


第3章 形式语言与自动机

宗成庆

中国科学院自动化研究所

cqzong@nlpr.ia.ac.cn

本章内容

- 
1. 形式语言
 2. 自动机
 3. 自动机在NLP中的应用
 4. 习题
 5. 附录

1. 形式语言

◆关于语言的解释

语言是人类所特有的用来表达意思、交流思想的工具，是一种特殊的社会现象，由语音、词汇和语法构成一定的系统。

—商务印书馆，《现代汉语词典》，1996/2019

语言可以被看成一个抽象的数学系统。

—吴蔚天，1994

按照一定规律构成的句子和符号串的有限或无限的集合。

— N. Chomsky

1. 形式语言

◆ 语言描述的三种途径

- ❖ 穷举法 — 只适合句子数目有限的语言。
- ❖ 语法描述 — 生成语言中合格的句子。
- ❖ 自动机 — 对输入的句子进行检验，区别哪些是语言中的句子，哪些不是语言中的句子。

1. 形式语言

◇ 诺姆·乔姆斯基(Noam Chomsky)

- 1928年12月生于美国费城
- 1944年(16岁) 进入UPenn 学习哲学、语言学和数学
- 1949年获学士学位、1951年获硕士学位
- 1952 起在哈佛认知研究中心研究员，后来获UPenn博士学位；1957年(29岁) MIT副教授，32岁成为现代语言学教授、47岁终生教授。



1. 形式语言

◆形式语言的直观意义

形式语言(formal language)是用来精确地描述语言（包括人工语言和自然语言）及其结构的手段。形式语言学 也称 代数语言学。

以重写规则 $\alpha \rightarrow \beta$ 的形式表示，其中， α, β 均为字符串。
顾名思义：字符串 α 可以被改写成 β 。一个初步的字符串通过不断地运用重写规则，就可以得到另一个字符串。通过选择不同的规则并以不同的顺序来运用这些规则，就可以得到不同的新字符串。

1. 形式语言

◆形式语法的定义

形式语法是一个4元组 $G=(N, \Sigma, P, S)$, 其中 N 是非终结符的有限集合(有时也叫变量集或句法种类集); Σ 是终结符的有限集合, $N \cap \Sigma = \Phi$; $V=N \cup \Sigma$ 称总词汇表; P 是一组重写规则的有限集合: $P=\{\alpha \rightarrow \beta\}$, 其中, α, β 是 V 中元素构成的串, 但 α 中至少应含有一个非终结符号; $S \in N$, 称为句子符或初始符。

例如: $G = (\{A, S\}, \{0, 1\}, P, S)$

$P: S \rightarrow 0 A 1$

$0 A \rightarrow 00A1$

$A \rightarrow 1$

1. 形式语言

◆ 推导的定义

设 $G=(N, \Sigma, P, S)$ 是一个文法, 在 $(N \cup \Sigma)^*$ 上定义关系 \Rightarrow_G (直接派生或推导)如下:

如果 $\alpha\beta\gamma$ 是 $(N \cup \Sigma)^*$ 中的符号串, 且 $\beta \rightarrow \delta$ 是 P 的产生式, 那么 $\alpha\beta\gamma \Rightarrow_G \alpha\delta\gamma$ 。

关于闭包的解释: 假设字符集 $\Sigma=\{a, b\}$, 那么

$$\Sigma^* = \{\epsilon, a, b, aa, bb, ab, ba, aab, bba, aba, bab, aabb, \dots\}$$

$$\Sigma^+ = \{a, b, aa, bb, ab, ba, aab, bba, aba, bab, aabb, \dots\}$$

1. 形式语言

用 $\xRightarrow{+}_G$ (按非平凡方式派生) 表示 \Rightarrow_G 的传递闭包, 也就是在集合 $(N \cup \Sigma)^*$ 上的符号串 ξ_i 到 ξ_{i+1} 的 n ($n \geq 1$) 步推导或派生。

用 $\xRightarrow{*}_G$ (派生) 表示 \Rightarrow_G 的**自反**和传递闭包, 即由 $(N \cup \Sigma)^*$ 上的符号串 ξ_i 到 ξ_{i+1} 经过 n ($n \geq 0$) 步的推导或派生。

如果清楚某个推导是文法 G 所产生的, 则符号 $\xRightarrow{*}_G$ 或 $\xRightarrow{+}_G$ 中的 G 可以省略不写。

1. 形式语言

◆最左推导、最右推导和规范推导

约定每步推导中只改写最左边的那个非终结符，这种推导称为“最左推导”。

约定每步推导中只改写最右边的那个非终结符，这种推导称为“最右推导”。

最右推导也称规范推导。

1. 形式语言

例1: 给定文法 $G(S)$:

$$\textcircled{1} S \rightarrow P NP \mid PP Aux NP$$

$$\textcircled{2} PP \rightarrow P NP$$

实际不使用
这种类型的
规则。

$$\textcircled{3} NP \rightarrow NN \mid NP Aux NP$$

$$\textcircled{4} P \rightarrow \text{关于}$$

$$\textcircled{5} NN \rightarrow \text{鲁迅} \mid \text{文章}$$

$$\textcircled{6} Aux \rightarrow \text{的}$$

短语“关于鲁迅的文章”的**最右推导**。

$$\textcircled{1} S \Rightarrow P \underline{NP} \Rightarrow P NP Aux \underline{NP} \Rightarrow P NP Aux \underline{NN}$$

$$\Rightarrow P NP \underline{Aux} \text{文章} \Rightarrow P \underline{NP} \text{的} \text{文章} \Rightarrow P \underline{NN} \text{的} \text{文章}$$

$$\Rightarrow \underline{P} \text{鲁迅} \text{的} \text{文章}$$

$$\Rightarrow \text{关于鲁迅的文章}$$

1. 形式语言

例1: 给定文法 $G(S)$:

- | | |
|-------------------------------------------------------|-------------------------------------|
| ① $S \rightarrow P \text{ NP} \mid PP \text{ Aux NP}$ | ② $PP \rightarrow P \text{ NP}$ |
| ③ $NP \rightarrow NN \mid NP \text{ Aux NP}$ | ④ $P \rightarrow \text{关于}$ |
| ⑤ $NN \rightarrow \text{鲁迅} \mid \text{文章}$ | ⑥ $\text{Aux} \rightarrow \text{的}$ |

短语 “关于鲁迅的文章” 的**最左**推导。

② $S \Rightarrow \underline{P} \text{ NP} \Rightarrow \text{关于} \underline{NP} \text{ Aux NP} \Rightarrow \text{关于} \underline{NN} \text{ Aux NP}$
 $\Rightarrow \text{关于鲁迅} \underline{\text{Aux}} \text{ NP} \Rightarrow \text{关于鲁迅的} \underline{NP}$
 $\Rightarrow \text{关于鲁迅的} \underline{NN}$
 $\Rightarrow \text{关于鲁迅的文章}$

1. 形式语言

◆句型与句子

一些特殊类型的符号串为文法 $G=(N, \Sigma, P, S)$ 的句子形式(句型):

- (1) S 是一个句子形式;
- (2) 如果 $\alpha\beta\gamma$ 是一个句子形式, 且 $\beta \rightarrow \delta$ 是 P 的产生式, 则 $\alpha\delta\gamma$ 也是一个句子形式;

文法 G 的不含非终结符的句子形式称为 G 生成的句子。

1. 形式语言

在例1中，下面是“关于鲁迅的文章”的最右推导：

① $S \Rightarrow P NP \Rightarrow P NP Aux NP \Rightarrow P NP Aux NN$
 $\Rightarrow P NP Aux \text{ 文章} \Rightarrow P NP \text{ 的 文章}$
 $\Rightarrow P NN \text{ 的 文章} \Rightarrow P \text{ 鲁迅 的 文章}$
 $\Rightarrow \text{关于鲁迅的文章}$

句型 \rightarrow $P NP Aux NN$
 \rightarrow $P NP Aux \text{ 文章}$
 \rightarrow $P NN \text{ 的 文章}$
 \rightarrow 关于鲁迅的文章 句子

1. 形式语言

◆文法生成的语言

由文法 G 生成的语言是指 G 生成的所有句子的集合，记作 $L(G)$ ，即 $L(G) = \{x \mid x \in \Sigma^+, S \xrightarrow{+}_G x\}$ 。

对于例1给定的文法 $G(S)$ ：

- | | |
|-------------------------------------------------------|-------------------------------------|
| ① $S \rightarrow P \text{ NP} \mid PP \text{ Aux NP}$ | ② $PP \rightarrow P \text{ NP}$ |
| ③ $NP \rightarrow NN \mid NP \text{ Aux NP}$ | ④ $P \rightarrow \text{关于}$ |
| ⑤ $NN \rightarrow \text{鲁迅} \mid \text{文章}$ | ⑥ $\text{Aux} \rightarrow \text{的}$ |

$L(G) = \{\text{关于}(\text{鲁迅的})^n \text{文章}, \text{关于}(\text{文章的})^m \text{鲁迅}, \text{关于鲁迅}, \text{关于文章}, \text{关于}(\text{文章的})^k \text{文章}, \text{关于}(\text{鲁迅的})^l \text{鲁迅}, \text{关于的鲁迅}, \text{关于的文章}, \text{关于鲁迅的文章的} \dots \text{鲁迅} \mid \text{文章}, \dots\}$

n, m, k, l 均为大于等于1的自然数。

1. 形式语言

◆ 乔姆斯基4类文法

- 正则文法(regular grammar, RG), 或称 3型文法
- 上下文无关文法(context-free grammar, CFG), 或称2型文法
- 上下文有关文法(context-sensitive grammar, CSG), 或称1型文法
- 无约束文法(unrestricted grammar), 或称0型文法

1. 形式语言

◆正则文法

如果文法 $G=(N, \Sigma, P, S)$ 的 P 中的规则满足如下形式: $A \rightarrow Bx$, 或 $A \rightarrow x$, 其中 $A, B \in N$, $x \in \Sigma$, 则称该文法为正则文法或称 3型文法。(左线性正则文法)

如果 $A \rightarrow xB$, 则该文法称为右线性正则文法。

例2: $G = (N, \Sigma, P, S)$, $N = \{S, A, B\}$, $\Sigma = \{a, b\}$,

P : (a) $S \rightarrow aA$ (b) $A \rightarrow aA$ (c) $A \rightarrow b b B$
(d) $B \rightarrow b B$ (e) $B \rightarrow b$

