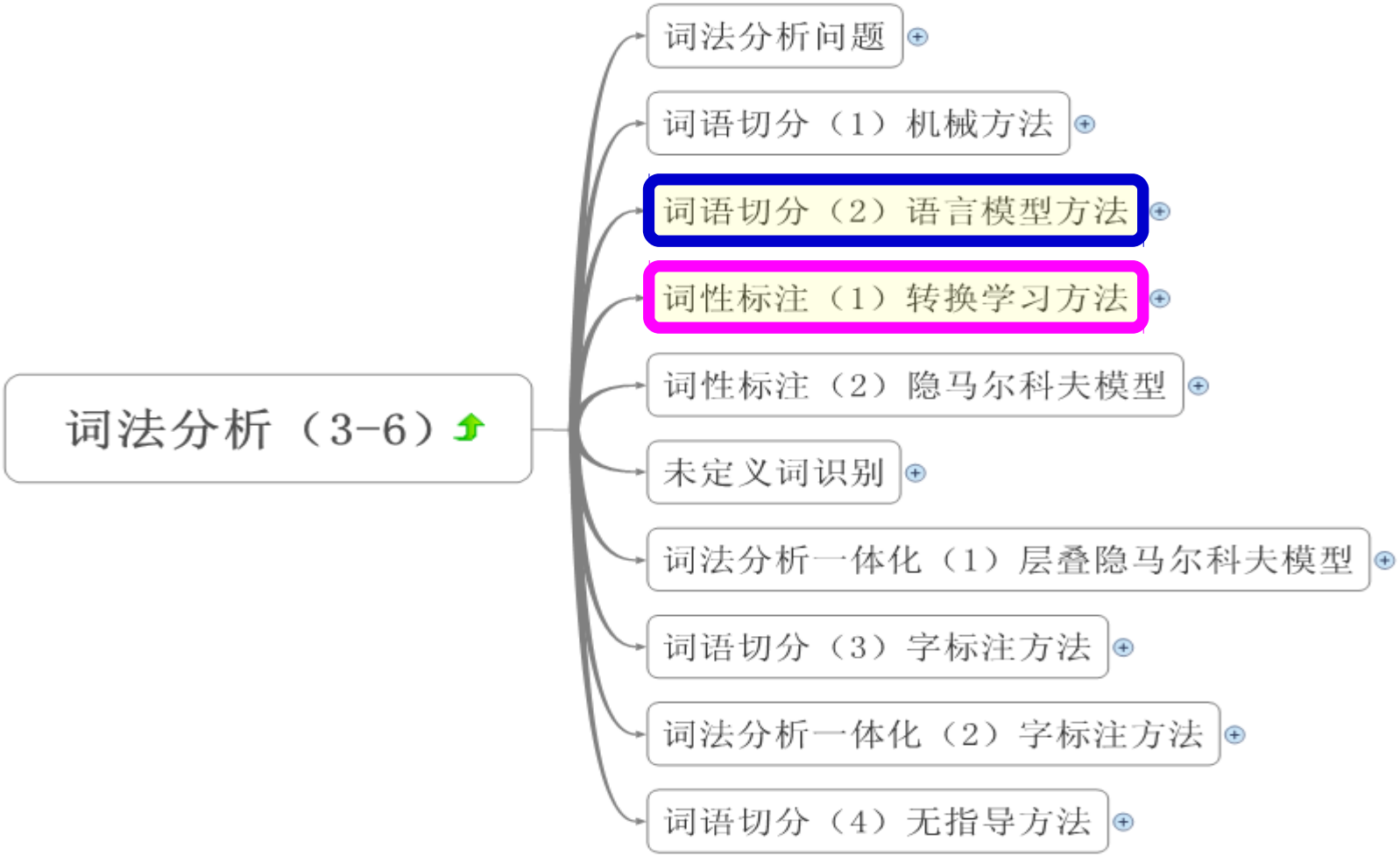
计算语言学

第 4 讲 词法分析（二）

刘群
中国科学院计算技术研究所
liuqun@ict.ac.cn

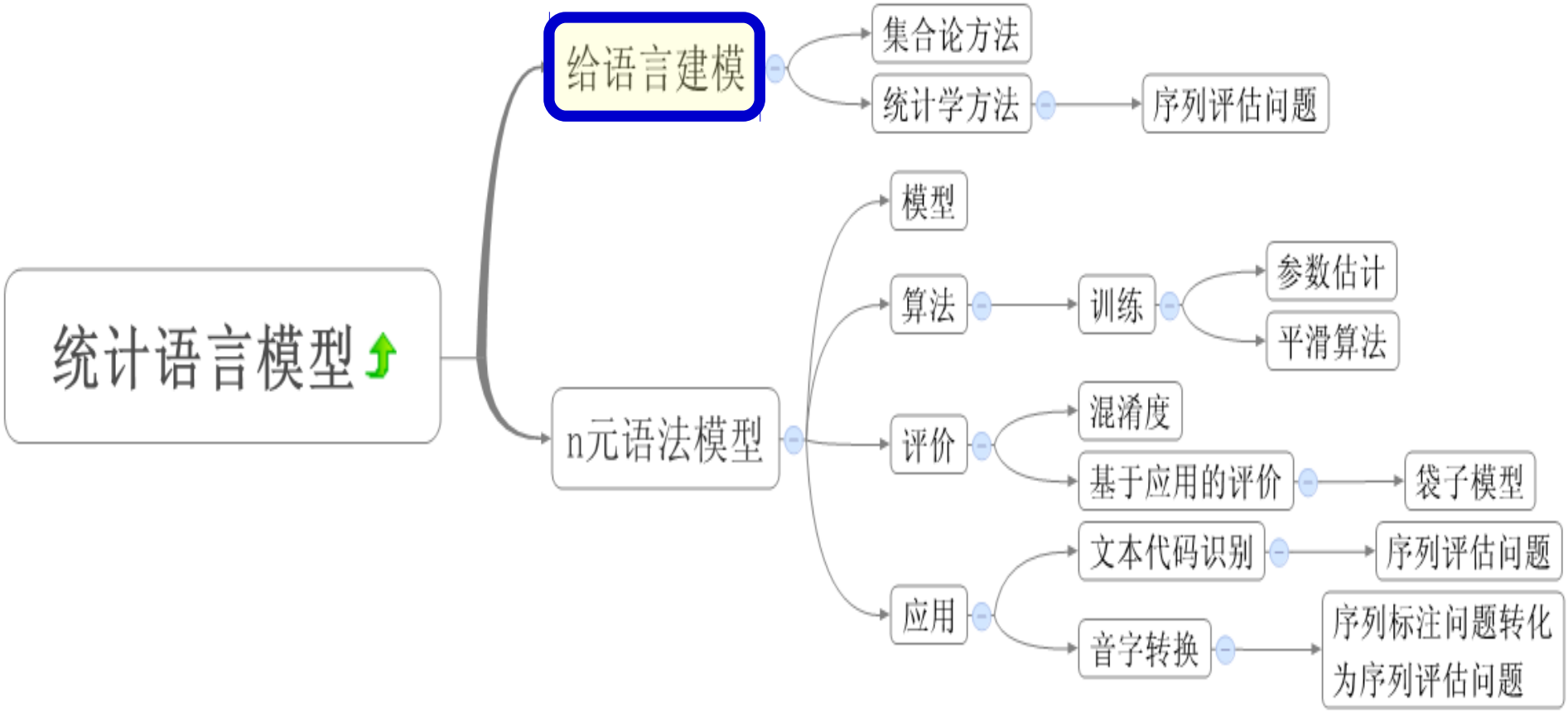
中国科学院研究生院 2012 年春季课程讲义

内容提要

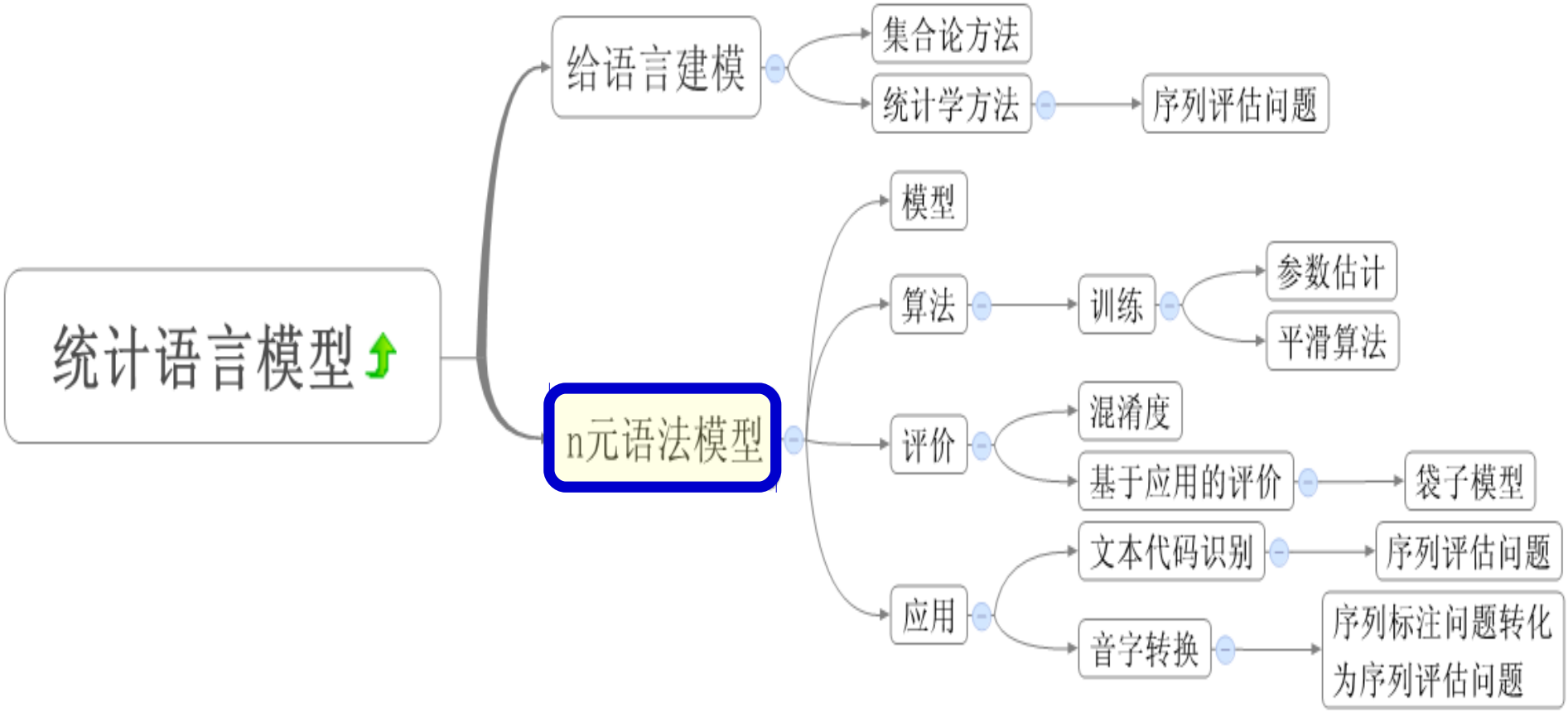

词法分析（3-6）
词法分析问题
词语切分（1）机械方法
词语切分（2）语言模型方法
词性标注（1）转换学习方法
词性标注（2）隐马尔科夫模型
未定义词识别
词法分析一体化（1）层叠隐马尔科夫模型
词语切分（3）字标注方法
词法分析一体化（2）字标注方法
词语切分（4）无指导方法

内容提要

内容提要


统计语言模型
给语言建模
集合论方法
统计学方法
序列评估问题
n元语法模型
模型
算法
训练
参数估计
平滑算法
评价
混淆度
基于应用的评价
袋子模型
应用
文本代码识别
序列评估问题
音字转换
序列标注问题转化为序列评估问题

内容提要


统计语言模型
给语言建模
集合论方法
统计学方法
序列评估问题
n元语法模型
模型
算法
训练
参数估计
平滑算法
评价
混淆度
基于应用的评价
袋子模型
应用
文本代码识别
序列评估问题
音字转换
序列标注问题转化为序列评估问题

什么是统计语言模型

- 语言模型给出任何一个句子的出现概率：

$$\Pr(Sentence=w_1 w_2 \ldots w_n)$$

归一化条件：$\sum_{Sentence}\Pr(Sentence)=1$

- 统计语言模型实际上就是一个概率分布，它给出了一种语言中所有可能的句子的出现概率
- 在统计语言模型看来，对于一种语言，任何一个句子都是可以接受的，只是接受的可能性（概率）不同
- 统计语言模型问题是一个典型的序列评估问题

