

# 自然语言处理 NLP 作业①

## MED 实现

姓名：罗福杰

学号：3120305208

班级：S0078

# 目录

1 作业要求：MED 实现.....	3
2 问题的数学描述 .....	3
2.1 最小编辑距离的概述 .....	3
2.2 问题描述 .....	3
2.3 数学语言描述 .....	4
3 编程实现 .....	5
3.1 编程思路 .....	5
3.2 核心代码（完整代码见附录） .....	6
3.2.1 实现最小距离的代码 .....	6
3.2.2 实现回溯计算的代码 .....	6
3.2.3 实现界面开发的核心代码 .....	8
3.3 界面设计与实验结果 .....	9
参考资料.....	15
附录.....	16
免责与版权.....	23

# 1 作业要求：MED 实现

作业具体要求：

- 实现英文（或中文）字符串的最小编辑距离，最小编辑路径的计算并显示结果。
- 可以输入不同字符串；可以通过修改操作代价来改变最小编辑距离。
- 需要提交报告、源代码和可执行程序。

## 2 问题的数学描述

### 2.1 最小编辑距离的概述

最小编辑距离（Edit Distance）是指两个字串之间，由一个转成另一个所需的最少编辑操作次数。允许对字符串中的字符进行的操作只有替换、插入、删除三种操作。

编辑距离是自然语言处理中的重要文本比较算法之一。也是从多个相似的字符串组中提取字符串的有利的武器。编辑距离算法，也称为 LD 算法。LD 算法就是自然语言处理(NLP)里的“编辑距离”算法。俄国科学家 Levenshtein 提出的，故又叫 Levenshtein Distance （LD 算法）。

最小编辑距离的用途：

- 计算衡量机器翻译和语音识别的好坏：将机器得到的字符串与专家写的字符串比较最小编辑距离，以一个单词为一个单位。
- 命名实体识别和链接：比如通过计算最小编辑距离，可以判定 IBM.Inc 和 IBN 非常相似，只有一个单词不同，所以认为这是指向同一个命名实体。

### 2.2 问题描述

给定两个字符串 a 和 b，如果要将字符串 a 转换为 b，转换过程中只有三种基本操作：

- 1) 插入 (insert) 字符;
- 2) 删除 (delete) 字符;
- 3) 替换 (replace) 字符。

例如从：单词 intention 通过删除 i 可以得到 ntention，通过插入 e 可以得到 eintention，通过将 i 换成 e 可以得到 entention。以下是从 intention 到叶子节点的任意一个单词经过的操作数就是一条路径。

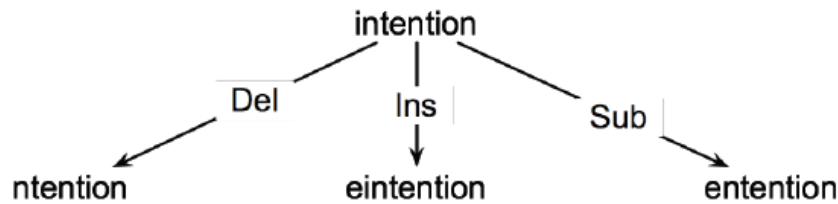


图 2.1 字符串编辑示例

可以发现枚举出所有可转变成的单词的花费是十分巨大的，我们不可能用枚举遍历的方式来寻找一条最短路径，一种解决方法是：使用剪枝，每层中有很多路径被剪枝了，只在每一层中保留最短的那条路径

通常每一个操作的 cost 值为 1，则最小编辑距离是字符串编辑距离的优化问题：如何组合这些操作从而使转换的代价 cost 最小。则可以转化为动态规划的最优值求解问题，下面给出其具体的数学语言描述过程。

## 2.3 数学语言描述

通过以上的分析，将两个字符串 a,b 的编辑距离 Levenshtein Distance 表示为  $lev_{a,b}(|a|, |b|)$  可用以下的数学语言描述：

$$lev_{a,b}(i, j) = \begin{cases} \max(i, j) & \text{if } \min(i, j) = 0 \\ \min \begin{cases} lev_{a,b}(i-1, j) + 1 \\ lev_{a,b}(i, j-1) + 1 \\ lev_{a,b}(i-1, j-1) + 1_{(a_i \neq b_i)} \end{cases} & \text{otherwise} \end{cases}$$

其中：

- $i, j$  是 a,b 的长度；定义的  $lev_{a,b}(i, j)$  是指 a 中前 i 个字符和 b 中的前 j 个字符之间的距离。这里的字符串的第一个字符 index 从 1 开始，因此在最后的编辑距离是  $i=|a|, j=|b|$  时的距离：  $lev_{a,b}(i, j)$ 。