$A \rightarrow b B'$
 $B' \rightarrow b B$

是正则文法吗? 请写出该文法生成的语言集合。

No

$L(G) = \{a^n b^m\}, n \geq 1, m \geq 3$

1. 形式语言

◆上下文无关文法

如果 P 中的规则满足如下形式: $A \rightarrow \alpha$, 其中 $A \in N$, $\alpha \in (N \cup \Sigma)^*$, 则称该文法为上下文无关文法 或称 2 型文法。

例3: $G = (N, \Sigma, P, S)$, $N = \{S, A, C\}$, $\Sigma = \{a, b, c\}$,

P : (1) $S \rightarrow A b C$ (2) $A \rightarrow a A \mid a$

(3) $C \rightarrow c C \mid c$

是上下文无关文法吗? **Yes**

请写出该文法生成的语言集合。

$$L(G) = \{a^n b c^k\}, n \geq 1, k \geq 1\}$$

1. 形式语言

◆CFG产生的语言句子的派生树表示

派生树也叫生成树，是用树状图表示的节点关系图。

CFG $G=(N, \Sigma, P, S)$ 产生一个句子的派生树由如下步骤构成：

- (1) 对于 $\forall x \in N \cup \Sigma$ 给一个标记作为节点, S 作为树的根节点。
- (2) 如果一个节点的标记为 A ，并且它至少有一个除它自身以外的后裔，则 $A \in N$ 。
- (3) 如果一个节点的标记为 A ，它的 k ($k > 0$) 个直接后裔节点按从左到右的次序依次标记为 A_1, A_2, \dots, A_k ，则 $A \rightarrow A_1 A_2 \dots A_k$ 一定是 P 中的一个产生式。

1. 形式语言

再看例1中对短语“关于鲁迅的文章”的最右推导：

- | | |
|--------------------------------------------|------------------------------|
| ① $S \rightarrow P\ NP\ \ PP\ Aux\ NP$ | ② $PP \rightarrow P\ NP$ |
| ③ $NP \rightarrow NN\ \ NP\ Aux\ NP$ | ④ $P \rightarrow \text{关于}$ |
| ⑤ $NN \rightarrow \text{鲁迅}\ \ \text{文章}$ | ⑥ $Aux \rightarrow \text{的}$ |

$S \Rightarrow P\ NP \Rightarrow P\ NP\ Aux\ NP \Rightarrow P\ NP\ Aux\ NN$

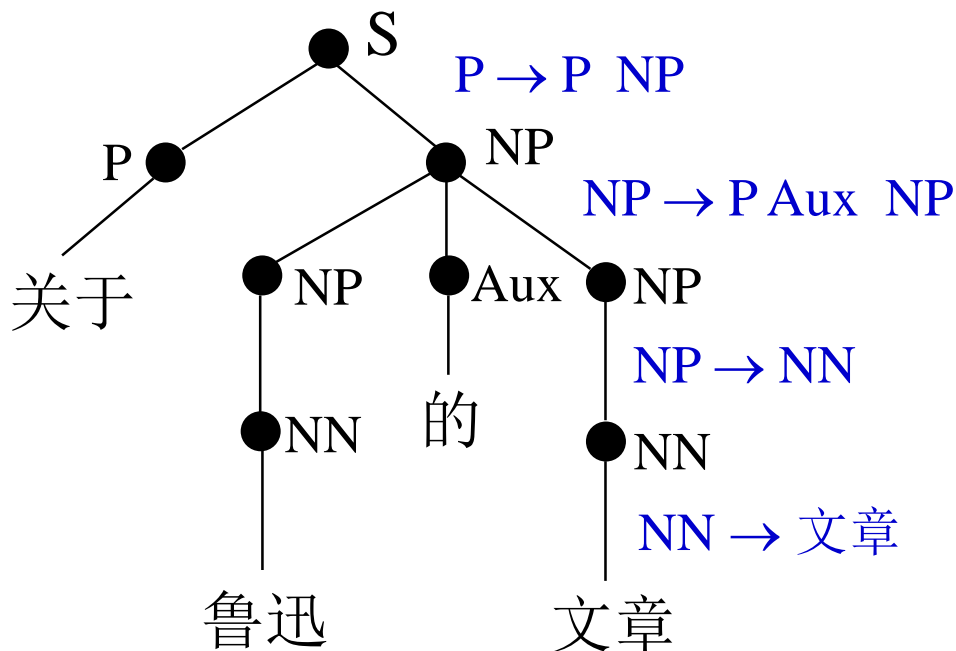
$\Rightarrow P\ NP\ Aux\ \text{文章} \Rightarrow P\ NP\ \text{的}\ \text{文章} \Rightarrow P\ NN\ \text{的}\ \text{文章}$

$\Rightarrow P\ \text{鲁迅}\ \text{的}\ \text{文章}$

$\Rightarrow \text{关于鲁迅的文章}$

1. 形式语言

短语“关于鲁迅的文章”的派生树：



短语的派生树一①

1. 形式语言

◆ 给定CFG，一个句子的派生树是否唯一呢？

再看例1，如果选择不同的规则对短语“关于鲁迅的文章”进行最右推导：

$$\textcircled{1} \underline{S} \rightarrow P \text{ NP} \mid \underline{PP} \text{ Aux NP}$$

$$\textcircled{2} PP \rightarrow P \text{ NP}$$

$$\textcircled{3} NP \rightarrow NN \mid NP \text{ Aux NP}$$

$$\textcircled{4} P \rightarrow \text{关于}$$

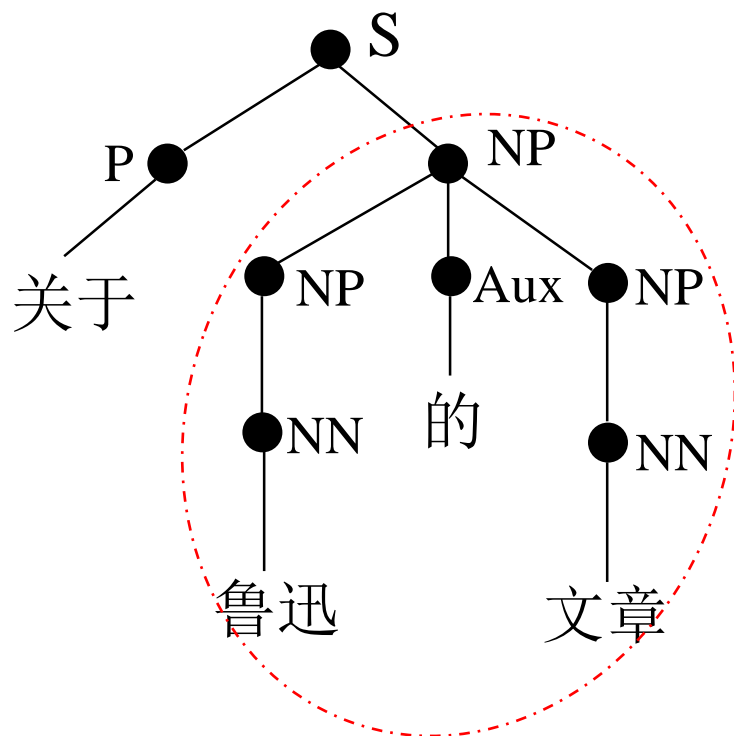
$$\textcircled{5} NN \rightarrow \text{鲁迅} \mid \text{文章}$$

$$\textcircled{6} \text{Aux} \rightarrow \text{的}$$

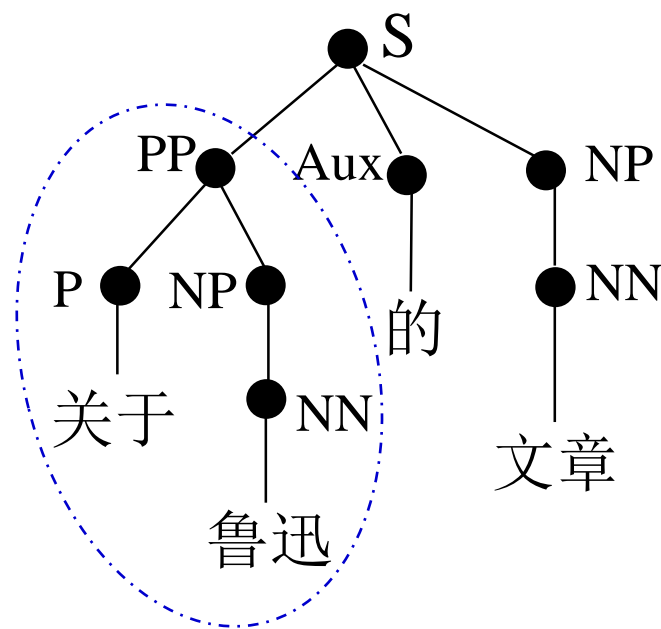
$$\begin{aligned} S &\Rightarrow PP \text{ Aux } \underline{NP} \Rightarrow PP \text{ Aux } \underline{NN} \Rightarrow PP \underline{\text{Aux}} \text{ 文章} \\ &\Rightarrow \underline{PP} \text{ 的 文章} \Rightarrow P \underline{NP} \text{ 的 文章} \Rightarrow P \underline{NN} \text{ 的 文章} \\ &\Rightarrow \underline{P} \text{ 鲁迅 的 文章} \\ &\Rightarrow \text{关于鲁迅的文章} \end{aligned}$$

1. 形式语言

短语“关于鲁迅的文章”的派生树：



短语的派生树—①



短语的派生树—②

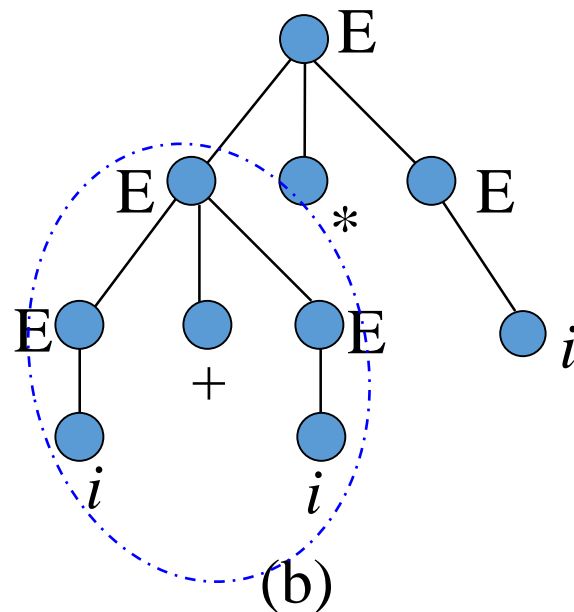
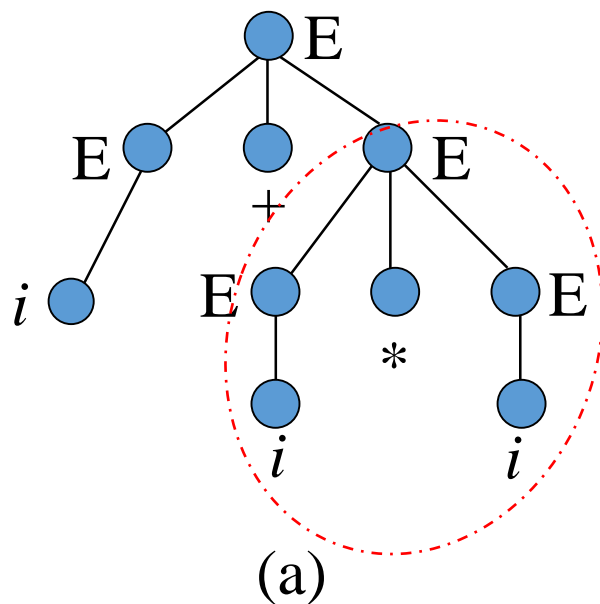
一个文法 CFG，如果存在某个句子有不只一棵分析树与之对应，那么称这个文法是二义的，或称歧义(ambiguous)文法。

1. 形式语言

请看如下文法:

$$G(E): E \rightarrow E + E \mid E * E \mid (E) \mid E - E \mid i$$

请画出句子 $i + i * i$ 对应的分析树。



1. 形式语言

◆上下文有关文法

如果 P 中的规则满足如下形式: $\alpha A \beta \rightarrow \alpha \gamma \beta$, 其中 $A \in N$, $\alpha, \beta, \gamma \in (N \cup \Sigma)^*$, 且 γ 至少包含一个字符, 则称该文法为上下文有关文法 或称 **1 型文法**。

或定义: if $x \rightarrow y$, $x \in (N \cup \Sigma)^+$, $y \in (N \cup \Sigma)^*$, 并且 $|y| \geq |x|$ 。

例4: $G = (N, \Sigma, P, S)$, $N = \{S, A, B, C\}$, $\Sigma = \{a, b, c\}$,

P : (a) $S \rightarrow A B C$ (b) $A \rightarrow a A \mid a$

(c) $B \rightarrow b B \mid b$ (d) $B C \rightarrow B c c$

是上下文有关文法吗? **Yes** 请写出该文法生成的语言集合。

$$L(G) = \{a^n b^m c^2\}, n \geq 1, m \geq 1$$

1. 形式语言

◆无约束文法(无限制重写系统)

如果 P 中的规则满足如下形式: $\alpha \rightarrow \beta$, α, β 是字符串, 则称 G 为**无约束文法**, 或称 **0 型文法**。

◆各类文法及生成语言集合之间的关系

每一个正则文法(3型)都是上下文无关文法(2型), 每一个上下文无关文法都是上下文有关文法(1型), 而每一个上下文有关文法都是无约束文法(0型), 即:

$$L(G_0) \supseteq L(G_1) \supseteq L(G_2) \supseteq L(G_3)。$$

如果一种语言能由几种文法所产生, 则把这种语言称为在这几种文法中受限制最多的那种文法所产生的语言。

本章内容

1. 形式语言

 2. 自动机

3. 自动机在NLP中的应用

4. 习题

5. 附录

2. 自动机

自动机是有穷地表示无穷语言的另一种方法。每一个语言的句子都能被某种自动机所接受。

- ◆ 有限(状态)自动机(finite automata, FA)
- ◆ 下推自动机(push-down automata, PDA)
- ◆ 线性带限自动机(linear bounded automata)
- ◆ 图灵机(Turing Machine)

2. 自动机

◆ 自动机与形式文法的对应关系

形式语言的类型	自动机类型
0 型	图灵机
1 型	线性带限自动机
2 型	下推自动机
3 型	有限状态自动机

- 确定性有限自动机(deterministic finite automata, DFA)
- 非确定性有限自动机(non-deterministic finite automata, NFA)

2. 自动机

● 确定的有限自动机(DFA)

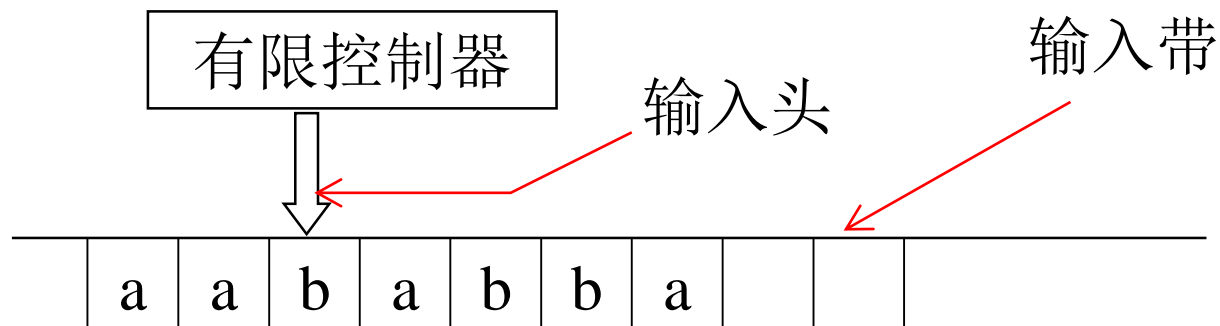
确定的有限自动机 M 是一个5元组:

$$M = (\Sigma, Q, \delta, q_0, F)$$

- Σ 是输入符号的有穷集合;
- Q 是状态的有限集合;
- $q_0 \in Q$ 是初始状态;
- F 是终止状态集合, $F \subseteq Q$;
- δ 是 Q 与 Σ 的直积 $Q \times \Sigma$ 到 Q (下一个状态) 的映射。它支配着有限状态控制的行为, 有时也称为状态转移函数。

2. 自动机

● DFA示意图

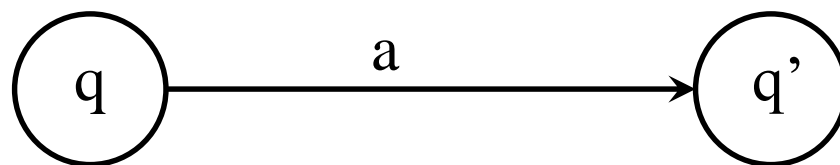


处在状态 $q \in Q$ 中的有限控制器从左到右依次从输入带上读入字符。开始时有限控制器处在状态 q_0 ，并注视 Σ^* 中一个链的最左符号。映射 $\delta(q, a) = q'$ ($q, q' \in Q, a \in \Sigma$)表示在状态 q 时，若输入符号为 a ，则自动机进入状态 q' 并且将输入头向右移动一个字符。

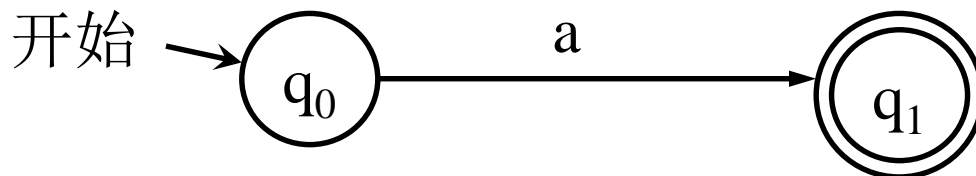
2. 自动机

● 状态变换图

映射 $\delta(q, a) = q'$ 可以由状态变换图描述。



为了明确起见, 终止状态用双圈表示, 起始状态用有“开始”标记的箭头表示。如:



2. 自动机

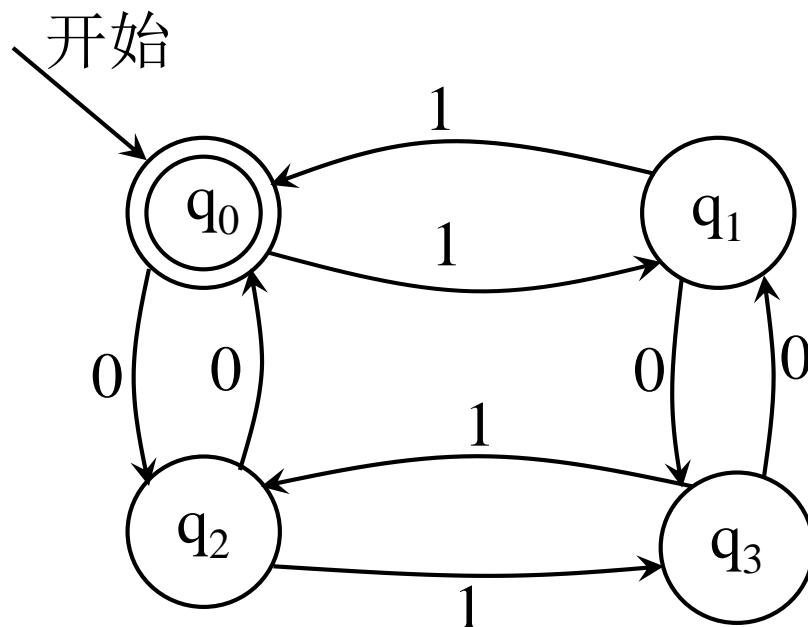
● DFA定义的语言

如果一个句子 x 使得有限自动机 M 有 $\delta(q_0, x) = p$, $p \in F$, 那么, 称句子 x 被 M 接受。由 M 定义的语言 $T(M)$ 就是被 M 接受的句子的全集。即:

$$T(M) = \{ x \mid \delta(q_0, x) \in F \}$$

2. 自动机

例5:



链 $x = 110101$ 是否被该 M 接受? **Yes!**

$T(M) = \{ \text{含偶数个0和偶数个1的链} \}$

2. 自动机

● 非确定的有限自动机(NFA)

NFA M 是一个5元组:

$$M = (\Sigma, Q, \delta, q_0, F)$$

其中, Σ 是输入符号的有穷集合;

Q 是状态的有限集合; $q_0 \in Q$ 是初始状态;

F 是终止状态集合, $F \subseteq Q$;

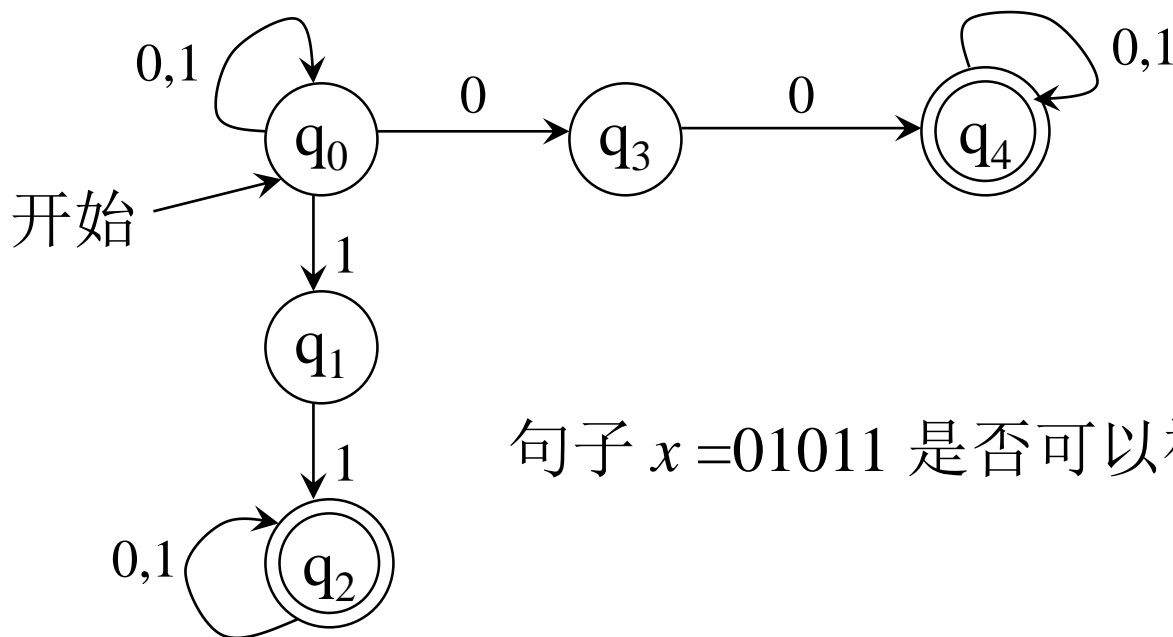
δ 是 Q 与 Σ 的直积 $Q \times \Sigma$ 到 Q 的幂集 2^Q 的映射。

2. 自动机

● NFA 与 DFA 的区别

NFA 与 DFA 的唯一区别是：在 NFA 中 $\delta(q, a)$ 是一个状态集合，而在 DFA 中 $\delta(q, a)$ 是一个状态。

例6:



句子 $x = 01011$ 是否可以被接受？

Yes!

2. 自动机

● NFA 与 DFA 的联系

定理 3.1: 设 L 是一个被 NFA 所接受的句子的集合, 则存在一个 DFA, 它能够接受 L 。

(证明略, 数学归纳法)

说明: 由于 DFA 与 NFA 所接受的是同样的链集, 所以一般情况下无需区分它们, 二者统称为有限自动机(FA)。

2. 自动机

- 正则文法与FA的关系

定理 3.2: 若 $G = (V_N, V_T, P, S)$ 是一个正则文法, 则存在一个有限自动机 $M = (\Sigma, Q, \delta, q_0, F)$, 使得: $T(M) = L(G)$ 。

2. 自动机

● 由正则文法构造FA的一般步骤(5步):

- (1) 令 $\Sigma = V_T$, $Q = V_N \cup \{ T \}$, $q_0 = S$, 其中, T 是一个新增加的非终结符。原文法的非终结符加上 T 作为有限状态机的状态节点, 原文法的终结符作为有向弧上的字符。
- (2) 如果在 P 中有产生式 $S \rightarrow \varepsilon$, 则
终止状态集合 $F = \{ S, T \}$, 否则 $F = \{ T \}$ 。
- (3) 如果在 P 中有产生式 $A \rightarrow a$, $A \in V_N$, $a \in V_T$, 则
 $T \in \delta(A, a)$ 。
- (4) 如果在 P 中有产生式 $A \rightarrow aB$, $A, B \in V_N$, $a \in V_T$, 则
 $B \in \delta(A, a)$ 。
- (5) 对于每一个非终结符 $a \in V_T$, 有 $\delta(T, a) = \phi$ 。

2. 自动机

例7: 给定正则文法 $G=(V_N, V_T, P, S)$, 其中, $V_N=\{S, B\}$,
 $V_T=\{a, b\}$, $P=\{S \rightarrow aB, B \rightarrow bS|aB|a\}$
 构造与 G 等价的 FA。

(1) 设 FA $M=(\Sigma, Q, \delta, q_0, F)$, 根据上述构造步骤有:

$$\Sigma = V_T = \{a, b\}$$

$$Q = V_N \cup \{T\} = \{S, B, T\}$$

$$q_0 = S$$

$$F = \{T\}$$

2. 自动机

(2) 映射 δ 为: $\delta(S, a) = \{B\}$ (因为有规则 $S \rightarrow aB$)

$$\delta(S, b) = \phi$$

$$\delta(B, a) = \{B, T\} \quad (\text{因为有 } B \rightarrow aB, B \rightarrow a)$$

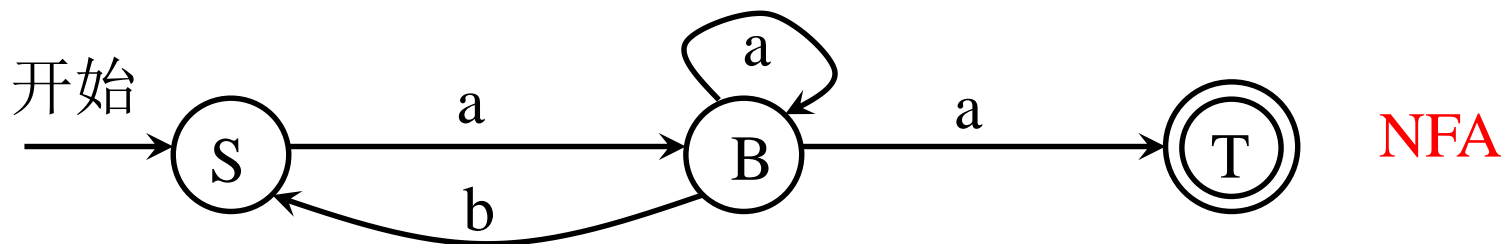
$$\delta(B, b) = \{S\} \quad (\text{因为有 } B \rightarrow bS)$$

$$\delta(T, a) = \phi$$

$$\delta(T, b) = \phi$$

$$P = \{S \rightarrow aB, B \rightarrow bS | aB | a\}$$

等价的 FA 的状态变换图为:



2. 自动机

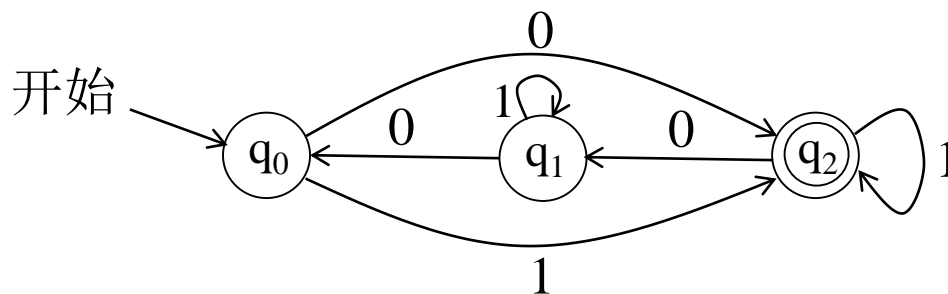
定理 3.3: 若 $M = (\Sigma, Q, \delta, q_0, F)$ 是一有限自动机, 则存在正则文法 $G = (V_N, V_T, P, S)$ 使 $L(G) = T(M)$ 。

● 由 M 构造 G 的一般步骤:

- (1) 令 $V_N = Q$, $V_T = \Sigma$, $S = q_0$;
- (2) 如果 $A \in \delta(B, a)$, $A, B \in Q$, $a \in \Sigma$, 则在 P 中有产生式
 $B \rightarrow aA$;
- (3) 如果 $C \in \delta(A, a)$, $C \in F$, 则在 P 中有产生式
 $A \rightarrow a$ 。

2. 自动机


例8：给定如下自动机：



请写出对应的正则文法。

$$\begin{aligned} G=(V_N, V_T, P, S): \quad & S = q_0, \quad A=q_1, \quad B=q_2 \\ & V_N=\{S, A, B\}, \quad V_T=\{0, 1\} \\ P: \quad & S \rightarrow 0B \mid 1B \\ & A \rightarrow 0S \mid 1A \\ & B \rightarrow 1B \mid 0A \end{aligned}$$

本章内容

1. 形式语言
2. 自动机
-  3. 自动机在NLP中的应用
4. 习题
5. 附录

3. 自动机在NLP中的应用

应用1：英语单词拼写检查 [Oflazer, 1996]

设 X 为拼写错误的字符串，其长度为 m ， Y 为 X 对应的正确的单词(答案)，其长度为 n 。则 X 和 Y 的编辑距离 $ed(X[m], Y[n])$ 定义为：从字符串 X 转换到 Y 需要的插入、删除、替换和交换两个相邻的基本单位(字符)的最小个数。如：

$$ed(\text{recog}\textcolor{red}{i}\text{nze}, \text{recogn}\textcolor{red}{i}\text{ze}) = 1$$

$$ed(\textcolor{red}{s}\text{ailn}, \textcolor{red}{f}\text{ailing}) = 3$$

$$ed(\text{sail}\textcolor{red}{n}, \text{sail}) = 1$$

3. 自动机在NLP中的应用

假设 $Z = z_1 z_2 \dots z_p$ 为字母表 A 上的 p 个字母构成的字符串， $Z[j]$ 表示含有 j ($j \geq 1$) 个字符的子串。 $X[m]$ 为拼写错误的字符串，其长度为 m ， $Y[n]$ 为与 X 串接近的字符串(一个候选)，其长度为 n 。则给定两个串 X 和 Y 的编辑距离 $ed(X[m], Y[n])$ 可以通过循环计算出从字符串 X 转换到 Y 需要进行插入、删除、替换和交换两个相邻的字符操作的最少次数。

