CS 562
Final report
Zhen Tang
3/14/16
Project report

For the testing project this term. I tested some functions from the python package "fuzzywuzzy". Fuzzywuzzy is a fuzzy string matching library built by the fine people at SeatGeek. My main goal of this project is to test the stability of this package. Generally speaking, what the package do is to take one (or more) string as input and have operations based on the similarity ratio between strings.

Because the library is used for string matching and it is actually not easy to check the correctness of the ratio. My test are mainly concerned about the stability of the functions. In the first several weeks, I used a base string as the basic of the input string I created and then use the base string as another input of the partial ratio function. That should guaranteed the output of the function would always be 100 which is easy for me to have assertion after each steps. In fact, it did pass all the tests. I think it would be interesting if I could find some different ways to construct strings to cover more cases. After finishing the testing about partial ratio function, I'm now working on construct appropriate test strings for testing token sort ratio function and token set ratio function. Until now, this system passed all of tests I created. I also have some effort on testing "process" function but there are not enough decent results.

The functions I have already tested are:

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. I used a basic string and some random strings to generate two test strings and use them as input for the partial_ratio function.

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).

Basically speaking, the test methods I used are similar to those I used above.

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.

process.extractOne

This function used to pick the closest option from a list of strings. In this test I created a list of four strings and then use a string as input to check the performance.

CS 562
Final report
Zhen Tang
3/14/16
During this term, I spend most of time tried to find the best way to create the input strings for the functions which I tested. I used both concatenate a lot of short strings to create a long string to test their performance under extreme size conditions and generate some random words to test their performance

Bug report:

Until now, I have not find any bugs from this package. Firstly, with all the random strings I tried as input, the function can work appropriately and produce the output value. However, it is not easy to check the correctness of the functions unless I can write my own functions which can do the same thing and then compare the results. So my test are mainly concerned about the stability of the functions.   In this term I can only show that these functions works well for most of simple inputs. And the result are generally shown as I expected.

The quality of the system

Basically speaking, the system under test did a good job. Actually, the functions in the package are quite simple and straightforward. Since I have not found any bug yet. I would like to agree that the system can works well in most cases. I already tested most of functions in the package. Although I didn't install the Levenshtein Distance tools, which would improve the performance of the system if installed, the system already shows a good performance. From my observation, the tester spend much more time initialize the context than doing calculate work.

The problems about TSTL

In this term, I found a problem about TSTL when running on the windows system. The system I used is windows 10 system. In this term, I tried both Git shell and Powershell to run the tstl programs. In order to fix the permission issue in coverage testing phase, I used administrator mode running both shells. However, every time when I run the randomtester, the system would crash. So I haven't successfully running the randomtester under administrator mode, which means that I couldn't fix the coverage issue. I hope the issue is just a special case that only shown on my system.

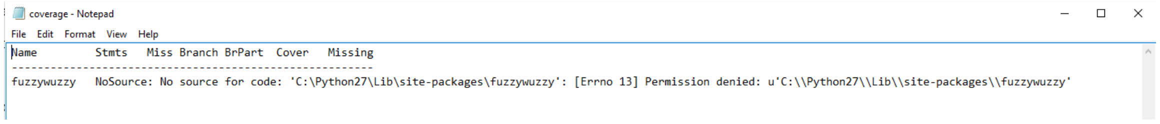
Another problem I faced is that it is not easy to find out where the bugs come from. While the tester crash during running, the system message would show the lines in the code. However, the lines it shows are where those bug codes are in the randomtester.py. So it is hard to figure out that is it a tester bug or system bug. I think it would be better to have a tool to analyze the system message.

The good areas about TSTL

The first good thing about TSTL is that it is easy for tester with less experience to create their test harness. As which is known to us, construct a test harness is a highly error prone work, especially for unexperienced programmers. With TSTL, people would avoid many problems that would have in constructing their test harness.

Coverage summaries

CS 562
Final report
Zhen Tang
3/14/16



```
Name         Stmts   Miss Branch BrPart  Cover   Missing
-------------------------------------------------------
fuzzywuzzy   NoSource: No source for code: 'C:\Python27\Lib\site-packages\fuzzywuzzy': [Errno 13] Permission denied: u'C:\\Python27\\Lib\\site-packages\\fuzzywuzzy'
```

As I mentioned above, I didn’t get the coverage report successfully due to the permission issue. I tried to running the tester under administrator mode but it makes my operation system crash. Thus I can only make an approximate calculate about the coverage. In my project I tested four of six functions shown in the document of the package.

As a result, I think my project is far from success. It is worth spending more time for me to get an fully understanding of TSTL and the FuzzyWuzzy system that I tested.