- 当 $\min(i, j) = 0$ 的时候, 对应着字符串 **a** 中前  $i$  个字符和 **b** 中的前  $j$  个字符, 此时的  $i, j$  有一个值是 0, 表示字符串 **a** 和 **b** 中有一个是空的字符串, 那么从 **a** 转换到 **b** 只需要进行  $\max(i, j)$  次单字符串的编辑操作, 所以他们之间的编辑距离为  $\max(i, j)$ , 即  $i, j$  中的最大者。
- 当 $\min(i, j) \neq 0$ 的时候,  $lev_{a,b}(|a|, |b|)$  有三种情况的最小值:  
这三种情况是:
  - 1)  $lev_{a,b}(i-1, j) + 1$  表示删除  $a_i$ ;
  - 2)  $lev_{a,b}(i, j-1) + 1$  表示插入  $b_i$ ;
  - 3)  $lev_{a,b}(i-1, j-1) + 1_{(a_i \neq b_i)}$  表示替换。
- $1_{(a_i \neq b_i)}$  是一个指示函数, 表示当  $a_i = b_i$  的时候取值为 0, 当  $a_i \neq b_i$  的时候取值为 1。

## 3 编程实现

### 3.1 编程思路

在完成本次作业过程中, 我主要是通过以下三个步骤来实现的:

- 1) 完成实现最小编辑距离的函数, `minDistance()`, 参数是输入的两个字符串, 输出是最小编辑距离和实现这一功能的二维矩阵;
- 2) 完成回溯的功能函数, `backtrackingPath()`, 参数是两个字符串, 输出是实现字符串编辑的具体步骤;
- 3) 界面开发, 为了拥有更友好的用户体验, 我开发了基于 windows 的“最小编辑距离计算器”软件, 通过该软件可以实现最小编辑距离的计算和实现的具体步骤。

## 3.2 核心代码（完整代码见附录）

### 3.2.1 实现最小距离的代码

```
def minDistance(w1,w2):
    w1=w1.strip()
    w2=w2.strip()
    m,n = len(w1),len(w2)
    if (w1==" " or m==0):
        messagebox.showinfo('Warning!!!','请在 Word1 中输入字符! ')
        return m
    if m>255:
        messagebox.showinfo('Warning!!!','字符长度不得超过 255! ')
        return m
    if (w2==" " or n==0):
        messagebox.showinfo('Warning!!!','请在 Word2 中输入字符! ')
        return n
    if n>255:
        messagebox.showinfo('Warning!!!','字符长度不得超过 255! ')
        return n

    # 生成全零矩阵, 形状是 (m+1, n+1)
    #step = [[0]*(n+1) for _ in range(m+1)]
    step = np.zeros([m+1,n+1])

    for i in range(1,m+1):
        step[i][0] = i
    for j in range(1,n+1):
        step[0][j] = j
    for i in range(1,m+1):
        for j in range(1,n+1):
            if w1[i-1] == w2[j-1]:
                diff = 0
            else:
                diff = 1
            step[i][j] = min(step[i-1][j-1],min(step[i-1][j],step[i][j-1]))+diff
    return step,int(step[m][n])
```

### 3.2.2 实现回溯计算的代码

```
def backtrackingPath(word1,word2):
    dp,mindista = minDistance(word1,word2)
    m = len(dp)-1
```

```

n = len(dp[0])-1
operation = []
spokenstr = []
writtenstr = []

operation_process = []
# 定义用来存放是否为最优值的矩阵
back_way = np.zeros([m+1,n+1])
back_way[m][n] = 1

while n>=0 or m>=0:
    if n and dp[m][n-1]+1 == dp[m][n]:
        processer="Insert : \""+(word2[n-1])+'\".'
        operation_process.append(processer)
        spokenstr.append("Insert")
        writtenstr.append(word2[n-1])
        operation.append("NULLREF:"+word2[n-1])
        n -= 1
        back_way[m][n] = 1
        continue
    if m and dp[m-1][n]+1 == dp[m][n]:
        processer="Delete : \""+(word1[m-1])+'\".'
        operation_process.append(processer)
        spokenstr.append(word1[m-1])
        writtenstr.append("Delete")
        operation.append(word1[m-1]+":NULLHYP")
        m -= 1
        back_way[m][n] = 1
        continue
    if dp[m-1][n-1]+1 == dp[m][n]:
        processer="Replace : \""+(word1[m-1])+'\" To \"'+(word2[n-1])+'\".'
        operation_process.append(processer)
        spokenstr.append(word1[m - 1])
        writtenstr.append(word2[n-1])
        operation.append(word1[m - 1] + ":" + word2[n-1])
        n -= 1
        m -= 1
        back_way[m][n] = 1
        continue
    if dp[m-1][n-1] == dp[m][n]:
        spokenstr.append(' ')
        writtenstr.append(' ')
        operation.append(word1[m-1])
        n -= 1

```

```
m -= 1
back_way[m][n] = 1
spokenstr = spokenstr[::-1]
writtenstr = writtenstr[::-1]
operation = operation[::-1]
# print(spokenstr,writtenstr)
# print(operation)
return spokenstr,writtenstr,operation,operation_process,back_way
```

