2019510094 - Elif Özker

ALGORITHM

1. Minimum Edit Distance Calculation: Inputs are two strings, word1 and word2. Output is a 2d array cost matrix representing the minimum edit distances between word1 and word2.First of all i initialized a 2d array costMatrix with dimensions m+1 and n+1. m+1 equal to word1 length and n+1 equal to word2 length. The minimum edit distances between the input word substrings are stored in a matrix that is initialized by this function. Iteratively going through each character in the words, it updates the matrix according to the least amount of work required for insertion, deletion, or substitution.

2. Word Distance Calculation: Inputs are a string inputWord and a list of strings wordList.Output a Map wordDistances representing the minimum edit distances between inputWord and each word in wordList.Algorithm is for each word in wordList, calculate the minimum edit distance using the minEditDistance method and store the distances in wordDistances with the word as the key and distance as the value.

3. Transformation of Words: Inputs are a 2D array costMatrix, and two strings. Output are a list of maps (operations) representing the transformation operations needed to convert word1 to word2.Algorithm is traverse the costMatrix and determine the operations (insert, delete, replace) needed to transform word1 into word2.

Functions/Procedures:

calculateAndDisplayDistances Function: Input is User-input word from the GUI (**inputWord**).Output is display top five words with the minimum edit distance in the GUI result area.Procedure is call **calculateDistances** to obtain word distances and display the top five words in the result area.

displayCostMatrix Function:Input is User-input words from the GUI (**word1** and **word2**).Output is Display the cost matrix and transformation operations in the GUI result area.Procedure is Call **minEditDistance** to get the cost matrix.It is calling **transformWords** to get the transformation operations and display the cost matrix and transformation operations in the result area.

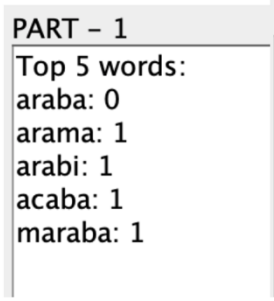
Total running time for Part 1 and Part 2 (for one run):

For part – 1 : 0.194332375

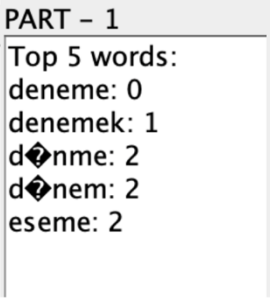
For part – 2 : 0.011806916

Part 1: words - alternative correct words list for at least 5 different words with 5 nearest words

araba



deneme

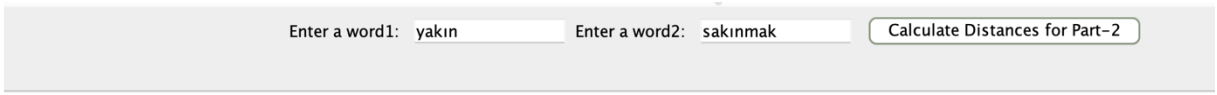


bilmiyorum

çözmek

telefon

Part 2: run the algorithm 2 times and write the results for any words you choose (screen shots of the solution for each run).



PART - 2

Cost Matrix:

0	1	2	3	4	5	6	7	8
1	1	2	3	4	5	6	7	8
2	2	1	2	3	4	5	6	7
3	3	2	1	2	3	4	5	6
4	4	3	2	1	2	3	4	5
5	5	4	3	2	1	2	3	4

oldCharacter --- y
newCharacter --- s
position --- 1
operation --- Replace
---
character --- a
operation --- Same word
---
character --- k
operation --- Same word
---
character --- ı
operation --- Same word
---
character --- n
operation --- Same word
---
character --- m
operation --- Insert
---
character --- a
operation --- Insert
---
character --- k
operation --- Insert