构造一个确定的有限状态自动机 R :

$$R = (Q, A, \delta, q_0, F)$$

其中， Q 表示状态集； A 表示输入字符集；

$$\delta : Q \times A \rightarrow Q$$

$q_0 \in Q$ 为起始状态；

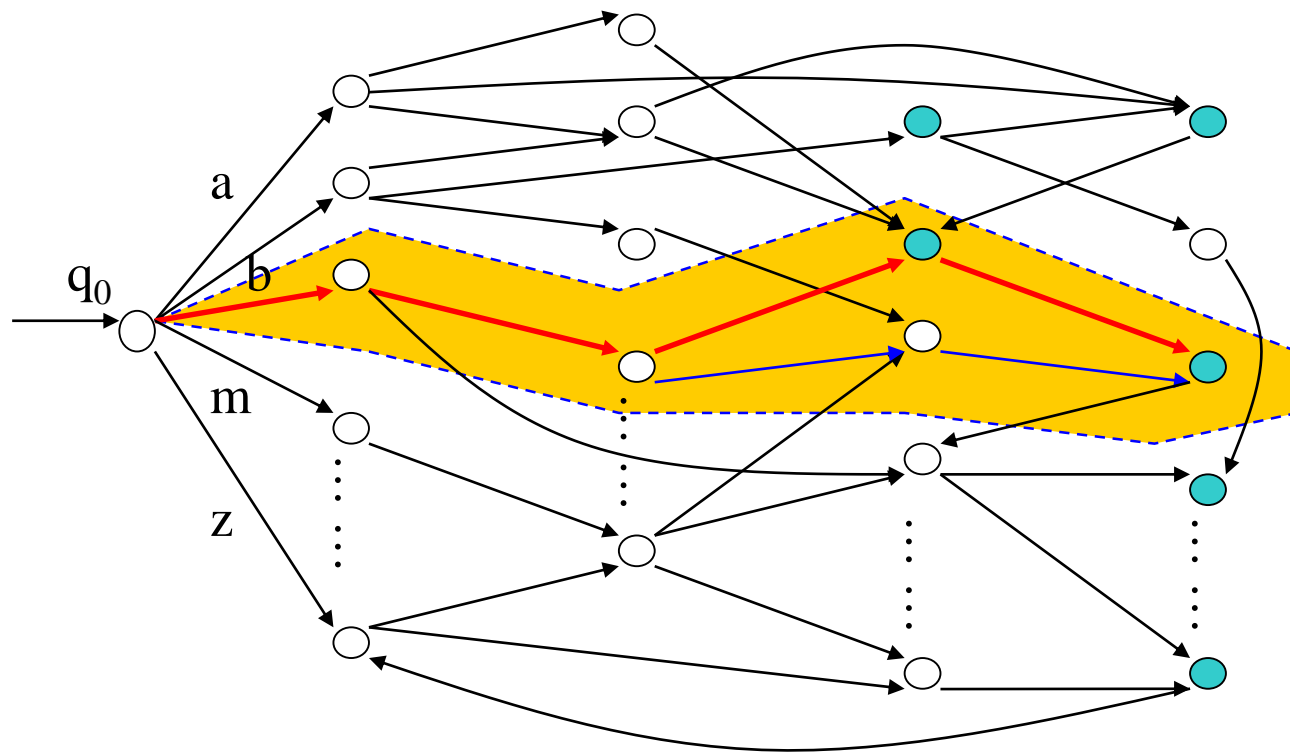
$F \subseteq Q$ 为终止状态集。

3. 自动机在NLP中的应用

如果 $L \subseteq A^*$ 表示有限状态机 R 接受的语言，字母构成的所有合法单词都是有限状态机中的一条路径。给定一个输入串，对其进行检查的过程就是在给定阈值 t ($t > 0$) 的情况下，寻找那些与输入串的编辑距离小于 t 的路径。那么，一个字符串 $X[m] \notin L$ 能够被 R 识别的条件是存在非空集合：

$$C = \{Y[n] \mid Y[n] \in L \quad \text{and} \quad ed(X[m], Y[n]) \leq t\}$$

3. 自动机在NLP中的应用



英文单词可用
键树(又称为数字查找树,
digital search trees)
存储。

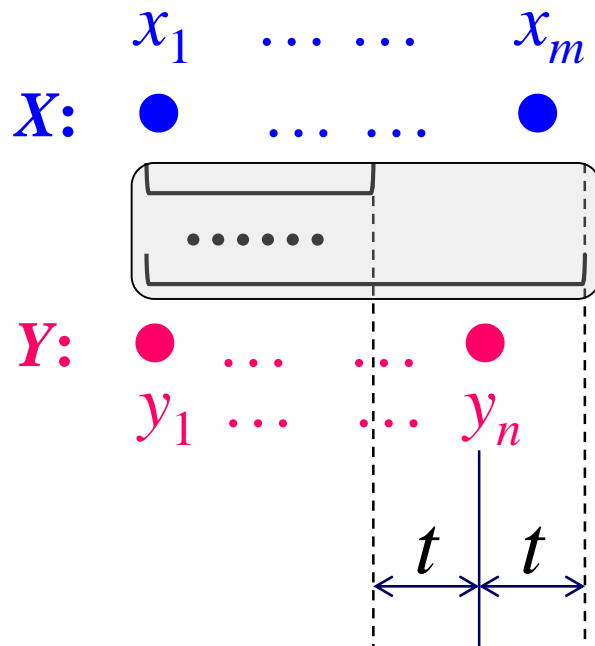
说明: 灰色节点表示终结节点, 下同。

对于某一输入的字符串 X , 搜索与之编辑距离最短的单词(路径)。

3. 自动机在NLP中的应用

定义: $cuted(X[m], Y[n]) = \min_{l \leq i \leq u} \{ed(X[i], Y[n])\}$

其中, $l = \max(1, n - t)$, $u = \min(m, n + t)$ 。



注意: 阈值 t 有两个用途:
确定截取 X 的范围; 限定编辑距离。

3. 自动机在NLP中的应用

例如: $t = 2$, $X = \text{reprter}$ ($m = 7$), $Y = \text{repo}$ ($n = 4$)

那么: $l = \max\{1, 4-2\} = 2$; $u = \min\{7, 4+2\} = 6$

$\text{cuted}(\text{reprter}, \text{repo})$

$= \min \{ \text{ed}(\text{re}, \text{repo}) = 2,$

$\text{ed}(\text{rep}, \text{repo}) = 1,$

$\text{ed}(\text{repr}, \text{repo}) = 1,$

$\text{ed}(\text{reprt}, \text{repo}) = 2,$

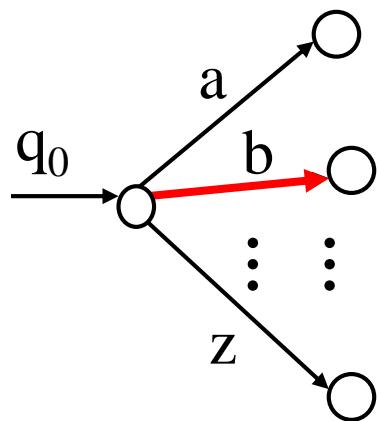
$\text{ed}(\text{reprte}, \text{repo}) = 3 \}$

$= 1$

结论: rep 和
repr 为最接近
的两个候选。

3. 自动机在NLP中的应用

采用深度优先搜索算法从自动机中选择路径。假设 $X=\text{bax}$, $t=2$ 。那么, $Y=\text{a/b/c/.../z}$, $l=\max\{1, 1-2\}=1$, $u=\min\{3, 1+2\}=3$ 。即从 X 中取长度在1~3个字符范围内的子串 $X'=\{\text{b}, \text{ba}, \text{bax}\}$, 分别计算与 Y 之间的编辑距离, 保留那些 $ed(X', Y) \leq t$ 的路径, 选择 ed 最小的路径继续扩展。

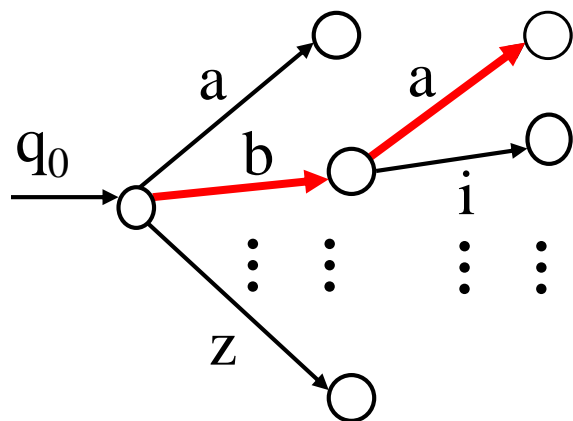


3. 自动机在NLP中的应用

$X = \text{bax}$, $t = 2$, $Y = \{\text{ba}, \text{bi}, \text{bo}, \dots\}$ 。

截取 X : $l = \max\{1, 2-2\} = 1$, $u = \min\{3, 2+2\} = 3$ 。 $X' = \{\text{b}, \text{ba}, \text{bax}\}$ 。

