Levenshteine and semantic scores for evaluating LLMS

- What is Levenshtine algorith
  - Levenshtine distance mesures how many edits (insertions, deletions, substitutions) you need to turn one string into another

Ex truth: "hello world" Diff: "e" → "a" (1 substitution)
model: "hallo word" missing "l" (1 deletion)

⇒ Levenshtine distance = 2

Ex ↳ if true had "worl" then 1 insertion of "d" is in Diff

- great way to mesure text simillarity (smaller diff = closer outputs)

- for LLMs using Levenshtine Algo
  - when generating text we want to know, how close is model output to reference answer
  - for factual, structured or short ans like (math, coding, translation) you can compare model output to ground truth using Levenshtine Distance

$$\text{simmilarity} = 1 - \frac{\text{levenshtein distance}}{\max(\text{len}(a), \text{len}(b))}$$

- gives score between 0 (totally diff) and 1 (identical)

⇒ 1.0 = perfect match
0.8 = almost same
0.4 = quite diff

- Why use Levenshtine vs. semantic score (BLEU, Rouge etc)
  - textual simmilarity ≠ meaning simmilarity

Ex Truth: "the capital of france is paris"
Model A: "paris is capitol of france" ✓
Model B: "Frances main city is paris" ✓ (same meaning, diff words)
model C: "Frances capital is lyon" ✗ (looks simillar, wrong meaning)

- Model A → high Levenshtine, high meaning score → almost exact match (good)
- Model B → low Levenshtine, high meaning score ⇒ Diff words, correct meaning (ok)
- Model C → high Levenshtine, low meaning score ⇒ close looking but wrong (Bad)

- in eval make 2D plots Levenshtine vs semantic like Rouge to mesure both correctly