Enter a word1: benzer Enter a word2: benze Calculate Distances for Part-2

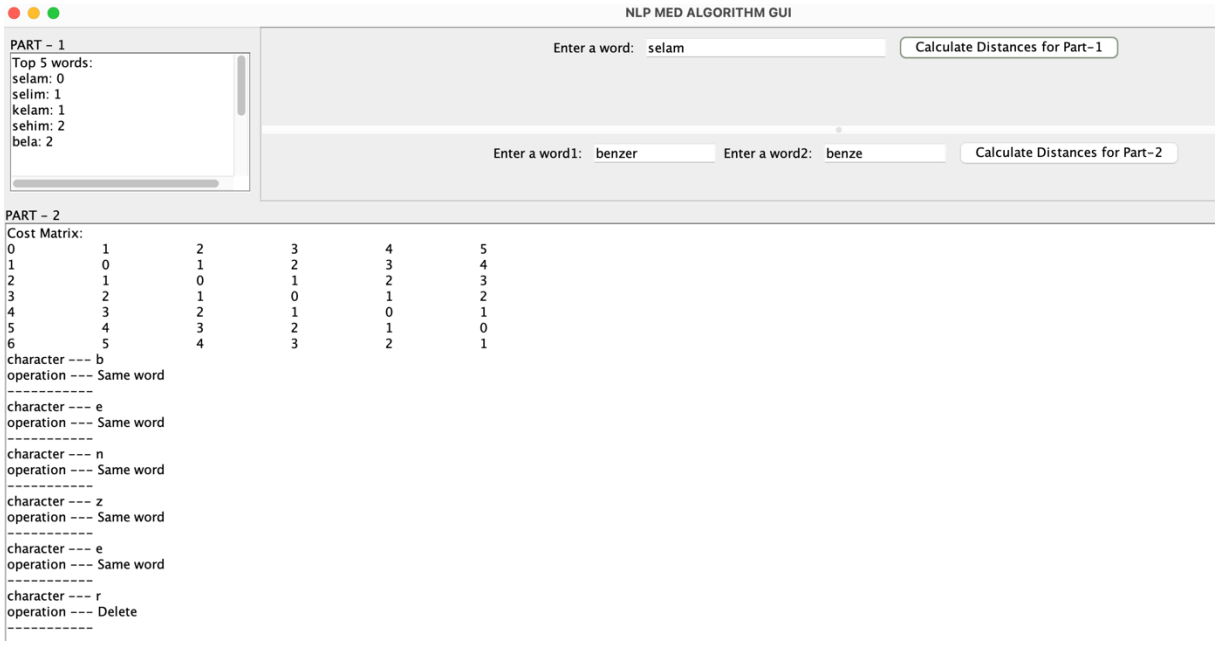
PART - 2

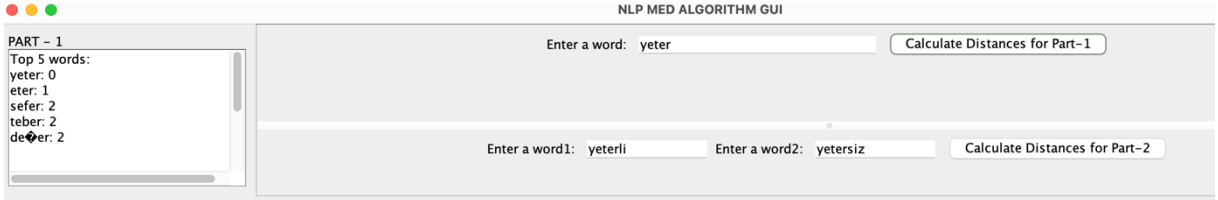
Cost Matrix:

0	1	2	3	4	5
1	0	1	2	3	4
2	1	0	1	2	3
3	2	1	0	1	2
4	3	2	1	0	1
5	4	3	2	1	0
6	5	4	3	2	1

character --- b
operation --- Same word
---
character --- e
operation --- Same word
---
character --- n
operation --- Same word
---
character --- z
operation --- Same word
---
character --- e
operation --- Same word
---
character --- r
operation --- Delete

SCREENSHOTS:





PART - 2

0	1	2	3	4	5	6	7	8
1	0	1	2	3	4	5	6	7
2	1	0	1	2	3	4	5	6
3	2	1	0	1	2	3	4	5
4	3	2	1	0	1	2	3	4
5	4	3	2	1	0	1	2	3
6	5	4	3	2	1	1	2	3
7	6	5	4	3	2	2	1	2

character --- y
operation --- Same word
---
character --- e
operation --- Same word
---
character --- t
operation --- Same word
---
character --- e
operation --- Same word
---
character --- r
operation --- Same word
---
oldCharacter --- l
newCharacter --- s
position --- 6
operation --- Replace
---
character --- i
operation --- Same word
---
character --- z
operation --- Insert
