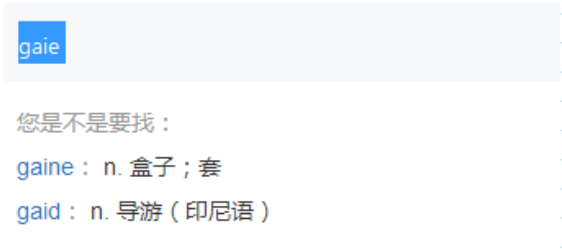
Task3-Q1

2019年10月31日 11:05

- **Q1**

利用动态规划算法求解编辑距离问题，给定两个字符串，求由一个转成另一个所需的最少编辑操作次数。许可的编辑操作包括将一个字符替换成另一个字符，插入一个字符，删除一个字符。

- **相关**



我们在使用词典app时，有没有发现即使输错几个字母，app依然能给我们推荐出想要的单词，非常智能。它是怎么找出我们想要的单词的呢？这里就需要BK树来解决这个问题了。在使用BK树之前我们要先明白一个概念，叫编辑距离，也叫Levenshtein距离。词典app是怎么判断哪些单词和我们输入的单词很相似的呢？我们需要知道两个单词有多像，换句话说就是两个单词相似度是多少。1965年，俄国科学家Vladimir Levenshtein给字符串相似度做出了一个明确的定义叫做Levenshtein距离，我们通常叫它“编辑距离”。字符串A到B的编辑距离是指，只用插入、删除和替换三种操作，最少需要多少步可以把A变成B。例如，从aware到award需要一步（一次替换），从has到have则需要两步（替换s为v和再加上e）。Levenshtein给出了编辑距离的一般求法，就是大家都非常熟悉的经典动态规划问题。

- **算法设计思路**

动态规划的思路就是为了求解当前问题的最优解，使用子问题的最优解然后综合处理，最后得到原问题的最优解。我们要求出word1转化为word2的最少操作数，九十八两个字符串一步步拆解开来，求出每一步的最少操作数，得到最优解。
以oppa变为apple为例，求解过程就是将以下表格填满的过程，表格右下角的值就是所求的最小编辑操作次数。

	null	a	p	p	l	e
null						
o						
P						
p						
a						

每个但此前都有一个空字符串null，代表一种边界情况，比如由null变为null，编辑距离为0，由null变为a或由a变为null，编辑距离为1，由null变为op，编辑距离为2，以此类推。可以对字符串进行三种操作：插入、删除和替换，可以将求最小编辑距离转换为求最

小编辑距离的子问题+1的过程，即求这三种操作中最小的一步，加1后就得到当前所求的最小编辑距离，以此类推，表格的右下角就是我们所求的值。将表格填满后结果如下所示，可知将oppa变为apple的最小编辑距离为3。

	null	a	p	p	l	e
null	0	1	2	3	4	5
o	1	1	2	3	4	5
p	2	2	1	2	3	4
p	3	3	2	1	2	3
a	4	3	3	2	2	3

- **源代码**

```
# -*- coding: utf-8 -*-
# @Time : 2019/12/1 17:25
# @Author : BaoBao
# @Mail : baobaotql@163.com
# @File : Edit_Distance.py
# @Software: PyCharm
'''
利用动态规划算法求解编辑距离问题
'''
def minDistance(word1, word2):
    '''

    :param word1: 传入字符串word1
    :param word2: 传入字符串Word2
    :return: 返回距离矩阵元素
    '''
    m = len(word1)
    n = len(word2)
    if m == 0:
        return n
    if n == 0:
        return m
    dp = [[0] * (n+1) for _ in range(m+1)] #初始化表格[m+1, n+1]
    # 计算边界
    for i in range(1, m+1):
        dp[i][0] = i
    for j in range(1, n+1):
        dp[0][j] = j
    for i in range(1, m+1): #计算dp
        for j in range(1, n+1):
            if word1[i-1] == word2[j-1]:
                dp[i][j] = dp[i-1][j-1]
            else:
                dp[i][j] = min(dp[i-1][j-1]+1, dp[i][j-1]+1, dp[i-1][j]+1)
    return dp[m][n]


if __name__ == "__main__":
    dis = minDistance('www.ccnu.edu.cn', 'www.neu.edu.cn')
    print("The longest Edit Distance is :", dis)
```

- **运行截图**

```
C:\Users\79453\Anaconda3\python.exe "D:/华师工程中心/研一/课程  算法设计/coding tests/LCS.py"
The length of longest common subsequence is: 13
The Longest Common Subsequence is: www.cn.edu.cn

Process finished with exit code 0
```