保留那些 $ed(X', Y) \leq t$ 的路径，选择 ed 最小的路径继续扩展。

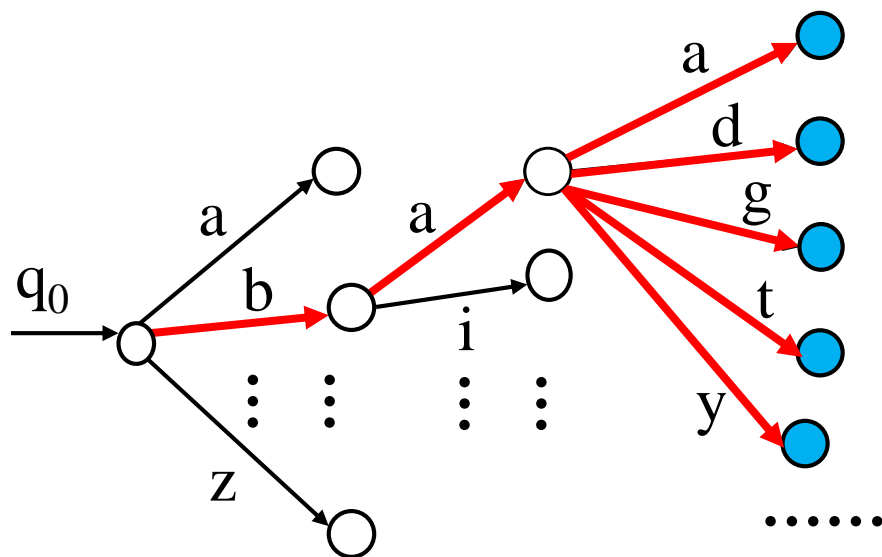


3. 自动机在NLP中的应用

$X = \text{bax}$, $t = 2$, $Y = \{\text{baa}, \text{bad}, \text{bag}, \text{bat}, \text{bay}\}$ 。

截取 X : $l = \max\{1, 3-2\} = 1$, $u = \min\{3, 3+2\} = 3$ 。 $X' = \{\text{b}, \text{ba}, \text{bax}\}$ 。

保留那些 $ed(X', Y) \leq t$ 的路径，选择 ed 最小的路径，继续扩展 Y 。

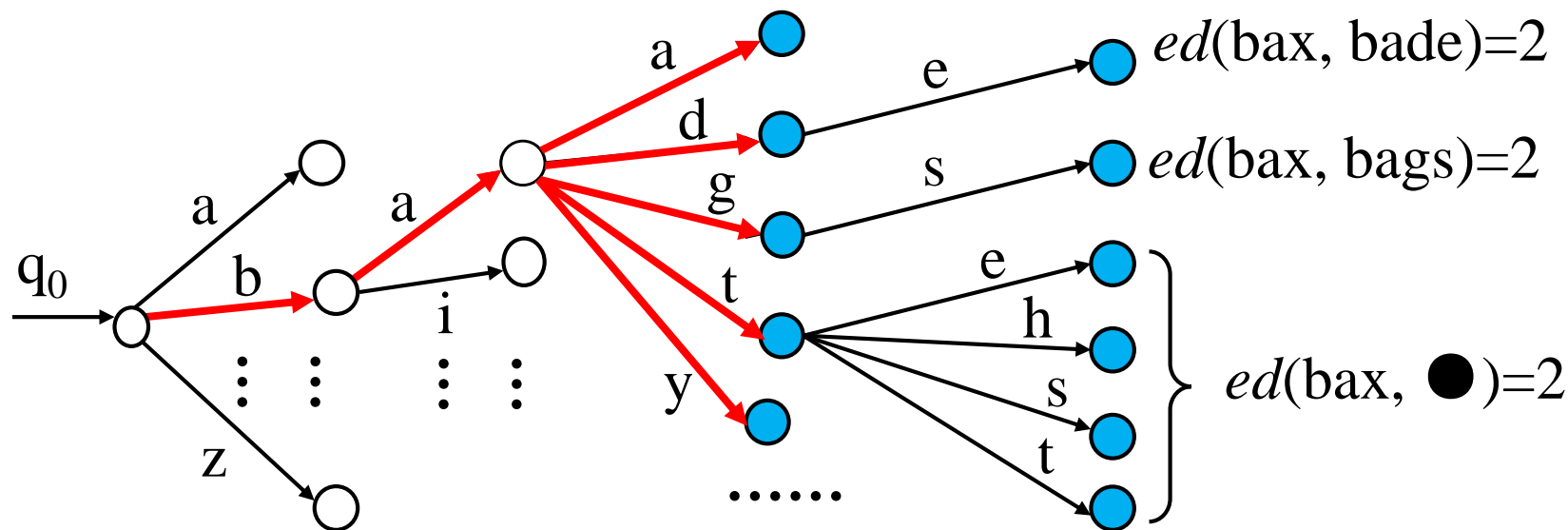


3. 自动机在NLP中的应用

$X=\text{bax}$, $t=2$, $Y=\{\text{bade}\}$ 。

截取 X : $l=\max\{1, 4-2\}=2$, $u=\min\{3, 4+2\}=3$ 。 $X'=\{\text{ba}, \text{bax}\}$ 。

保留那些 $ed(X', Y) \leq t$ 的路径, 选择 ed 最小的路径。



3. 自动机在NLP中的应用

算法描述:

```
push( $\epsilon$ ,  $q_0$ ); /* 将一个空的字符串和初始节点压栈 stack */
while stack  $\neq$  NULL {
    pop( $Y'$ ,  $q_i$ ); /* 从栈顶弹出一个部分字符串  $Y'$  和状态节点 */
    for all  $q_j$ ,  $a$ :  $\delta(q_i, a) = q_j$  {
         $Y = \text{concat}(Y', a)$ ; /* 扩展候选串, 把字符  $a$  连接到  $Y'$  上。 */
        if ( $\text{cuted}(X, Y) \leq t$ ); /* 保证剪除距离在限定的范围 */
        push( $Y$ ,  $q_j$ );
        if ( $\text{ed}(X, Y) \leq t$ ) and  $q_j \in F$ 
            output( $Y$ ); /* 输出  $Y$  */
    }
}
```

算法的具体解释
请见本章附录2。

说明: 由于该算法采用深度优先搜索策略,
因此输出结果未必是所有路径中最优的。

3. 自动机在NLP中的应用

应用2：英文单词的形态分析[Allen, 1995]

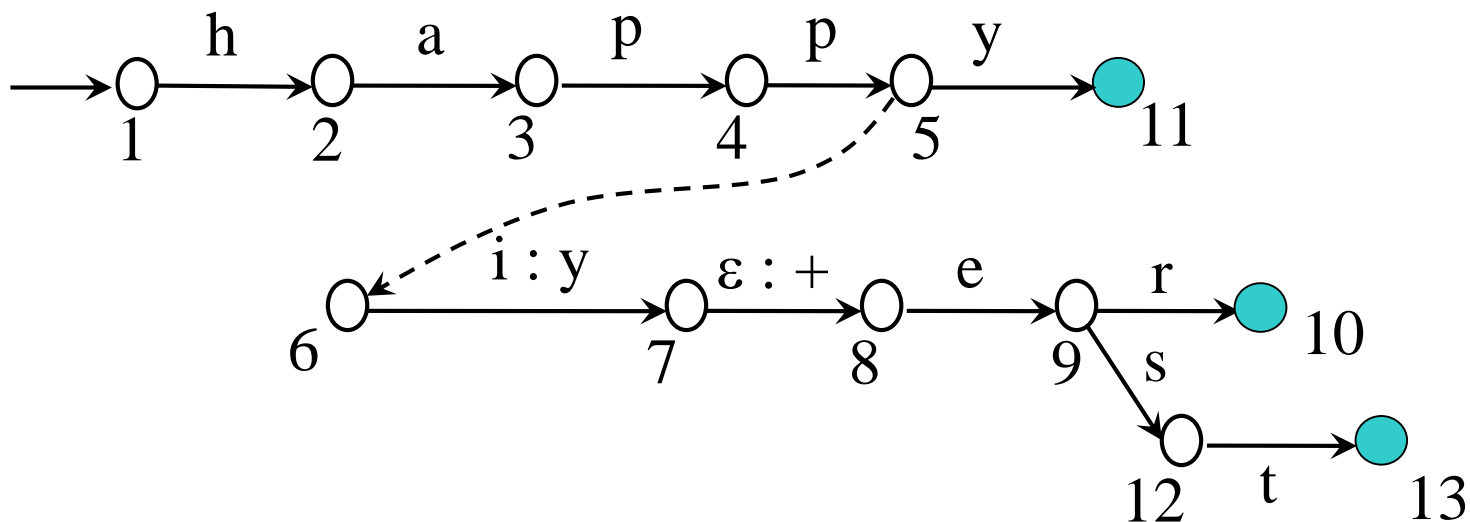
英语单词形态变化非常普遍，例如：

eat: eats, eating, ate, eaten

happy: happier, happiest

在实际应用中，除了有限状态机以外，也常使用有限状态转换机(finite state transducer, FST)的概念。粗略地讲，有限状态转换机与有限自动机(或有限状态机)的区别在于：FST 在完成状态转移的同时产生一个输出，而 FA只实现状态转移，不产生任何输出。

3. 自动机在NLP中的应用



识别单词: happy, happier, happiest

可转换的形式: happier \rightarrow happy + er


happiest \rightarrow happy + est

3. 自动机在NLP中的应用

通常具有相同的前缀或词根，词缀不同的单词可以共用一个有限状态转移机，共享其中的某些状态节点。如：tie, ties, trap, traps, try, tries, to, torch, torches, toss, tosses 等。