语言模型的类型

- 理论上，单词串的任何一种概率分布，都是一个语言模型。
- 实际上， N 元语法模型是最简单也是最常见的语言模型。
- N 元语法模型由于没有考虑任何语言内部的结构信息，显然不是理想的语言模型。
- 其他语言模型：
  - 隐马尔科夫模型（ HMM ）（加入词性标记信息）
  - 概率上下文无关语法（ PCFG ）（加入短语结构信息）
  - 概率链语法（ Probabilistic Link Grammar ）
    （加入链语法的结构信息）
- 目前为止，其他形式的语言模型效果都不如 N 元语法模型
- 统计机器翻译研究中开始有人尝试基于句法的语言模型

N 元语法模型—概念辨析

- N 元语法模型： N-Gram Model 。
- 所谓 N-Gram ，指的是由 N 个词组成的串，可以称为“ N 元组”，或“ N 元词串”。
- 基于 N-Gram 建立的语言模型，称为“ N 元语法模型 (N-Gram Model)”。
- Gram 不是 Grammar 的简写。在英文中，并没有 N-Grammar 的说法。
- 在在汉语中，单独说“ N 元语法”的时候，有时指“ N 元组 (N-Gram)”，有时指“ N 元语法模型 (N-Gram Model)”，请注意根据上下文加以辨别。

N 元语法模型—定义

- N 元语法模型（ N-gram Model ）

$$P(w)=p(w_1)\,p(w_2|w_1)\ldots p(w_n|w_1 w_2\ldots w_{n-1})$$

$$=\prod_{i=1}^{n} p(w_i|w_1 w_2\ldots w_{i-1})$$

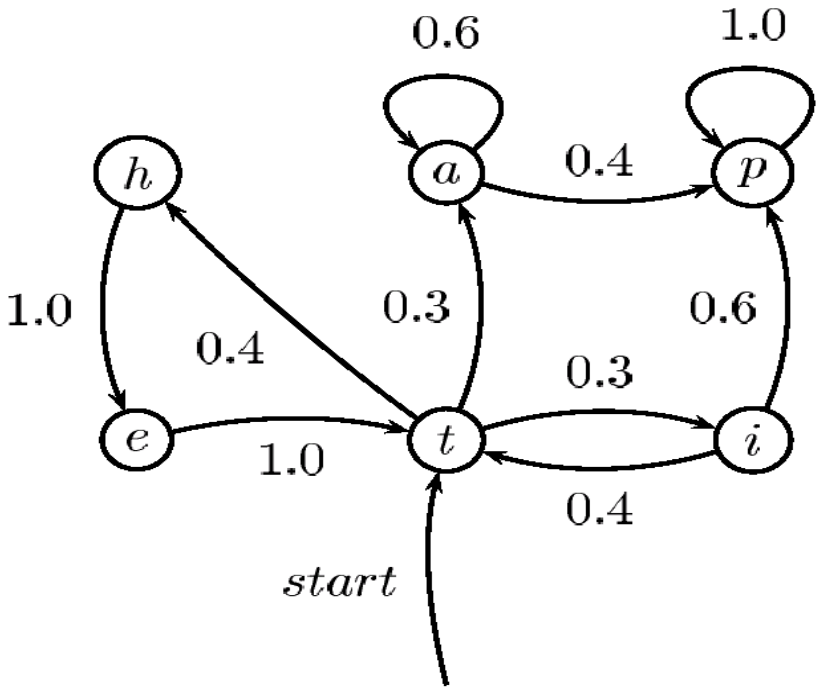
$$\approx\prod_{i=1}^{n} p(w_i|w_{i-N+1} w_{i-N+2}\cdots w_{i-1})$$

- 假设：单词 w_i 出现的概率只与其前面的 N-1 个单词有关

N 元语法模型—举例

- N=1 时：一元语法模型
  - 相当于词频表，给出所有词出现的频率
- N=2 时：二元语法模型
  - 相当于一个转移矩阵，给出每一个词后面出现另一个词的概率
- N=3 时：三元语法模型
  - 相当于一个三维转移矩阵，给出每一个词对儿后面出现另一个词的概率
- 在自然语言处理中，N 元语法模型可以在汉字层面，也可以在单词层面，还可以在概念层面……

二元语法模型—图示



$P(t\text{-}i\text{-}p) = p(X_1 = t)p(X_2 = i|X_1 = t)p(X_3 = p|X_2 = i)$

$= 1.0 \times 0.3 \times 0.6 = 0.18$

袋子模型 Bag Model (1)

- 将一个英语句子中所有的单词放入一个袋子中
- 用 N 元语法模型试图将其还原
  - 对于这些单词的任何一种排列顺序根据 N 元语法模型计算其出现概率
  - 取概率最大的排列方式

袋子模型 Bag Model (2)

- 实验：取 38 个长度小于 11 个单词的英语句子，实验结果如下：

Exact reconstruction (24 of 38)

Please give me your response as soon as possible.
⇒ Please give me your response as soon as possible.

Reconstruction preserving meaning (8 of 38)

Now let me mention some of the disadvantages.
⇒ Let me mention some of the disadvantages now.

Garbage reconstruction (6 of 38)

In our organization research has two missions.
⇒ In our missions research organization has two.

代码识别问题 (1)

- 给出一段汉语文本，需要识别出其是 GB 码还是 BIG5 码

$$
\begin{aligned}
code &= \arg\max_{code} P(code|text) \\
&= \arg\max_{code} \frac{P(text|code)P(code)}{P(text)} \\
&= \arg\max_{code} P(text|code)P(code) \\
&\approx \arg\max_{code} P(text|code)
\end{aligned}
$$

假设 GB 码的文本和 BIG 码的文本出现概率相同

代码识别问题 (2)

- 为 GB 码和 BIG5 码分别建立一元统计语言模型，也就是为两种代码分别建立字频表
- 将代码 text 按照 GB 码和 BIG5 码分别识别成不同的汉字串，计算其中所有汉字频率的乘积
- 算法的优点：简单、高效，通过很短的一段文本就可以识别出其代码类型

音字转换 (1)

- 给出一段拼音，要求转换成汉字
  pinyin = woaini
  汉字=我爱你、窝爱霓、我挨你……

$$
\begin{aligned}
text &= \arg\max_{text} P(text|pinyin) \\
&= \arg\max_{text} \frac{P(pinyin|text)P(text)}{P(pinyin)} \\
&= \arg\max_{text} P(pinyin|text)P(text) \\
&\approx \arg\max_{text} P(text)
\end{aligned}
$$

不考虑同音字，即认为 *P(pinyin|text)* 为常量

音字转换 (2)

- 一元语法模型：
  - 空间： n （汉字总数）
  - 同音字中总是会选择最高频的汉字，不合适
- 二元语法模型：
  - 空间： n^2
  - 效果比一元语法模型有较大提高
- 估计对于汉语而言四元语法模型效果较好
- 实用系统：智能狂拼，微软拼音

N 元语法模型的参数估计

- 最大似然估计：
  选择参数，使得训练语料出现的概率最大

$$p(w_n|w_1 w_2 \ldots w_{n-1}) = \frac{f(w_{1.} .. w_n)}{f(w_{1.} .. w_{n-1})}$$

用实际样本中事件出现的频率来估计该事件的概率

数据平滑

- 数据稀疏问题
  - 如果 $f(w_1 \ldots w_n) = 0$，那么出现零概率，导致整个文本的出现概率为零
- 解决办法：劫富济贫
- 约束：概率的归一性

$$\sum_{w_n} p(w_n | w_1 w_2 \ldots w_{n-1}) = 1$$

平滑算法—加 1 法

- 给每个事件出现的词数加 1

$$p(w_n|w_1 w_2 \ldots w_{n-1}) = \frac{f(w_{1.}..w_n)+1}{f(w_{1.}..w_{n-1})+m}$$

m 为单词（ word type ）的个数

- 会导致未出现的事件概率过高
- 可以将 1 改成一个更小的常数 ε

平滑算法— Good-Turing 法 (1)

- 样本数据的大小为 N
- 样本中出现 r 次的样本数为 n_r，且 r 最大值为 R，于是有：

$$\sum_{r=1}^{R} r \cdot N_r = N$$

- 估计：

$$P_r = \frac{r+1}{N} \frac{n_{r+1}}{n_r}$$

平滑算法— Good-Turing 法 (2)

- 修正：
  - n_r=0 时， P_r=0
  - n_{r+1}=0 时，用下一个不为 0 的 n_{r+k} 取代 n_{r+1}
- 直观理解：

  用出现 r+1 次的事件出现的总概率平均分配到出现 r 次的事件上面

$$P_r = \frac{r+1}{N}\frac{n_{r+1}}{n_r}$$

平滑算法— Good-Turing 法 (3)

- 优点：
  - 有很好的理论基础：理论上，可以采用留一方法（交叉检验的一种），通过最大似然估计推导出这种方法
  - 其它平滑技术的基础
- 缺点：
  - 无法保证概率估计的“有序性”，即出现次数多的事件的概率大于出现次数少的事件的概率。
  - p_r 与 r/N 不能很好地近似，好的估计应当保证 $p_r<=r/N$ 。
- 适用范围：对 $0<r<6$ 的小计数事件进行估计。

平滑算法— Good-Turing 法 (4)

- 问题：对于最高频事件（最大的 r ），由于 $N_{r+1}=0$ ，会导致 $P_r=0$ ，这显然是不合理的。
- 解决办法：
  - 只对低频事件进行平滑（比如说 $r<10$ ），对高频时间不做平滑。为保持概率的归一性，需要进行再次归一化操作。
  - 对最高频事件（ $r=R$ ）不做平滑，对其他时间都进行平滑：

$$P_r=\begin{cases} P_R=\dfrac{R}{N}, \text{当} r=R \\ (1-n_R P_R)\dfrac{r+1}{N}\dfrac{n_{r+1}}{n_r}, \text{当} 0\leqslant r\leqslant R-1 \end{cases}$$

  - 用一个函数 S(r) 对 N_r 进行拟合

（详见《统计自然语言处理基础》一书第 6.2.5 节）

平滑算法—绝对减值法

- 绝对减值法

$$P_r=\begin{cases}\dfrac{r-b}{N}, \text{当} r>0\\ \dfrac{b(K-n_0)}{Nn_0}, \text{当} r=0\end{cases}$$

K 为所有可能的事件数目

- 参数 b 的上限 $b\le\dfrac{n_1}{n_1+2\mathrm{n}_2}$

平滑算法—线性减值法

- 线性减值法

$$P_r = \begin{cases} \dfrac{(1-\alpha)r}{N}, \text{当} r > 0 \\ \dfrac{\alpha}{n_0}, \text{当} r = 0 \end{cases}$$

自由参数 α 的优化值为： n_1/N

平滑算法—回退（ Back-off ）法

当某一事件的频率小于 K 时，用 n-1 元语法来代替 n 元语法

$$P(x_n|x_1 \ldots x_{n-1}) = \begin{cases} (1-\alpha(f(x_1 \ldots x_n)))\dfrac{f(x_1 \ldots x_n)}{f(x_1 \ldots x_{n-1})}, \text{当 } f(x_1 \ldots x_n) > K \\ \alpha(f(x_1 \ldots x_{n-1}))P(x_n|x_2 \ldots x_{n-1}), \text{当 } f(x_1 \ldots x_n) \le K \end{cases}$$

α 是归一化因子

平滑算法—删除插值法（Deleted Interpolation）

- 将高阶语法与低阶语法混合使用

$$P(w_3|w_1 w_2)=\lambda_3 P'(w_3|w_1 w_2)+\lambda_2 P'(w_3|w_2)+\lambda_1 P'(w_3)$$

其中，$\lambda_1+\lambda_2+\lambda_3=1$

- 理论上与回退法等价

平滑算法的参数估计

- 将训练语料分成两部分
  - 第一部分用于估计模型参数
  - 第二部分语料用于估计平滑参数
- 如果直接用第一部分估计平滑参数
  - 得出的结论将是：不平滑最好
  - 因为第一部分语料中本来就不会出席需要平滑的现象

句子首尾标记处理

- 在语言模型训练时，要注意给每一个句子加上句子首尾标记，通常用 <s> 和 </s> 来表示，应用语言模型时，也要把句子首尾标记考虑进来，否则会影响模型应用的效果。
- 在考虑句子首尾标记的情况下，用三元语法模型计算一个句子 $w_1 w_2 w_3$ 的概率应该是：

$$p(\text{<s>}\, w_1 w_2 w_3 \,\text{</s>}) = p(w_1|\text{<s>}) \times p(w_2|\text{<s>}\, w_1) \times p(w_3|w_1 w_2) \times p(\text{</s>}|w_2 w_3) \times p(\text{</s>}|w_3)$$

