词形态分析
Lexical Morphological Analysis - 1

王小捷
智能科学与技术中心
北京邮电大学

CL技术内容=语言+计算

- 问题：从符号匹配到自然语言匹配
  - 正则表达式：符号模式匹配的有效工具
  - 用于自然语言模式匹配时会遇到一些挑战：

- 搜索have时，如何包含下面：
  - I’ve read the book. (have)
- 搜索“因为…所以”时，如何排除下面：
  - 我的病因为何总查不出来? 你这卫生所以后我再也不来了。 (因为…所以)
- 一种办法是设计更复杂的正则表达式
  - 如have|’ve

- 另一个解决办法：标出词边界：
  - I’ve read the book. (have)
  - → I have read the book.

- 我的病因为何总查不出来? 你这卫生所以后我再也不来了。 (因为…所以)
- →我/的/病因/为何/总/查/不/出来/?/ 你/这/卫生所/以后/我/再/也/不/来/了/。

- 另一类问题
- 如何包括：
  - He got the message. (get)
- 一种解决方法：get|got

- 另一个解决办法：解析出词结构
  - He got the message. (get)
  - → He get+V+Past the message

- 标出词边界、解析词结构：
- 从面向符号串的匹配
- →
- 面向语言符号串的分析
  - 首先是词汇层的分析

- 词汇层的分析首先是词形态分析(Lexical Morphological Analysis)
- 词形态分析有两个基本任务：
  - 断词(Tokenization):
    - 在符号序列中找出词序列，确定词的外部边界。
  - 词形还原(Lemmatization)
    - 词内部结构分析：dogs → dog + PL(复数)

- 不同语种中两个任务的差异
  - 英语
    - 断词: 相对简单 (tokenization)
    - 词形还原 ➔ 形态分析(morphological analysis)
  - 汉语
    - 断词 ➔ 切分(segmentation)
    - 词形还原: 基本无此任务

英语-断词(Tokenization):

- 平凡任务?
  - I think it is a trivial task.
- 一些小小的例外
  - Words including punctuations or spaces (easy)
    - abbreviation： I'm, i.e. , Ph.D. , Google.com
    - With spaces: per se, ad hoc
  - Mixing Numerals\letters\ punctuations (a little boring)
    - ACL2012, AK-47, 11/02/2000, 123,456.78, …
  - Words Including some special characters (for debate)
    - AT&T, Micro$oft, one-third
  - …… (something more difficult?)

英语词形还原Lemmatization(Morphological Analysis)

- 英语词汇结构模式
  - {prefix}＋ {stem} ＋ {suffix}
- 例子：
  - Impossible (prefix + stem)
  - Happiness (noun + suffix)
  - Boys (plural suffix）
  - Housewarmings (compund＋复数词尾）

有些语言的内部结构可能更丰富

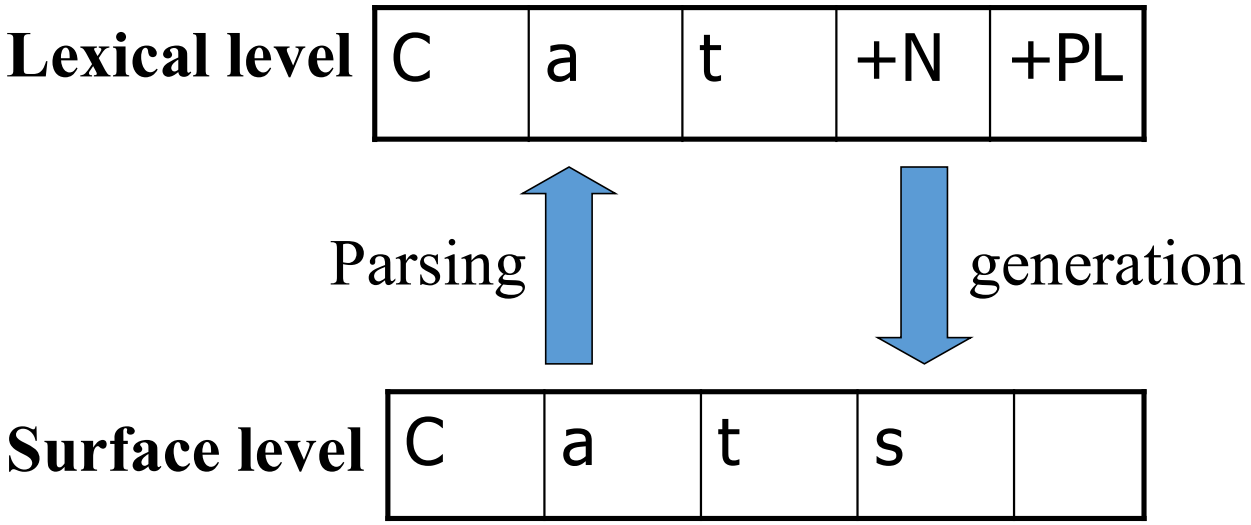
- English (数量巨大) (屈折语)
  - {prefix}＋ {stem} ＋ {suffix}
  - >100 >10000 >100
- Turkish word (黏着语)
  - Uygarlastiramadiklarimizdanmissinizcasina
    - Uygar(civilized)+las(become)+tir(cause)+ama(not able)+dik(past) +lar(plural)+imiz(1pl)+ dan(abl)+ mis(past)+ siniz (2pl)+ casina(as if)
  - `(behaving) as if you are among those whom we could not civilize'

Morphological Analysis

- 剖析 (Parsing)：获得词的内部结构
  - cats → cat + N + PL
  - caught → catch + V +Past
  - ……
- 也希望反向地能从结构信息来拼写出词
- 生成 (Generation)
  - cat + N + PL → cats
  - catch + V +Past → caught
  - ……

剖析(Parsing)/生成(Generation)

