CSE 654/484

NATURAL LANGUAGE PROCESSİNG

HW1 Report

COŞKUN ŞALTU

1801046231

1) Problem Definition

In this assignment, random sections of 20 different textbooks were taken and 100 different text files, each containing 400 lines, were created. The same lines are placed in each of the created text files. Then, each text file was compared with each other and similar lines were found.

2) Problem Solution

Smith-Waterman algorithm was used to solve the problem. The Smith-Waterman algorithm simply aims to take two strings and find the common areas in these two strings.

In order to observe the difference between substrings we will first have to define a scoring function so that we can determine which substrings match more closesly than others. In this scoring function we will give any two matching characters a +2 if the characters at some index match and -1 otherwise. A single-character change still costs -1. Notice that if two characters are not aligned (contributing a score of -1 to our scoring function), but we can align them with a single-

character change, we can, in effect, “pay” a cost, -1, in order to gain a +2 from the two-characters becoming aligned.

So,

Score Match: +2

Score Dismatch: -2

Score Insertion and Deletion: -1

Using these scoring methods, a 2-dimensional matrix was obtained. Matching strings were found by applying the **traceback** process in this matrix.

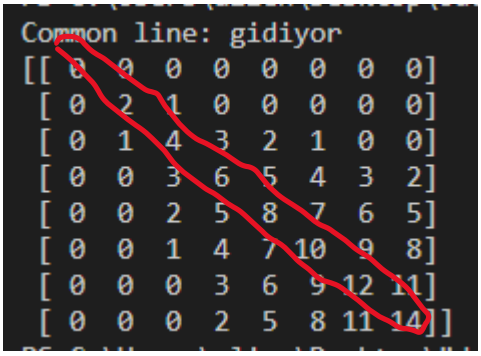
3) Test Algorithm

The algorithm was tested same strings:

```
if __name__ == "__main__":

    # compareTwoFiles("txts/3.txt", "txts/50.txt")
    smith_waterman("gidiyor", "gidiyor")
```

Result:



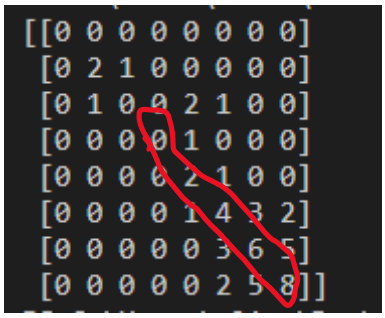
As you see, The matrix is constantly increased by 2 due to the matching of all characters.

The algorithm was tested by different strings:

```
if __name__ == "__main__":

    # compareTwoFiles("txts/3.txt", "txts/50.txt")
    smith_waterman("gidiyor", "geliyor")
```

Result:



Just -iyor is matched.

Algorithm was tested two files: