**Fuzzy Search Algorithm Implementation**

In our daily life we see the implementation of fuzzy search algorithms everywhere on the internet which makes our life easier, and we use this often without even knowing the underlying implementation of how this works. So, let’s understand,

* **What is Fuzzy search?**
  + This is a technique that finds strings that are approximate match to a pattern, it need not be the exact match.
  + This is also known as approximate string matching.
  + This algorithm gives us the flexibility to search for terms without an exact match.
  + These are implemented in search engines.
* **What problem does Fuzzy Search Algorithm solve?**
  + Nobody is perfect and when we go on the web to search for something we often end up with misspellings, typos, variations in word order and various other lexical errors.
  + Fuzzy search algorithms approximate the word correctly, for example users may type TV for television, info for information, iphne for iphone, AMAZON for amazon, all these examples are different examples for misspelt words, synonyms, typos, case mismatches.
  + Without fuzzy search the results will not be the expected ones and they lead to bad user experience.
  + Fuzzy search addresses the limitations of exact match search.
* Fuzzy employs various algorithms like
  + **Levenshtein Distance** to calculate the minimum edits needed to transform one word to another.
    - This helps with handling minor spelling errors and typographical errors.
  + **Jaccard similarity** measuresthesimilarity between two sets by comparing the size of their intersection divided by size of their union.
    - Works well for comparing large text bodies or documents where set-based operations can be used.
  + **Soundex** encodes words based on their phonetic sound. Similar sounding words have the same Soundex code.
    - Very helpful for matching homophones (similar sounding words may be spelled the same or different example: **bat(bird) : bat(cricket equipment)**, same spelling and pronunciation with different meaning, **sell(to sell something) : cell(like a mobile phone)**, here these two sound similar though they are spelled different and their meaning is also different)
  + **N-gram similarity** breaks down words or phrases into smaller chunks or n-grams and compares them to find similarities.
    - This handles partial matches and helps in autocompletion and correction features.
  + **TF-IDF (Term Frequency – Inverse Document Frequency)** calculates the frequency of the word in a document to determine the importance of the word to the collection or document.
    - Prioritizes terms that are significant in a context to enhance search relevance.
  + **Cosine Similarity** measures the cosine angle between two vectors, to check their relation. Here the text is first encoded using one hot encoding and then cosine similarity is checked against the 2 words.
    - A value of 1 means the 2 words are similar and a value of 0 indicates the 2 words are not similar.
    - This is mostly used in natural language processing and information retrieval to compare documents or queries.
* **Where is this fuzzy search used?**
  + It is used everywhere from google to autocorrect our search queries, in YouTube to suggest the right video that we are looking for, in amazon to fetch the right product that we are looking for etc....
* **Implementation of Fuzzy Search:**

Implementation of fuzzy search involves several steps:

1. Tokenization: break down the user query into tokens
2. Normalization: converting these tokens to a standard form. (removing stop words, special characters, converting them to lower case etc.)
3. Similarity computation: calculating levelshtein distance or n-gram similarity to compare the query tokens against the database index.
4. Ranking and Scoring: then prioritizes results based on the similarity scores
5. Suggestions and Corrections: also offers autocorrected query and autocompletion based on the fuzzy matches.

Example implementation of fuzzy search:

Prerequisites: python, fuzzywuzzy library

Code:

#!pip install fuzzywuzzy

#!pip install python-levenshtein

from fuzzywuzzy import fuzz, process

# list of product names

products = ["Samsung Galaxy Phone", "Apple iphone", "Google Pixel Phone", "Sony Bluetooth Speaker"]

# User's search query with typos

query = "samsng galay phone"

#finding the closest match

match = process.extractOne(query, products)

print(f'Query: {query}')

print(f'Best Match: {match[0]} with a confidence score of {match[1]}%')