N元语法模型的参数存储

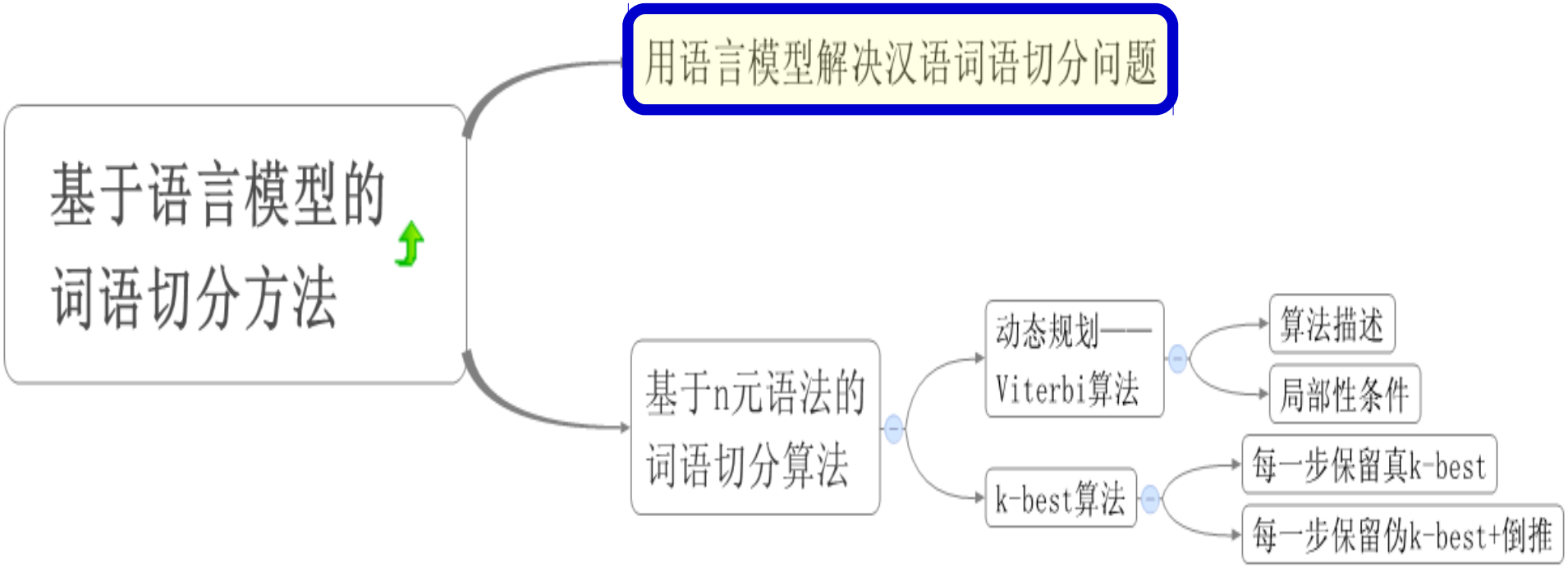
- 假设词表大小为 S，那么 N 元语法模型的参数空间理论上就是 S^N，然而实际上，由于这些参数绝大部分都是零，数据稀疏非常严重，因此如果采用 N 维数组形式存放将会严重浪费空间
- 通常采用 TRIE 树的形式来存放 N 元语法模型的参数
- 与词典 TRIE 树的区别在于：词典 TRIE 树上每个结点对应于一个字母（或者汉字），而 N 元语法模型 TRIE 树上一个结点对应于一个词

N元语法模型工具

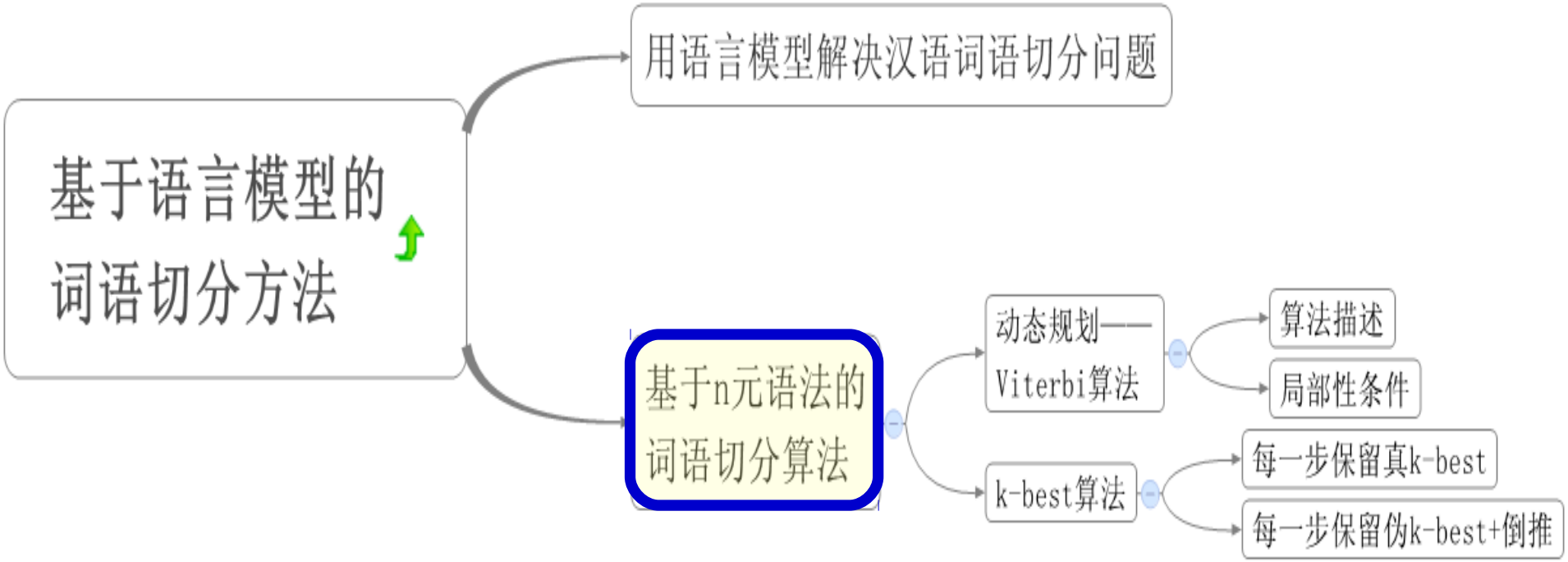
- 开源工具：
  - SRI Language Model
  - IRST Language Model (in Moses)

内容提要

内容提要


用语言模型解决汉语词语切分问题
基于语言模型的词语切分方法
基于n元语法的词语切分算法
动态规划——Viterbi算法
算法描述
局部性条件
k-best算法
每一步保留真k-best
每一步保留伪k-best+倒推

内容提要


基于语言模型的词语切分方法
用语言模型解决汉语词语切分问题
基于n元语法的词语切分算法
动态规划——Viterbi算法
算法描述
局部性条件
k-best算法
每一步保留真k-best
每一步保留伪k-best+倒推

基于 N 元语法的词语切分

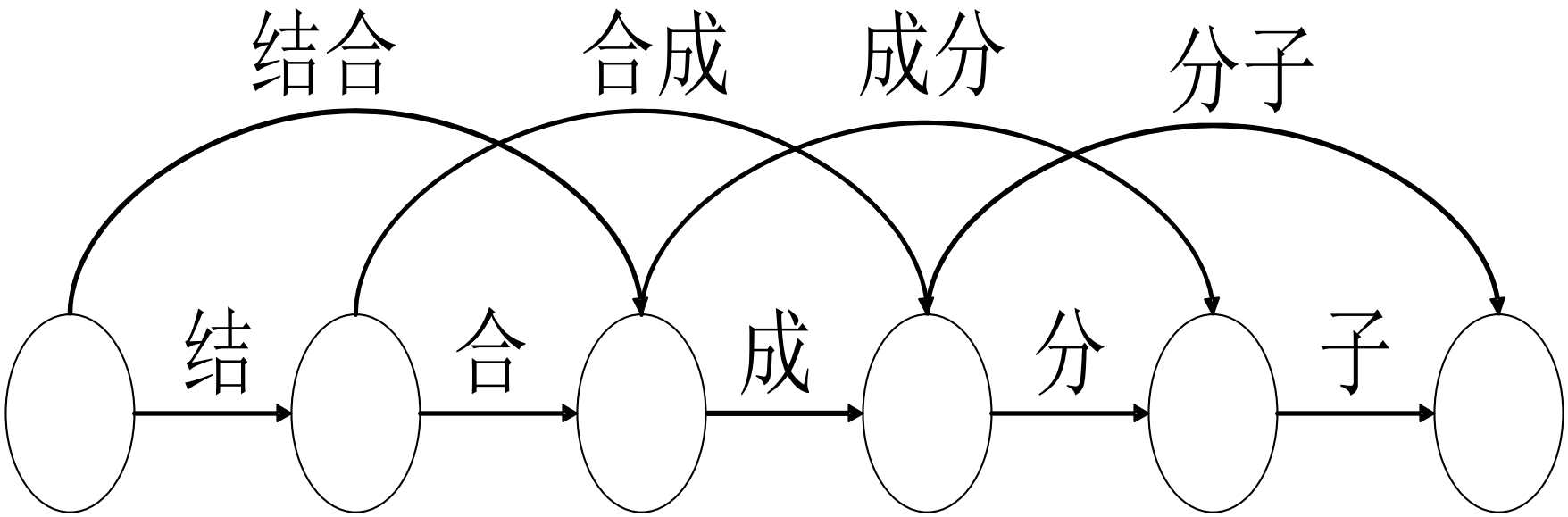
- 对于每一个切分结果，采用 n 元语法模型计算其概率，并输出概率最大的切分结果

$$
\begin{aligned}
W^* &= \underset{W}{argmax}\, P(\text{<s>}\, W\, \text{</s>}) \\
&= \underset{W}{argmax}\, P(\text{<s>}\, w_1 \ldots w_l\, \text{</s>}) \\
&= p(w_1|\text{<s>})\, p(w_2|\text{<s>}\, w_1) \ldots p(w_{N-1}|\text{<s>}\, w_1 w_2 \ldots w_{N-2}) \\
&\quad p(w_N|w_1 w_2 \ldots w_{N-1}) \ldots p(w_l|w_{l-N+1} \ldots w_{l-1}) \\
&\quad p(\text{</s>}|w_{l-N+2} \ldots w_l) \ldots p(\text{</s>}|w_l)
\end{aligned}
$$

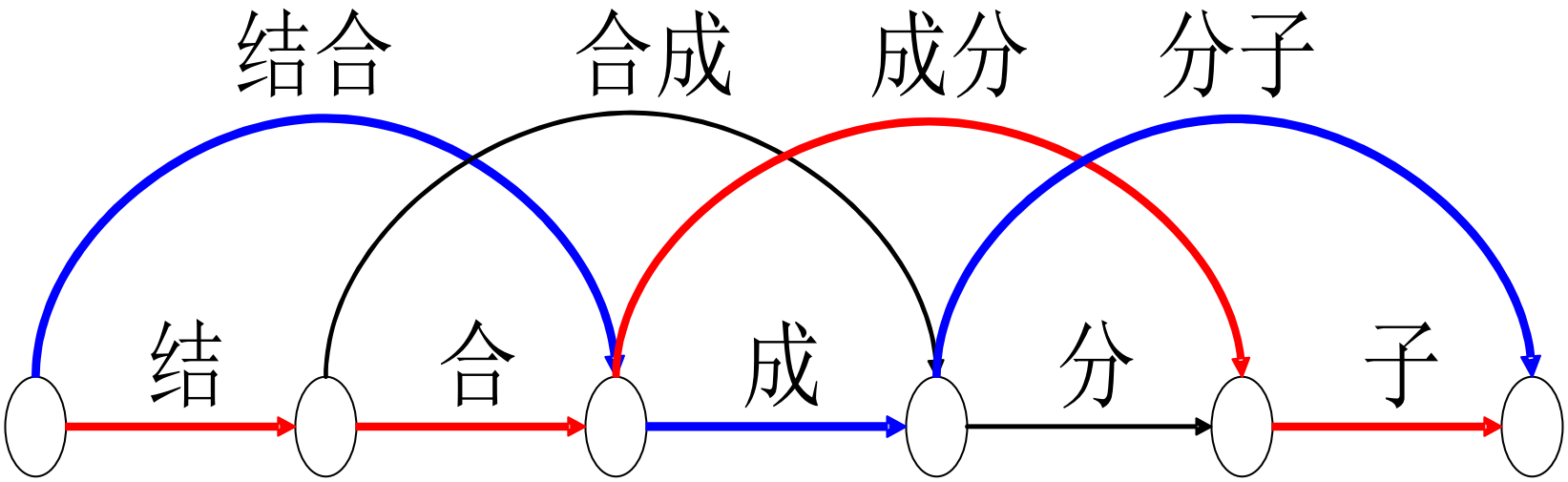
一元切分词图

- 对任何一个未切分句子，可以构造一个一元切分词图
- 一元切分词图为一个有向图：
  - 结点：相邻两个汉字之间的间隔
  - 边：一个候选的词语
- 在一元切分词图上，从句子起始位置开始，到句子结束位置中止的任何一条路径，对应着一种可能的词语切分结果。

