Zahory Velazquez

04/21/2019

This assignment was really fun to do, especially the database portion. I struggled the most in the database portion because I wasn’t familiar with sqlite formatting. I used stackoverflow and sqlite documentation to try to understand how to insert or select rows, but it wasn’t until I found a youtube video where I fully understood why we need to use ? or :some\_key as placeholders rather than using python formatting. I learned it was necessary to use ? or :some\_key to prevent SQL injection (which we had covered in Web Apps, so it made sense). I mainly used the :some\_key format because it makes it more readable than ? as placeholders.

I was comparing the results given from the AutoCorrect.exe to my results. We didn’t get all of the similar words for the possible correct words, but the main words always showed up so I’m not sure if the order of the rest mattered. I created classifyEditDistance() function to accept the incorrect word and from there I created a dictionary where the keys are the edit distance and the values are a list of all the words that share the same edit distance. As python supports sorting keys, I inputted all the values with smallest edit distance into a list of tuples. If the size of the list was less than 10, I went to the next key and filled my list until it reached size 10. That’s the only possibility I can think of that made me not get the exact possible correct words. The reason why I questioned if it mattered was because I still got the correct output files.

In this assignment, I did incorporate some techniques that I learned from David’s python class, such as the defaultdict, string formatting, \*args, sort, None, keyword arguments. Since I am using python, I thought I tried writing the “pythonic way” (didn’t go that well haha). There was instance where I wanted to use the next() function in which I learned in David’s, but I wasn’t sure how to create the generator for it. I wanted to use it in my destinationFile() for when it came to writing the correct words to a new file. If I could use the next() function then I wouldn’t have to separate the last element in my list from the rest.

I thought it would be a great idea if I could get all the data from the files to be corrected and insert it into a list. However, this meant that I needed to separate the punctuations from the word. I used findall() from the re (regular expressions) module. Understanding how the patterns worked was honestly difficult. The r stands for raw, \w stands for any word character, + will match 1 or more repetition, | think of it as or, and anything in the braces [] is our set. I’m not explaining it thoroughly, but the only reason why it began to make sense was because we learned regular expressions with David during Discrete Math. It was really great seeing how the material we learned in other classes being applied during this program.