### 3.2.3 实现界面开发的核心代码

本软件是用 Python 中的 tkinter 框架进行开发的，核心代码如下：

```
import tkinter as tk
from tkinter import ttk
import numpy as np
from tkinter import messagebox
window = tk.Tk()
window.title("NLP 作业① 最小编辑距离计算器 by 罗福杰")
#设置窗口的长和宽不可调节
window.resizable(0,0)
# 设置初始界面的长和高
window.geometry('700x600')

def compute():
    word1=str(tx_word1.get(1.0, "end"))
    word2=str(tx_word2.get(1.0, "end"))
    step,mindis=minDistance(word1,word2)
    result_value.set(mindis)
    btn_show.place(x=200,y=400)

btn_compute = tk.Button(window,text='Start Compute',width=15,height=5,command=compute)
btn_compute.place(x=30,y=400)
btn_show = tk.Button(window,text='Show Operation',width=15,height=5,command=show_operation)
#btn_show.place(x=200,y=400)
btn_show.place_forget()
window.mainloop()
```



### 3.3 界面设计与实验结果

本实验首先由 Python 编写各个功能的程序，然后用 pyinstaller 进行打包，将文件打包成可执行文件（.exe 文件）

下面是实验的结果：

- 打包后的文件示意图

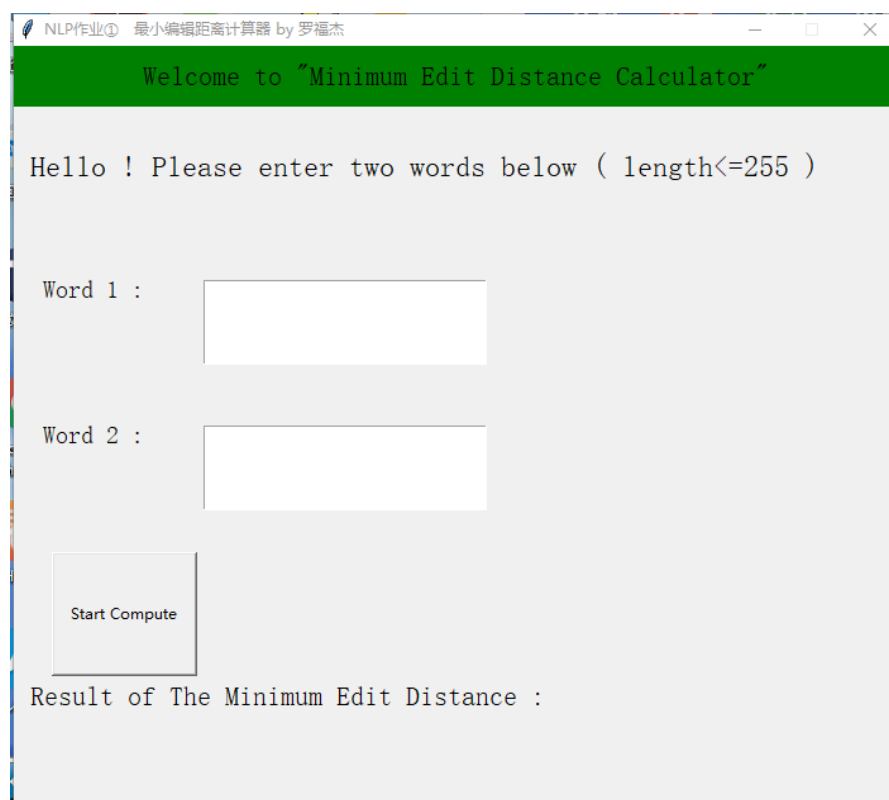
名称	修改日期	类型	大小
__pycache__	2020-09-19 23:02	文件夹	
build	2020-09-19 23:02	文件夹	
dist	2020-09-19 23:03	文件夹	
bitbug.ico	2020-09-19 22:46	ICO 文件	17 KB
NLP01_MED_FujieLuo.py	2020-09-19 15:00	PY 文件	9 KB
NLP01_MED_FujieLuo.spec	2020-09-19 23:02	SPEC 文件	1 KB

进入 dist 文件夹后：

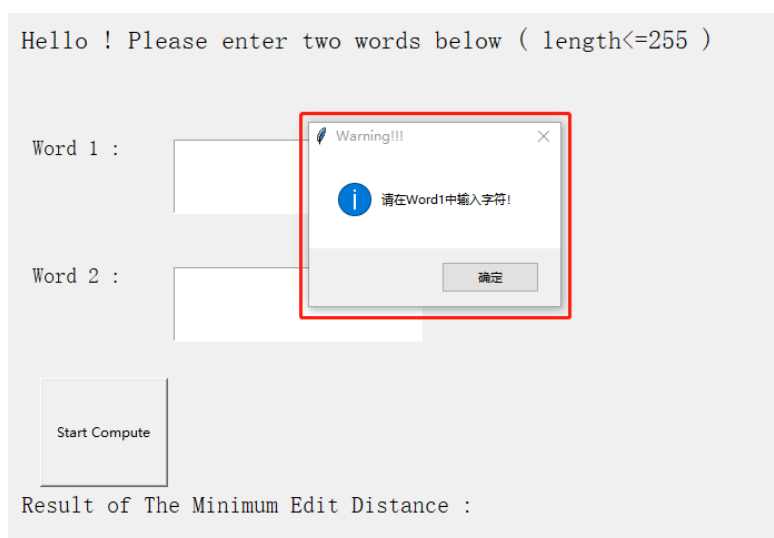
dist	名称	修改日期	类型	大小
	NLP01_MED_FujieLuo.exe	2020-09-19 23:03	应用程序	30,184 KB

