NATURAL LANGUAGE PROCESSING

UNIT I: INTRODUCTION

What is NLP?

- **Natural Language Processing (NLP)** is a branch of **Artificial Intelligence (AI)** that enables **computers to understand, interpret, and generate human language**.
- It acts as a **bridge between computers and human (natural) languages** like English, Hindi, Telugu, etc.

Origins of NLP

1. Early Days: Rule-Based Systems (1950s–1970s)

- Based on **hand-coded grammar rules**.
- Inspired by **linguistics** and **formal grammars** (used for translating Russian to English).
- Example: Using **syntax trees** to parse sentences like "The boy eats an apple."

2. Machine Translation (Cold War era)

- In 1950s–60s, **US and USSR** motivated research in **automatic translation**.
- **Georgetown-IBM experiment (1954)** was an early success but couldn't scale.
- Failed because **language is ambiguous and context-sensitive**.

3. Statistical NLP (1980s–1990s)

- Shift from rules to **probability & statistics**.
- Based on large **corpora** (text datasets).
- Used concepts like:
    - **n-grams** (predict next word)
    - **Hidden Markov Models (HMMs)**
    - **Part-of-Speech (POS) tagging**.

4. Machine Learning Era (2000s)

- NLP problems modeled as **machine learning tasks**.
- Algorithms: **Naïve Bayes, Decision Trees, SVMs**.
- Improved results with more **training data** and **feature engineering**.

5. Deep Learning and Neural NLP (2010s–present)

- Use of **Neural Networks** and **word embeddings** (like Word2Vec, GloVe).
- Major breakthrough: **Transformers** (e.g., BERT, GPT).
- These models understand **context** better and handle **longer sentences**.

Challenges of NLP

1. Ambiguity

- Words or sentences can have **multiple meanings**.
- Example:
    - "I saw the man with the telescope." → Who has the telescope?
    - Types: **Lexical, syntactic, semantic, and pragmatic ambiguity**.

2. Language is Highly Contextual

- Meaning depends on **situation, tone, or culture**.
- "He is *cold*." → Sick? Emotionless? Temperature?

3. Variation in Language

- Multiple ways to express the same idea.
    - "Can you help me?" = "Would you assist?" = "Could you give me a hand?"

4. Data Sparsity

- Many words or sentence combinations may be **rare or unseen**.
- Especially hard for **low-resource languages**.

5. Syntax and Grammar Differences

- Languages differ structurally.
    - English: Subject-Verb-Object → "I eat mango."
    - Hindi: Subject-Object-Verb → "Main aam khata hoon."

6. Named Entity Recognition (NER) Issues

- Distinguishing names from common words is hard.
    - "Apple" → Fruit or Company?

7. Sarcasm, Irony, Slang

- Hard to detect with literal meaning.
    - "Oh great! Another exam!" (actually negative feeling)

8. Multilinguality and Code-Mixing

- People switch between languages mid-sentence.
    - "Aaj mood off hai, I'll skip class."

LANGUAGE MODELING

What is Language Modeling (LM)?

- A **Language Model (LM)** is a system that **predicts the next word** (or sequence of words) in a sentence.
- It helps computers understand **how language is structured**.
- It assigns **probabilities** to word sequences:

"I am going to the **market**" → High probability
"Going to market the am I" → Low probability

Types of Language Models

1. Grammar-Based Language Model

Definition:

- Based on **formal grammar rules** of a language (like English grammar).
- Works by **generating valid sentences** using **syntax rules**.

Example:

Grammar Rule:

S → NP + VP

NP → Det + Noun

VP → Verb + NP

Sentence Generated:

The dog ate food

Features:

- Uses **Context-Free Grammar (CFG)**.
- Good for **small or well-defined domains**.
- Deterministic and interpretable.

Limitations:

- Cannot handle **uncertainty or probability**.
- Fails with **unstructured, real-world text**.

2. Statistical Language Model (SLM)

Definition:

- Assigns **probabilities** to sequences of words using **statistics from large text corpora**.

Example:

Bigram model:
P("I want food") = P("I") × P("want" | "I") × P("food" | "want")

Types of Statistical Models:

Model Type	**Meaning**	**Example**
Unigram	Each word is independent	P("I want food") = P("I") × P("want") × P("food")
Bigram	Each word depends on the previous one	P("want"
Trigram	Depends on two previous words	P("food"

Features:

- More **flexible** than grammar-based models.
- Used in **speech recognition, text prediction**.

Limitations:

- Struggles with **long-distance dependencies**.
- Needs a **lot of training data**.
- **Sparsity**: Some word combinations may never appear in training data.

Regular Expressions (Regex)

What is it?

- A **pattern-matching technique** used to **identify strings** in text.
- Very useful for **searching, extracting, validating** textual data.

Example Patterns:

Pattern	**Meaning**	**Matches**
\d+	One or more digits	123, 99
\w+	One or more word characters (a-z, A-Z, 0–9, _)	Hello, Word123
[A-Za-z]+	Only letters	apple, Banana
^A.*e$	Starts with A, ends with e	Apple, Age
\bcat\b	Whole word "cat" only	cat, not in "category"

Use Cases:

- **Tokenization**: Splitting text into words or sentences.
- **Validation**: Checking if an email/phone number is valid.
- **Preprocessing**: Removing unwanted patterns like HTML tags or numbers.

Finite-State Automata (FSA)

What is FSA?

- A **Finite-State Automaton (FSA)** is a simple abstract machine used in **language processing**.
- It is called "finite-state" because it operates with a **finite number of states**.
- The machine reads input **symbol by symbol** and **changes its state** according to defined rules.
- If the FSA **ends in a final (accepting) state**, the input string is accepted.

