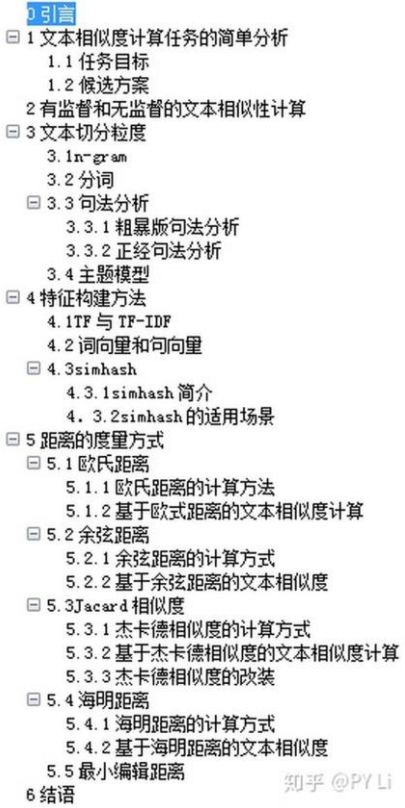
传统方法进行文本分析

文本相似度计算——编辑距离

东北大学自然语言处理实验室

文本相似度计算方法



https://github.com/NLP-lecture/part-1

- 学习最小编辑距离有哪些好处？
  - 能用到数据清洗步骤中（去重）
  - 经典的算法题（大厂面试）

去重：在机器翻译任务中的数据集中常常出现一个现象，即源语言和目标语言有多个非常相似的句子。

I left your paper on your desk.
I left your paper on your computer.
我把你的论文放到了你的桌子上。

最小编辑距离是啥

给定两个字符串 str1 和 str2，计算出将 str1 转换成str2所使用的最少操作数 。
你可以对一个字符串进行如下三种操作:
---插入一个字符
---删除一个字符
---替换一个字符
可以看下面的例子:

str1:	I give an apple to you	编辑距离是1	编辑距离也可以是3，先增加一个没用的字符，再删去，再替换
str2:	I gave an apple to you		
str1:	I give an apple to you	编辑距离是1	
str2:	I given an apple to you		
str1:	I give an apple to you	编辑距离是1	但最小编辑距离都是1
str2:	I give a apple to you		

最小编辑距离是啥

https://github.com/NLP-lecture/part-1

要是操作数再多一点呢?

str1: apple
str2: oppaple

编辑距离是3

要是字符串再长一点呢?

str1: applesadeawfasdfadwfa
str2: oppapleasdwasxcwxfsdfsd

编辑距离是???

编辑距离是15

DP求解思路

DP问题的求解一般遵循3个步骤:
定义数组元素的含义
找出初始值
找出数组元素之间的关系

https://github.com/NLP-lecture/part-1

str2

		C	C	A	C	A
	0	1	2	3	4	5
A	1	?				
A	2					
T	3					
C	4					
T	5					

str1

DP求解思路

https://github.com/NLP-lecture/part-1

DP问题的求解一般遵循3个步骤：

定义数组元素的含义：dp[i][j]表示 str1前i个字符转换到str2前j个字符时所需最小操作数

找出初始值：在两个字符串前，都加上同一个符号

找出数组元素之间的关系：下移表示删除、右移表示插入、斜移动表示替换

str2

str1		0	1	2	3	4	5	6
	0		空格	C	C	A	C	A
	1	空格	0	1	2	3	4	5
	2	A	1	?				
	3	A	2					
	4	T	3					
	5	C	4					
	6	T	5					

比如dp[1][1]表示：从“空格”变成“空格”需要0个操作。此时**两个字符串已经变成一样了**。

比如dp[1][6]表示：从“空格”变成“空格 C C A C A”需要5个操作。此时**两个字符串已经变成一样了**。

DP求解思路

https://github.com/NLP-lecture/part-1

DP问题的求解一般遵循3个步骤：

定义数组元素的含义：dp[i][j]表示 str1前i个字符转换到str2前j个字符时所需最小操作数

找出初始值：在两个字符串前，都加上同一个符号

找出数组元素之间的关系：下移表示删除、右移表示插入、斜移动表示替换

		str2					
		0	1	2	3	4	5	6
	0		空格	C	C	A	C	A
	1	空格	0	1	2	3	4	5
	2	A	1	?				
str1	3	A	2					
	4	T	3					
	5	C	4					
	6	T	5					

