**SEARCH ENGINE USING PYTHON**

**Problem Statement:**

The task is to design a typo-tolerant search algorithm to identify the top matches from the given set of company names for the given user query. Consider these cases for the dataset that follows,

**For example:**

Case 1:

For a given user query is “ynos”, the closest match among the above would be “(3) YNOS Venture Engine.”

Case 2:

With a typo in the query such as “ynot”, the top match should still be “(3) YNOS Venture Engine”.

In these two cases, the input string has changed. Since it is relevant to the company name, the program should interpret and find the closest company name associated with the input i.e. ‘YNOS Venture Engine”.

The **Porter stemming algorithm** (or '**Porter stemmer**') removes the commoner morphological and inflexional endings from words in English. Its main use is as part of a term normalization process that is usually done when setting up Information Retrieval systems.

Cosine Similarity: Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the document's size), chances are they may still be oriented closer together. The smaller the angle, the higher the cosine similarity.

Levenstein Distance: The Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other.

**Dependencies Used:**

1. NLTK
2. Keras
3. RE (Regular Expression library)
4. Pandas
5. Numpy

**Approach 1 : Cosine Similarity**

**Step 1:**  The data is taken into a CSV format file, and it is preprocessed for the next steps.

**Step 2:**  Necessary modules are imported for preprocessing and analysis.

**Step 3:** The CSV file is loaded into the program using Pandas library and it is converted into an array format.

**Step 4:** The array is then passed through Porter Stemmer to remove any redundancy.

**Step 5:**  One- hot Encoding is performed to convert the words into vectorized format.

**Step 6:** The input word/ sentence is taken and preprocessed, same as the data.

**Step 7:** The cosine similarity is being calculated on the input word and the sentences in the input file. The cosine similarity function only takes vectorized data.

**Step 8:** The highest cosine similarity index would be considered as the necessary output.

**Step 9:** The output is printed onto the console.

**Approach 2: Levenshtein Distance**

**Step 1:**  The data is taken into a CSV format file, and it is preprocessed for the next steps.

**Step 2:**  Necessary modules are imported for preprocessing and analysis.

**Step 3:** The CSV file is loaded into the program using Pandas library, and it is converted into an array format.

**Step 4:** The array is then passed through Porter Stemmer to remove any redundancy.

**Step 5:** The Edit distance between the input word and words of sentences are calculated using the NLTK library.

**Step 6:** The least Edit distance istaken into consideration.

**Step 7:** The words with the least Edit distance are taken as the result, and the sentence is printed as a search query.

**Conclusion:**

1. Out of two approaches, the second approach, i.e. Lenstein Distance, has given accurate results. This method can be applied to search the queries in the present problem statement.
2. The time-space complexity plays a significant role in this program. Since many iterations are used, the processing may vary with an increase in the dataset. A Machine Learning model has to be trained and tuned to get accurate and faster results to resolve this issue.
3. The cosine similarity can be applicable if the preprocessing can be done in a different approach, i.e. splitting the words in each sentence and applying pre processing operations on it and calculating the cosine similarity word to word.

References:

1. [www.cuelogic.com/blog/the-levenshtein-algorithm#:~:text=The%20Levenshtein%20distance%20is%20a,one%20word%20into%20the%20other.](https://www.cuelogic.com/blog/the-levenshtein-algorithm#:~:text=The%20Levenshtein%20distance%20is%20a,one%20word%20into%20the%20other.)
2. <https://www.machinelearningplus.com/nlp/cosine-similarity/>