一元切分词图



一元切分词图

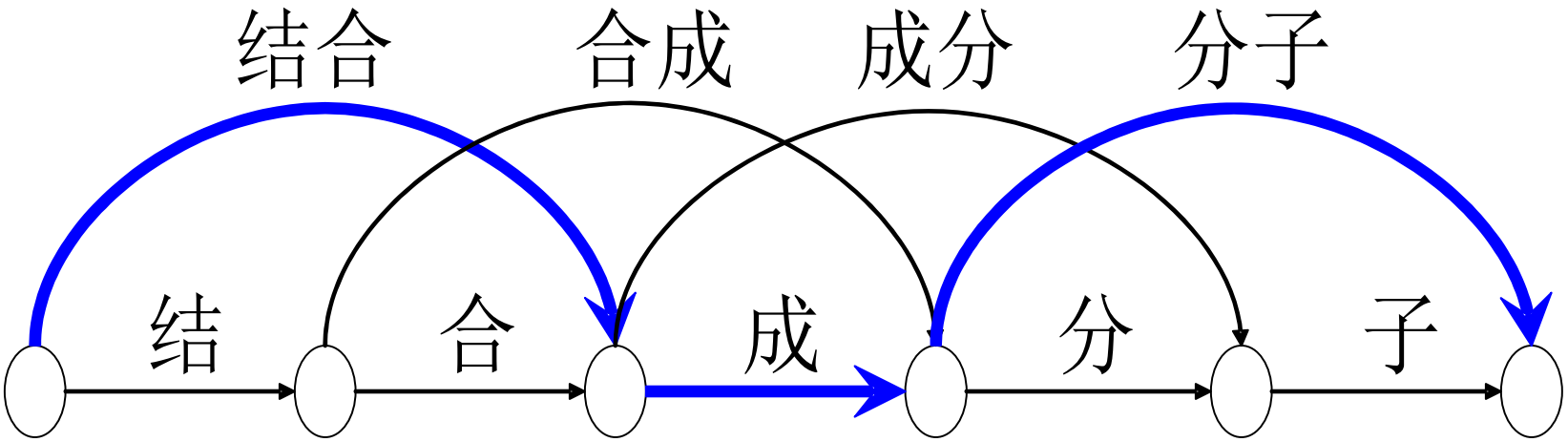


路径即切分：

- 红色路径： 结 . 合 . 成分 . 子
- 蓝色路径： 结合 . 成 . 分子

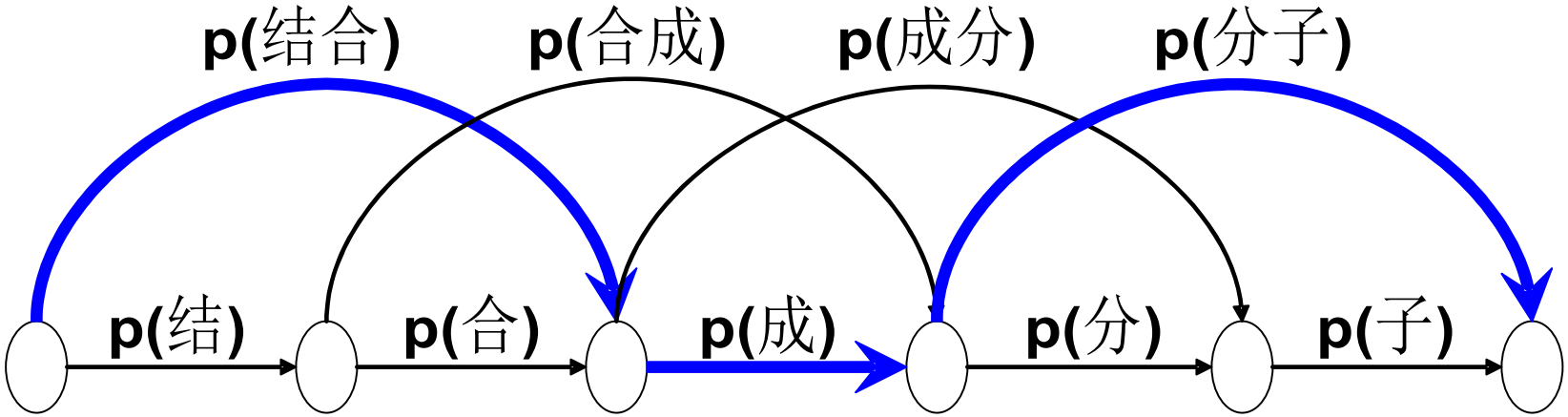
切分表示为词图的路径

- 词语切分可以转化为在切分词图上寻找概率最大的最优路径问题。



基于一元语法模型的词语切分

- 给词图上每一条边赋予相应词语的一元语法模型概率，一条切分路径的概率表示为路径上所有边的概率乘积。

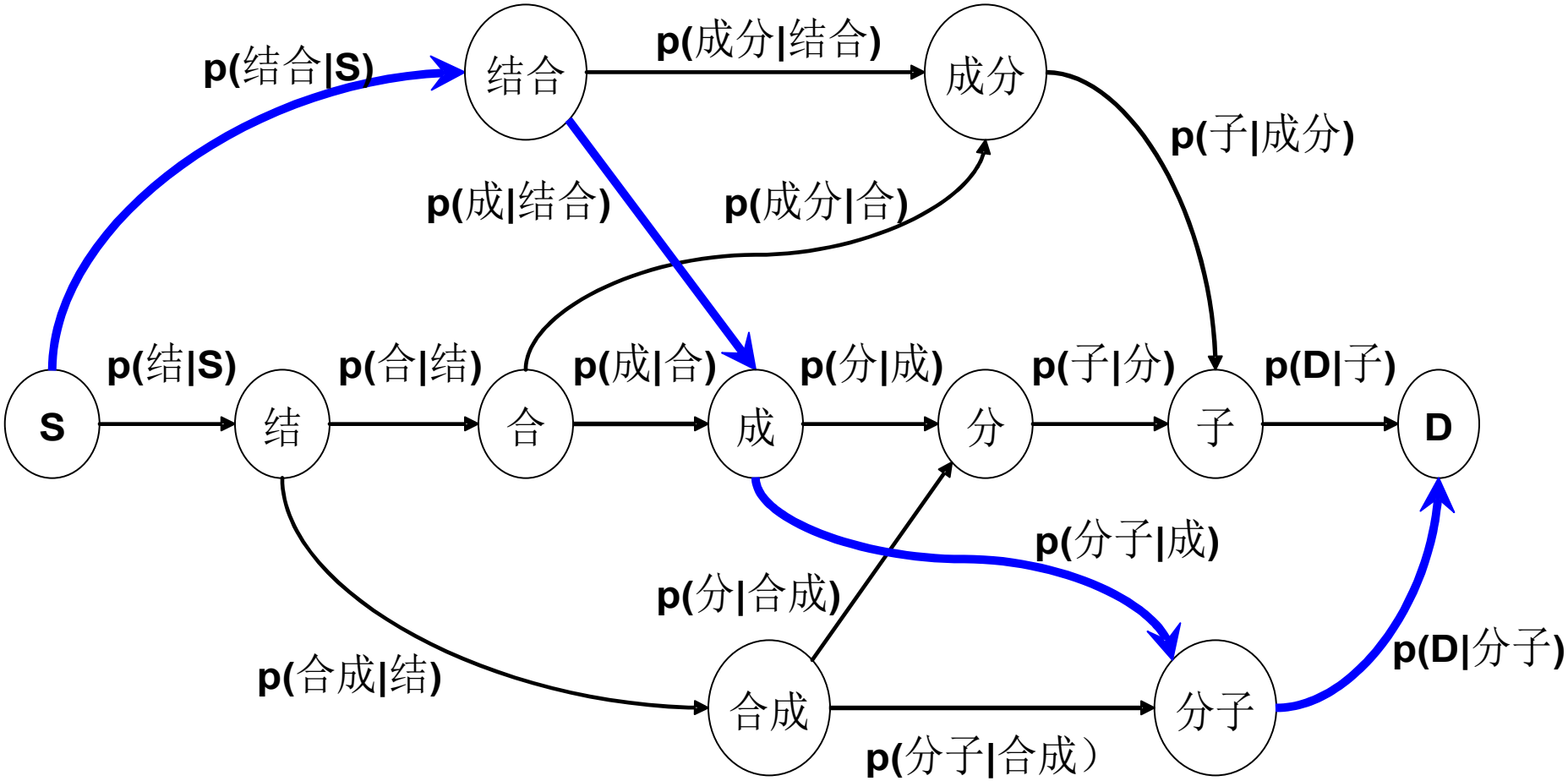


$$p(结合\cdot成\cdot分子)=p(结合)p(成)p(分子)$$

二元切分词图

- 一元切分词图无法表示二元语法模型所需的二元词语转移概率
- 二元切分词图定义为如下的有向图：
  - 结点：任何一个可能的候选词语 (W_i)
  - 边：相邻两个词语的接续关系 (W_{i-1})→(W_i)
- 在二元切分词图的每一条边上标记二元词语转移概率 $P(W_i|W_{i-1})$
- 任何一个词语切分可以表示为二元切分词图上的一条起始结点到结束结点的路径
- 路径上所有边的概率之积就是该切分结果对应的二元语法模型概率。

二元切分词图

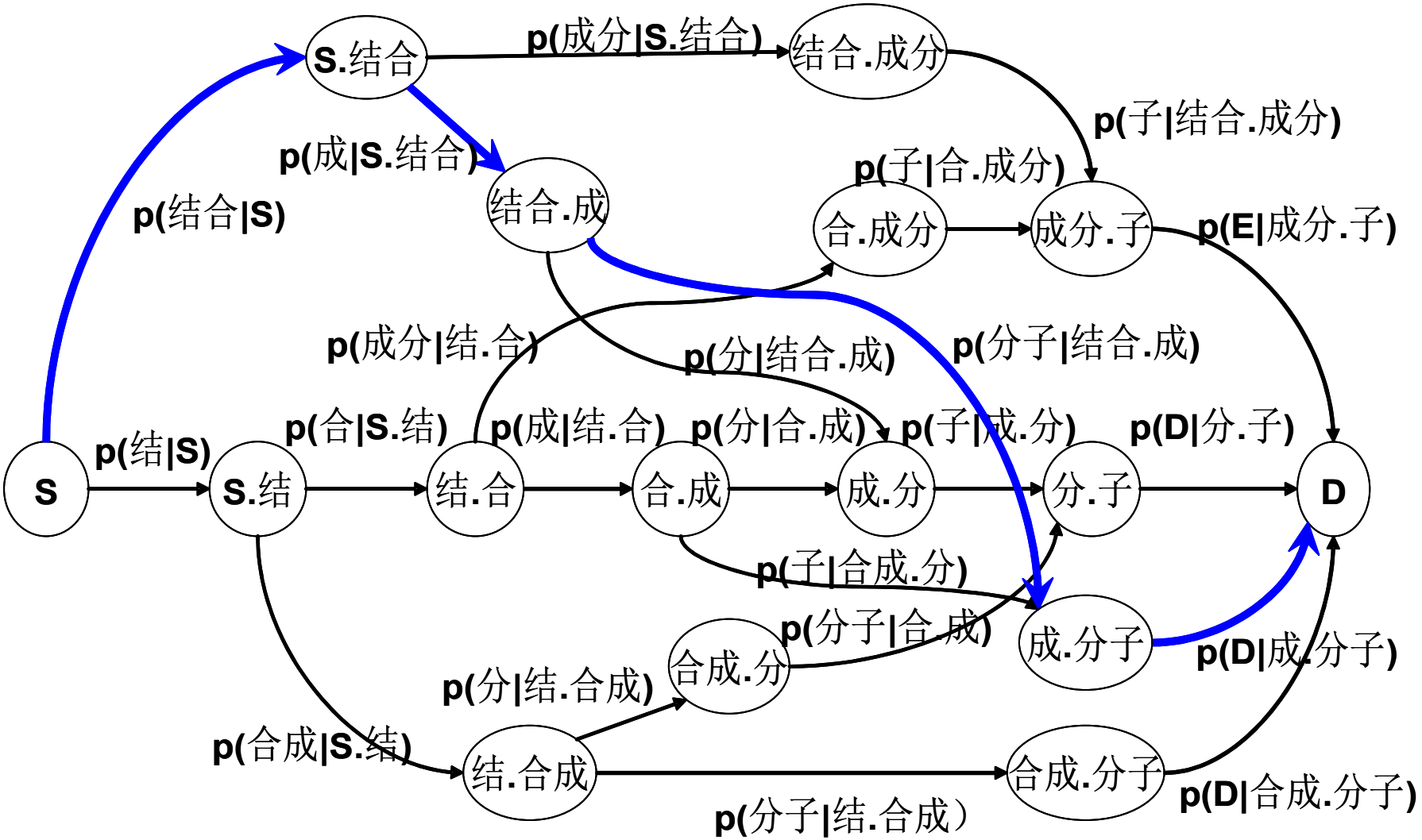


p(结合·成·分子)=p(结合|S)p(成|结合)p(分子|成)p(D|分子)