加上初始值不影响最小编辑距离结果

为什么要加初始值？方便算法计算

DP求解思路

https://github.com/NLP-lecture/part-1

DP问题的求解一般遵循3个步骤：

定义数组元素的含义：dp[i][j]表示 str1前i个字符转换到str2前j个字符时所需最小操作数

找出初始值：在两个字符串前，都加上同一个符号

找出数组元素之间的关系：下移表示删除、右移表示插入、斜移动表示替换

str2

str1		0	1	2	3	4	5	6
	0		空格	C	C	A	C	A
	1	空格	0	1	2	3	4	5
	2	A	1	?				
	3	A	2					
	4	T	3					
	5	C	4					
	6	T	5					

前面说了一个格表示此状态两个字符串经过格子里面的数值已经变成两个一样的字符串了。基于这个思想我们就能进行状态之间改变。

DP求解思路

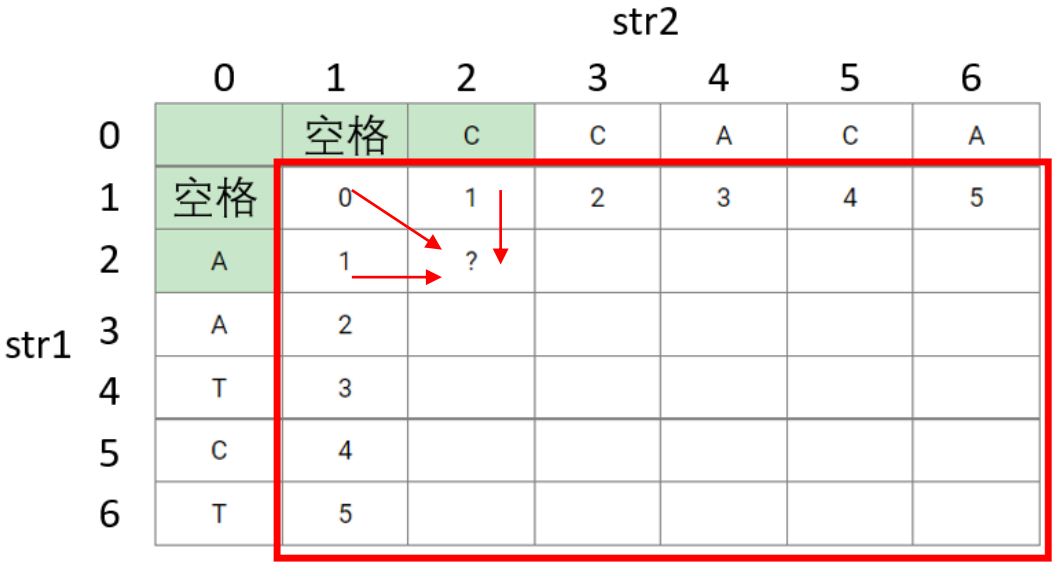
https://github.com/NLP-lecture/part-1

DP问题的求解一般遵循3个步骤：

定义数组元素的含义：dp[i][j]表示 str1前i个字符转换到str2前j个字符时所需最小操作数

找出初始值：在两个字符串前，都加上同一个符号

找出数组元素之间的关系：下移表示删除、右移表示插入、斜移动表示替换



下移表示删除：dp[1][2]变到dp[2][2]，因为dp[1][2]时str1已经变成“空格 C”了，此时dp[2][2]又让str1往后看了一个新的字符“A”变成“空格 C A”，而str2还是“空格 C”。此时str1变成str2的最少操作就是删除“A”

