogue-based summarisation. It has been expanded for multilingual tasks.

5. OpenSubtitles:

- OpenSubtitles contains a vast collection of movie subtitles in multiple languages. It's useful for summarisation systems focused on dialogue and informal language.