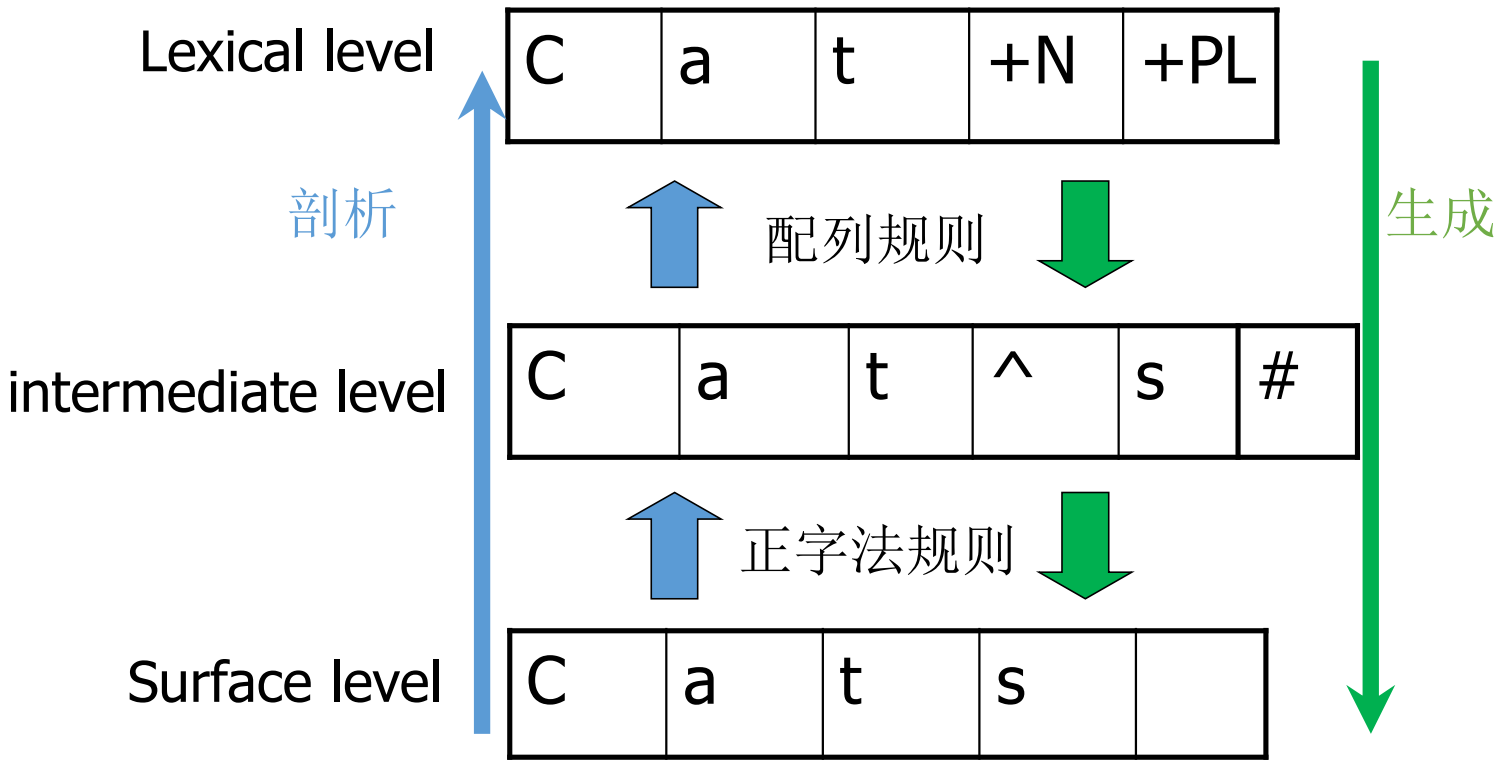
■ “cats” ←→ “cat +N +PL”



词形态分析需要什么知识？

- 三类知识:
  - 词汇(Lexicon)
    - stem list
    - affix list
  - 配列规则 (Morphotactics)
    - 词缀的使用次序
      - E.g: 词缀+复数，动词+tion+al变为形容词
  - 正字法 (Orthographic(Phonological) Rules)
    - 拼写规则(Spelling rules):
      - e.g.复数时怎样是+es，怎样是去y改i +es …

利用这三类知识的过程

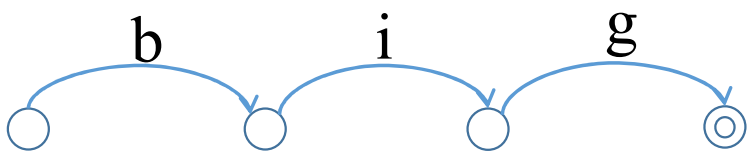

Lexical level
C a t +N +PL
剖析
配列规则
生成
intermediate level
C a t ^ s #
正字法规则
Surface level
C a t s

三类知识的数学模型

- 词汇：符号描述
- 配列规则：规则描述
- 正字法：规则描述

- 计算模型？

Lexicon知识可以用FSA来表示



- 配列规则、正字法规则使用什么模型？
  - 规则 不同于 词表
- 有限状态转录机 (Finite State Transducers, FST)

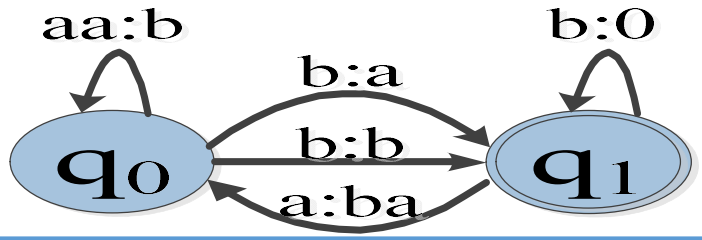
有限状态转录机

- 简单地说
  - 增加一条带子、在原带子接收符号输入的同时在增加的带子上输出符号
  - 例如 正字法规则
  - 输入带子读：“cat^s#”
  - 输出带子写：“cats”

有限状态转录机: vs 有限状态自动机

FST	FSA
状态集: Q	状态集: Q
初始状态: $q_0 \in Q$	初始状态: $q_0 \in Q$
终止状态集: F sub Q	终止状态集: F sub Q
输入字母集: $\sum$	输入字母集: $\sum$
转移函数: $\delta(q,w)$: $Q \times \sum^* \rightarrow 2^Q$	转移函数: $\delta(q,w)$: $Q \times \sum^* \rightarrow 2^Q$
输出字母集: Δ	
输出函数: $\sigma(q,w)$: $Q \times \sum^* \rightarrow 2^{\Delta^*}$	