除了单词拼写检查、形态分析以外，有限状态自动机还曾广泛应用于词性标注、句法分析、短语识别、机器翻译甚至语音识别等很多方面。

本章内容

1. 形式语言
2. 自动机
3. 自动机在NLP中的应用
-  4. 习题
5. 附录

4. 习题

3-1. 构造上下文无关文法用以产生：

(a) 有相同数目的 0 和 1 的所有 0, 1 符号串。

(b) $\{a_1a_2...a_na_n...a_2a_1 | a_i \in \{0,1\}, 1 \leq i \leq n\}$ 。

3-2. 有以下文法： $G = (\{S,B,C\}, \{a,b,c\}, P, S)$, 其中：

$P: S \rightarrow aSBC \mid abC$

$CB \rightarrow BC$

$bB \rightarrow bb$

$bC \rightarrow bc$

$cC \rightarrow cc$

请写出 $L(G)$ 。

3-3. 编写并实现程序，模拟一个确定性的有限状态自动机。

3-4. 编写程序实现编辑距离计算方法：对于任意给定的两个相似的英语单词计算出其编辑距离。

4. 习题

3-5. 设文法 G 由如下规则定义：

$$S \rightarrow AB$$

$$A \rightarrow Aa|bB$$

$$B \rightarrow a|Sb$$

请画出下列句子形式的派生树：

(1) baabaab

(2) bBABb

3-6. 写一个程序以正则文法 G 作为输入，构造 G 相应的有限自动机。

本章小结

◆形式语言

- 4 类形式文法的定义
- 语言推导（规范推导）
- 文法的二义性

◆自动机

- 自动机的定义
- 自动机与正则文法

◆有限自动机与状态转移机的应用

- 英文单词拼写检查
- 英文单词形态分析

本章内容

1. 形式语言
2. 自动机
3. 自动机在NLP中的应用
4. 习题

 5. 附录

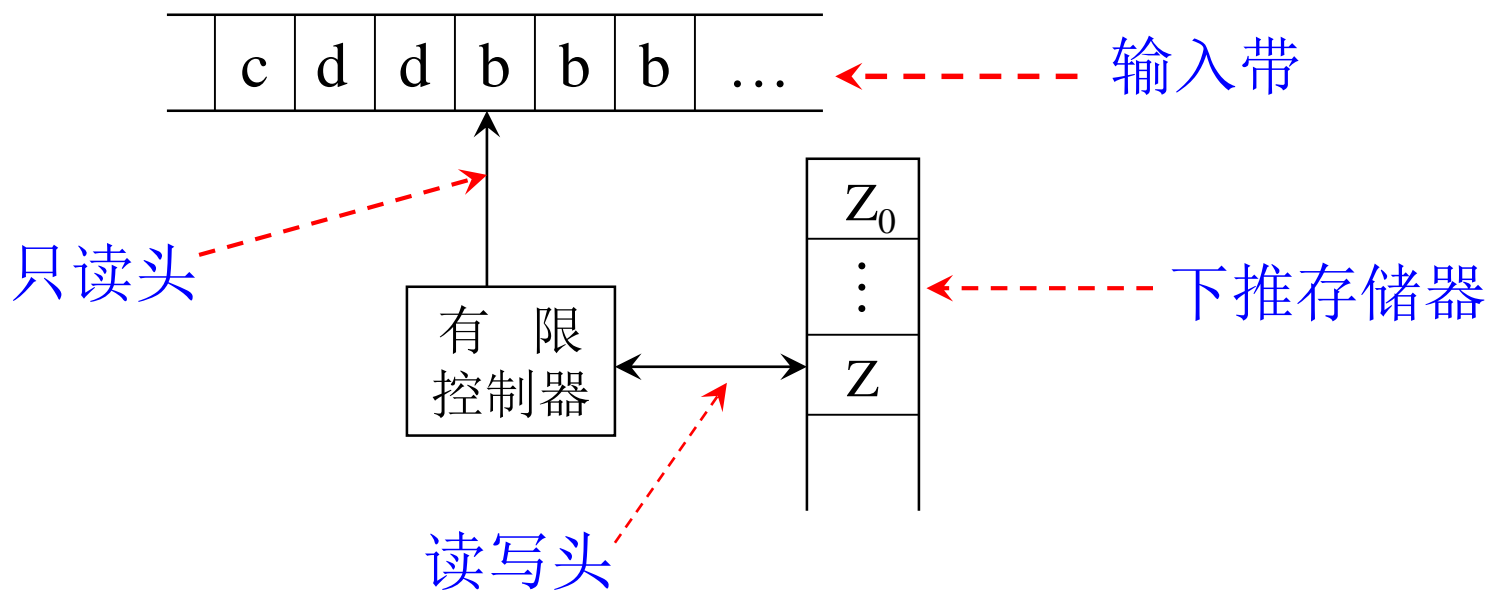
5. 附录

1. 下推自动机与CFG
2. 编辑距离计算方法的具体解释

5. 附录

1. 下推自动机(push-down automata, PDA)

PDA 可以看成是一个带有附加的下推存储器的有限自动机，下推存储器是一个栈。如下图所示：



5. 附录

● PDA 的定义

一个不确定的PDA可以表达成一个7元组：

$$M = (\Sigma, Q, \Gamma, \delta, q_0, Z_0, F)$$

其中， Σ 是输入符号的有穷集合；

Q 是状态的有限集合； $q_0 \in Q$ 是初始状态；

Γ 为下推存储器符号的有穷集合；

$Z_0 \in \Gamma$ 为最初出现在下推存储器顶端的符号；

F 是终止状态集合， $F \subseteq Q$ ；

δ 是从 $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$ 到 $Q \times \Gamma^*$ 子集的映射。

5. 附录

● 映射关系 δ 的解释

映射关系 $\delta(q, a, Z) = \{(q_1, \gamma_1), (q_2, \gamma_2), \dots, (q_m, \gamma_m)\}$

其中, $q_1, q_2, \dots, q_m \in Q, a \in \Sigma, Z \in \Gamma, \gamma_1, \gamma_2, \dots, \gamma_m \in \Gamma^*$ 。

该映射的含义是：当PDA处于状态 q ，面临输入符号 a 时，自动机将进入 $q_i, i = 1, 2, \dots, m$ 状态，并以 γ_i 来代替下推存储器(栈)顶端符号 Z ，同时将输入头指向下一个字符。当 Z 被 γ_i 取代时， γ_i 的符号按照从左到右的顺序依次从下向上推入到存储器。

特殊情况下，当 $\delta(q, \varepsilon, Z) = \{(q_1, \gamma_1), (q_2, \gamma_2), \dots, (q_m, \gamma_m)\}$ 时，输入头位置不移动，只用于处理下推存储器内部的操作，叫作“ ε 移动”。

5. 附录

● 符号约定

设有序对 (q, γ) , $q \in Q, \gamma \in \Gamma^*$, 对于 $a \in (\Sigma \cup \{\varepsilon\}), \beta \in \Gamma^*, Z \in \Gamma$
如果 $(q', \beta) \in \delta(q, a, Z), q', q \in Q$, 则表达式

$$a: (q, Z\gamma) \mid_{\overline{M}} (q', \beta\gamma)$$

表示根据下推自动机的状态变换规则, 输入 a 能使下推自动机 M 由格局 $(q, Z\gamma)$ 变换到格局 $(q', \beta\gamma)$, 或称 $a: (q, Z\gamma) \mid_{\overline{M}} (q', \beta\gamma)$ 为合法转移。零次或多次合法转移, 记为:

$$a: (q, Z\gamma) \mid_M^* (q', \beta\gamma), M \text{ 可以省略不写。}$$

5. 附录

- 下推自动机接受的语言

下推自动机 M 所接受的语言定义为:

$$T(M) = \{x | x: (q_0, Z_0) \xrightarrow{*}_M (q, \gamma), \gamma \in \Gamma^*, q \in F \}$$

Definition 1: A PDA accepts its input by consuming it and entering an accepting state.

Definition 2: The set of strings that cause the PDA to empty its stack, starting from the initial instantaneous description (ID).

5. 附录

例7: 下推自动机 $M = (\Sigma, Q, \Gamma, \delta, q_0, Z_0, F)$ 接受语言
 $L = \{wcw^R | w \in \{a, b\}^*\}$, 其中, $Q = \{0, 1\}$, $\Sigma = \{a, b, c\}$,
 $\Gamma = \{A, B\}$, $q_0 = 0$, $Z_0 = \#$, $F = \{1\}$, δ 定义如下:

- (1) $\delta(0, a, \varepsilon) \vdash_M \{(0, A)\}$ (2) $\delta(0, b, A) \vdash_M \{(0, AB)\}$
(3) $\delta(0, b, B) \vdash_M \{(0, BB)\}$ (4) $\delta(0, c, B) \vdash_M \{(1, B)\}$
(5) $\delta(1, b, B) \vdash_M \{(1, \varepsilon)\}$ (6) $\delta(1, a, A) \vdash_M \{(1, \varepsilon)\}$

更全面、通用的形式为:

$\delta(q_0, X, Y) \vdash_M \{q_0, XY\}$, 对于一切 $X \in \{a, b\}$, $Y \in \{Z, a, b\}$, $Z \in \Gamma$ 。

本例见如下参考文献:

翁富良、王野翊著: 计算语言学导论, 中国社科出版社, 1998, pages 40-42

[美]A. V. 阿霍, J. D. 厄尔曼著, 石青云译: 形式语言及其句法分析, 科学出版社, 1987, pages 200-201

5. 附录

对于输入 abbcbbba 下推自动机 M 的处理步骤为：

状态	输入	栈	运用的规则
0	abbcbbba	#	—
0	bcbba	A#	$\delta(0, a, \varepsilon) \Rightarrow \{(0, A)\}$
0	cbba	BA#	$\delta(0, b, A) \Rightarrow \{(0, AB)\}$
0	cbba	BBA#	$\delta(0, b, B) \Rightarrow \{(0, BB)\}$
1	bba	BBA#	$\delta(0, c, B) \Rightarrow \{(1, B)\}$
1	ba	BA#	$\delta(1, b, B) \Rightarrow \{(1, \varepsilon)\}$
1	a	A#	$\delta(1, b, B) \Rightarrow \{(1, \varepsilon)\}$
1	ε	#	$\delta(1, a, A) \Rightarrow \{(1, \varepsilon)\}$

5. 附录

● PDA 与 CFG 的关系

定理 3.4: 假设 $G=(V_N, V_T, P, S)$ 是 CFG, 由 G 可以构造一个 PDA M , 使 $T(M) = L(G)$ 。

定理 3.5: 假设 $M=(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ 是 PDA, 则可以构造一个 CFG G , 使 $L(G) = T(M)$ 。

两个定理的证明分别见石青云译:《形式语言及其句法分析》, 科学出版社, 1987.9, P191 和 P199。

(原著: A. V. Aho and J. D. Ullman, The Theory of Parsing, Translation, and Compiling, 1972.)