三元切分词图

- 三元切分词图
  - 结点：任何一个可能的词语对 ($W_{i-1}W_i$)
  - 边：两个结点之间的有向边，有向边的起点的后一个词语与终点的前一个词语相同 ($W_{i-2}W_{i-1}$)→($W_{i-1}W_i$)
- 在三元切分词图的每一条边上标记三元词语转移概率 $P(W_i|W_{i-2}W_{i-1})$
- 任何一个词语切分可以表示为三元切分词图上的一条起始结点到结束结点的路径
- 路径上所有边的概率之积就是该切分结果对应的三元语法模型概率。

三元切分词图


S.结合
p(成分|S.结合)
结合.成分
p(子|结合.成分)
p(成|S.结合)
结合.成
p(子|合.成分)
合.成分
成分.子
p(E|成分.子)
p(结合|S)
p(成分|结.合)
p(分|结合.成)
p(分子|结合.成)
p(合|S.结)
p(成|结.合)
p(分|合.成)
p(子|成.分)
p(D|分.子)
p(结|S)
S
S.结
结.合
合.成
成.分
分.子
D
p(子|合成.分)
p(分子|合.成)
成.分子
p(D|成.分子)
合成.分
p(分|结.合成)
p(合成|S.结)
结.合成
合成.分子
p(D|合成.分子)
p(分子|结.合成)

N 元切分词图

- N 元切分词图
  - 结点：任何一个可能的 N-1 词语对 ($W_{i-N+2}\cdots\cdots W_i$)
  - 边：两个结点之间的有向边，有向边的起点的后 N-2 个词语与终点的前 N-2 个词语相同 ($W_{i-N+1}\cdots\cdots W_{i-1}$)→($W_{i-N+2}\cdots\cdots W_i$)
- 在 N 元切分词图的每一条边上标记 N 元词语转移概率 $P(W_i|W_{i-N+1}\cdots\cdots W_{i-1})$
- 任何一个词语切分可以表示为 N 元切分词图上的一条起始结点到结束结点的路径
- 路径上所有边的概率之积就是该切分结果对应的 N 元语法模型概率。

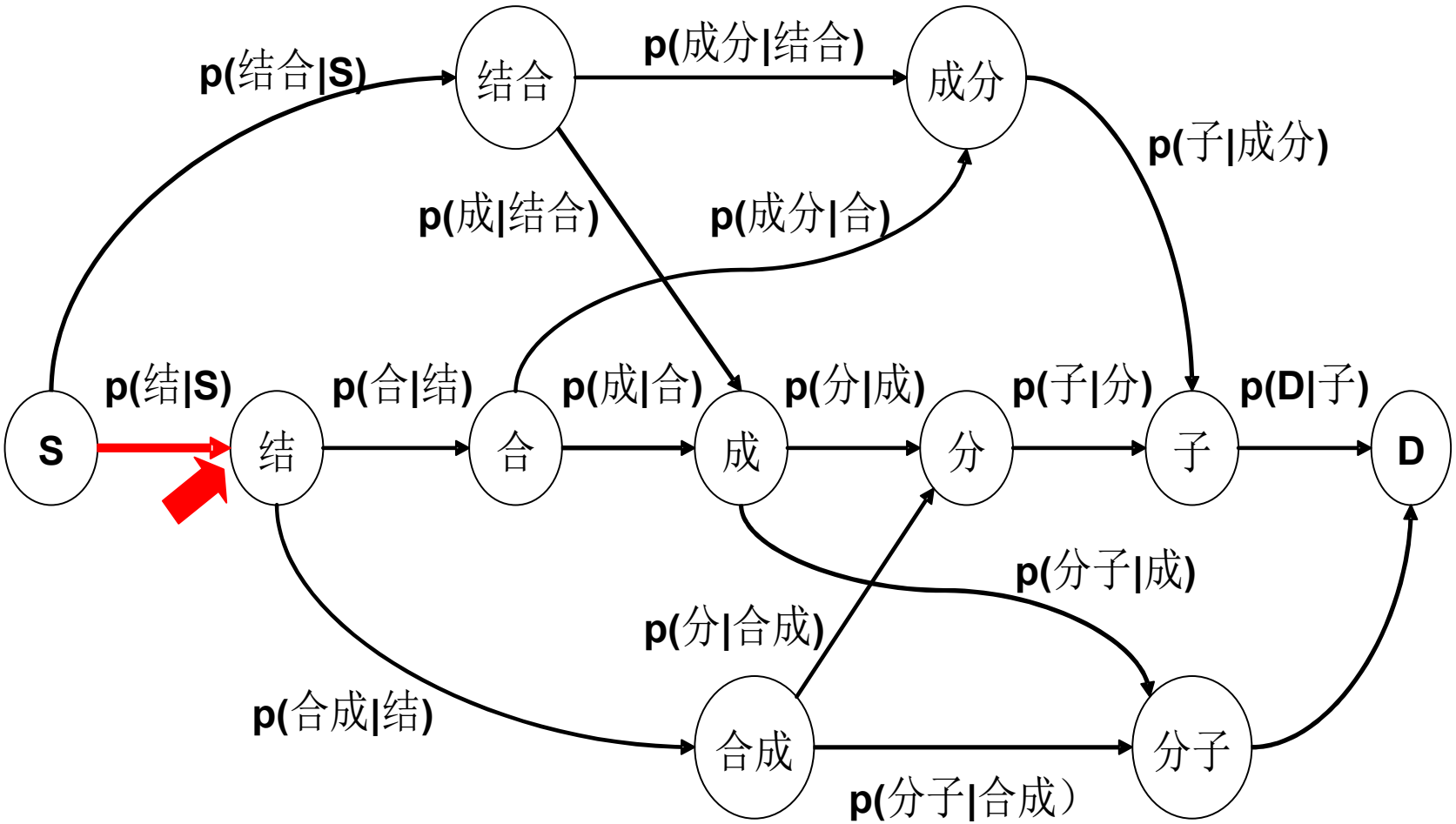
N元切分词图的构造

- 先构造 N-1 元词图
- 对于 N-1 元词图上每一条边：在 N 元词图上添加一个结点；
- 对于 N-1 元词图上每一个结点：假设该结点有 S 条入边 ($e_{i1},\ldots,e_{iS}$) 和 T 条出边 ($e_{o1},\ldots,e_{oT}$)，那么对于对于该结点的每一对入边和出边的组合 (e_{is},e_{ot})，在 N 元词图上增加一条边，该边的起点和终点分别是 e_{is} 和 e_{ot} 在 N 元词图上对应的结点

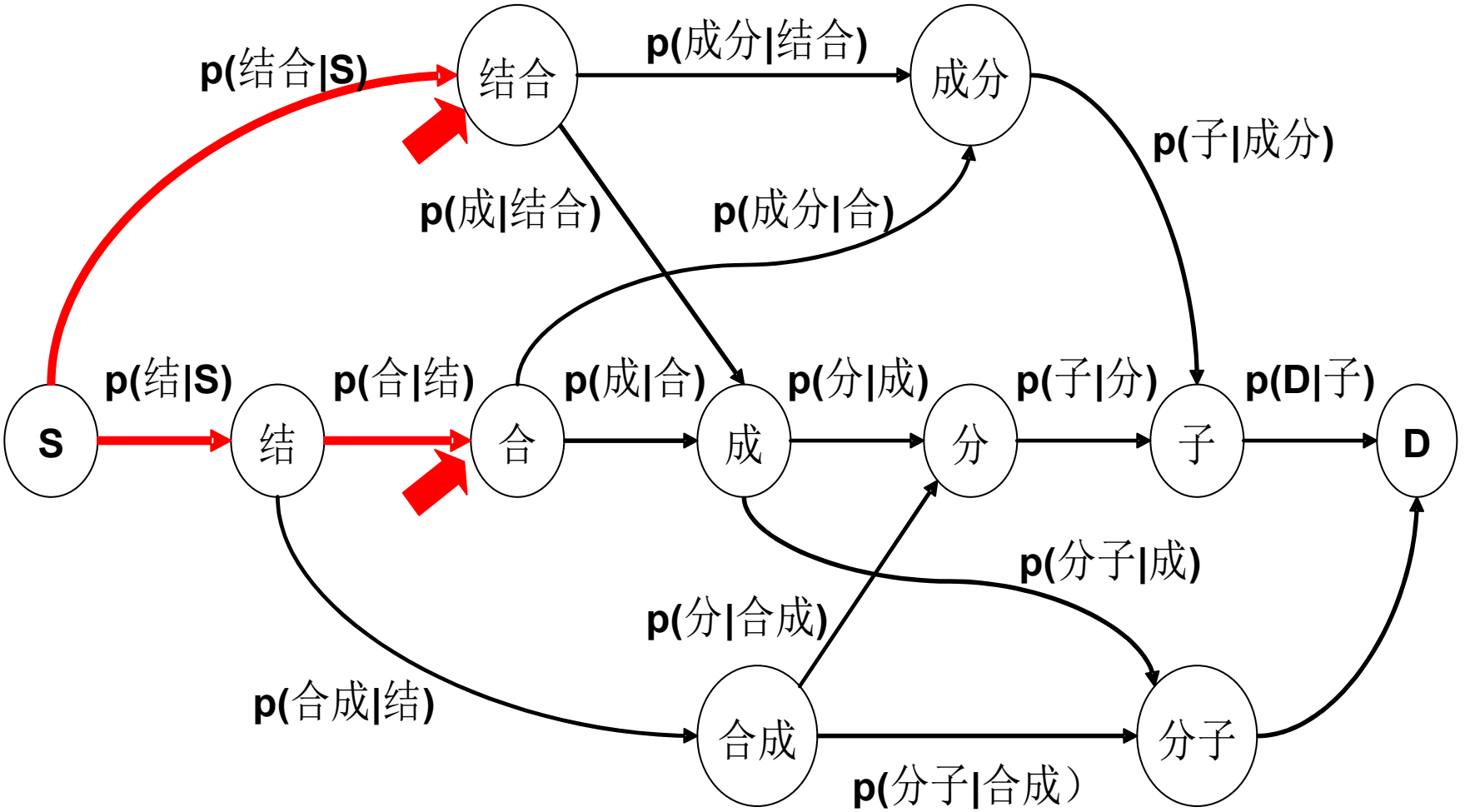
在词图上搜索最优路径： Viterbi 算法

- Viterbi Search Algorithm
  - Assign p(source_node)=1
  - For each node ***n*** in the word lattice from source to destination in a **topological order**
    - p(***n***)=0, previous_edge(***n***)=ø
    - For each edge ***e*** directed to ***n*** from ***n***'
      - p'(***n***)=p(***n***')*p(***e***)
      - If p'(***n***) > p(***n***) then p(***n***)=p'(***n***), previous_edge(***n***)=***e***
  - Let ***best_segmentation*** is a empty array of edges
  - Let node ***n*** is the destination node
  - Repeat until ***n*** is the source node
    - Push previous_edge(***n***) to the head of ***best_segmentation***
    - Let ***n*** be the node where ***e*** start from
  - Return ***best_segmentation***

