SIMILARITY CHECK

1. Fuzzy Algorithm for Titles that Look Similar, f

- Purpose: To identify titles or strings that are visually or contextually similar but not exactly the same.
- Method: Utilizes fuzzy matching techniques to handle slight variations, typos, or misspellings in titles.
- Common Libraries/Tools:
  - Python: FuzzyWuzzy, RapidFuzz
  - Scoring: Provides a similarity score (0 to 100) based on approximate matching; more the better.

2. Levenshtein Distance,l

- Purpose: To measure the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one string into another.
- Method: Calculates the distance between two strings to quantify their similarity.
- Common Libraries/Tools:
  - Python: Levenshtein, difflib
- Use Case: Titles that differ slightly in spelling but are conceptually the same.
- Range: 0-n, n=maximum distance from the obtained distance; less the better.

3. Cosine Similarity,d

- Purpose: To measure the similarity between two strings by converting them into vector representations and calculating the cosine of the angle between them.
- Method: Text is transformed into vector space, and the cosine of the angle between vectors is computed.
- Common Libraries/Tools:
  - Python: scikit-learn
- Use Case: Titles with similar word composition but potentially different word order.
- Priority: Title(eg. =1) and Metaphone(eg. = 0.5), then range = [0,1.5); more the better.

4. Order Possibility

- f >= d>= l
- f >= l>= d
- d >= f >= l