Oregon stste
Cs 562
Zhen Tang

Testing proposal for python library “Fuzzywuzzy”

For the testing project this term. I am planning to test the python package “fuzzywuzzy”.

Fuzzywuzzy is a fuzzy string matching library built by the fine people at SeatGeek. It is, at the beginning, used for find sports and concert tickets. As it explained in the designer’s blog, the original use case of this package was the problem that some events or sports games would sometimes have many different names or labels. They may contains or hide some information in the label such as the address, date or venue, which makes people hard to figure out whether two ticket listings are for the same event. Basically, fuzzywuzzy implements things like string comparison ratios, token ratios, and plenty of other matching metrics.

I would pick some functions from this system to test if they can deal with input data appropriately and if the system is sturdy enough.

1) partial_ratio

partial_ratio is a function used to calculate the similarity score between two strings. The more similar the two string are, the higher the score would be. If one string is exactly same to a part of another, it would get full mark. For example, if we use "ACME Factory", "ACME Factory Inc." as input, the output should be 100. So for testing, I would use random function to generate string pairs that one string is consist by another string and some other parts. But the shorter string would appears at different part of another string, which could be the beginning, the end, or other random positions. By definition, all those test case should get 100 at output.

2) fuzz.token_sort_ratio

fuzz.token_sort_ratio function would split strings on white spaces, lowercase everything and ignore non-alpha non-numeric characters, which means punctuation is ignored (as well as weird unicode symbols).

For instance,

‶

```
>>> fuzz.token_sort_ratio("fuzzy wuzzy was a bear", "wuzzy fuzzy was a bear")
    100
```

‶

After splitting and sorting, both of those two strings would become a set with exactly same substrings.

So in order to test this function, I would choose two methods. First, generate wo strings with same substring but arrange those substrings in different order. Second, insert some Unicode or other non-alpha non-numeric characters in strings and then use these strings as input of the function. If the function works as it explains, we should always get 100 with those inputs.

3) fuzz.token_set_ratio

The function fuzz.token_set_ratio is also used for compare two string while ignore non-alpha non-numeric characters. Instead of compare the whole string. Token_set_ratio function would split the input string then create them in a set of substring. That same substring would only appears ones. Thus, the testing of this function could use steps below.

Oregon stste
Cs 562
Zhen Tang

1. Generate a set of strings which does not contain space or non-alpha non-numeric characters.
2. Use substrings in the set to create 2 different strings. Make sure every substring are used and may be used several times in both of strings.
3. Insert spaces and special characters to each string.
4. Put the string created as the input of the function.

If works correctly, the function should output 100 because they are exactly generated by the same set of substring.