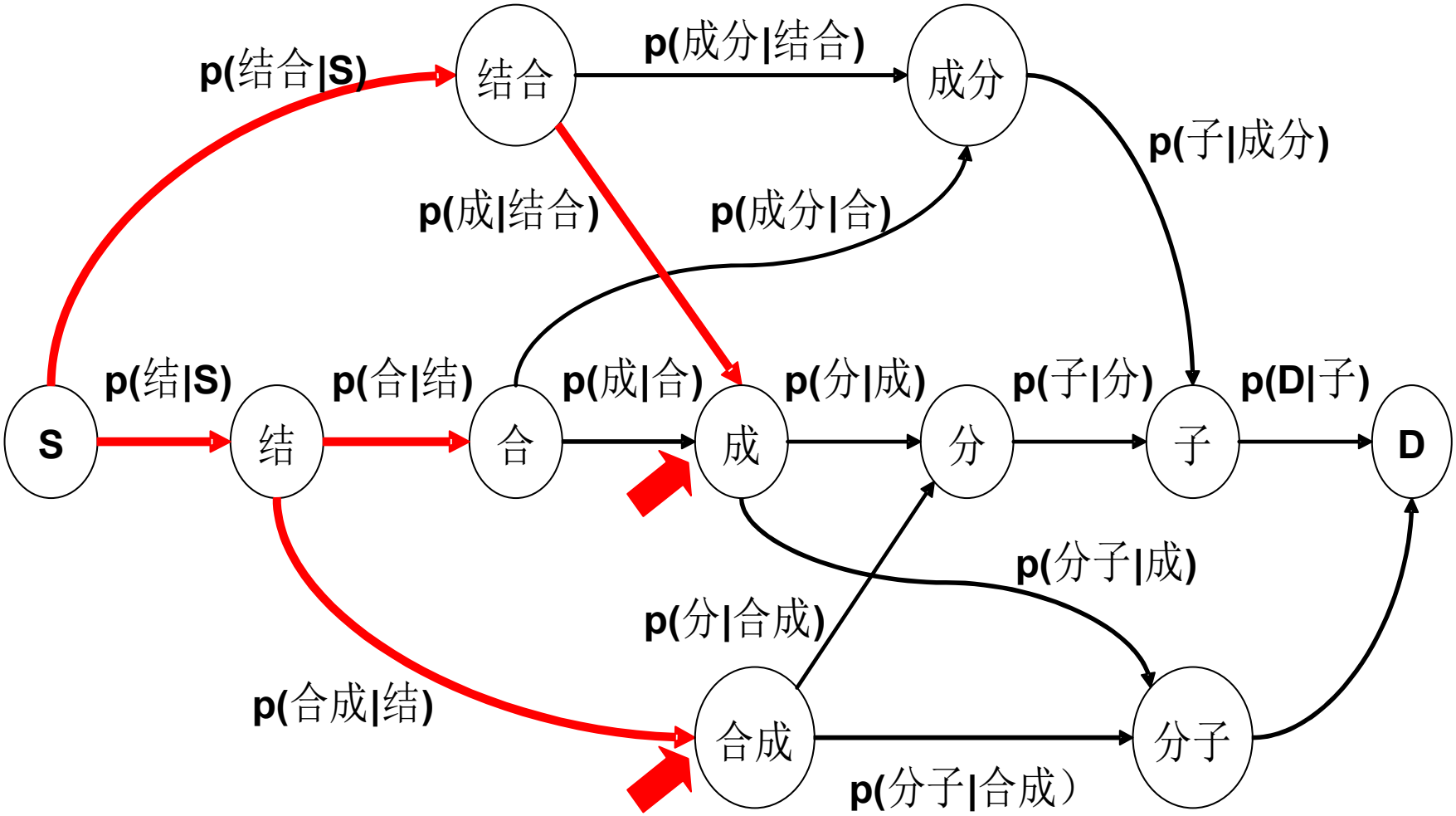
在词图上搜索最优路径： Viterbi 算法



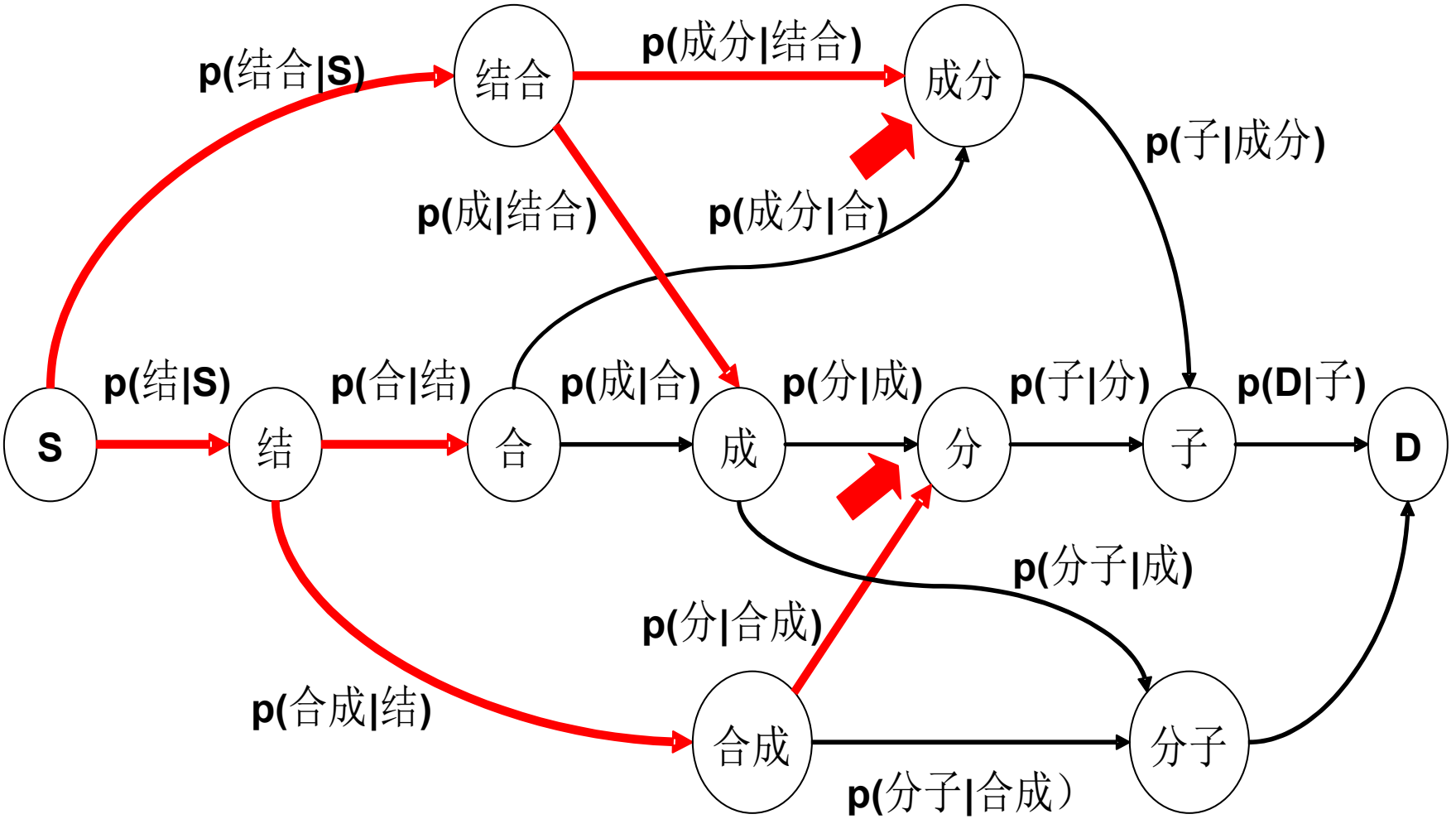
在词图上搜索最优路径： Viterbi 算法



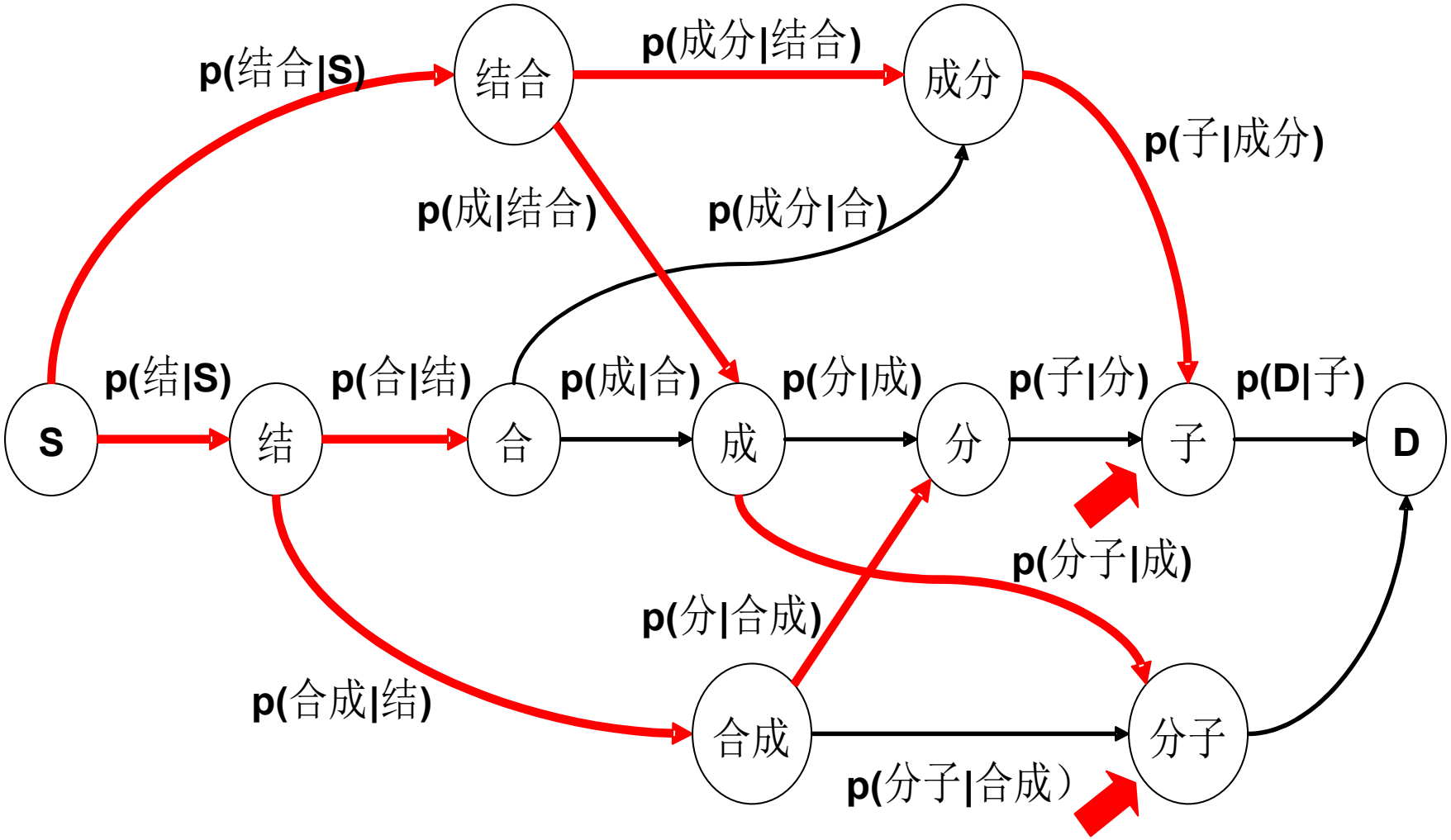
在词图上搜索最优路径： Viterbi 算法



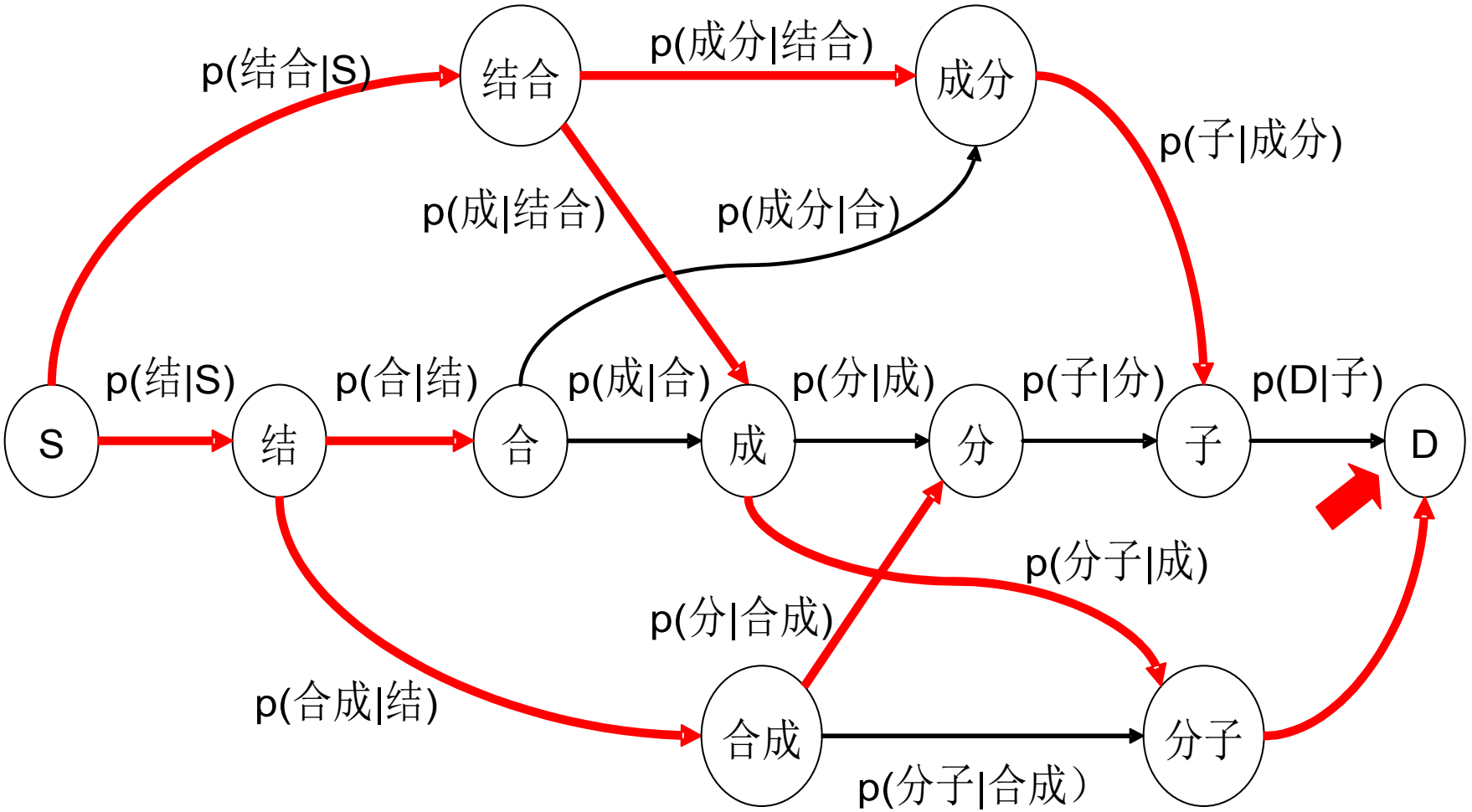
在词图上搜索最优路径： Viterbi 算法



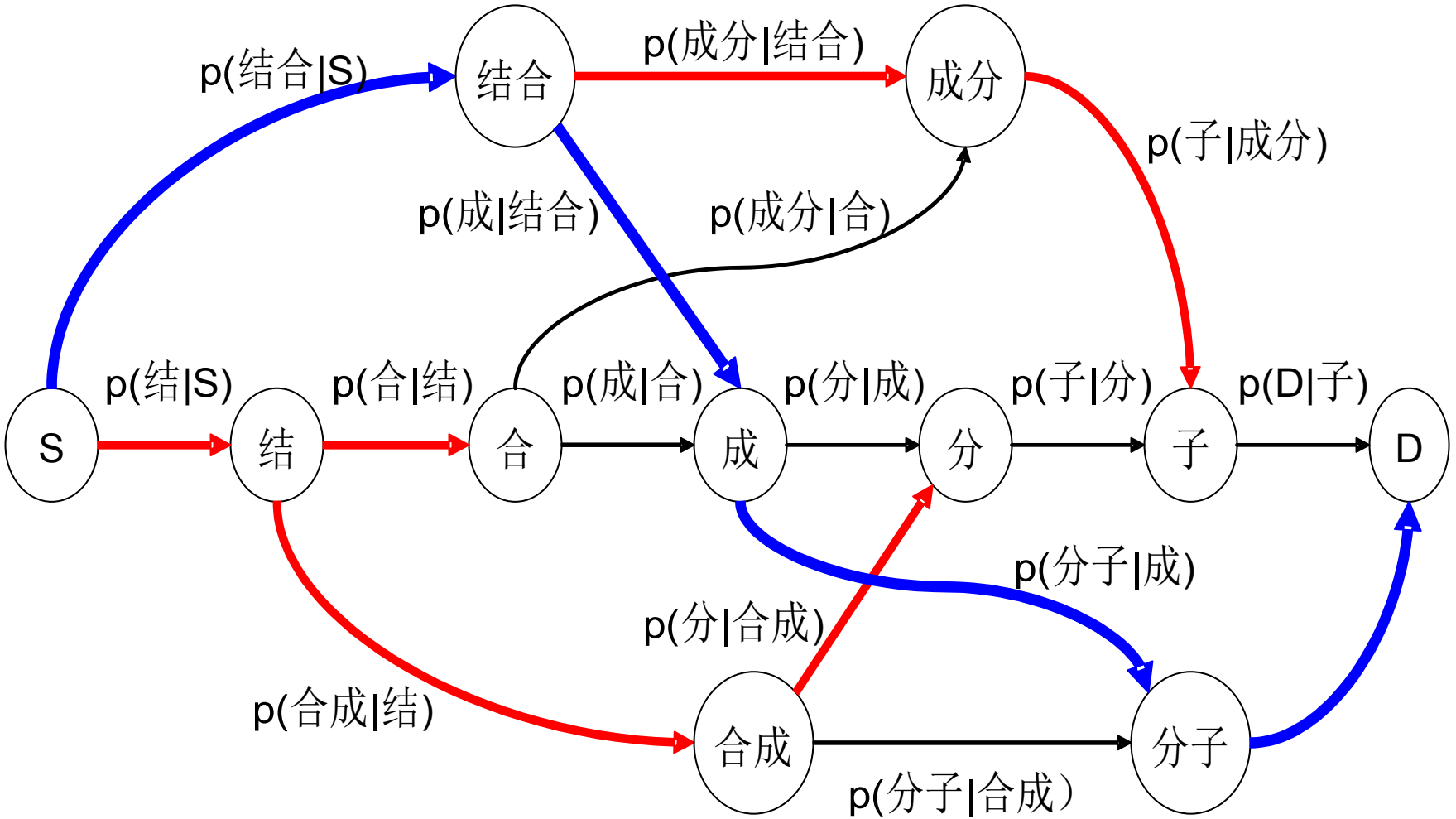
在词图上搜索最优路径： Viterbi 算法



在词图上搜索最优路径： Viterbi 算法



在词图上搜索最优路径： Viterbi 算法



基于 n 元语法模型的词语切分

- 采用一元语法
  - 把词频的负对数理解成“代价”，这种方法也可以理解为最短路径法的一种扩充
  - 正确率可达到 92%
  - 简便易行，效果一般好于基于词表的方法
- 采用三元语法
  - 实验表明，在较大规模数据上，采用三元语法进行词语切分正确率可以达到 98% 以上
- 缺点：无法识别未定义词

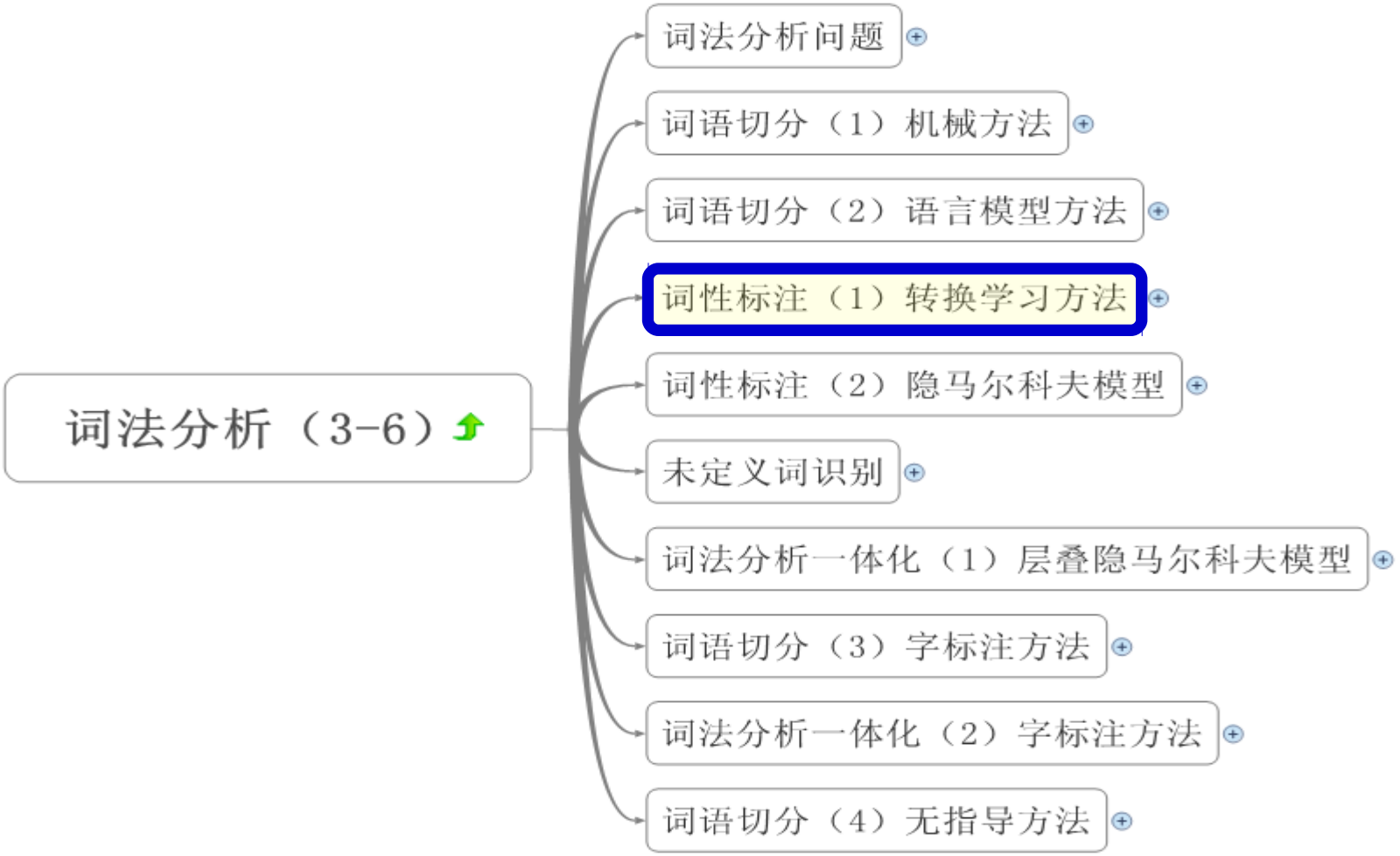
在词图上搜索 n-best 路径

- （略）

在词图上搜索最优路径： Dijkstra 算法