- 双击软件即可运行：

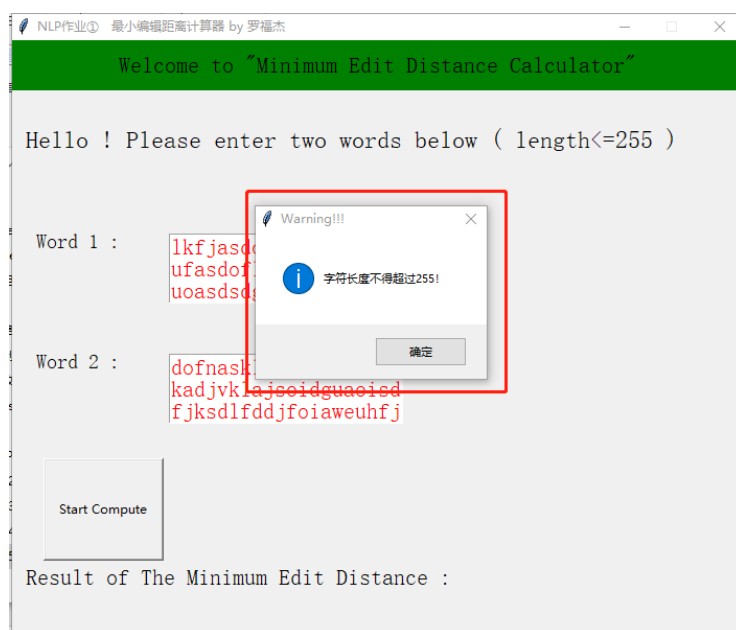
该软件有两个输入框，在里面输入有效的字符才能进行计算，一个字符串的长度不超过 255。输入完成后，点击“Start Compute”开始计算。