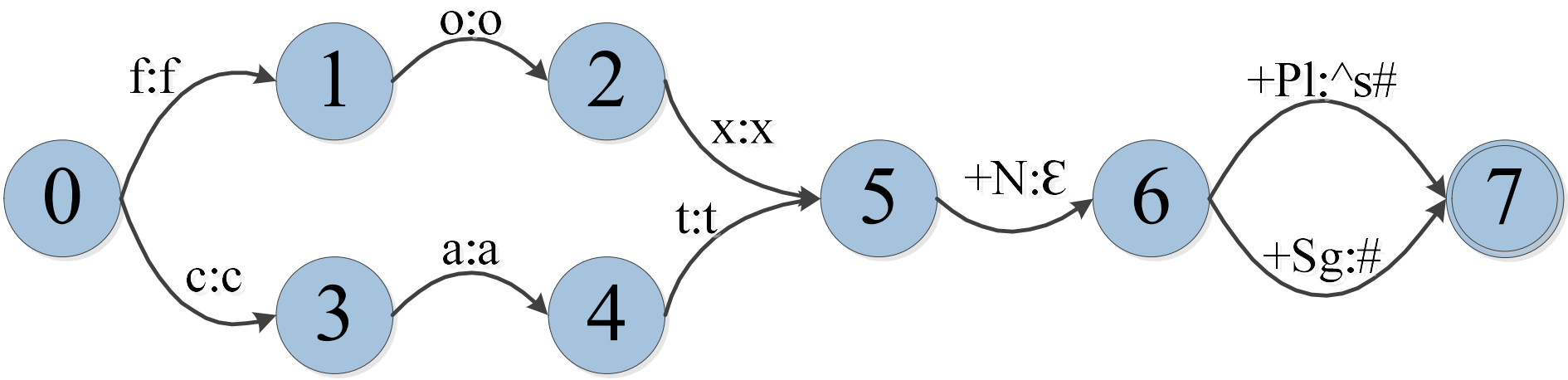
FST: 示例



FST	In above example
状态集: Q	$Q=\{q_0,q_1\}$
初始状态: $q_0 \in Q$	$q_0 \in Q$
终止状态集: F sub Q	$F=\{q_1\}$
输入字母集: ∑	{aa,b,a}
转移函数: delta(q,w): $Q \times \sum^* \rightarrow 2^Q$	$(q_0,aa)\rightarrow q_0$, $(q_1,aa)\rightarrow ø$ $(q_0,b)\rightarrow q_1$, $(q_1,b)\rightarrow q_1$ $(q_0,a)\rightarrow ø$, $(q_1,a)\rightarrow q_0$
输出字母集: ⊿	{b,a,ba,0}
输出函数: Sigma(q,w): $Q \times \sum^* \rightarrow 2^{⊿*}$	$(q_0,aa)\rightarrow b$, $(q_1,aa)\rightarrow ø$ $(q_0,b)\rightarrow$ **{a,b}**, $(q_1,b)\rightarrow 0$ $(q_0,a)\rightarrow ø$, $(q_1,a)\rightarrow ba$

配列规则FST

名词加复数：fox+N+PL ➔ fox^s#　(生成方向)



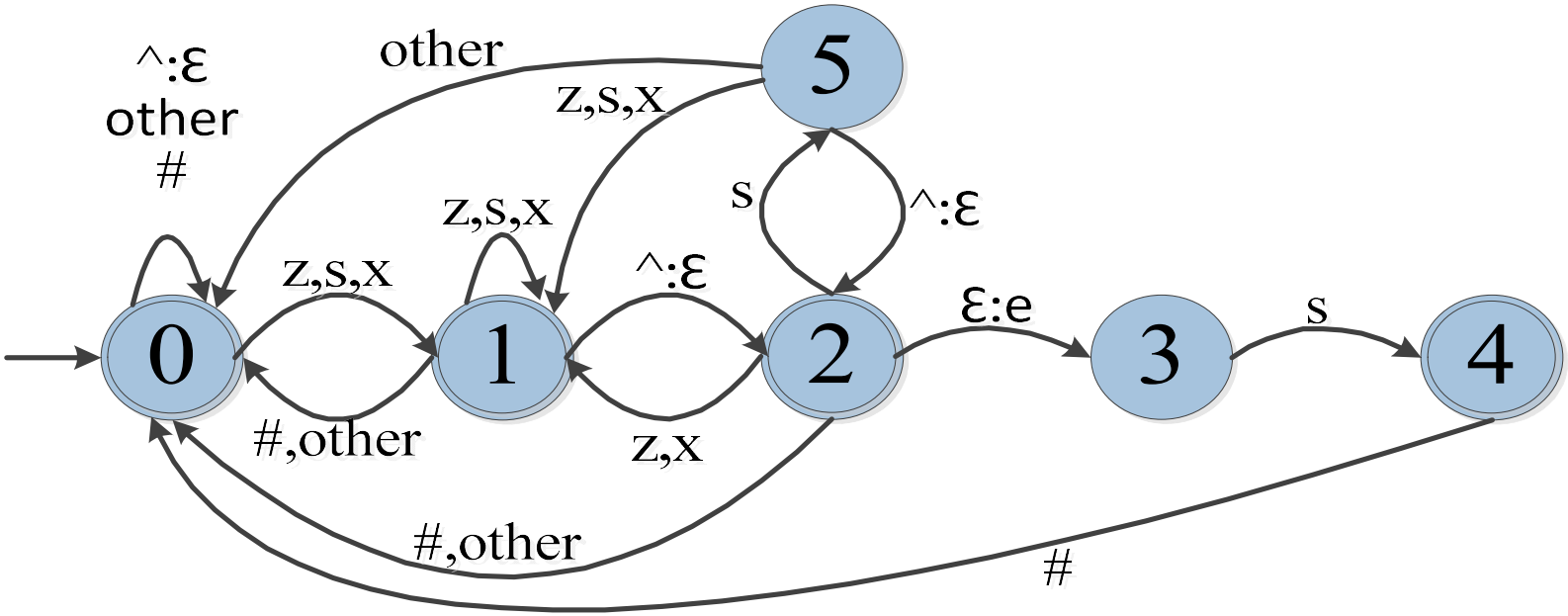
反之，剖析方向，即要实现 fox^s# ➔ fox+N+PL 时，FST结构不变，只需要输入-输出换位即可