- （略）

内容提要


词法分析（3-6）
词法分析问题
词语切分（1）机械方法
词语切分（2）语言模型方法
词性标注（1）转换学习方法
词性标注（2）隐马尔科夫模型
未定义词识别
词法分析一体化（1）层叠隐马尔科夫模型
词语切分（3）字标注方法
词法分析一体化（2）字标注方法
词语切分（4）无指导方法

内容提要

- 汉语词性标注
  - 汉语词性的特点
  - 词的兼类现象与词性标注
  - 基于规则的词性标注方法
  - 基于转换的错误驱动的词性标注方法
  - 基于隐马尔科夫模型的词性标注方法
- 汉语词法分析系统的集成

关于汉语词性 (1)

- 英语的词性
  - 英语的词性主要是由词的变化形式决定的
  - 英语的词性与词的句法功能存在着比较明确的一一对应关系
- 汉语的词性
  - 汉语词几乎没有形态变化
  - 汉语词性与所充当的语法功能不存在明确的一一对应关系
- 问题：如何确定汉语的词性？

关于汉语词性 (2)

- 问题 1：词性判定的依据是什么？
  - 句法功能 √
  - 语义 ×
  - ➢金、银 — 铜、铁、锡
  - 红、黑、紫、灰、粉 — 红色、咖啡色
  - 战争 — 打仗

关于汉语词性 (3)

- 问题 2：同一个词在充当不同句子成分时词性是否发生变化？
  - 我们调查了这件事。
  - 调查很重要。
  - 这件事很困难。
  - 我们不怕困难。
  - 去是有道理的。
  - 暂时不去是有道理的。