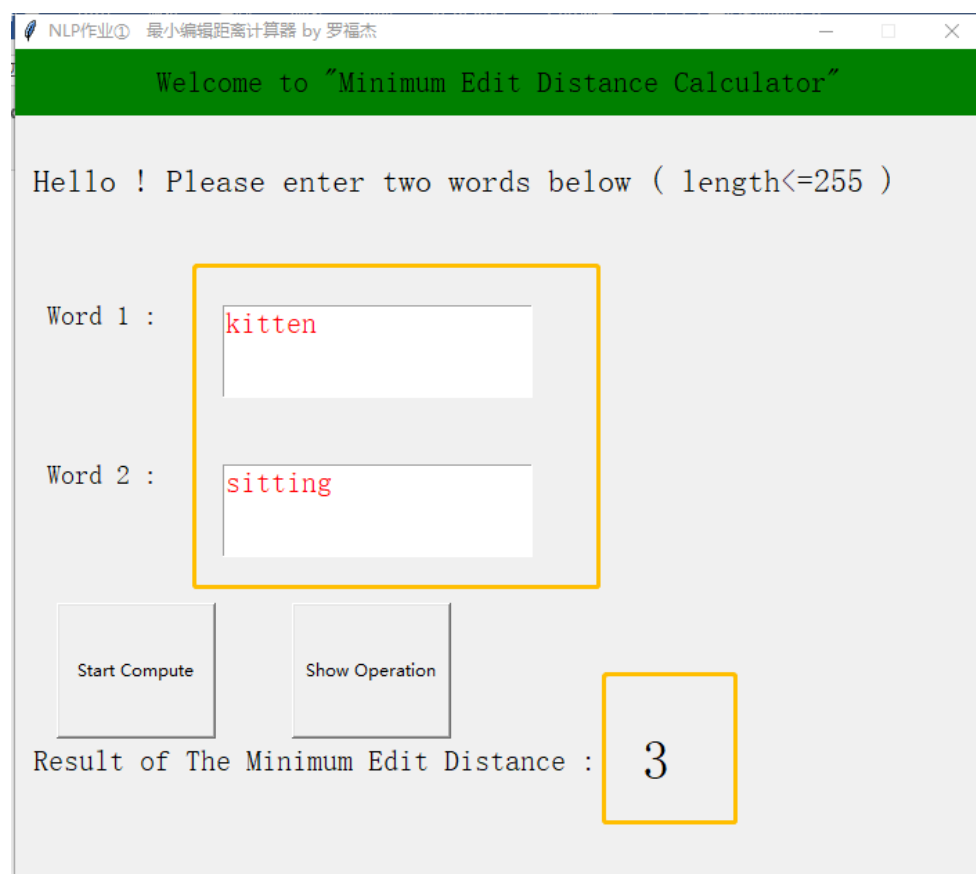
- 当没输入任何信息时，提示输入有效的字符



- 当输入的字符数大于 255 时，提示输入过长

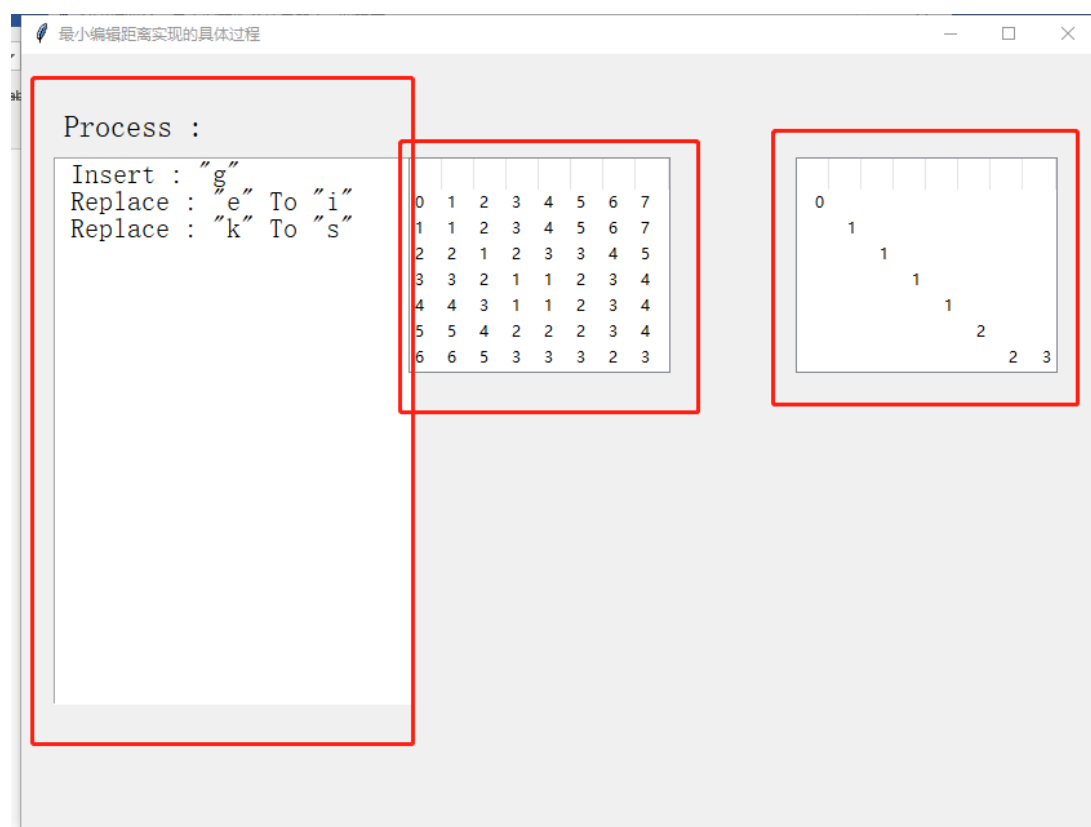


- 当输入有效的字符后，开始计算，并将最小编辑距离打印输出，示例如下：

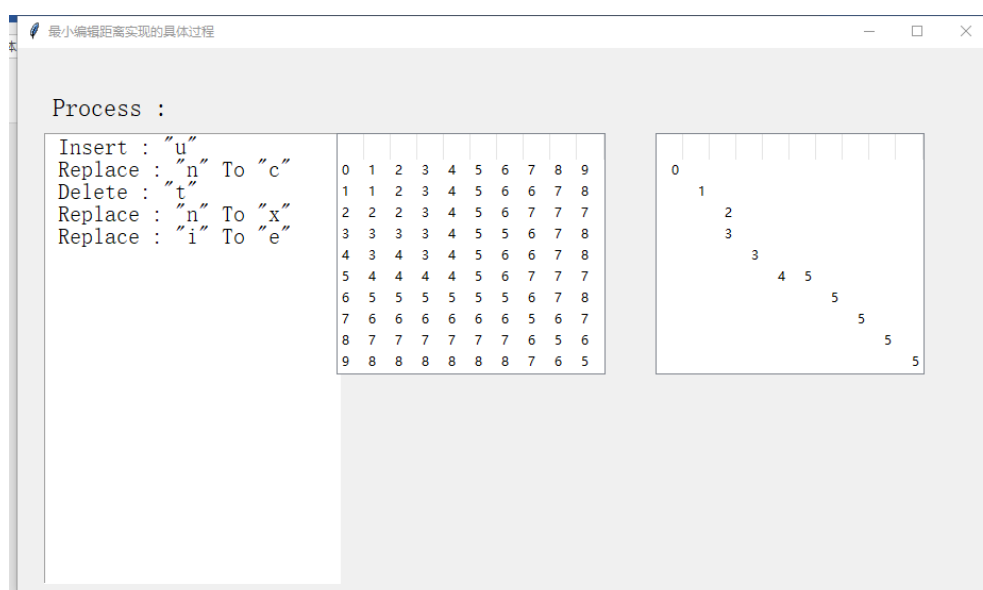
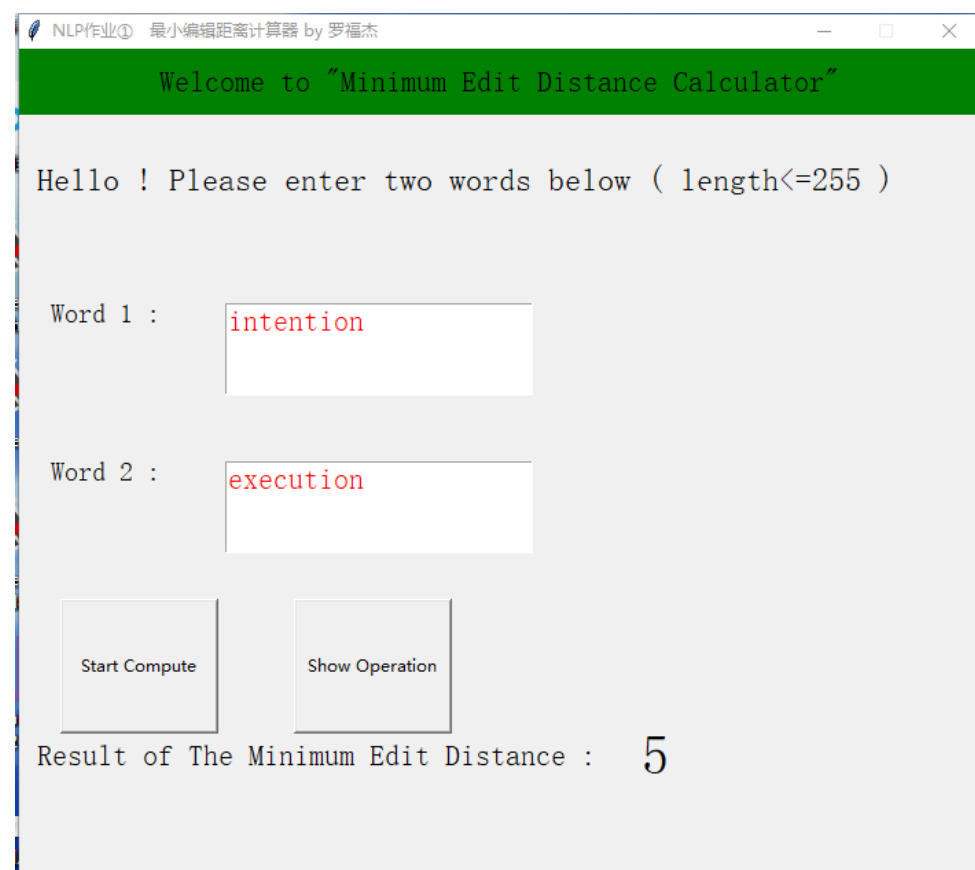


- 这个时候可以查看如何进行编辑操作的，当点击“Show Operation”即可弹出另一个窗口，显示了如何进行字符编辑的、字符编辑表格和字符编辑的回溯路线。如下图中红色框内部分所示，是由字符串“kitten”转换成“sitting”

的操作结果。



- 示例二的结果：从字符串“intention”转换为“execution”的结果。



- 中文字符示例一个中文汉字可以看作是一个字符，结果如下：

NLP作业① 最小编辑距离计算器 by 罗福杰

Welcome to "Minimum Edit Distance Calculator"

Hello ! Please enter two words below ( length<=255 )

Word 1 :

Word 2 :

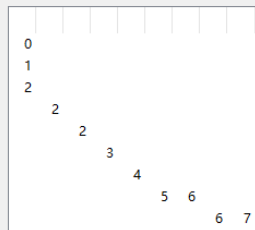
Result of The Minimum Edit Distance : 7

最小编辑距离实现的具体过程

Process :

Insert : "进"  
Insert : "天"  
Replace : "向" To "每"  
Replace : "天" To "国"  
Replace : "天" To "强"  
Delete : "好"  
Delete : "好"

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



## 参考资料

- [1] [https://blog.csdn.net/weixin\\_40446557/article/details/103456906](https://blog.csdn.net/weixin_40446557/article/details/103456906)
- [2] <https://blog.csdn.net/chl0000/article/details/7657887>
- [3] [https://blog.csdn.net/weixin\\_45926547/article/details/108397563](https://blog.csdn.net/weixin_45926547/article/details/108397563)
- [4] [https://blog.csdn.net/qq\\_42382909/article/details/82883237](https://blog.csdn.net/qq_42382909/article/details/82883237)
- [5] [https://blog.csdn.net/vivian\\_ll/article/details/93168926](https://blog.csdn.net/vivian_ll/article/details/93168926)

