词法、语法及统计思想基础

肖桐　朱靖波

xiaotong@mail.neu.edu.cn
zhujingbo@mail.neu.edu.cn

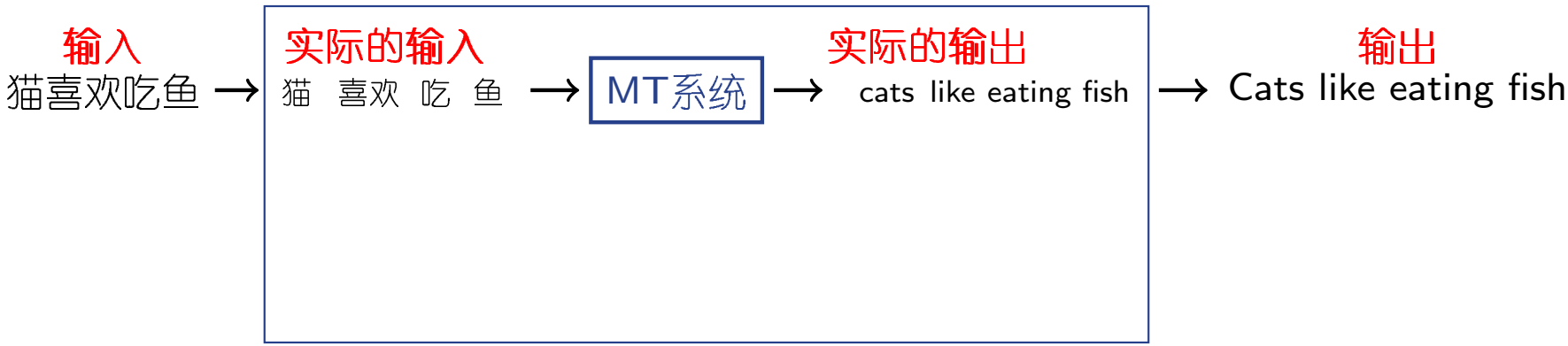
东北大学 自然语言处理实验室
http://www.nlplab.com

输入和输出是啥样子？

我们看到的机器翻译(MT)系统 -

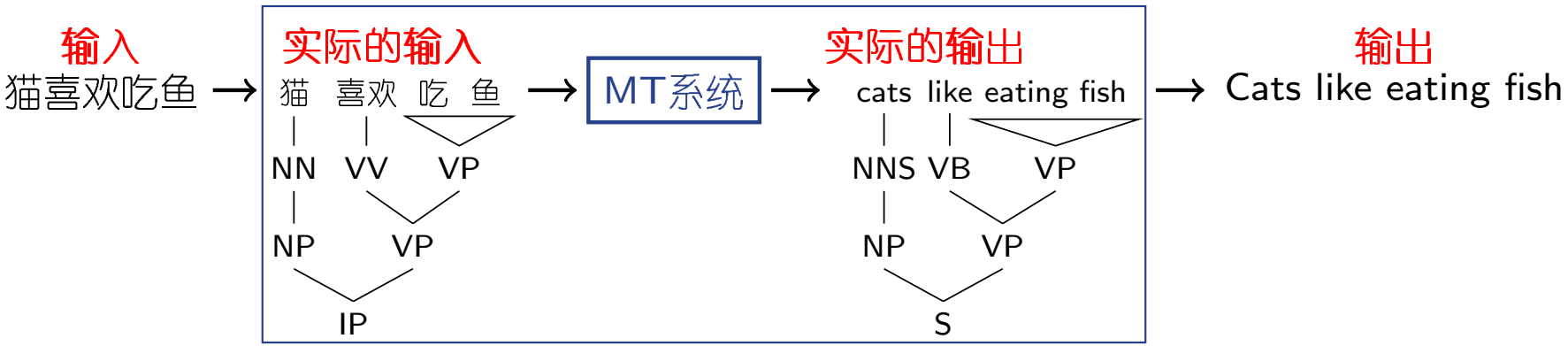
