CSE654

Introduction to Natural Language Processing

Fall 2022

Homework - I

Çağla Şahin
Instructor: Yusuf Sinan Akgül
9 November 2022

In this homework, I am supposed to download the standard textbooks from internet, convert them to txt files, insert some same lines of texts into random positions of random text documents and try to find out it with the algorithm called Smith-Waterman.

First of all, I downloaded 10 books and divide them 10 different txt files and truncate them to make my process faster. Here, the list of books I used.

- 1.txt-10.txt - Çağdaş Türk Tarihi
- 11.txt-20.txt - Felsefe
- 21.txt-30.txt - Psikoloji
- 31.txt-40.txt - Sosyoloji
- 41.txt-50.txt - Sanat Tarihi
- 51.txt-60.txt - Biyoloji
- 61.txt-70.txt - Coğrafya
- 71.txt-80.txt - Dinler Tarihi
- 81.txt-90.txt - Kimya
- 91.txt-100.txt - TC İnkılap Tarihi ve Atatürkçülük

After preparing the test data, I have written the Smith-Waterman algorithm which is given on the lecture slides.

In [31]:
```
def smith_waterman_algorithm(text1, text2, match=3, mismatch=-1, insertion=-1, deletion=-1):
    # if text-2 is longer than text-2, then swith them for searching.
    if(len(text2) > len(text1)):
        text1, text2 = text2, text1

    # need a distance matrix to calculate how similar these two texts.
    # matrix should be one index larger for both column and row for gap on start.
    distance_matrix = np.zeros((len(text2), len(text1)))

    for i in range(0,len(text2)):
        for j in range(0,len(text1)):
            # situation-1

            distance_matrix[i, j] = max(
                # negative values are not allowed
                0,
                # if previous character matches increase the score by match, else decrease it by mismatch
                distance_matrix[i - 1, j - 1] + (match if text1[j - 1] == text2[i - 1] else mismatch),
                # one character is missing in seq2, so decrease the score by deletion
                distance_matrix[i - 1, j] + deletion,
                # one additional character is in seq2, so decrease the scare by insertion
                distance_matrix[i, j - 1] + insertion
            )
            #print(text2[i], text1[j])
            #print(distance_matrix[i, j])



    df = pd.DataFrame(distance_matrix, columns = text1, index=text2)

    #print(df)
    return np.max(distance_matrix)
```

To test the algorithm on the text files, I created some scenarios and run the code.

1. Scenario:

Test how the algorithm works on the text files which belongs to same topic, find out how similar the text files when they includes similar topics.

To see the results, I tried the algorithm on each 10 text files seperately and got the results as below.

```
In [13]: """
         1-10 - cagdas turk tarihi - list1
         11-20 - felsefe -list2
         21-30 - psikoloji -list3
         31-40 sosyoloji -list4
         41-50 sanat tarihi - list5
         51-60 biyoloji -list6
         61-70 cografya -list7
         71-80 dinler tarihi -list8
         81-90 kimya - list9
         91-100 tc inkılap tarihi ve atatürkçülük -list10
         """

         print(list1_scores)
         print(list2_scores)
         print(list3_scores)
         print(list4_scores)
         print(list5_scores)
         print(list6_scores)
         print(list7_scores)
         print(list8_scores)
         print(list9_scores)
         print(list10_scores)

         [12.0, 12.0, 9.0, 12.0, 11.0, 9.0, 9.0, 9.0, 11.0]
         [8.0, 9.0, 9.0, 9.0, 9.0, 6.0, 8.0, 6.0, 1293.0]
         [12.0, 7.0, 9.0, 8.0, 8.0, 7.0, 6.0, 6.0, 5.0]
         [12.0, 11.0, 6.0, 9.0, 12.0, 12.0, 12.0, 11.0, 9.0]
         [23.0, 24.0, 73.0, 11.0, 28.0, 67.0, 10.0, 45.0, 45.0]
         [12.0, 12.0, 12.0, 28.0, 44.0, 12.0, 12.0, 14.0, 12.0]
         [6.0, 9.0, 8.0, 12.0, 6.0, 6.0, 15.0, 20.0, 20.0]
         [10.0, 9.0, 12.0, 7.0, 47.0, 26.0, 27.0, 27.0, 21.0]
         [15.0, 15.0, 15.0, 15.0, 15.0, 18.0, 15.0, 10.0, 6.0]
         [12.0, 16.0, 8.0, 13.0, 9.0, 6.0, 29.0, 6.0, 19.0]
```

After this point, I felt compelled to create second scenario which is;

2. Scenario:

Test how the algorithm works on the text files which belongs to different topic, find out how similar the text files when they includes completely different topics.

To test this scenario, I should have paired two different course books. I paired *Biyoloji* and *Dinler Tarihi*, *Coğrafya* and *Kimya, Psikoloji* and *TC İnkılap*

Tarihi ve Atatürkçülük. I paired these subjects considering how different their fields are. To get a general result of this testing, I tried every two pair in two different 10 text files. E.g: first text from Biyoloji and paired text from Dinler Tarihi.

```
In [25]:
"""
to get a better prediction result, i will compare each text on same indexes of different lists.
"""

#biyoloji and dinler tarihi
mixedlist_scores = []
i = 0
while i < 10:
    score = smith_waterman_algorithm(list6[i], list8[i])
    mixedlist_scores.append(score)
    i+=1
    #print(score)

print("biyoloji-dinler tarihi avg score: ",sum(mixedlist_scores)/len(mixedlist_scores))


#cografya and kimya
mixedlist_scores = []
i = 0
while i < 10:
    score = smith_waterman_algorithm(list7[i], list9[i])
    mixedlist_scores.append(score)
    i+=1
    #print(score)

print("cografya-kimya avg score: ",sum(mixedlist_scores)/len(mixedlist_scores))


#psikoloji and inkılap tarihi
mixedlist_scores = []
i = 0
while i < 10:
    score = smith_waterman_algorithm(list3[i], list10[i])
    mixedlist_scores.append(score)
    i+=1
    #print(score)

print("psikoloji-inkılap tarihi  avg score: ",sum(mixedlist_scores)/len(mixedlist_scores))

biyoloji-dinler tarihi avg score:  6.2
cografya-kimya avg score:  5.1
psikoloji-inkılap tarihi  avg score:  5.4
```

Looking at these results, I could seen that the highest score average is still under the results of the comparison of the text files belongs to same topic.