# 附录

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Sep 17 22:29:38 2020

@author: fujie
"""

import tkinter as tk
from tkinter import ttk
import numpy as np
from tkinter import messagebox

window = tk.Tk()
window.title("NLP 作业① 最小编辑距离计算器 by 罗福杰")
#设置窗口的长和宽不可调节
window.resizable(0,0)
# 设置初始界面的长和高
window.geometry('700x600')

#lb_title = tk.Label(window,text='Welcome to "最 小 编 辑 距 离 计 算 器" haha',bg='green',font
=('Arial',16),width=80,height=2)
lb_title = tk.Label(window,text='Welcome to "Minimum Edit Distance Calculator"',bg='green',font =('Time New
Roman',16),width=80,height=2)
#lb_title = tk.Label(window,text='欢迎使用最小编辑距离计算器',font =('Arial',18),width=40,height=20)
lb_title.pack(side='top')
#lb_title.pack(side='bottom')

lb_mode = tk.Label(window,text = 'Hello ! Please enter two words below ( length<=255 )',font =('Time New
Roman',17))
#lb_mode.pack(side='left')
#设置标签的位置
lb_mode.place(x=10,y=80,anchor='nw')

#cmb = ttk.Combobox(window)
#cmb.place(x = 210,y = 85,anchor='nw')
#cmb['value']=('English','中文')
#设置默认值
#cmb.current(0)
```



```
lb_char1=tk.Label(window,text='Word 1 :',font=('Time New Roman',14))
lb_char1.place(x=20,y=180)
lb_char2=tk.Label(window,text='Word 2 :',font=('Time New Roman',14))
lb_char2.place(x=20,y=295)
lb_result = tk.Label(window,text='Result of The Minimum Edit Distance : ',font=('Time New Roman',16))
lb_result.place(x=10,y=500)

result_value = tk.StringVar()
lb_result_value = tk.Label(window,textvariable=result_value,font=('Time New Roman',30))
lb_result_value.place(x=450,y=490)

result_lb = tk.StringVar()
lb_result_lb=tk.Label(window,textvariable=result_lb,font=('Time New Roman',20))
lb_result_lb.place(x=500,y=100)

result_process_lb=tk.StringVar()
lb_result_process=tk.Label(window,textvariable=result_process_lb,font=('Time New Roman',20))
lb_result_process.place(x=350,y=150)

#entry_word1=tk.Entry(window,bd=3,xscrollcommand=True)
#entry_word1.place(x=150,y=185)
#entry_word2=tk.Entry(window,bd=3)
#entry_word2.place(x=150,y=245)

f=tk.Frame(window)
s1 = tk.Scrollbar(f,orient=tk.VERTICAL)
tx_word1=tk.Text(window,autoseparators=2,font=('Time New Roman',16),height=3,width=20 ,fg='red')
tx_word1.place(x=150,y=185)
tx_word2=tk.Text(window,autoseparators=2,font=('Time New Roman',16),height=3,width=20 ,fg='red')
tx_word2.place(x=150,y=300)

def minDistance(w1,w2):
    w1=w1.strip()
    w2=w2.strip()
    m,n = len(w1),len(w2)
    if (w1=="" or m==0):
        messagebox.showinfo('Warning!!!','请在 Word1 中输入字符! ')
        return m
    if m>255:
        messagebox.showinfo('Warning!!!','字符长度不得超过 255! ')
        return m
    if (w2=="" or n==0):
        messagebox.showinfo('Warning!!!','请在 Word2 中输入字符! ')
        return n
```

```
if n>255:
    messagebox.showinfo('Warning!!!','字符长度不得超过 255! ')
    return n

# 生成全零矩阵，形状是 (m+1, n+1)
#step = [[0]*(n+1) for _ in range(m+1)]
step = np.zeros([m+1,n+1])

for i in range(1,m+1):
    step[i][0] = i

for j in range(1,n+1):
    step[0][j] = j

for i in range(1,m+1):

    for j in range(1,n+1):
        if w1[i-1] == w2[j-1]:
            diff = 0
        else:
            diff = 1
        step[i][j] = min(step[i-1][j-1],min(step[i-1][j],step[i][j-1]))+diff

#return step[m][n]
return step,int(step[m][n])

def backtrackingPath(word1,word2):
    dp,mindista = minDistance(word1,word2)
    m = len(dp)-1
    n = len(dp[0])-1
    operation = []
    spokenstr = []
    writtenstr = []

    operation_process = []

    back_way = np.zeros([m+1,n+1])
    back_way[m][n] = 1

    while n>=0 or m>=0:

        if n and dp[m][n-1]+1 == dp[m][n]:
            processer="Insert : \"\"+(word2[n-1])+\\""
```

