Python Test

Nikhil Sebastian

October 26, 2024

Methodology

Broadly, to measure the similarity between two firm names, I first quantify how similar the two strings are. This measure is based on the Levenshtein distance which measures how many single-character edits (insertions, deletions, or substitutions) are required to change one string into another. The fuzzy ratio normalizes this distance into a score between 0 and 100, where:

- 100 means that the strings are identical
- 0 means that there is no similarity at all

Before I calculate the distance, I run the vector of strings through a pre-processing function which:

1. removes punctuation and non-alphanumeric characters using regular expressions
2. converts all characters in a string to lower case
3. removes stop words from a list of custom stopwords
4. removes white spaces

After the pre-processing step, and the computation of the measure of similarity - I add a column with values yes/no for firm names that have to be changed. In particular, the firm name has to be changed if the similarity score is greater than or equal to 80. If the firm name has to be changed, it looks for the nearest value 'no' above it, and takes the corresponding firm name from that row (after the data is sorted by country and firm name).

A key assumption is that the values occur in even counts - for instance, on sorting, the algorithm would not be perfect if there were three counts of the same firm with different names as the current algorithm does a sequential pairwise comparison.

Optimization

Ideally, I would have liked to use a larger collection of stopwords - possibly a different set for each country to include differences in language. If there was more time, I would have liked to construct a distance matrix to compute the distance between two strings more efficiently than the sequential pair-wise matching.

The code is also available in the GitHub rep: https://github.com/nikhilsebastiank/PythonTest2024.