输入和输出是啥样子？

真实的机器翻译(MT)系统 - 词序列？



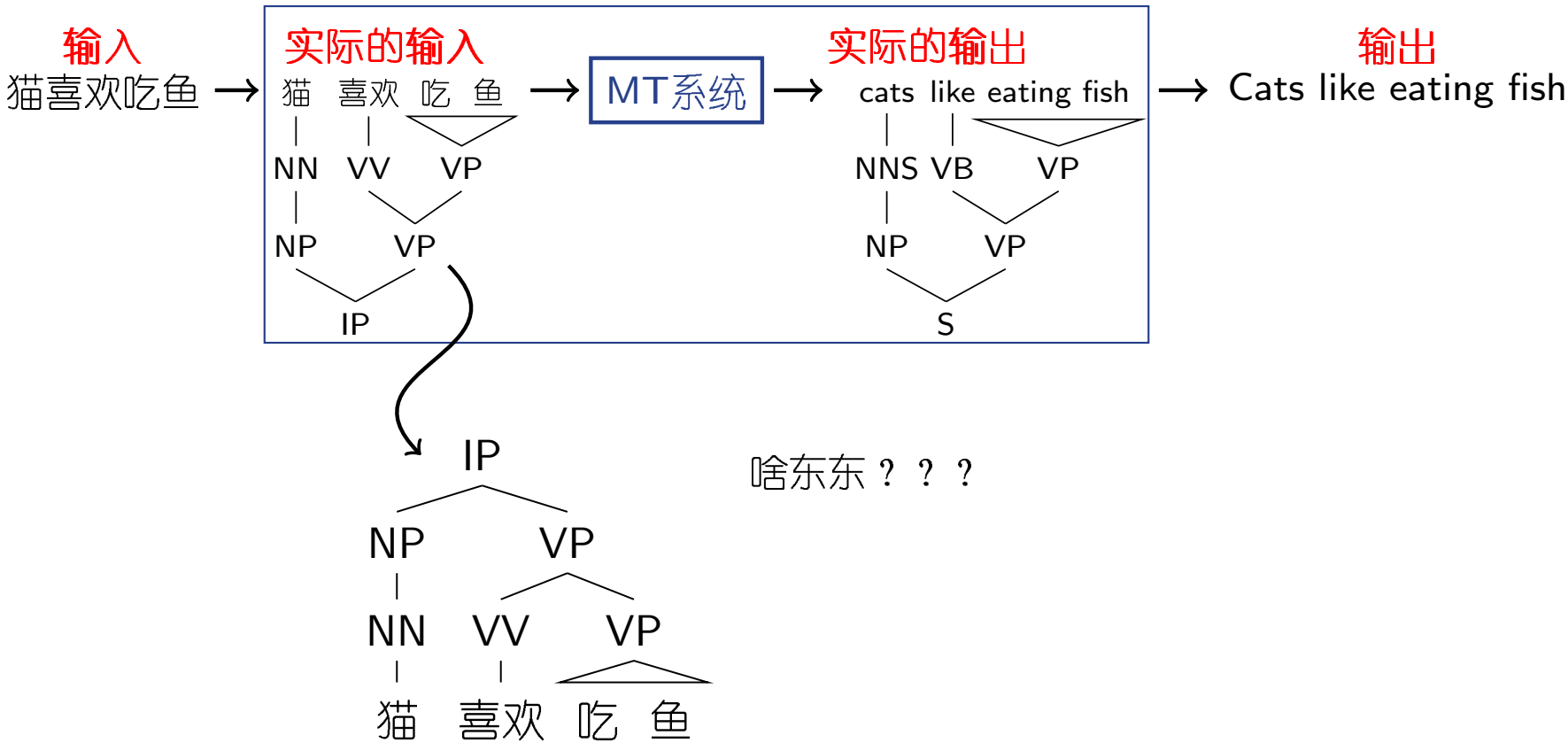
输入和输出是啥样子？

真实的机器翻译(MT)系统 - 词序列？树？



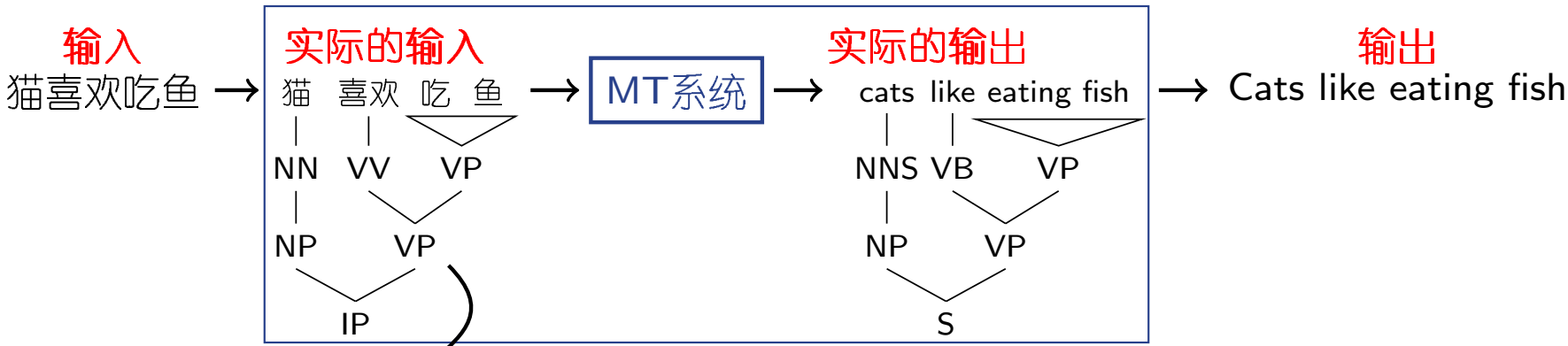
输入和输出是啥样子？

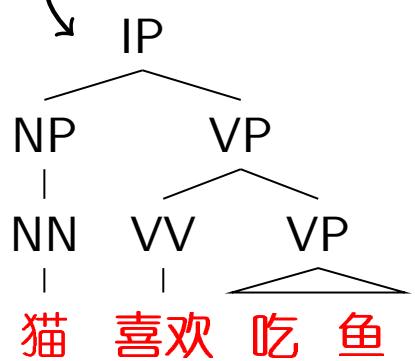
