◆ Q.5 Compare Edit Distance and Weighted Edit Distance Approaches in Spelling Correction

Also: Under what scenarios does weighted edit distance yield better results?

☑️ A. What is Edit Distance?

Edit Distance (Levenshtein Distance) is the **minimum number of operations** (insertion, deletion, substitution) required to transform one word into another.

- All operations are considered **equal cost** (typically 1).
- Example:
  `"kitten"` → `"sitting"` = 3 operations
  (k→s, e→i, +g)

☑️ B. What is Weighted Edit Distance?

Weighted Edit Distance assigns **different costs** to each operation (and even specific character pairs). This reflects **real-world likelihoods** — e.g., typos on keyboard or phonetic errors.

Example:

- Substituting ‘a’ with ‘s’ may have a cost of 0.5 (they’re adjacent on QWERTY)
- Substituting ‘q’ with ‘z’ may have a cost of 2.0 (more rare or distant)

📊 Comparison Table:

Feature	Edit Distance	Weighted Edit Distance
Cost Type	Fixed (1 per operation)	Variable based on characters/operation
Realism	Low	High (models human errors better)
Accuracy	Basic corrections	More accurate for noisy/spelling inputs
Complexity	Simple	Slightly more complex
Use Case	General typo correction	Domain-specific or realistic error modeling

---

☑️ C. Scenarios Where Weighted Edit Distance Performs Better:

1. **Keyboard Proximity Errors**
   - “gril” → “girl”
   - ‘r’ is adjacent to ‘i’ → lower substitution cost
2. **Phonetic/Transcription Errors**
   - “nife” → “knife”
   - Helps match similar-sounding words with plausible errors
3. **Language Learner Mistakes**
   - Learners confuse ‘b’ and ‘v’ → assign lower penalty
4. **Medical, Legal, or Domain-Specific Terms**
   - Technical terms often have predictable typo patterns
5. **Optical Character Recognition (OCR)**
   - Misread ‘0’ as ‘O’ or ‘1’ as ‘l’

🧠 Conclusion:

Weighted edit distance is **more intelligent and realistic**, especially in **high-noise environments**, or where domain-specific errors are frequent. It yields better results when **error patterns are known or predictable**.

🔹 Q.6 How Can a Neural Network Be Used to Enhance Machine Translation Applications?

Machine Translation (MT) aims to convert text from one language to another (e.g., English → Hindi). Neural networks (especially **Neural Machine Translation - NMT**) have revolutionized this field by learning complex patterns **end-to-end** without rule-based programming.

✅ A. Key Neural Network Models Used:

1. **Recurrent Neural Networks (RNNs)**
   - First used in early NMT systems
   - Sequence-to-sequence models with encoder-decoder structure
   - Limitation: struggles with long-term dependencies
2. **Long Short-Term Memory (LSTM) / GRU**
   - Handle long sequences better than plain RNNs
   - Still used in small or mid-scale translation systems
3. **Transformer Models (e.g., BERT, GPT, T5)** ✅

- Current **state-of-the-art**
- Use **self-attention** to capture relationships between all words in a sentence
- Can learn **context better** and **translate idioms, syntax, grammar** accurately

---

✅ B. How Neural Networks Improve Translation:

1. **Context Awareness**
   - Understand word meaning based on surrounding text
   - e.g., “bank” (riverbank vs. finance)
2. **Fluent Sentence Generation**
   - Produces grammatically correct, natural-sounding translations
3. **Handling Rare or Unseen Words**
   - Through subword units (Byte Pair Encoding or SentencePiece)
4. **Learning Syntax and Semantics Automatically**
   - No handcrafted grammar rules needed
   - Learns from large parallel corpora (e.g., English-French datasets)

🌐 Example:

English:

“I am going to the market.”

Hindi via NMT (Transformer):

“मैं बाज़ार जा रहा हूँ।”

Earlier rule-based or statistical models might have mistranslated "going" or messed up the verb-object order. Neural models **learn the correct syntax** automatically.

✅ C. Tools and Frameworks:

- Google Translate (uses Transformer-based models)
- OpenNMT, Fairseq, MarianNMT
- Hugging Face models (T5, mBART)

🧠 Summary:

Neural networks have transformed machine translation by making it **more accurate, scalable, and fluent**. Transformers especially dominate this field by learning **deep context, grammar, and semantic**

structures, all without explicit linguistic rules.