ШИНЖЛЭХ УХААН ТЕХНОЛОГИЙН ИХ СУРГУУЛЬ

Мэдээлэл, холбооны технологийн сургууль

Тайлан

Хэл ба мэдээлэл

2021-2022 хичээлийн жилийн намрын улирал

Шалгасан багш: Батзолбоо

Гүйцэтгэсэн: B200900802 Анужин Баатарцогт

Улаанбаатар хот
2022 он

Levenshtein: matrix[x,y] буюу матриц үүсгэнэ. Левенштеин нь хамгийн бага утгийг авдаг учраас матрицаасаа min авна.

```
# min(matrix[x-1,y] + 1,
    matrix[x-1,y-1] + 0,
    matrix[x,y-1] + 1)
```

Хэрэв ижил тэмдэгт бол matrix[x-1,y-1] + 0, өөр өөр тэмдэгтүүд бол 2-г нэмнэ. Матрицаа буцаана. Коммандуудаа олохдоо матрицаа ухрааж хамгийн багийг нь хайж бодно. Ухрааж олох хамгийн бага утгийг

```
# res_min = min(res[j-1][i-1] + (0 if word1[j - 1] == word2[i - 1] else 2),
res[j-1][i] + 1, res[j][i-1] + 1)
```

гэж олно.

Хэрэв res_min нь res[j][i] буюу өөрчлөгдөхгүй бол тэмдэгтүүд ижил учраас комманд байхгүй гэж үзнэ.

res_min нь 1-н өөрчлөлттэй бол нэмэх эсвэл устгах гэсэн 2 коммандууд байна. Доошоо 1-р өөрчлөгдөж байвал устгах, зүүн тийшээ 1-р өөрчлөгдөж байвал нэмэх гэсэн коммандууд байна.

res_min нь 2 өөрчлөлттэй бол солих гэсэн комманд байна.

