◆ Q.3 Why is Natural Language Processing (NLP) Considered Difficult Compared to Structured Data Processing?

Natural Language Processing is significantly more challenging than traditional structured data processing because **human language is inherently ambiguous, unstructured, and context-dependent**.

☑ Linguistic Challenges:

---

1. Ambiguity

Words or phrases can have multiple meanings depending on context.

- **Lexical ambiguity**:
  "*bank*" → financial institution **or** riverbank
- **Syntactic ambiguity**:
  "*I saw the man with the telescope*"
  → Who has the telescope? Me or the man?

---

2. Context Dependence

Meaning changes depending on surrounding words or conversation history.

- "*He is running*" → Could mean a race, a company, or for office
- **Pronoun resolution**:
  "*John told Mike that he won.*" → Who is “he”?

---

3. Morphological Variability

Words appear in many forms (tenses, plurals, derivations).

- *“run”*, *“running”*, *“ran”*, *“runner”* → same root, different forms
- Language rules vary by **language**, making cross-lingual NLP even harder.

☑ Computational Challenges:

---

1. Data Sparsity

- Many possible word combinations (bigrams, trigrams, etc.)
- Many may never occur in training data but may appear during testing

---

2. Scalability and Memory

- NLP models (especially deep learning ones like BERT) are **computationally expensive**
- Require **high processing power, GPUs, and memory** for training

---

3. Noisy and Unstructured Input

- Especially true for real-world data like:
  - Social media posts (e.g., “ur” instead of “you are”)
  - Code-mixed language (Hindi + English)
  - Spelling mistakes

🧠 Summary:

Aspect	Traditional Structured Data	Natural Language
Data Format	Tables, rows, numeric	Free-form text
Structure	Well-defined schema	Unstructured
Ambiguity	Rare	Frequent
Context Dependence	Low	High
Processing Complexity	Relatively simple	Computationally intensive

---

🔹 Q.4 Using the Edit Distance Algorithm, Compute the Distance Between “intention” and “execution”

☑️ Edit Distance (Levenshtein Distance)

It measures the **minimum number of operations** (insertions, deletions, substitutions) needed to transform one string into another.

💉 Words:

- Source = `"intention"`
- Target = `"execution"`

📌 Operations Allowed:

- **Insertion**
- **Deletion**
- **Substitution**

☑️ Step-by-Step Transformation:

We transform `"intention"` → `"execution"` using **substitutions, insertions, and deletions**.

Step	Transformation	Operation
1	intention → extention	**Substitute** 'i' → 'e'
2	extention → exention	**Substitute** 't' → 'x'
3	exention → exection	**Substitute** 'n' → 'c'
4	exection → executon	**Substitute** 'e' → 'u'
5	executon → execution	**Insert** 'i' before 'o'

🔢 Final Edit Distance = 5

🧠 Matrix View (Optional for Deep Understanding):

You can compute this using a dynamic programming matrix where:

- Rows = characters of `"intention"`
- Columns = characters of `"execution"`
- Each cell (i, j) = minimum edit distance to reach that prefix

(Let me know if you want the **DP matrix** visualized or included.)