For the task given on homework paper, I chose a text which includes 130 words which is, *"There are many ways to align two protein sequences against each other. First, however, we must remember that an alignment generated by software will represent only one of many different possible alignments. The alignment software sorts the generated alignments according to a calculated score, with the output being the one with the highest score. This suggests that the alignment score is essential, and its calculation needs careful consideration. The most straightforward score to assess how closely related two sequences are can be based on the number of identical amino*

acids that align against each other. Using this number, we can count the percentage of identical residues – called the percentage of sequence identity. The higher this percentage, the closer the compared sequences will be in terms of their evolutionary origin."

I pasted this text into first index of both list1 and list2. After that, I run the one more time to see the results.

```
In [34]: # now i will paste the same text inside of two text files.
         # i chose list1[0] and list2[0] to paste this text:

         string =  "There are many ways to align two protein sequences against each other. First, however, we must remember
         string= string.split()
         len(string)

Out[34]: 130

In [36]: score = smith_waterman_algorithm(list1[0], list2[0])
         score

Out[36]: 390.0

In [ ]: """
        Match value has been assigned as 3 before on algorithm.
        Here, I can precisely say that same text has been found on list1[0] and list2[0].
        """
```

Clearly I could see that I found out that text in determined text files.

Matrix Result:

```
In [6]: score = smith_waterman_algorithm(list1[0], list2[0])
        score

                    Gazavât/Müridizm  Direniş  Hareketi     On  sekizinci  \
                 0.0               0.0      0.0       0.0    0.0        0.0
Felsefe          0.0               3.0      2.0       1.0    0.0        0.0
dersinin         0.0               2.0      2.0       1.0    0.0        0.0
ilk              0.0               1.0      1.0       1.0    0.0        0.0
ünitesi        0.0               0.0      0.0       0.0    0.0        0.0
...              ...               ...      ...       ...    ...        ...
ile            104.0             103.0    103.0     103.0  102.0      101.0
bilgi,         104.0             103.0    102.0     102.0  102.0      101.0
var            104.0             103.0    102.0     101.0  101.0      101.0
olanı          104.0             103.0    102.0     101.0  100.0      100.0
tanımaktır.    103.0             103.0    102.0     101.0  100.0       99.0

                yüzyılda  Çarlık  Rusyası'nın  Kafkasya'yı  ...     ve  1925'te  \
                     0.0     0.0          0.0          0.0  ...    0.0      0.0
Felsefe              0.0     0.0          0.0          0.0  ...    0.0      0.0
dersinin             0.0     0.0          0.0          0.0  ...    0.0      0.0
ilk                  0.0     0.0          0.0          0.0  ...    0.0      0.0
ünitesi            0.0     0.0          0.0          0.0  ...    0.0      0.0
...                  ...     ...          ...          ...  ...    ...      ...
ile                101.0   101.0        101.0        101.0  ...  107.0    107.0
bilgi,             100.0   100.0        100.0        100.0  ...  106.0    106.0
var                100.0    99.0         99.0         99.0  ...  105.0    105.0
olanı              100.0    99.0         98.0         98.0  ...  104.0    104.0
tanımaktır.         99.0    99.0         98.0         97.0  ...  103.0    103.0

                Türkiye'ye  gelmiştir.   Yurt   dışı     ve   yurt  içinde  \
                       0.0         0.0    0.0    0.0    0.0    0.0     0.0
Felsefe                0.0         0.0    0.0    0.0    0.0    0.0     0.0
dersinin               0.0         0.0    0.0    0.0    0.0    0.0     0.0
ilk                    0.0         0.0    0.0    0.0    0.0    0.0     0.0
ünitesi              0.0         0.0    0.0    0.0    0.0    0.0     0.0
...                    ...         ...    ...    ...    ...    ...     ...
ile                  107.0       107.0  107.0  107.0  107.0  107.0   106.0
bilgi,               106.0       106.0  106.0  106.0  106.0  106.0   106.0
var                  105.0       105.0  105.0  105.0  105.0  105.0   105.0
olanı                104.0       104.0  104.0  104.0  104.0  104.0   104.0
tanımaktır.          103.0       103.0  103.0  103.0  103.0  103.0   103.0

                birçok
                   0.0
Felsefe            0.0
dersinin           0.0
ilk                0.0
ünitesi          0.0
...                ...
ile              105.0
bilgi,           105.0
var              105.0
olanı            104.0
tanımaktır.      103.0

[604 rows x 619 columns]

Out[6]: 390.0
```

About Match-mismatch and Insertion-Deletion values;

While doing homework, I had a lot of time to search about sequential alignment and scoring methods. I found out that while aligning long sequences, to get a general insight about the inputs and their similarity, having a value bigger than 1 for match variable is better. I kept mismatch, ins/del steady as 1.