Report

1. Obstacles

The logic within the rate function was too complicated for me at first. It seemed that separating words in document out was a very difficult aim. So I planned out the steps I need to achieve before I attempt to write each small manipulation of the document.

So here is my preparation:

In order to get the score of a document, I need first to make it into proper syntax with only alpha and space in it;

Second, I need to make the new document into an array of cstring, so that I can get the separations between each word more efficiently;

Third, I drew a picture of the relationships between the four arrays: wordlist, word1, word2, separations;

Fourth, I found that matching the word in word1 with wordlist and then determine the separation between this word and the same word in wordlist and word2 is the fastest method for rating.

Hence, I knew how to cooperate my manipulations in each step in an efficient manner without fumbling around.

1. Pseudocode
2. Set a function for calculating the absolute value of the difference between two integers.

Return the value of the bigger integer minus the smaller integer

1. Set a function of making proper a collection of patterns.

Set the number of patterns to 0 if it is less than 0;

Repeatedly in each pattern:

Set the isValid as false if any of the two words in the pattern is null

(contains no characters) or the separation is less than one;

Repeatedly:

Set the isValid as false if any character in the words in the pattern

isn’t alpha or space;

Otherwise make every alpha in the words to lowercase;

Delete the words and separation in the pattern if isValid is false.

Get the number of remaining patterns as count;

Initialize the isValid.

Repeatedly:

If there are two patterns containing two same words in either exchanged

sequence in the two wordlists or the same sequence,

set the isValid as false;

change the smaller separation value equal to the bigger one;

delete the pattern first examined;

get the number of remaining patterns as count;

initialize the isValid;

1. Set a function rating a document according to a collection of patterns.

Set the number of patterns to 0 if it is less than 0;

Declare a new document;

Repeatedly:

Store document’s character with lowercase in the new document if the

character is an alpha or is a space before an alpha;

delete the first character in the new document if the first character is a space;

declare an array of words (each word is an array of characters) and set it to null;

repeatedly for the characters in the new document:

store the character of the new document into the array of words if it is not a space;

stop storing the character and step to the next word in the array if a character in the new document is a space;

repeatedly in each pattern:

repeatedly:

if a word in word1 equals to a word in the array of words:

repeatedly:

if word2 equals to another word in the array of words and the separations between this word and the word previous corresponding word in the array is less than or equal to the required separation in the pattern:

increase score incrementally and set signal as true;

break;

set signal as false and break if signal is true;

return score;

1. Test Data

**Test makeProper Function**

//1. an already proper collection of patterns with uppercase character in words.

const int TEST1\_NRULES = 4;

char test1w1[TEST1\_NRULES][MAX\_WORD\_LENGTH+1] = {"mad","deranged","NEFARIOUS","have"};

char test1w2[TEST1\_NRULES][MAX\_WORD\_LENGTH+1] = {"scientist","robot","PLOT","mad"};

int test1dist[TEST1\_NRULES] = {1,3,2,1};

//2. containing a same pattern with exactly same words with same positions.

const int TEST2\_NRULES = 5;

char test2w1[TEST2\_NRULES][MAX\_WORD\_LENGTH+1] = {"mad","deranged","NEFARIOUS","deranged","have"};

char test2w2[TEST2\_NRULES][MAX\_WORD\_LENGTH+1] = {"scientist","robot","PLOT","robot","mad"};

int test2dist[TEST2\_NRULES] = {1,3,0,2,1};

//3. containing same patterns with exactly same words with exchanged positions in word1 and word2.

const int TEST3\_NRULES = 6;

char test3w1[TEST3\_NRULES][MAX\_WORD\_LENGTH+1] = {"mad","deranged","NEFARIOUS","robot","have", "MAD"};

char test3w2[TEST3\_NRULES][MAX\_WORD\_LENGTH+1] = {"scientist","robot","PLOT","deranged","mad", "hAVe"};

int test3dist[TEST3\_NRULES] = {1,3,0,2,1,2};

//4. containing word with characters other than alpha

const int TEST4\_NRULES = 5;

char test4w1[TEST4\_NRULES][MAX\_WORD\_LENGTH+1] = {"mad","deranged","NEFARIOUS","half-witted","ROBOT"};

char test4w2[TEST4\_NRULES][MAX\_WORD\_LENGTH+1] = {"scientist","robot","PLOT","assistant","deranged"};

int test4dist[TEST4\_NRULES] = {1,3,0,2,1};

//5. containing word with same pattern appearing more than twice

const int TEST5\_NRULES = 7;

char test5w1[TEST5\_NRULES][MAX\_WORD\_LENGTH+1] = {"mad","MAD","Scientist","have", "HAVE", "MAd","mad!@#"};

char test5w2[TEST5\_NRULES][MAX\_WORD\_LENGTH+1] = {"scientist","ScientIst","maD","mad","Mad","hAve","Have"};

int test5dist[TEST5\_NRULES] = {1,3,2,1,3,4,3};

//6. containing many error simultaneously

const int TEST6\_NRULES = 11;

char test6w1[TEST6\_NRULES][MAX\_WORD\_LENGTH+1] = {"ai","bi","ci@@","D","D","D","E","","",""," "};

char test6w2[TEST6\_NRULES][MAX\_WORD\_LENGTH+1] = {"AI","BQ","Cassf21423","E","E","D","",""," "};

int test6dist[TEST6\_NRULES] = {-1,-2,3,4,5,4,3,2,1,-2,0};

//7 containing testrules equals to zero

const int TEST7\_NRULES = 0;

char test7w1[TEST7\_NRULES][MAX\_WORD\_LENGTH+1] = {};

char test7w2[TEST7\_NRULES][MAX\_WORD\_LENGTH+1] = {};

int test7dist[TEST7\_NRULES] = {};

//8 containing separations less than zero

const int TEST8\_NRULES = 5;

char test8w1[TEST8\_NRULES][MAX\_WORD\_LENGTH+1] = {"mad","deranged","NEFARIOUS","deranged","have"};

char test8w2[TEST8\_NRULES][MAX\_WORD\_LENGTH+1] = {"scientist","robot","PLOT","robot","mad"};

int test8dist[TEST8\_NRULES] = {1,-3,0,-2,-1};

**Test rate Function**

int score1, score2, score3, score4, score5, score6, score7, score8;

//1. Normal document with lowercase in key words

score1 = rate("The mad UCLA scientist unleashed a deranged evil giant robot.",test1w1, test1w2, test1dist, TEST1\_NRULES);

//2. Normal document with uppercase in key words

score2 = rate("The MaD UCLA SCientIST unleashed a dERanged evil giant rOBot.",test1w1, test1w2, test1dist, TEST1\_NRULES);

//3. Document with special characters and numbers

score3 = rate("The mad UC#$#&LA scientis123t unleashed @#$% a deran#)$(ged evil giant123 robot.",test1w1, test1w2, test1dist, TEST1\_NRULES);

//4. Document with more than one space separating words

score4 = rate("The mad UCLA scientist unleashed a deranged evil giant robot.",test1w1, test1w2, test1dist, TEST1\_NRULES);

//5. Document with space characters at the beginning

score5 = rate(" The mad UCLA scientist unleashed a deranged evil giant robot.",test1w1, test1w2, test1dist, TEST1\_NRULES);

//6. Document with words satisfying a pattern more than once

score6 = rate("The mad scientist mad scientist scientist unleashed a deranged evil robot robot.",test1w1, test1w2, test1dist, TEST1\_NRULES);

//7. key words with special character connecting tham instead of space

score7 = rate("The mad-scientist unleashed a-deranged evil giant-robot.",test1w1, test1w2, test1dist, TEST1\_NRULES);

//8. document containing no words matching the patterns

score8 = rate("12e32rwaedsf asw321 1323 32asf sfs#@#!#!",test1w1, test1w2, test1dist, TEST1\_NRULES);