关于汉语词性 (4)

- 问题 1 的答案：词性判定的依据只能是句法功能；
- 问题 2 的答案：同一个词在充当不同句子成分时词性不发生变化。

参考文献：朱德熙，《语法答问》，商务印书馆，1985

汉语词性标记集

- 几个典型的词性标记集
  - 北京大学《人民日报》语料库标记集
  - 清华大学《汉语树库》词性标记集
  - 语用所《信息处理用现代汉语词类及词性标记集规范》
  - 宾州树库规范
  - 计算所词性标记集（ V3.0 ）
- 参考：词性标记集对照表

词的兼类现象 (1)

兼类数	兼类词数	百分比	例词及词性标记
5	3	0.01%	和 c-n-p-q-v
4	20	0.04%	光 a-d-n-v
3	126	0.23%	画 n-q-v
2	1475	2.67%	锁 n-v
合计	1624	2.94%	总词数： 55191

数据来源：北大计算语言所《现代汉语语法信息词典》 1997 年版

词的兼类现象 (2)

兼类	词数	百分比	例词
n-v	613	42%	爱好，把握，报道
a-n	74	5%	本分，标准，典型
a-v	217	15%	安慰，保守，抽象
b-d	103	7%	长期，成批，初步
n-q	64	4%	笔，刀，口
a-d	30	2%	大，老，真
合计	1101	75%	兼两类词数： 1475

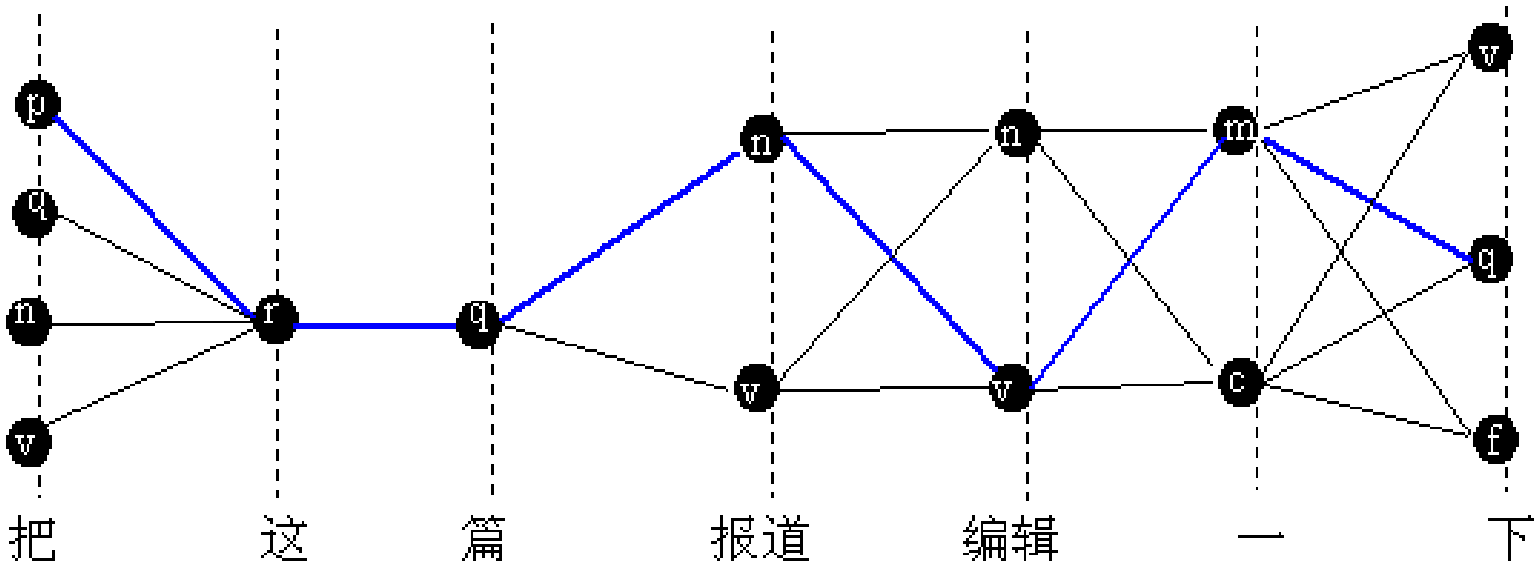
词的兼类现象 (3)
（English data, from Brown corpus）

引自： http://www.cs.columbia.edu/~becky/cs4999/04mar.html

10.4 percent of the lexicon is ambiguous as to part-of-speech (types)
40 percent of the words in the Brown corpus are ambiguous (tokens)

Degree of ambiguity	Total frequency (39,440)
1 tag	35,340
2-7 tags	4,100
2	3,760
3	264
4	61
5	12
6	2
7	1

词性标注：寻找最优路径



4×1×1×2×2×2×3 ＝ 96 种可能性，哪种可能性最大？

词性标注（ POS-Tagging ）

- 语法体系 —— 词性标记集的确定
- 一词多类现象
  - Time flies like an arrow.

    Time/n-v-a flies/v-n like/p-v an/Det arrow/n

  - 把这篇报道编辑一下

    把 /q-p-v-n 这 /r 篇 /q 报道 /v-n 编辑 /v-n 一 /m-c 下 /f-q-v

词性标注方法回顾

序号	作者 / 标注项目	标记集	方法，特点	处理语料规模	精确率
1	Klein&Simmons (1963)	30	手工规则	百科全书小样本	90%
2	TAGGIT (Greene&Rubin, 1971)	86	人工规则 (3300 条)	Brown 语料库	77 %
3	CLAWS (Marshall,1983; Booth, 1985)	130	概率方法 效率低	LOB 语料库	96 %
4	VOLSUNGA (DeRose,1988)	97	概率方法 效率高	Brown 语料库	96 %
5	Eric Brill's tagger (1992-94)	48	机器规则 (447 条) 效率高	UPenn WSJ 语料库	97 %

规则方法进行词性标注示例

@@ 信 (n-v)

CONDITION

FIND(L,NEXT,X){%X.yx= 的 | 封 | 写 | 看 | 读 }

SELECT n

OTHERWISE SELECT v-n

@@ 一边 (c-s)

CONDITION

FIND(LR,FAR,X) {%X.yx = 一边 }

SELECT c

OTHERWISE SELECT s

基于转换的错误驱动的词性标注方法

Eric Brill (1992,1995)

- Transformation-based error-driven part of speech Tagging
- 基于转换规则，先给出初始标记，然后不断修正
- 有监督的学习，通过语料库学习转换规则

基本思想：

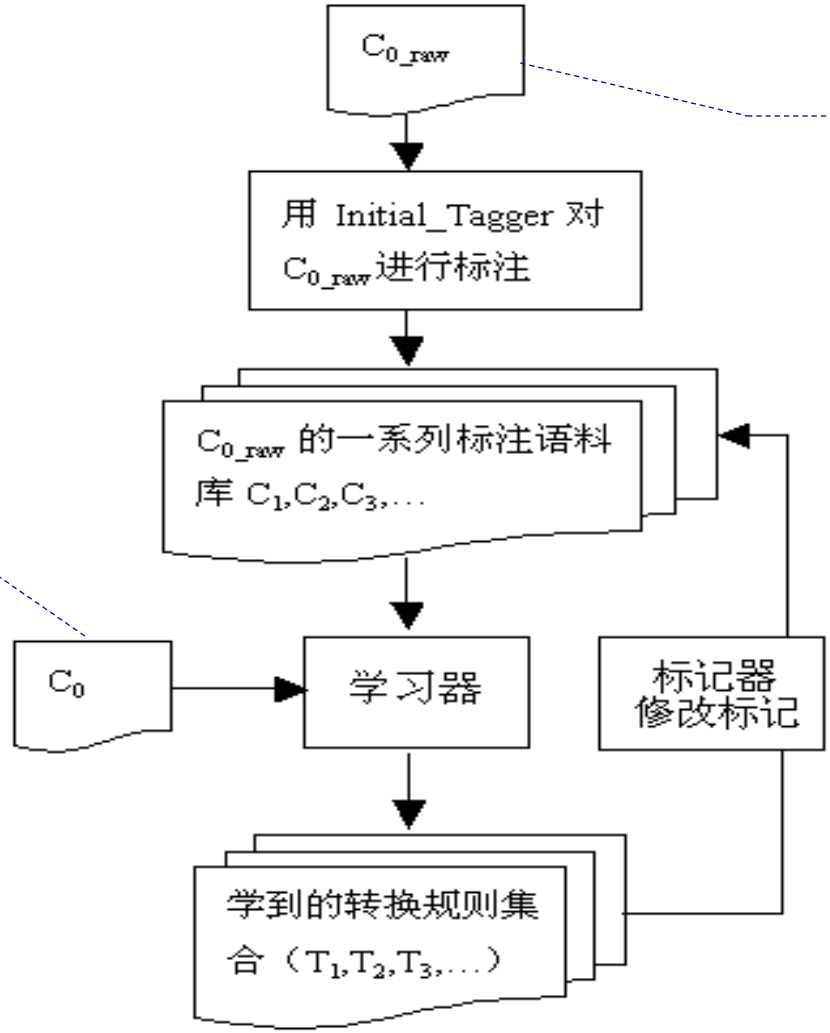
（1）正确结果是通过不断修正错误得到的

（2）修正错误的过程是有迹可循的

（3）让计算机学习修正错误的过程，这个过程可以用转换规则（transformation）形式记录下来，然后用学习得到转换规则进行词性标注

下载 Brill's tagger: http://research.microsoft.com/~brill/

转换规则的学习流程



C_{0_raw} 表示 C_0 拿掉正确标注后的生语料库

C_0 表示带有正确标注的语料库

转换规则学习器算法描述

1） 首先用初始标注器对 C_{0_raw} 进行标注，得到带有词性标记的语料 $C_i(i=1)$；

2） 将 C_i 跟正确的语料标注结果 C_0 比较，可以得到 C_i 中总的词性标注错误数，并根据规则模板，产生一个可应用的候选规则集合；

3） 依次从候选规则中取出一条规则 $T_m(m=1,2,\ldots)$，每用一条规则对 C_i 中的词性标注结果进行一次修改，就会得到一个新版本的语料库，不妨记做 $C_i^m(m=1,2,3,\ldots)$，将每个 C_i^m 跟 C_0 比较，可计算出每个 C_i^m 中的词性标注错误数。假定其中错误数最少的那个是 C_i^j（且 C_i^j 中的错误数少于 C_i 中的错误数），产生它的规则 T_j 就是这次学习得到的转换规则；此时 C_i^j 成为新的待修改语料库，即 $C_{i+1}=C_i^j$ 。

4） 重复第 3 步的操作，得到一系列的标注语料库 $C_2^k, C_3^l, C_4^m, \ldots$ 后一个语料库中的标注错误数都少于前一个中的错误数，每一次都学习到一条令错误数降低最多的转换规则。直至运用所有规则后，都不能降低错误数，学习过程结束。这时得到一个有序的转换规则集合 $\{T_a, T_b, T_c, \ldots\}$

转换规则的形式

- 转换规则由两部分组成
  - 改写规则（ rewriting rule ）
  - 激活环境（ triggering environment ）
- 一个例子：转换规则 T_1

改写规则：将一个词的词性从动词（ v ）改为名词（ n ）；

激活环境：该词左边第一个紧邻词的词性是量词（ q ），
第二个词的词性是数词（ m ）

S0: 他 /r 做 /v 了 /u 一 /m 个 /q 报告 /v

S1: 他 /r 做 /v 了 /u 一 /m 个 /q 报告 /n

转换规则的模板（ template ）

改写规则：将词性标记 x 改写为 y

激活环境：

（ 1 ）当前词的前（后）面一个词的词性标记是 z ；
（ 2 ）当前词的前（后）面第二个词的词性标记是 z ；
（ 3 ）当前词的前（后）面两个词中有一个词的词性标记是 z ；
……

其中 x ， y ， z 是任意的词性标记代码。

规则模板决定了所有可能的规则构成的一个规则空间，规则模板定义的好坏，直接影响最终系统的性能

根据模板可能学到的转换规则

T_1：当前词的前一个词的词性标记是量词（q）时，
将当前词的词性标记由动词（v）改为名词（n）；

T_2：当前词的后一个词的词性标记是动词（v）时，
将当前词的词性 标记由动词（v）改为名词（n）；

T_3：当前词的后一个词的词性标记是形容词（a）时，
将当前词的词 性标记由动词（v）改为名词（n）；

T_4：当前词的前面两个词中有一个词的词性标记是名词（n）时，
将当前词的词性标记由形容词（v）改为数词（m）；