Components of an FSA:

Component	**Description**
Q	Set of all **states** (e.g., q0, q1, q2)
Σ (Sigma)	**Alphabet** – the set of allowed **input symbols** (like characters or words)
δ	**Transition function** – describes how to move from one state to another using an input
q_o	**Start state** – where the machine starts reading input
F	**Final states** – accepting states, meaning valid input

Example: Recognizing the word "cat"

- **States**: q0, q1, q2, q3
- **Start state**: q0
- **Final state**: q3
- **Transitions**:
    - q0 --c--> q1
    - q1 --a--> q2
    - q2 --t--> q3

So, if input = cat, and the FSA ends in q3, it **accepts** the string.

Applications of FSA in NLP:

- **Tokenization** – splitting text into words
- **Lexical analysis** – checking word validity
- **Syntax checking** – basic grammar pattern validation
- **Morphological parsing** – identifying root + suffix

English Morphology

What is Morphology?

- **Morphology** is the study of the **internal structure of words**.
- Words are made up of **morphemes**, which are the **smallest meaningful units** of language.

Types of Morphemes:

Type	**Description**	**Example**
Root / Base	Main part of a word	"run", "happy"
Prefix	Added at the beginning	"**un**happy"
Suffix	Added at the end	"walk**ed**"

Why is Morphology Important in NLP?

- Helps in **understanding root forms** of words.

"running", "ran", "runs" → root = **run**

- Enables **search engines**, **chatbots**, and **grammar checkers** to process natural language more intelligently.

3. Transducers for Lexicon and Rules

What is a Finite-State Transducer (FST)?

- A **Finite-State Transducer** is like an FSA **with output**.
- Instead of just accepting input, it **translates input into output**.

Think of it as a machine that takes in one form of a word and gives out **another form**.

Simple Example:

Let's say input word is: **"cats"**
Transducer output:

- Root = "cat"

- Suffix = “+PL” (plural marker)

So the mapping becomes:
cats → cat +PL

How Does an FST Work?

Input	Transition	Output
"c"	q0 → q1	"c"
"a"	q1 → q2	"a"
"t"	q2 → q3	"t"
"s"	q3 → q4	"+PL"

Final output: **cat +PL**

Use of Transducers in NLP:

Function	Description
Morphological Analysis	Breaking word into root + affixes
Morphological Generation	Producing correct word form from root
Spelling Correction	Converting misspelled word to correct one
Lexicon Lookup	Matching input with known words in dictionary

ADVANCED PREPROCESSING TOPICS

1. Tokenization

What is Tokenization?

- **Tokenization** is the process of **splitting text into smaller units** called **tokens**.
- Tokens can be:
    - **Words** → “I love NLP” → [“I”, “love”, “NLP”]
    - **Sentences** → “Hello! How are you?” → [“Hello!”, “How are you?”]

Why is Tokenization Important in NLP?

Reason	Explanation
Text Preprocessing	First step before any NLP task (POS tagging, parsing, etc.)
Input for ML Models	Helps convert raw text into structured format
Language Understanding	Helps in semantic, syntactic analysis

Types of Tokenization:

Type	Description	Example
Word Tokenization	Splits text into words	“Let’s learn NLP.” → [“Let’s”, “learn”, “NLP”, “.”]
Sentence Tokenization	Splits text into sentences	“Hi. I’m Sam.” → [“Hi.”, “I’m Sam.”]
Subword Tokenization	Splits into morphemes or sub-parts	“unbreakable” → [“un”, “break”, “able”]
Character Tokenization	Each letter is a token	“cat” → [“c”, “a”, “t”]

Challenges in Tokenization:

- **Punctuation**: “U.S.A.” vs “USA”
- **Contractions**: “don’t” → “do”, “not”?
- **Hyphenated words**: “well-known”

2. Detecting and Correcting Spelling Errors

Why is it Needed?

- People make **typos**: “teh” instead of “the”, or “recieve” instead of “receive”.
- **Autocorrect systems**, **search engines**, and **text editors** use NLP to fix them.

Types of Spelling Errors:

Type	Example	Description
Insertion	“taable”	Extra letter added
Deletion	“tabl”	Letter missing
Substitution	“tabke”	Wrong letter used
Transposition	“taBle” → “tbale”	Two letters swapped

How to Detect & Correct Errors:

- Compare misspelled word with **dictionary words**.
- Use **edit distance** (explained below) to find the **closest valid word**.

Techniques:

Technique	Description
Dictionary Lookup	Compare word with known word list
Noisy Channel Model	Use probability to correct most likely word
Minimum Edit Distance	Count changes needed to fix the word
Spell Checker Libraries	Tools like TextBlob, SymSpell, pyspellchecker

3. Minimum Edit Distance (MED)

What is it?

- **Minimum Edit Distance** is the **minimum number of operations** needed to change one word into another.

Allowed Operations:

Operation	Description	Example
Insertion	Add a character	"cat" → "cart"
Deletion	Remove a character	"cart" → "cat"
Substitution	Replace a character	"cat" → "cut"
(Optional) Transposition	Swap two letters	"acress" → "caress"

How is It Calculated?

- Usually calculated using **Dynamic Programming**.
- The output is a **distance score**.
- Example:
    - "kitten" → "sitting"
    - MED = 3 (k → s, e → i, add g)

Applications in NLP:

Use Case	Explanation
Spell Correction	Suggest closest word
Speech Recognition	Match spoken word with written
Plagiarism Detection	Compare similarity of sentences

DNA Sequence Matching	Bioinformatics uses MED heavily