Weighted: Датагаа бэлдэнэ. Бэлдсэн датагаа Json хэлбэрээр хадгална. Json нь dictionary хэлбэртэй учраас key-р утга руу нь нэвтэрч хамгийн бага утгийг минимум функцээр олно.

```
#
if seq1[x-1] == seq2[y-1]:
  matrix [x,y] = min(matrix[x-1, y] + insertion[word2[y-1]],
  matrix[x-1, y-1] + sub[word1[x-1]][word2[y-1]],
 matrix[x, y-1] + deletion[word1[x-1]])
else:
matrix [x,y] = min(matrix[x-1,y] + insertion[word2[y-1]],
matrix[x-1,y-1] + sub[word1[x-1]][word2[y-1]],
matrix[x,y-1] + deletion[word1[x-1]]
```

```
import numpy as np
import json

# Opening JSON file
```

```
# opening JSON file
f1 = open('confusing_row.json')
f2 = open('confusing_column.json')
f3 = open('confusing_matrix.json')

# returns JSON object as
# a dictionary
deletion = json.load(f1)
insertion = json.load(f2)
sub = json.load(f3)


def levenshtein(seq1, seq2):
    size_x = len(seq1) + 1
    size_y = len(seq2) + 1
    matrix = np.zeros ((size_x, size_y))
    for x in range(size_x):
        matrix [x, 0] = x
    for y in range(size_y):
        matrix [0, y] = y

    for x in range(1, size_x):
        for y in range(1, size_y):
            if seq1[x-1] == seq2[y-1]:
                matrix [x,y] = min(
                    matrix[x-1, y] + 1,
                    matrix[x-1, y-1],
                    matrix[x, y-1] + 1
                )
            else:
                matrix [x,y] = min(
                    matrix[x-1,y] + 1,
                    matrix[x-1,y-1] + 2,
                    matrix[x,y-1] + 1
                )
    return matrix

def weighted(seq1, seq2):
    size_x = len(seq1) + 1
    size_y = len(seq2) + 1
    matrix = np.zeros ((size_x, size_y))
    for x in range(size_x):
        matrix [x, 0] = x
    for y in range(size_y):
        matrix [0, y] = y

    for x in range(1, size_x):
        for y in range(1, size_y):
            if seq1[x-1] == seq2[y-1]:
                matrix [x,y] = min(
                    matrix[x-1, y] + insertion[word2[y-1]],
                    matrix[x-1, y-1] + sub[word1[x-1]][word2[y-1]],
```

```
                    matrix[x, y-1] + deletion[word1[x-1]]
                )
            else:
                matrix [x,y] = min(
                    matrix[x-1,y] + insertion[word2[y-1]],
                    matrix[x-1,y-1] + sub[word1[x-1]][word2[y-1]],
                    matrix[x,y-1] + deletion[word1[x-1]]
                )
    return matrix

word1 = "intention"
word2 = "execution"
res = levenshtein("intention", "execution")
res2 = weighted("intention", "execution")
j = res.shape[0]-1
i = res.shape[1]-1
res_lst = []
while i != 0 or j != 0:
    if i != 0 and j == 0:
        while i > 0:
            res_lst.append('insertion')
            i -= 1

    if j != 0 and i == 0 :
        while j > 0 :
            res_lst.append('remove')
            j -= 1

    res_min = min(res[j-1][i-1] + (0 if word1[j - 1] == word2[i - 1] else 2), res[j-1][i] + 1
    if (word1[j - 1] == word2[i - 1] and res_min == res[j][i]):
        res_lst.append('')
        i -= 1
        j -= 1
    elif res_min == res[j - 1][i - 1] + 2:
        res_lst.append('switch')
        i -= 1
        j -= 1
    elif res_min == res[j-1][i] + 1:
        res_lst.append('deletion')
        j -= 1
    elif res_min == res[j][i-1] + 1:
        res_lst.append('insertion')
        i -= 1
    #print(i, j, res_min)
res_lst.reverse()
print(res)
print(res_lst)
print(res2)
```

```
[[ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9.]
 [ 1.  2.  3.  4.  5.  6.  7.  6.  7.  8.]
 [ 2.  3.  4.  5.  6.  7.  8.  7.  8.  7.]
 [ 3.  4.  5.  6.  7.  8.  7.  8.  9.  8.]
 [ 4.  3.  4.  5.  6.  7.  8.  9. 10.  9.]
 [ 5.  4.  5.  6.  7.  8.  9. 10. 11. 10.]
 [ 6.  5.  6.  7.  8.  9.  8.  9. 10. 11.]
 [ 7.  6.  7.  8.  9. 10.  9.  8.  9. 10.]
 [ 8.  7.  8.  9. 10. 11. 10.  9.  8.  9.]
 [ 9.  8.  9. 10. 11. 12. 11. 10.  9.  8.]]
['remove', 'switch', 'switch', '', 'insertion', 'switch', '', '', '', '']
[[0.         1.         2.         3.         4.         5.
  6.         7.         8.         9.        ]
 [1.         0.03356322 0.11931034 0.20505747 0.2908046  0.37655172
  0.46229885 0.54804598 0.6337931  0.71954023]
 [2.         0.20505747 0.03402299 0.08298851 0.13195402 0.18091954
  0.22988506 0.27885057 0.32781609 0.37678161]
 [3.         0.37655172 0.03655172 0.03563218 0.08505747 0.13195402
  0.18091954 0.22988506 0.27954023 0.32896552]
 [4.         0.54804598 0.03908046 0.03655172 0.03632184 0.08850575
  0.13333333 0.20137931 0.25126437 0.28068966]
 [5.         0.71954023 0.0416092  0.03977011 0.03793103 0.03632184
  0.08528736 0.13333333 0.18229885 0.23126437]
 [6.         0.89103448 0.04413793 0.04321839 0.04183908 0.03793103
  0.03632184 0.08528736 0.13448276 0.18344828]
 [7.         1.06252874 0.04666667 0.07770115 0.04321839 0.05264368
  0.03816092 0.03632184 0.09655172 0.13448276]
 [8.         1.23402299 0.0491954  0.07333333 0.06965517 0.05218391
  0.05586207 0.04390805 0.03632184 0.09655172]
 [9.         1.40551724 0.05172414 0.04988506 0.07471264 0.06965517
  0.0537931  0.05586207 0.04390805 0.05425287]]
```

✓ 0s completed at 3:35 PM