```

        operation_process.append(processer)
        spokenstr.append("Insert")
        writtenstr.append(word2[n-1])
        operation.append("NULLREF:"+word2[n-1])
        n -= 1
        back_way[m][n] = 1
        continue

    if m and dp[m-1][n]+1 == dp[m][n]:
        processer="Delete : \"'+(word1[m-1])+'\".'"
        operation_process.append(processer)
        spokenstr.append(word1[m-1])
        writtenstr.append("Delete")
        operation.append(word1[m-1]+":NULLHYP")
        m -= 1
        back_way[m][n] = 1
        continue

    if dp[m-1][n-1]+1 == dp[m][n]:
        processer="Replace : \"'+(word1[m-1])+'\" To \"'+(word2[n-1])+'\".'"
        operation_process.append(processer)
        spokenstr.append(word1[m - 1])
        writtenstr.append(word2[n-1])
        operation.append(word1[m - 1] + ":" + word2[n-1])
        n -= 1
        m -= 1
        back_way[m][n] = 1
        continue

    if dp[m-1][n-1] == dp[m][n]:
        spokenstr.append(' ')
        writtenstr.append(' ')
        operation.append(word1[m-1])

    n -= 1
    m -= 1
    back_way[m][n] = 1

    spokenstr = spokenstr[::-1]
    writtenstr = writtenstr[::-1]
    operation = operation[::-1]
    # print(spokenstr, writtenstr)
    # print(operation)
    return spokenstr, writtenstr, operation, operation_process, back_way

def compute():

```

```
#word1=entry_word1.get()
#word2=entry_word2.get()
word1=str(tx_word1.get(1.0, "end"))
word2=str(tx_word2.get(1.0, "end"))

step,mindis=minDistance(word1,word2)
result_value.set(mindis)
btn_show.place(x=200,y=400)

def show_operation():
    compute()
    word1=tx_word1.get(1.0, "end")
    word2=tx_word2.get(1.0, "end")

    window_show=tk.Toplevel(window)
    window_show.geometry('800x600')
    window_show.title('最小编辑距离实现的具体过程')

    lb_show_Process=tk.Label(window_show,text='Process : ',font=('Time New Roman',18))
    lb_show_Process.place(x=30,y=40)

    tx_show=tk.Text(window_show,autoseparators=2,font=('Time New Roman',16),height=20,width=25)
    tx_show.place(x=25,y=80)

    spokenstr,writtenstr,operation,operation_process,back_way=backtrackingPath(word1,word2)
    #result_process_lb.set(operation_process)

    tx_show.insert(1.0," ")
    process=str(operation_process)
    process=process.replace('.',',', '\n')
    process=process.replace('.',\]', '')
    process=process.replace("\", "")
    process=process.replace('[', '')
    tx_show.insert(1.1,process)

    stepff,_=minDistance(word1, word2)
    lines,column_num=stepff.shape[0],stepff.shape[1]

    columnssk=[str(int(stepff[0][i])) for i in range(0,column_num)]
    tree = ttk.Treeview(window_show, show = "headings", columns = columnssk,height=lines, selectmode =
tk.BROWSE)
    tree["columns"]=columnssk
    #设置每一列的属性
```

```
for i in range(0,column_num):
    tree.column(columnskk[i], width=25)
# 开始填充数据
for i in range(0,lines):
    step_line_val=[str(int(stepff[i][j])) for j in range(0,column_num)]
    tree.insert('', 'end', values=step_line_val)

tree.place(x=300,y=80)

back_way_only_1 = np.zeros([lines,column_num])
for i in range(0,lines):
    for j in range(0,column_num):
        back_way_only_1[i][j]=stepff[i][j]*back_way[i][j]

back_way_only_1_str = [['0']*column_num for _ in range(lines)]
for i in range(0,lines):
    for j in range(0,column_num):
        if back_way[i][j]==1:
            back_way_only_1_str[i][j]=str(int(stepff[i][j]))
        else:
            back_way_only_1_str[i][j]=' '
back_way_only_1_str[0][0]='0'

tree2 = ttk.Treeview(window_show, show = "headings", columns = columnskk,height=lines, selectmode =
tk.BROWSE)
tree2["columns"]=columnskk
#设置每一列的属性
for i in range(0,column_num):
    tree2.column(columnskk[i], width=25,anchor='e')

for i in range(0,lines):
    tree2.insert('', 'end', values=back_way_only_1_str[i])

tree2.place(x=600,y=80)

btn_compute = tk.Button(window,text = 'Start Compute',width=15,height=5,command=compute)
btn_compute.place(x=30,y=400)
btn_show = tk.Button(window,text = 'Show Operation',width=15,height=5,command=show_operation)
#btn_show.place(x=200,y=400)
btn_show.place_forget()

window.mainloop()
```



## 免责与版权

1. 该软件由罗福杰同学（邮箱：1626027173@qq.com）开发与维护，版权归其所有；
2. 该软件仅供学习交流使用，禁止商业用途，如有因学习以外的用途产生的问题，软件开发人不承担法律责任。