如果按删除操作来的话，dp[2][2]=dp[1][2]+1=2

DP求解思路

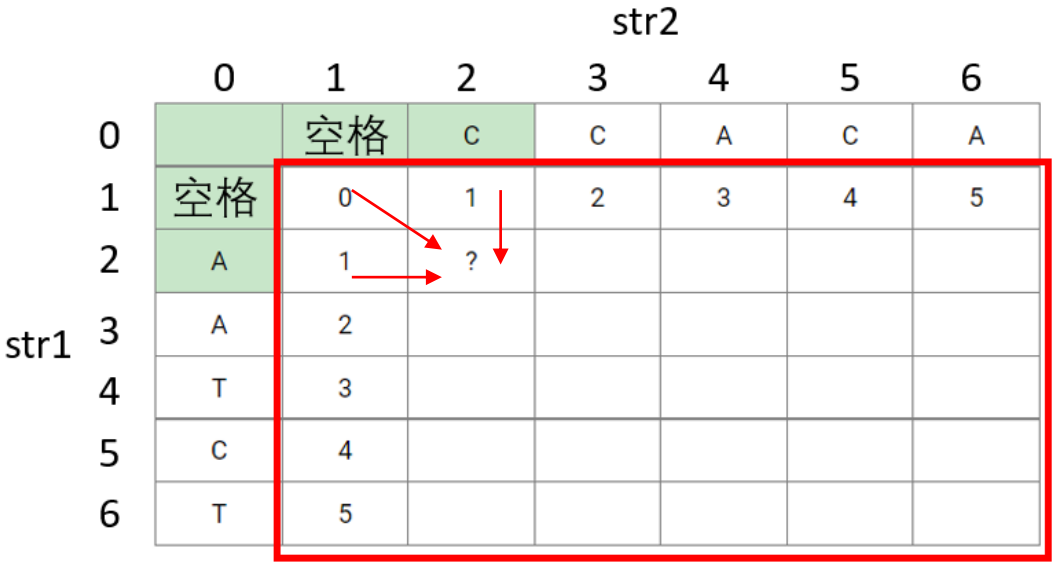
https://github.com/NLP-lecture/part-1

DP问题的求解一般遵循3个步骤：

定义数组元素的含义：dp[i][j]表示 str1前i个字符转换到str2前j个字符时所需最小操作数

找出初始值：在两个字符串前，都加上同一个符号

找出数组元素之间的关系：下移表示删除、右移表示插入、斜移动表示替换



		0	1	2	3	4	5	6
			空格	C	C	A	C	A
0								
1	空格	0	1	2	3	4	5	
2	A	1	?					
3	A	2						
4	T	3						
5	C	4						
6	T	5						

右移表示插入：dp[2][1]变到dp[2][2]，因为dp[2][1]时str1已经变成“空格”了，此时dp[2][2]又让str2往后看了一个新的字符“C”变成“空格 C”，而str1还是“空格”。此时str1变成str2的最少操作就是插入“C”

如果按插入操作来的话，dp[2][2]=dp[2][1]+1=2

DP求解思路

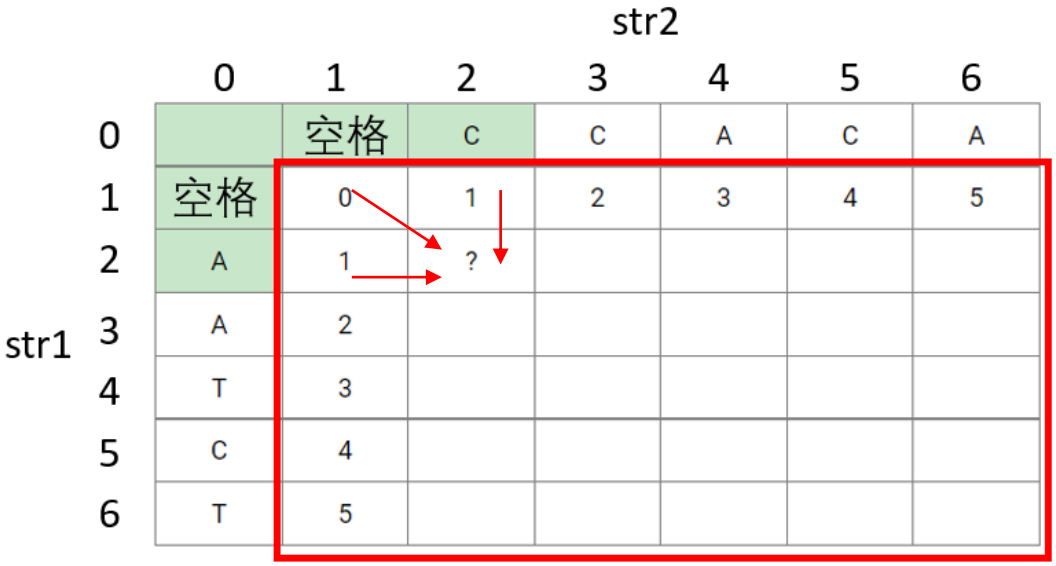
https://github.com/NLP-lecture/part-1

DP问题的求解一般遵循3个步骤：

定义数组元素的含义：dp[i][j]表示 str1前i个字符转换到str2前j个字符时所需最小操作数

找出初始值：在两个字符串前，都加上同一个符号

找出数组元素之间的关系：下移表示删除、右移表示插入、斜移动表示替换



斜移动表示替换：dp[1][1]变到dp[2][2]，因为dp[1][1]时str1和str2都是“空格”，此时dp[2][2]又让str1往后看了一个字符“A”变成“空格 A”；str2往后看了一个新的字符“C”变成“空格 C”。此时str1变成str2的最少操作就是把“A”替换成“C”