正字法规则FST

加复数时的 e-insertion规则：

$$\epsilon \rightarrow e / \begin{Bmatrix} x \\ s \\ z \end{Bmatrix} \text{^} \_\_\_\_ s\#$$

fox^s# ➔foxes (生成方向)



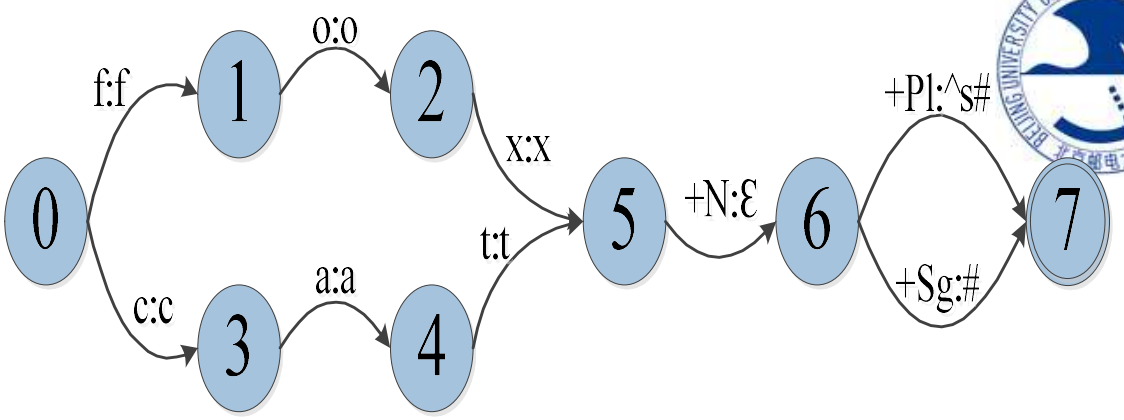
反之，剖析方式，即要实现 foxes ➔ fox^s# 时，
结构不变，输入-输出换位即可