5. 附录

- 图灵机

- ✧ 图灵机与0型文法等价；

- ✧ 图灵机与有限自动机 (FA) 的区别：图灵机可以通过其读/写头改变输入带的字符。

- 线性带限自动机

- ✧ 线性带限自动机与1型文法等价；

- ✧ 线性带限自动机是一个确定的单带图灵机，其读写头不能超越原输入带上字符串的初始和终止位置，即线性带限自动机的存储空间被输入符号串的长度所限制。

5. 附录

● 各类自动机之间的主要区别

各类自动机的主要区别是它们能够使用的信息存储空间的差异：有限状态自动机只能用状态来存储信息；下推自动机除了可以用状态以外，还可以用下推存储器(栈)；线性带限自动机可以利用状态和输入/输出带本身。由于输入/输出带没有“先进后出”的限制，因此，其功能大于栈；而图灵机的存储空间没有任何限制。

5. 附录

1. 下推自动机与CFG

2. 编辑距离计算方法的具体解释

5. 附录

2. 编辑距离计算方法的具体解释

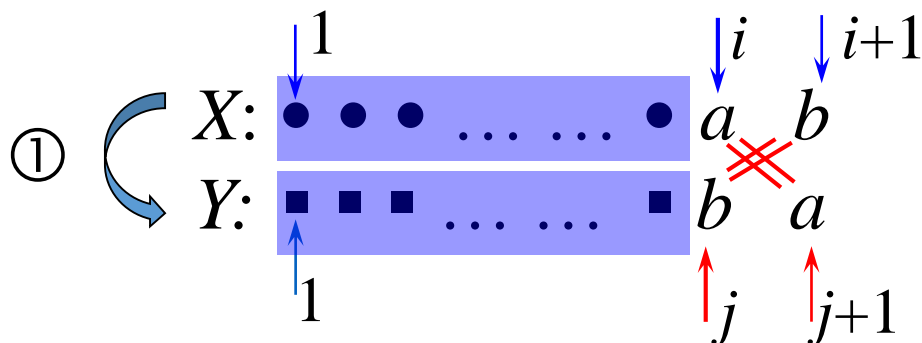
(1) 如果 $x_{i+1} = y_{j+1}$ (两个串的最后一个字母相同), 则

$$ed(X[i+1], Y[j+1]) = ed(X[i], Y[j]);$$

(2) 如果 $x_i = y_{j+1}$, 并且 $x_{i+1} = y_j$ (最后两个字符需要交换位置), 则

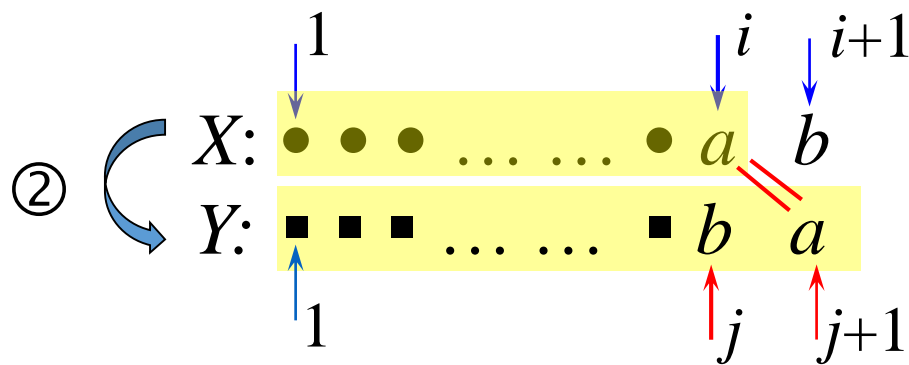
$$ed(X[i+1], Y[j+1]) = 1 + \min \{ ed(X[i-1], Y[j-1]), \\ ed(X[i], Y[j+1]), \\ ed(X[i+1], Y[j]) \}$$

5. 附录



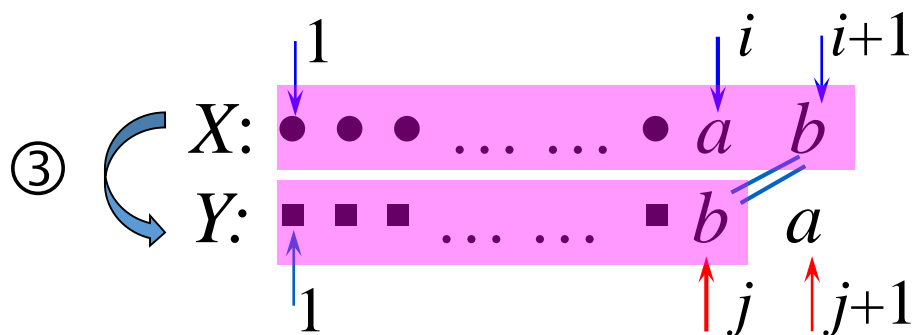
$ed(X[i-1], Y[j-1])$

(交换X中最末端的两个字符)



$ed(X[i], Y[j+1])$

(删除 x_{i+1})



$ed(X[i+1], Y[j])$

(在 x_{i+1} 之后插入 y_{j+1})

5. 附录

(3)其它情况下($x_{i+1} \neq y_{j+1}$ 且 ($x_i \neq y_{j+1}$ 或 $x_{i+1} \neq y_j$))

$$ed(X[i+1], Y[j+1]) = 1 + \min \{ ed(X[i], Y[j]), \\ ed(X[i], Y[j+1]), \\ ed(X[i+1], Y[j]) \}$$

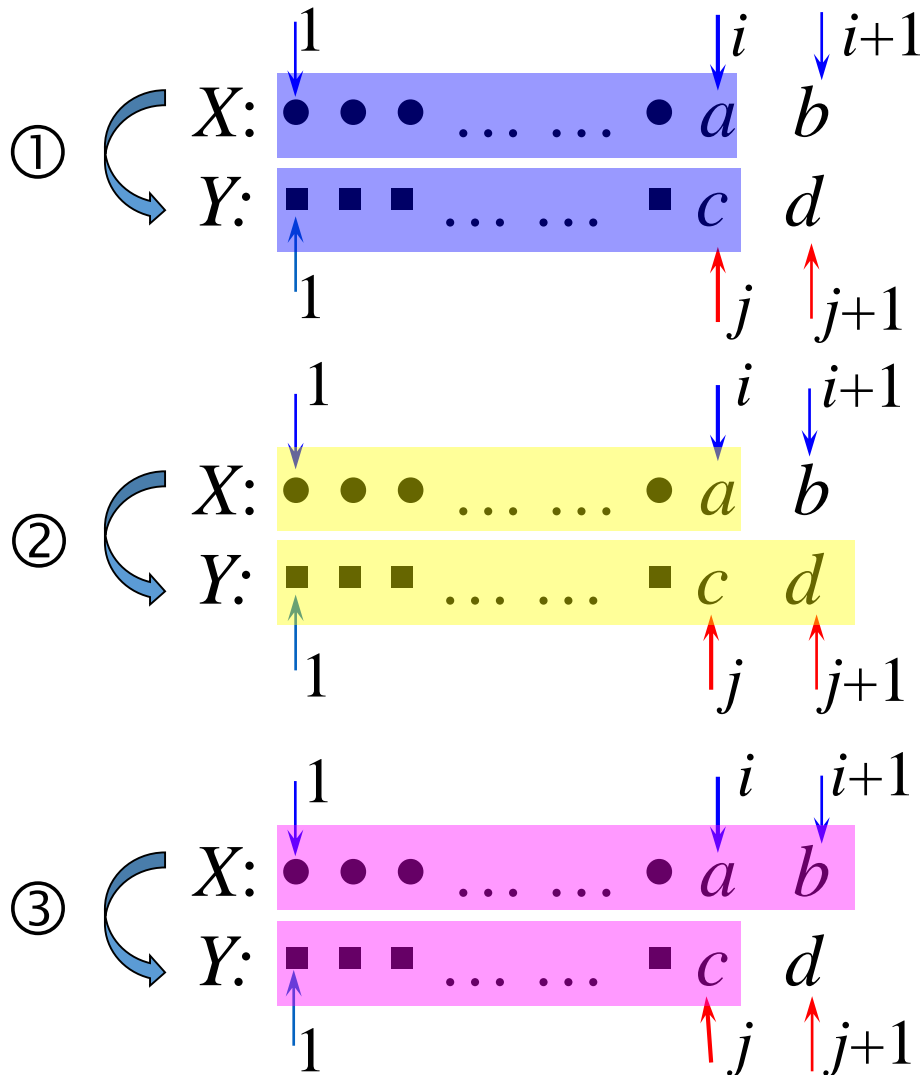
其中,

$$ed(X[0], Y[j]) = j \quad (0 \leq j \leq n) \quad (X \text{长度为} 0)$$

$$ed(X[i], Y[0]) = i \quad (0 \leq i \leq m) \quad (Y \text{长度为} 0)$$

$$ed(X[-1], Y[j]) = ed(X[i], Y[-1]) = \max\{m, n\} \\ (\text{边界约定})$$

5. 附录



$ed(X[i], Y[j])$

$(x_{i+1} \neq y_{j+1}, \text{修改 } x_{i+1})$

$ed(X[i], Y[j+1])$

$(x_{i+1} \neq y_{j+1}, \text{且 } x_{i+1} \neq y_j, \text{删除 } x_{i+1})$

$ed(X[i+1], Y[j])$

$(x_{i+1} \neq y_{j+1}, \text{且 } x_i \neq y_{j+1}, \text{在 } x_{i+1} \text{ 之后插入字符 } y_{j+1})$

5. 附录

算法具体实现时，通过构造一个 $m \times n$ 的 H 矩阵快速地计算剪除距离， H 矩阵的元素 $H(i, j) = ed(X[i], Y[j])$ 。根据前面给出的编辑距离的定义，矩阵元素 $H(i+1, j+1)$ 的计算仅递归地依赖于 $H(i, j)$ ， $H(i, j+1)$ ， $H(i+1, j)$ 和 $H(i-1, j-1)$ ，参照下面的图示：

$$\begin{matrix} & & & & & \\ & & & & & \\ & & & & & \\ m & \left(\begin{array}{ccccc} \dots & \dots & \dots & \dots & \dots \\ \dots & H(i-1, j-1) & \dots & \dots & \dots \\ \dots & \dots & H(i, j) & H(i, j+1) & \dots \\ \dots & \dots & H(i+1, j) & H(i+1, j+1) & \dots \\ \dots & \dots & \dots & \dots & \dots \end{array} \right) & \\ & & & & & \\ & & & & & n \end{matrix}$$

更详细解释请参阅：

宗成庆，统计自然语言处理，清华大学出版社，2013，Pages 45-48。

K. Oflazer. Error-tolerant Finite State Recognition with Applications to Morphological Analysis and Spelling Correction, Computational Linguistics, Vol. 22, No. 1, 1996. Pages 1-18

谢谢!

Thanks!