如果按替换操作来的话，

dp[2][2]=dp[1][1]+1=1

DP求解思路

如果按删除操作来的话，dp[2][2]=dp[1][2]+1=2
如果按插入操作来的话，dp[2][2]=dp[2][1]+1=2
如果按替换操作来的话，dp[2][2]=dp[1][1]+1=1
从前面的推导可以看出，我们计算dp[2][2]通过dp[1][1]+1来得到数值是最小的

https://github.com/NLP-lecture/part-1

str2

str1		0	1	2	3	4	5	6
	0		空格	C	C	A	C	A
	1	空格	0	1	2	3	4	5
	2	A	1	?				
	3	A	2					
	4	T	3					
	5	C	4					
	6	T	5					

因此，我们可以确定，当计算dp[i][j]时，如果i和j对应字符**不同**的时候，应该选下移、右移、斜移动中数值最小的

DP求解思路

如果计算dp[i][j]时，i和j对应字符相同我们该怎么办？
凉拌！
我们还按照原来的思路，看看下移计算出来的结果是啥？

https://github.com/NLP-lecture/part-1

str2

str1		0	1	2	3	4	5	6
	0		空格	C	C	A	C	A
	1	空格	0	1	2	3	4	5
	2	A	1	1	2	?		
	3	A	2					
	4	T	3					
	5	C	4					
	6	T	5					

下移的话， dp[1][4]时，str1已经变成str2，即为“空格CCA”了， dp[2][4]意味着str1又往后看了一位字符“A”， 变成“空格CCAA”， 而str2还是“空格CCA”， 所以str1变成str2所需最少操作是删除“A”。
如果按下移来的话，
dp[2][4]=dp[1][4]+1=4

DP求解思路

https://github.com/NLP-lecture/part-1

如果计算dp[i][j]时，i和j对应字符相同我们该怎么办？
凉拌！
我们还按照原来的思路，看看右移计算出来的结果是啥？

		str2					
		0	1	2	3	4	5	6
	0		空格	C	C	A	C	A
	1	空格	0	1	2	3	4	5
	2	A	1	1	2	?		
str1	3	A	2					
	4	T	3					
	5	C	4					
	6	T	5					

右移的话， dp[2][3]时，str1已经变成str2，即为“空格CC”了，dp[2][4]意味着str2又往后看了一位字符“A”，变成“空格CCAA”，而str1还是“空格CCA”，所以str1变成str2所需最少操作是插入“A”。
如果按右移来的话，
dp[2][4]=dp[2][3]+1=3

DP求解思路

如果计算dp[i][j]时，i和j对应字符相同我们该怎么办？
凉拌！
我们还按照原来的思路，看看斜移动计算出来的结果是啥？

https://github.com/NLP-lecture/part-1

		str2					
		0	1	2	3	4	5	6
	0		空格	C	C	A	C	A
	1	空格	0	1	2	3	4	5
	2	A	1	1	2	?		
str1	3	A	2					
	4	T	3					
	5	C	4					
	6	T	5					

斜移动的话， dp[1][3]时，str1已经变成str2，即为“空格CC”了，dp[2][4]意味着str1和str2都往后看了一位字符“A”，都变成“空格CCA”，那还需要什么操作吗？当然不需要str1已经跟str2一样了。
如果按斜移动来的话，
dp[2][4]=dp[1][3]+0=2

DP求解思路

如果按下移来的话，dp[2][4]=dp[1][4]+1=4
如果按右移来的话，dp[2][4]=dp[2][3]+1=3
如果按斜移动来的话，dp[2][4]=dp[1][3]+0=2
从前面的推导可以看出，我们计算dp[2][4]通过dp[1][3]+0来得到数值是最小的

https://github.com/NLP-lecture/part-1

		str2					
		0	1	2	3	4	5	6
	0		空格	C	C	A	C	A
	1	空格	0	1	2	3	4	5
	2	A	1	?				
str1	3	A	2					
	4	T	3					
	5	C	4					
	6	T	5					

因此，我们可以确定，当计算dp[i][j]时，如果i和j对应字符**相同**的时候，应该直接选择斜移动的数值即可

终于，我们形成了一个算法

https://github.com/NLP-lecture/part-1

1、先构建一个状态转移矩阵
2、进行初始化
3、进行移动，分成两种情况：

当计算dp[i][j]时，如果i和j对应字符**不同**的时候，应该选下移、右移、斜移动中数值最小的

当计算dp[i][j]时，如果i和j对应字符**相同**的时候，应该直接选择斜移动的数值即可

4、当我们移动到矩阵右下角时，完成了全部推导，右下角的值就是最小编辑距离

至于现在还没明白算法为什么这么做的同学可以自行翻看前面ppt，自己试着推一边

下面我们做一个练习：
https://alchemist-al.com/algorithms/edit-distance

把算法变成程序

https://github.com/NLP-lecture/part-1

edit_distance.py