按序使用两个FST进行生成：

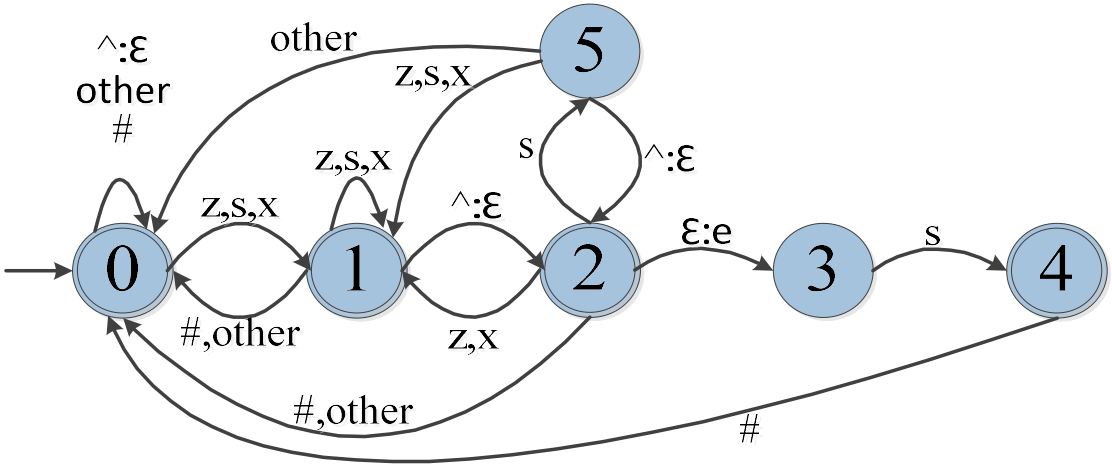
fox+N+PL

➔ fox^s#

➔ foxes



1、配列规则 (生成): fox+N+PL➔fox^s#



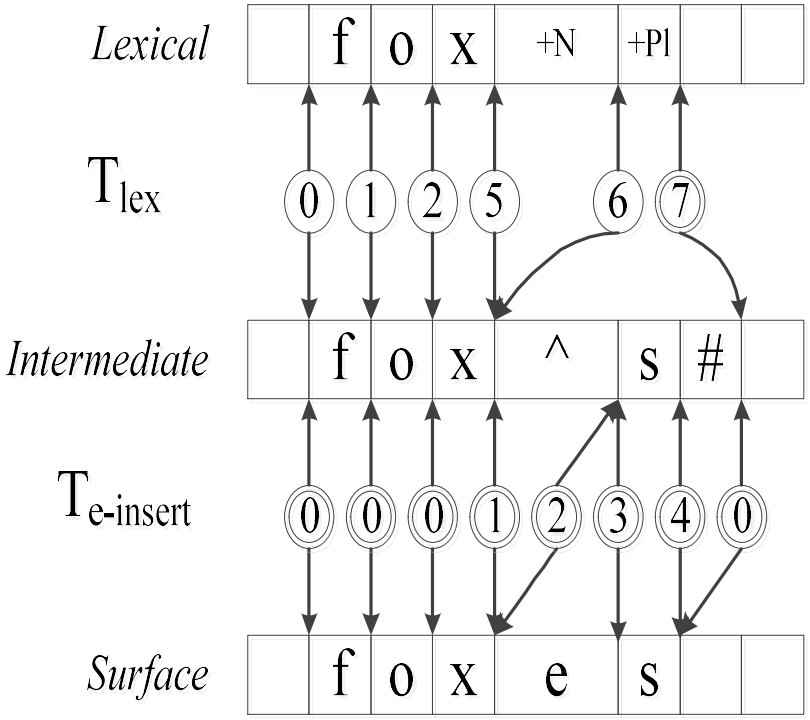
2、正字法规则(生成)：fox^s# ➔foxes

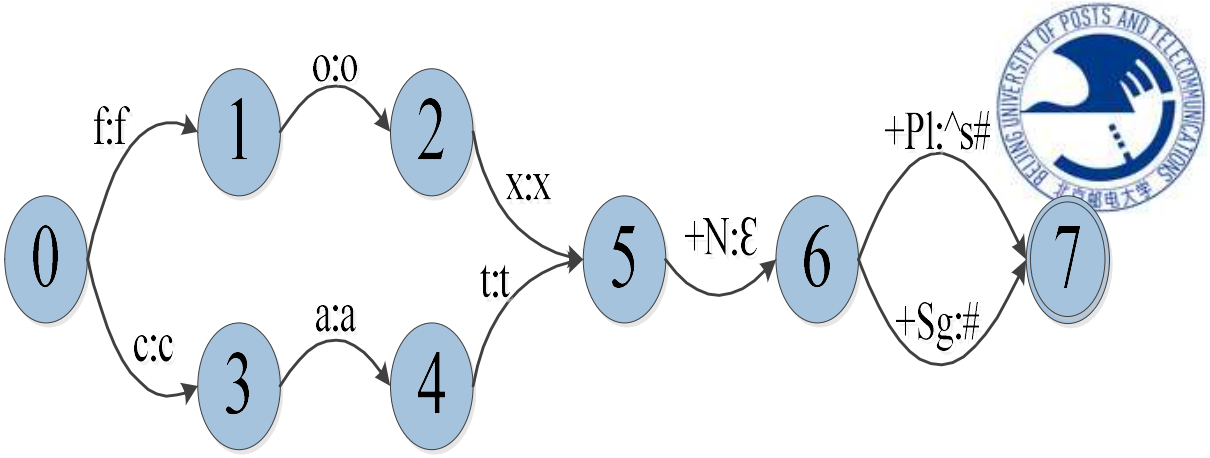
按序使用两个FST进行生成：

fox+N+PL

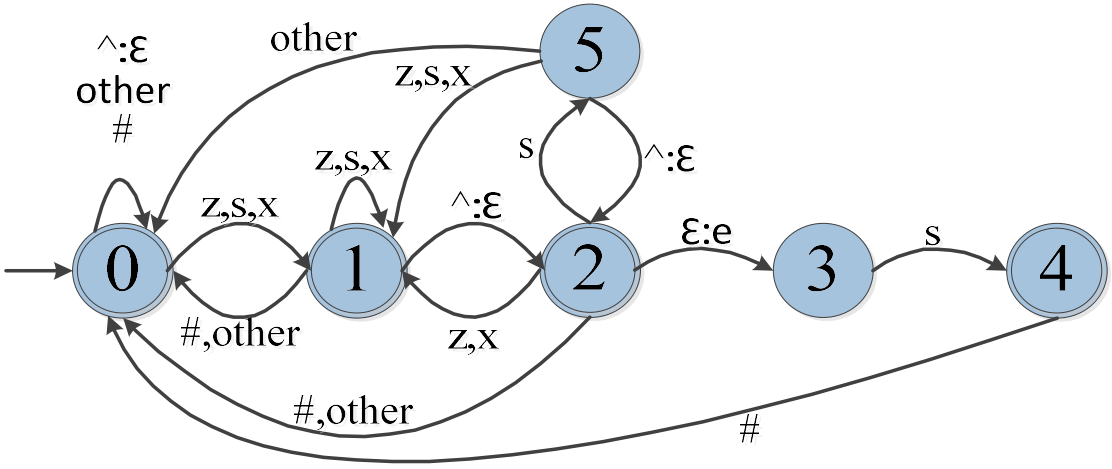
➔ fox^s#

➔ foxes



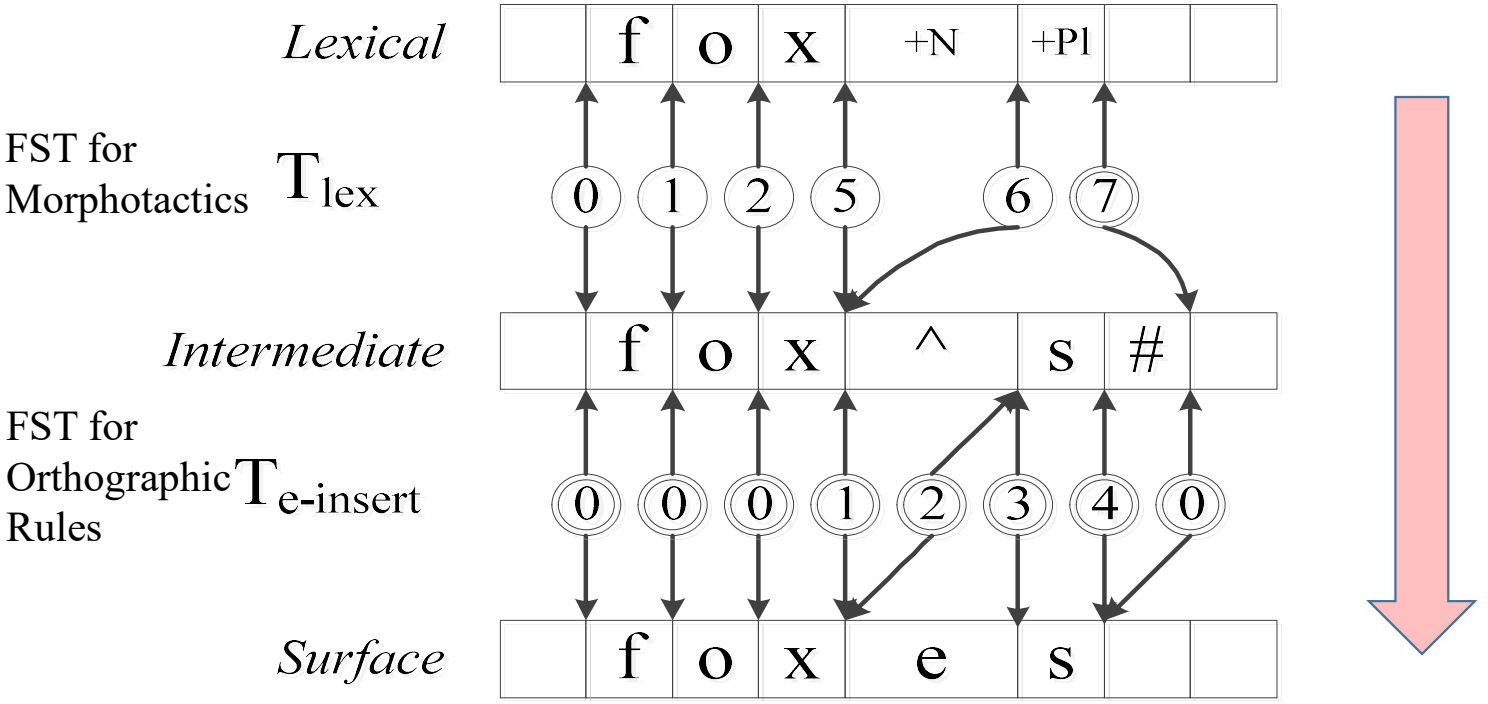


1、配列规则 (生成): fox+N+PL➔fox^s#



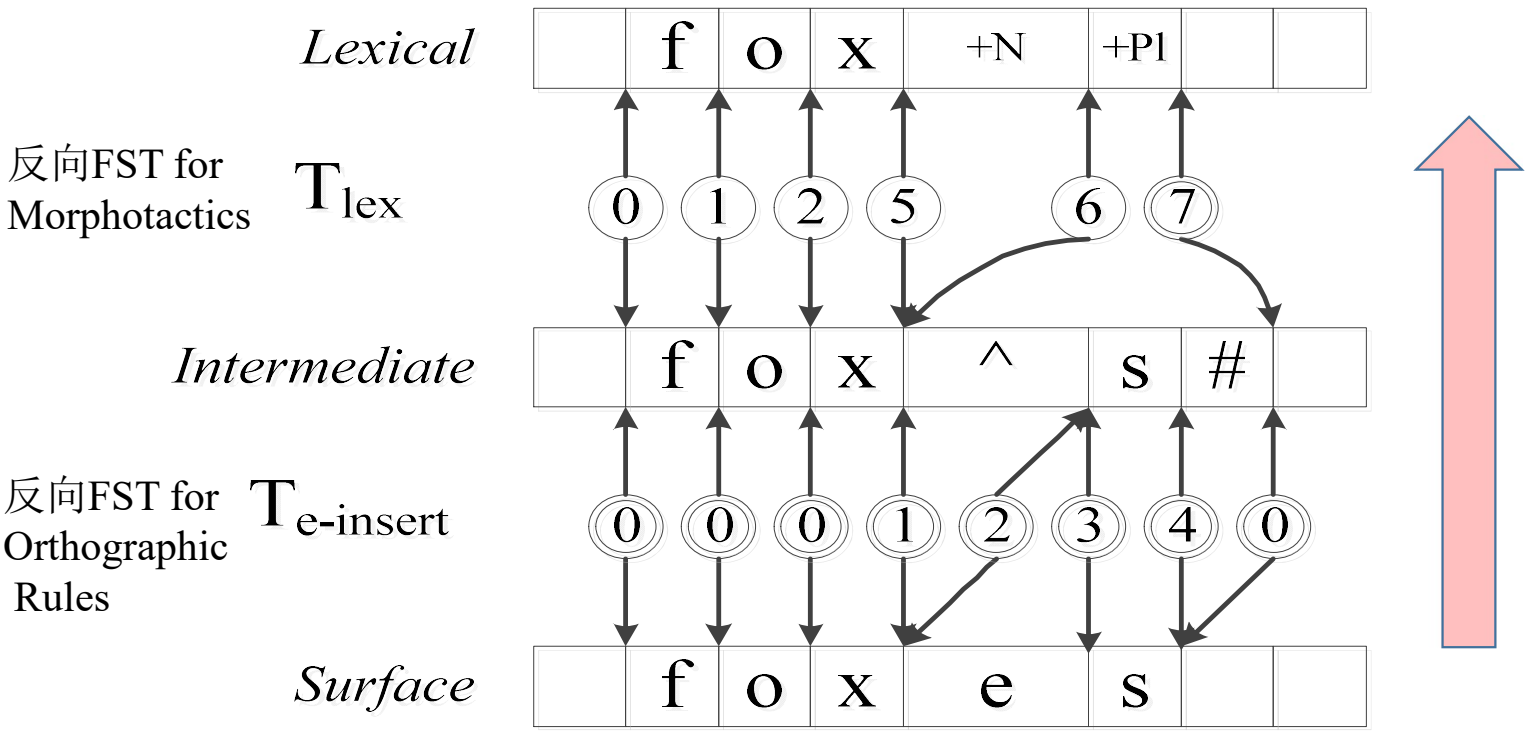
2、正字法规则(生成)：fox^s# ➔foxes

组合 FST 进行 生成 和 剖析



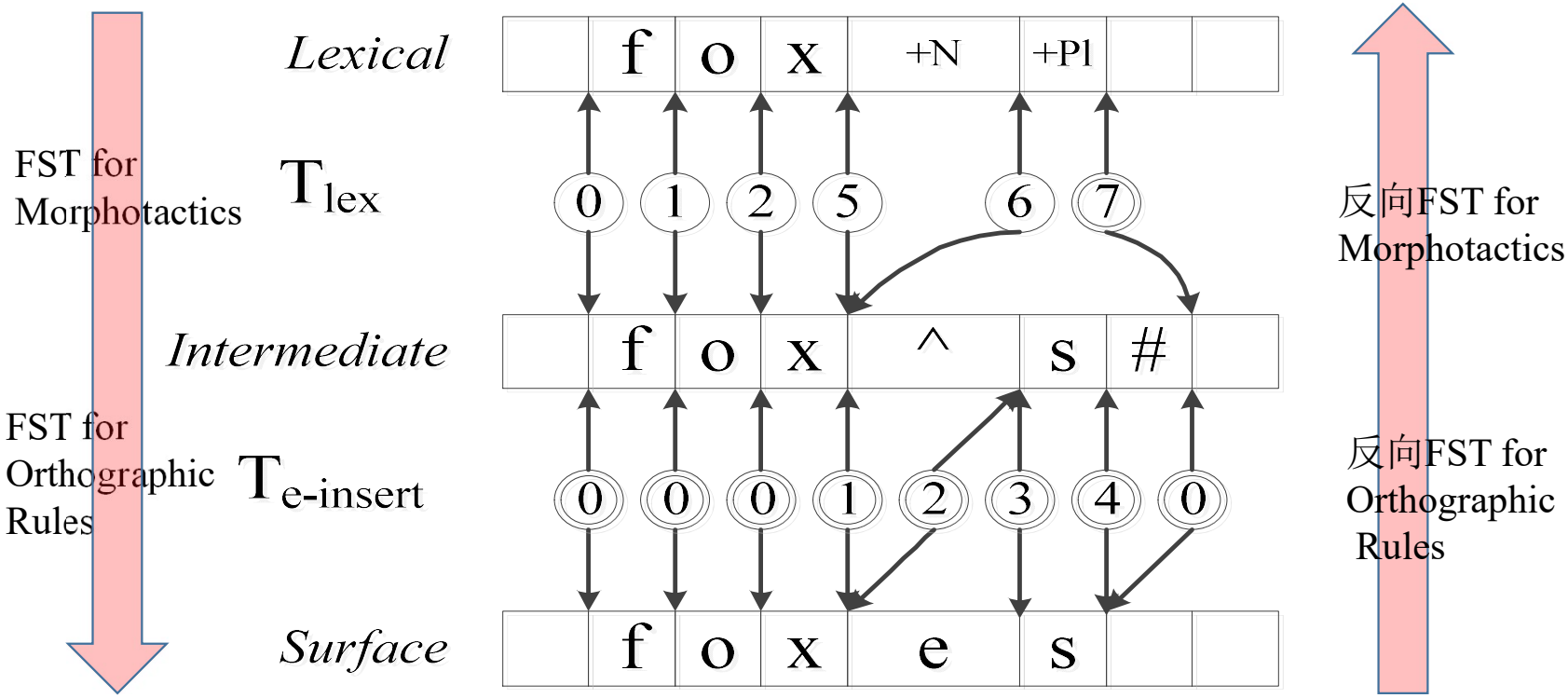
Generation: fox+N+PL ➔ fox^s# ➔ foxes

组合 FST 进行 生成 和 剖析



Parsing: foxes➔ Fox^s#➔ fox+N+PL

组合 FST 进行 生成 和 剖析



Generation:

fox+N+PL ➔ fox^s#➔foxes

parsing:

foxes➔ Fox^s#➔ fox+N+PL

词形态分析之应用

- 非词错误检测
  - It are you | It ih you.
  - FSF based recognizor
- 非词错误纠正
  - 检测出来之后，推荐最相似的正确词
  - 那个词最相似?
    - 距离最近：
      - 最小编辑距离(Minimum Edit Distance, MED)
    - 与错词具有最小的MED的词作为推荐词

编辑距离

- 从一个串变换到另一个串的编辑操作次数来计算两个字符串间的距离
  - Insertion; Deletion; Substitution
    - Levenshtein distance：sub+1 / sub +2
  - Example：intention -- execution
    - 编辑操作序列：s,s,s,s,s: 总距离2+2+2+2+2=10

S	i	n	t	e	n	t	i	o	n
T	e	x	e	c	u	t	i	o	n
O	s	s	s	s	s				

最小编辑距离

- 达成目标的编辑操作序列可以有多种
  - Example：intention -- execution

S	i	n	t	e		n	t	i	o	N
T		e	x	e	c	u	t	i	o	n
O	d	s	s		i	s				

  - 编辑操作序列d,s,s,i,s: 总编辑距离： 1+2+2+1+2=8
- 最小编辑距离：
  - 所有可能编辑操作序列中的最小距离

- 应用：将两个词的最小编辑距离作为两个词的距离，那么在拼写推荐中，推荐与错词的最小编辑距离最小的词作为候选。

- 如何求最小编辑距离?

- 穷尽方法：
- 找出所有可能的操作序列，比较大小
  - 数量大
  - 重复操作

- 基于动态规划的方法
  - 原理：将求A和B的最小编辑距离➔求从A转化为B的最优编辑操作序列(即产生最小编辑距离的操作序列) 对应的每次操作结果构成了串变化的最优路径。
  - 例如：求intention 与 execution的最小编辑距离

串序列 (串序列变化最优路径)	操作序列 (最优操作序列)
intention	d (删除i)
ntention	s (用e替换n)
etention	s (用x替换t)
exention	i (插入c)
execntion	s (用u替换n)
execution	

- 方法：定义串的状态以及串变化时状态间的关系，基于关系进行递推计算
- 串的状态：
  - dis [i,j]：目标串前i个字符和源串前j个字符间的距离
- 串状态间的关系：

$$dis[i,\ j] = \min\begin{cases} dis[i-1,\ j] + ins(t[i]) \\ dis[i-1,\ j-1] + sub(s[j], t[i]) \\ dis[i,\ j-1] + del(s[j]) \end{cases}$$

