Report on Assignment 5

1. **Notable difficulties that I overcame:**
2. It took me a long to time to debug the program: there was one error that I had, in which for the normalizeRules function will go in an infinite loop if the last rule had an empty string. It took me a while to figure out where the problem was, and I used the debugger and cerr outputs to find it.
3. Separating the C-String with spaces was hard: What I did instead was I looked up the documentation for C-Strings, and found a function strtok() that would separate the C-String into substrings with the space between them.
4. **Pseudo-code for my program:**

removeRule: Used for removing the rule at position

Shifts all rules after position up one, replacing the rule to be removed

Decreases the number of good rules by 1

normalizeRules:

number of good rules = number of rules

For each rule:

If the rule has non-letter characters, empty strings or a non-positive distance

Invoke the removeRule function on the rule

For each rule:

Turn every letter in the rule to lower case

Switch the word1 and word2 such that word1 < word2

For each rule:

If there is another rule with the same word1 and word2

Compare the distance of the two rules, and invoke removeRule on the rule with the smaller distance

Return the number of good rules

calculateSatisfaction:

Satisfaction score= 0;

Check if document is empty, return 0 if true

Create a copy of the document

Turn all letters in the copy to lowercase

For each letter in the copy

If the letter is a non-letter (except space)

Delete it by shifting all characters after it to the left by one position (also shifts the null byte)

Separate the words in the copy using the space character and the function strtok(), then place the words into a 2-D char array.

For each word in the char array:

Compare against each word1 in the rules, if same:

Search words in front of it (bounded by distance of this rule/number of words in document) for the word2 in this rule, if found:

Increase satisfaction score by 1

Do the same for words behind it (bounded by distance of this rule)

Increase satisfaction score by 1 if word2 is found

Return satisfaction score;

1. **Test cases for my program:**

normalizeRules:

Test 1:

|  |  |  |
| --- | --- | --- |
| Word1 | Word2 | Distance |
| alpha | beta | 2 |
| alpha | beta | 3 |
| beta | gamma | 5 |
| gamma | delta | 4 |
| beta | gamma | 2 |

Successful, check whether if the function is able to delete duplicate rules

Test 2:

|  |  |  |
| --- | --- | --- |
| Word1 | Word2 | Distance |
| alpha | BEta | 2 |
| A!PHa | beta | 3 |
| beta | gamma | 5 |
| gamma | !!! | 4 |
| betA | gamma | 2 |

Successful, check whether if the function is able to handle words with uppercase/words with non-letters

Test 3:

|  |  |  |
| --- | --- | --- |
| Word1 | Word2 | Distance |
| alpha | beta | -4 |
| alpha | BEta | 2 |
| AlPHa | beta | 3 |
| beta | gamma | 5 |
| Bad | Rule | 0 |
| gamma | dElta | 4 |
| betA | gamma | 2 |
| Terrible | Project | -9999 |

Successful, check whether if the function is able to handle non-positive distance and remove rules at start, middle and end.

Test 5:

|  |  |  |
| --- | --- | --- |
| Word1 | Word2 | Distance |
|  | beta | 5 |
| alpha | BEta | 2 |
| AlPHa | beta | 3 |
| beta | gamma | 5 |
| BadRule |  | 7 |
| gamma | dElta | 4 |
| betA | gamma | 2 |
|  |  | 5 |

Successful, check whether if the function is able to handle empty strings in rules

Test 6:

|  |  |  |
| --- | --- | --- |
| Word1 | Word2 | Distance |
|  | beta | 5 |
| alpha | BEta | 2 |
| beta | Alpha | 2 |
| beta | gamma | 5 |
| gamma | dElta | 4 |
| gamma | beta | 2 |
|  |  | 5 |

Successful, check whether if the function is able to handle identical rules with word 1 and word 2 switched, and rules with identical word 1, word 2 and distance

Test 7:

|  |  |  |
| --- | --- | --- |
| Word1 | Word2 | Distance |
|  |  | 5 |
|  |  | 5 |

Successful, test whether if function is able to return zero rules

Test 8:

|  |  |  |
| --- | --- | --- |
| Word1 | Word2 | Distance |
|  | scientist | 2 |
| deranged | robot | -4 |
| ! | wRong | 2 |
| plot | nafarious | 1 |
| have | mad | 13 |
| mad | have | 4 |
| plot | nefaRious | 5 |
|  | badRule | 4 |

Successful, mixture of all cases above

calculateSatisfaction:

Rules1:

|  |  |  |
| --- | --- | --- |
| Word1 | Word2 | Distance |
| mad | scientist | 2 |
| deranged | robot | 4 |
| nefarious | plot | 1 |
| have | mad | 13 |

Test1, using Rules1:

|  |
| --- |
| "I'm upset that on Nov. 15th, 2016, my 2 brand-new BMW M760Lis were stolen!!" |

Successful, test whether if function can handle documents with non-letters and turn them into words

Test2, using Rules1:

|  |
| --- |
| “H!ELP I'm being oppressed!!!” |

Successful, test whether if function can handle documents with non-letters and multiple spaces between words and turn them into words

Test3, using Rules1:

|  |
| --- |
| "The mad UCLA scientist unleashed a deranged evil giant robot." |

Successful, test whether if function can return a correct satisfaction score

Test4, using Rules1:

|  |
| --- |
| "The mad UCLA scientist unleashed a deranged robot..." |

Successful, test whether if function can return a correct satisfaction score when the document has non-letters and multiple spaces

Test5, using Rules1:

|  |
| --- |
| "\*\*\*\* 2016 \*\*\*\*" |

Successful, test whether if function can return 0 when there are no actual words in the document

Test6, using Rules1:

|  |
| --- |
| "" |

Successful, test whether if function can return 0 when the document is an empty string

Test7, using Rules1:

|  |
| --- |
| " That plot: NEFARIOUS!" |

Successful, test whether if function can return a correct satisfaction when the document has uppercase letters and is shorter than the distance of rules

Test8, using Rules1:

|  |
| --- |
| "deranged deranged robot deranged robot robot" |

Successful, test whether if function can return a correct satisfaction when the document has multiple instances of the same rule

Test9, using Rules1:

|  |
| --- |
| "deranged deranged robot deranged robot robot" |

Successful, test whether if function can return a correct satisfaction when the document has multiple instances of the same rule

Test10, using Rules1:

|  |
| --- |
| "That scientist said two mad scientists suffer from deranged-robot fever." |

Successful, test whether if function is able to search within the distance of the rule correctly, and whether if it treats words separated only by a non-letter (except spaces) correctly

Test11, using Rules1:

|  |
| --- |
| "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" |

Successful, test whether if function can handle document at character limit (200)

Test12, using Rules1:

|  |
| --- |
| " a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a aa" |

Successful, test whether if function can handle document with the maximum possible amount of words (100)