Q7. Evaluate the Effectiveness of the Noisy Channel Model in Spelling Correction

◆ Overview

The **Noisy Channel Model (NCM)** treats spelling correction as a decoding problem: we assume the observed (possibly misspelled) word is a "noisy" version of an intended correct word. The model tries to find the most probable intended word based on the observed input.

◆ Formula

$$\hat{c} = \arg\max_{c} P(e \mid c) \cdot P(c)$$

- e: observed (possibly misspelled) word
- c: candidate correct word
- $P(e \mid c)$: probability that c was transformed into e (error model)
- $P(c)$: probability that c appears in natural language (language model)

◆ How It Balances the Probabilities

- **Language Model** $P(c)$ ensures that common and likely words are favored.
- **Error Model** $P(e \mid c)$ favors words that are phonetically or typographically close to the input.
- The best correction is the word that is both **likely to appear** and **likely to be mistyped as the input word**.

◆ Strengths

- Captures realistic human spelling errors.
- Works well for correcting typos and common misspellings.
- Balances likelihood of the word with the likelihood of making a mistake.

◆ Weaknesses

- Requires a large error corpus to estimate $P(e|c)$.
- Ignores sentence context (e.g., grammar and semantics).
- Not effective for creative spelling, abbreviations, or slang.

◆ Example:

Misspelled word: `"recieve"`

Candidate words and probabilities:

- $P(\text{"receive"}) = 0.0015, P(\text{"recieve"} \mid \text{"receive"}) = 0.9$
- $P(\text{"recipe"}) = 0.0011, P(\text{"recieve"} \mid \text{"recipe"}) = 0.1$

Score:

- receive = 0.00135
- recipe = 0.00011

✅ Chosen correction: "**receive**"

Q8. Design a Hybrid Spelling Correction System (Edit Distance + N-Gram Language Model)

🔹 Motivation

Real-world texts (especially on social media, messaging, etc.) often have:

- Typographical errors
- Context-sensitive ambiguity
- Informal grammar and abbreviations

Using **only edit distance** may return the closest word without considering the sentence.
Combining **edit distance** with **N-gram language models** improves correction by evaluating both form and context.

🔹 Architecture

1. **Candidate Generation using Edit Distance**
   - Generate all words within a small edit distance (e.g., ≤2).
   - Use Levenshtein or weighted edit distance.
2. **Contextual Ranking using N-Gram Language Model**
   - Use bigram or trigram probabilities to determine the best correction based on sentence fluency.

🔹 Example:

Sentence: **"He will adress the audience."**

Step 1: Candidates for "adress":

- "address"
- "adore"
- "dress"

Step 2: N-gram probabilities:

- "will address the" → high score ✅
- "will adore the" → low probability ❌
- "will dress the" → possible, but lower fluency ❌

✅ Final correction: **"He will address the audience."**

🔹 Advantages

- Corrects typos while understanding sentence structure.
- Useful for noisy text (e.g., tweets, chats, emails).
- Reduces false positives by scoring candidates in context.

🔹 Weaknesses

- N-gram models are limited to short context windows (usually 2–3 words).
- Requires large corpus to train language model.
- May miss corrections for domain-specific or rare words.