

2. 最小编辑距离

WU Xiaokun 吴晓埜

xkun.wu [at] gmail

2022/03/11

定义

用例：拼写

拼写校正

- 用户写“graffe”，下面那个正确？
 - graf, graft, grail, giraffe

用例：拼写

拼写校正

- 用户写“graffe”，下面那个正确？
 - graf, graft, grail, giraffe

中文：拼音输入法

严谨作弊？
严禁作弊！

用例：拼写

拼写校正

- 用户写“graffe”，下面那个正确？
 - graf, graft, grail, giraffe

中文：拼音输入法

严谨作弊？
严禁作弊！

模糊搜索：返回相近的搜索

用例：计算生物学

- RNA 序列匹配：

```
AGGCTATCACCTGACCTCCAGGCCGATGCCC  
TAGCTATCACGACCGCGGTCGATTTGCCCGAC
```

```
-AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---  
TAG-CTATCAC--GACCGC--GGTCGATTTGCCCGAC
```

用例：共指

共指 **coreference**: 表述之间是否指向同一个实体

中国

中华人民共和国

大中华地区

中國（佛教）：釋迦牟尼出生地

震旦：漢傳佛教經典中，對古代中國的稱呼

阮朝：越南阮朝朝廷在國內有時也自稱為「中國」

China, la Chine, ...

用例：共指

共指 **coreference**: 表述之间是否指向同一个实体

中国
中华人民共和国
大中华地区
中國（佛教）：釋迦牟尼出生地
震旦：漢傳佛教經典中，對古代中國的稱呼
阮朝：越南阮朝朝廷在國內有時也自稱為「中國」
China, la Chine, ...

其他应用：机器翻译，信息提取，语言识别

度量相似度

从计算的角度讲，很多NLP任务的核心是度量字符串的相似度

- 如何度量相似度？观察打字与修改
 - 字符串是由编辑操作构造的
 - 每一步只涉及一个字符

例如最简单的拉丁字母序列

```
intention  
execution
```

编辑距离

从计算的角度讲，很多NLP任务的核心是度量字符串的相似度

- 如何度量相似度？观察打字与修改
- 例如最简单的拉丁字母序列

```
intention  
execution
```

编辑距离 edit distance：编辑操作的数量

- 插入字符
- 删除字符
- 替换字符

对齐可视化

对齐 alignment: 将尽可能多的相同字符对应起来

- 插入、删除字符时插入空字符*

d	s	s		i	s														
I	N	T	E	*	N	T	I	O	N										
*	E	X	E	C	U	T	I	O	N										

- 编辑距离: 简单数一下编辑操作

对齐可视化

对齐 alignment: 将尽可能多的相同字符对应起来

- 插入、删除字符时插入空字符*

d	s	s		i	s							
I	N	T	E	*	N	T	I	O	N			
*	E	X	E	C	U	T	I	O	N			

- 编辑距离: 简单数一下编辑操作

最小编辑距离: 对齐最多的字母; 最少的编辑操作

编辑成本

Levenshtein 距离：字符串之间的加权编辑距离

- 每种操作的“成本”可以不一样

d	s	s		i	s														
I	N	T	E	*	N	T	I	O	N										
*	E	X	E	C	U	T	I	O	N										

- 等权重：5
- 替换成本翻倍：8
 - 替换可以看成：删除+插入

MED 应用

评测机器翻译、语音识别

	d	d				i	i
He	to	reporters	introduced	main	content	*	*
He	*	*	introduced	main	content	to	reporters
他	向	记者	介绍了	主要	内容	*	*

MED 应用

评测机器翻译、语音识别

	d	d				i	i
He	to	reporters	introduced	main	content	*	*
He	*	*	introduced	main	content	to	reporters
他	向	记者	介绍了	主要	内容	*	*

命名实体识别、共指消解

中华人民共和国 简称 中国

如何计算MED?

简单搜索：搜索空间随字符长度指数级增长

- D: 1
- I: 26
- S: 26, 或禁止替换操作

如何计算MED?

简单搜索：搜索空间随字符长度指数级增长

- D: 1
- I: 26
- S: 26, 或禁止替换操作

大量路径通往相同字符串

d	s	s		i	s							
I	N	T	E	*	N	T	I	O	N			
*	E	X	E	C	U	T	I	O	N			

s	d	s		s	i							
I	N	T	E	N	*	T	I	O	N			
E	*	X	E	C	U	T	I	O	N			

- 不需要记住所有路径
- 只需要其中一条最短路径

如何计算MED?