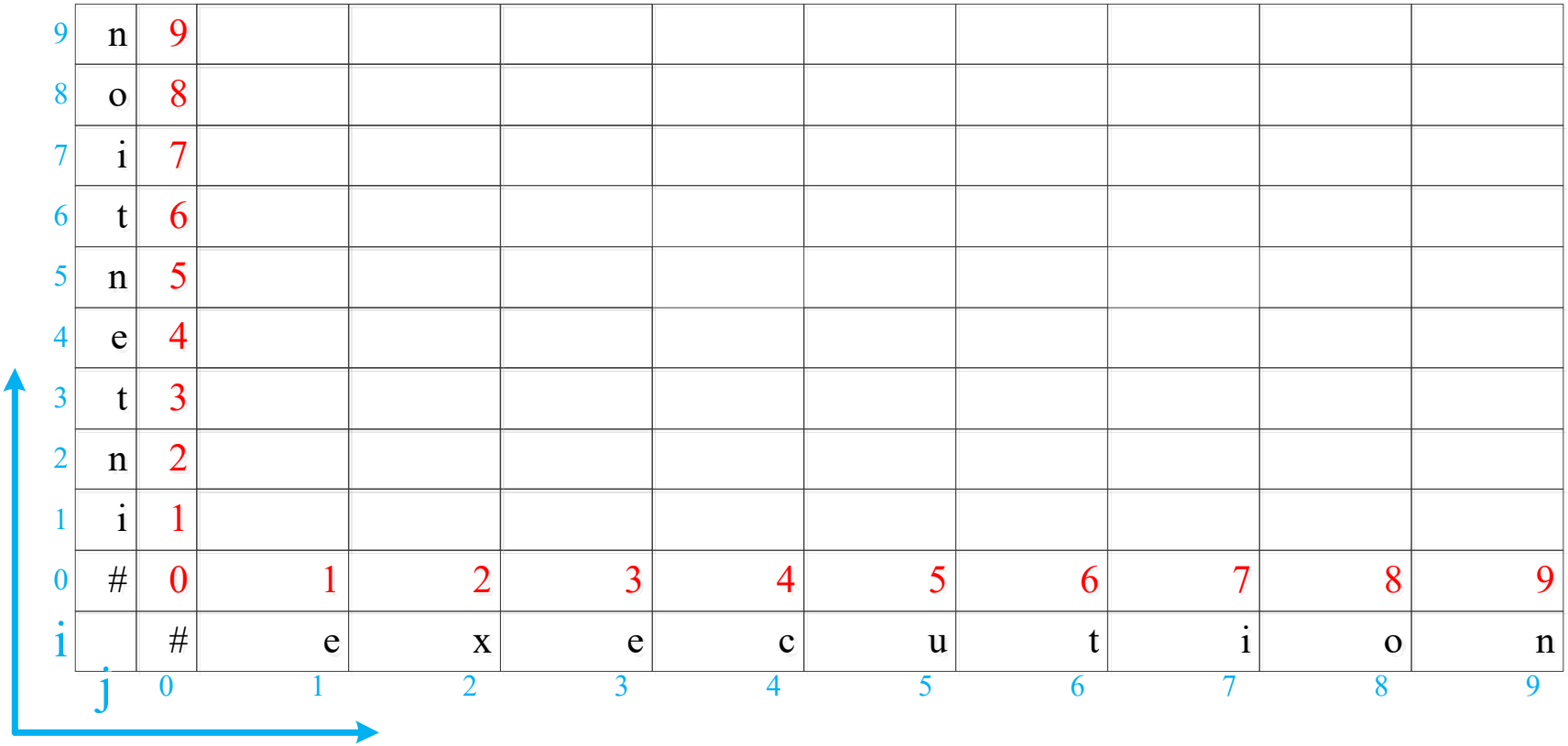
MED 求解算法

```
function Min-Edit-Distance(target,source) return min-dis
 n ← Length(target)
 m← Length(source)
 create a distance matrix dis[n+1,m+1]
 Initialize the zeroth row and column to be the distance from the empty string
   dis[0,0]=0
   for each column i from 1 to n do
      dis[i,0]← dis[i-1,0]+ins(target[i])
   for each row j from 1 to m do
      dis[0,j]← dis[0, j-1]+del(source[j])
   for each column i from 1 to n do
      for each row j from 1 to m do
         dis[i,j]←Min(dis[i-1, j]+ins(target[i]),
                      dis[i-1,j-1]+sub(source[j],target[i]),
                      dis[i,j-1]+del(source[j]))
 return dis[n,m]
```

算法示例

intention与execution的MED

$$dis[i,j] \leftarrow \text{Min}(dis[i\text{-}1, j]+ins(target[i]),$$
$$dis[i\text{-}1,j\text{-}1]+sub(source[j],target[i]),$$
$$dis[i,j\text{-}1]+del(source[j]))$$



i		j=0	1	2	3	4	5	6	7	8	9
9	n	9									
8	o	8									
7	i	7									
6	t	6									
5	n	5									
4	e	4									
3	t	3									
2	n	2									
1	i	1									
0	#	0	1	2	3	4	5	6	7	8	9
		#	e	x	e	c	u	t	i	o	n

■ 前向计算

初始化：

$$dis[i,0] \leftarrow dis[i\text{-}1,0]+ins(target[i])$$
$$dis[0,j] \leftarrow dis[0, j\text{-}1]+del(source[j])$$

算法示例

intention与execution的MED

$$dis[i,j] \leftarrow \mathrm{Min}(dis[i\text{-}1,j]+ins(target[i]),\ dis[i\text{-}1,j\text{-}1]+sub(source[j],target[i]),\ dis[i,j\text{-}1]+del(source[j]))$$

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

$$dis[1,1] \leftarrow \mathrm{Min}(dis[0,1]+ins(target[1])=1+1,\ dis[0,0]+sub(source[1],target[1])=0+2,\ dis[1,0]+del(source[1]))=1+1$$

■ 前向计算

算法示例

intention与execution的MED

$$dis[i,j] \leftarrow \text{Min}(dis[i\text{-}1,j]+ins(target[i]),\ dis[i\text{-}1,j\text{-}1]+sub(source[j],target[i]),\ dis[i,j\text{-}1]+del(source[j]))$$

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

■ 前向计算

算法示例

intention与execution的MED

$$dis[i,j] \leftarrow \mathrm{Min}(dis[i\text{-}1,j]+ins(target[i]),\ dis[i\text{-}1,j\text{-}1]+sub(source[j],target[i]),\ dis[i,j\text{-}1]+del(source[j]))$$

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

- 反向获取最优操作序列，获得MED

思考

- 最小编辑距离度量的词间距离是什么？
  - wrong vs prong； wrong vs. mistake
- 汉字
  - 假设字的笔画集合唯一决定一个汉字，当已知两个字的笔画时，如何基于此度量两个字的形态学距离?
  - 假设字的笔画次序唯一决定一个汉字，当已知两个字的笔画次序时，如何基于此度量两个字的形态学距离?
  - 当两个字书写完成后，如何度量两个字的形态学距离?
- 汉字串

选读材料

- FST 与 RNN(Recurrent Neural Network)
  - 从RNN的状态来抽取FST
    - C. W. Omlin and C. L. Giles. Constructing deterministic finite-state automata in Recurrent Neural Network. JACM, 43(6):937–972, 1996.
    - G. Weiss, Y. Goldberg, and E. Yahav. Extracting automata from recurrent neural networks using queries and counterexamples. In ICML2018, 5244–5253.
  - 关联RNN状态和FST状态
    - P. Tiˆno, B. G. Horne, and C. L. Giles. Finite state machines and recurrent neural networks—automata and dynamical systems approaches. In Neural Networks and Pattern Recognition, 171 – 219. 1998.
    - Joshua J. Michalenko, et al. Representing Formal Languages: A Comparison Between Finite Automata and Recurrent Neural Networks. ICLR2019.

谢谢！