简单搜索：搜索空间随字符长度指数级增长

- D: 1
- I: 26
- S: 26, 或禁止替换操作

大量路径通往相同字符串

d	s	s		i	s					
I	N	T	E	*	N	T	I	O	N	
*	E	X	E	C	U	T	I	O	N	

s	d	s		s	i					
I	N	T	E	N	*	T	I	O	N	
E	*	X	E	C	U	T	I	O	N	

- 不需要记住所有路径
- 只需要其中一条最短路径

听起来像是动态规划的思想?

计算MED

问题定义

字符串:

- $x = x_1x_2\dots x_n$
- $y = y_1y_2\dots y_m$

问题定义

字符串:

- $x = x_1x_2\dots x_n$
- $y = y_1y_2\dots y_m$

子串:

- $x[1..i] = x_1x_2\dots x_i$
- $y[1..j] = y_1y_2\dots y_j$

问题定义

字符串:

- $x = x_1x_2\dots x_n$
- $y = y_1y_2\dots y_m$

子串:

- $x[1..i] = x_1x_2\dots x_i$
- $y[1..j] = y_1y_2\dots y_j$

距离函数:

- 字符串距离: $D(i, j) = MED(x[1..i], y[1..j])$
- 编辑距离: $D(n, m)$

动态规划

动态规划 **dynamic programming**: 基于表格查找

- 将问题拆解成一系列子问题
- 自底向上合并子问题的解

动态规划

动态规划 dynamic programming: 基于表格查找

- 将问题拆解成一系列子问题
- 自底向上合并子问题的解

Bellman 方程: 递推关系 recurrence relation

$$D(i, j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min\{\alpha_{x_i, y_j} + D(i-1, j-1), \delta + D(i-1, j), \delta + D(i, j-1)\} & \text{otherwise} \end{cases}$$

- 插入、删除花费: $\delta = 1$; 替换花费: $\alpha_{pq} = \{0, 2\}$

MED表：初始化

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

MED表：填表 I

I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
#	E	X	E	C	U	T	I	O	N	

I	I	I	I	I	I	
*	*	*	*	*	*	
E	X	E	C	U	T	I

MED表：填表 E

E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
#	E	X	E	C	U	T	I	O	N	

d	d	d	
I	N	T	E
*	*	*	E

MED表：完整版

N	9	8	9	10	11	12	11	10	9	8
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
#	E	X	E	C	U	T	I	O	N	

回溯

对齐方案

注意：任务通常并不仅是计算编辑距离

- 大多数情况需要字符对齐的具体方案

N	9	8	9	10	11	12	11	10	9	8
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
#	E	X	E	C	U	T	I	O	N	

s	d	s		s	i					
I	N	T	E	N	*	T	I	O		
E	*	X	E	C	U	T	I	O		

回溯法

回溯法 **backtrace**: 在计算每个表格时, 记录路径来源

- 算法结束后, 从终点回溯到起点
- 注意: 路径不唯一, 取决于代码执行顺序
 - 但关键节点一定会通过

E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
#	E	X	E	C	U	T	I	O	N	

MED回溯表

n	9	↓ 8	↙←↓ 9	↙←↓ 10	↙←↓ 11	↙←↓ 12	↓ 11	↓ 10	↓ 9	↙ 8
o	8	↓ 7	↙←↓ 8	↙←↓ 9	↙←↓ 10	↙←↓ 11	↓ 10	↓ 9	↙ 8	← 9
i	7	↓ 6	↙←↓ 7	↙←↓ 8	↙←↓ 9	↙←↓ 10	↓ 9	↙ 8	← 9	← 10
t	6	↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ 8	↙←↓ 9	↙ 8	← 9	← 10	←↓ 11
n	5	↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ 8	↙←↓ 9	↙←↓ 10	↙←↓ 11	↙↓ 10
e	4	↙ 3	← 4	↙← 5	← 6	← 7	←↓ 8	↙←↓ 9	↙←↓ 10	↓ 9
t	3	↙←↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ 8	↙ 7	←↓ 8	↙←↓ 9	↓ 8
n	2	↙←↓ 3	↙←↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ 8	↓ 7	↙←↓ 8	↙ 7
i	1	↙←↓ 2	↙←↓ 3	↙←↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙ 6	← 7	← 8
#	0	1	2	3	4	5	6	7	8	9
	#	e	x	e	c	u	t	i	o	n