真实的机器翻译(MT)系统 - 词序列？树？



输入和输出是啥样子？

真实的机器翻译(MT)系统 - 词序列？树？



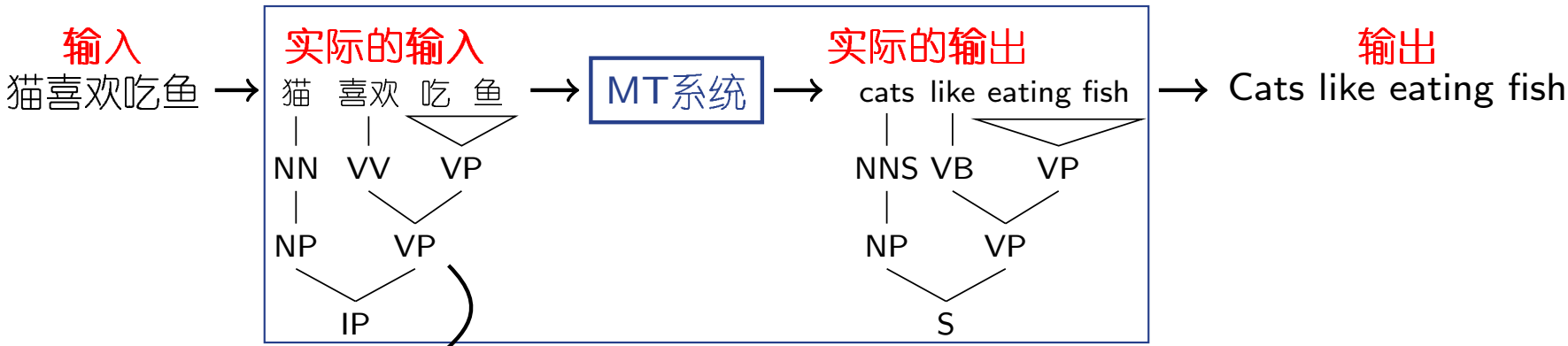


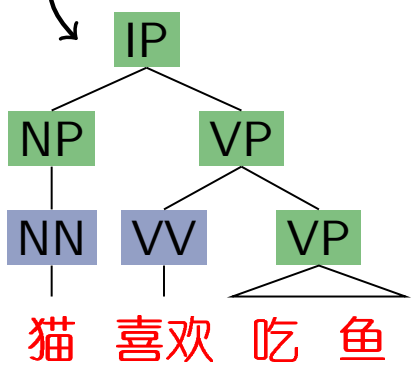
啥东东？？？

1. 句子按单词切开

输入和输出是啥样子？

真实的机器翻译(MT)系统 - 词序列？树？



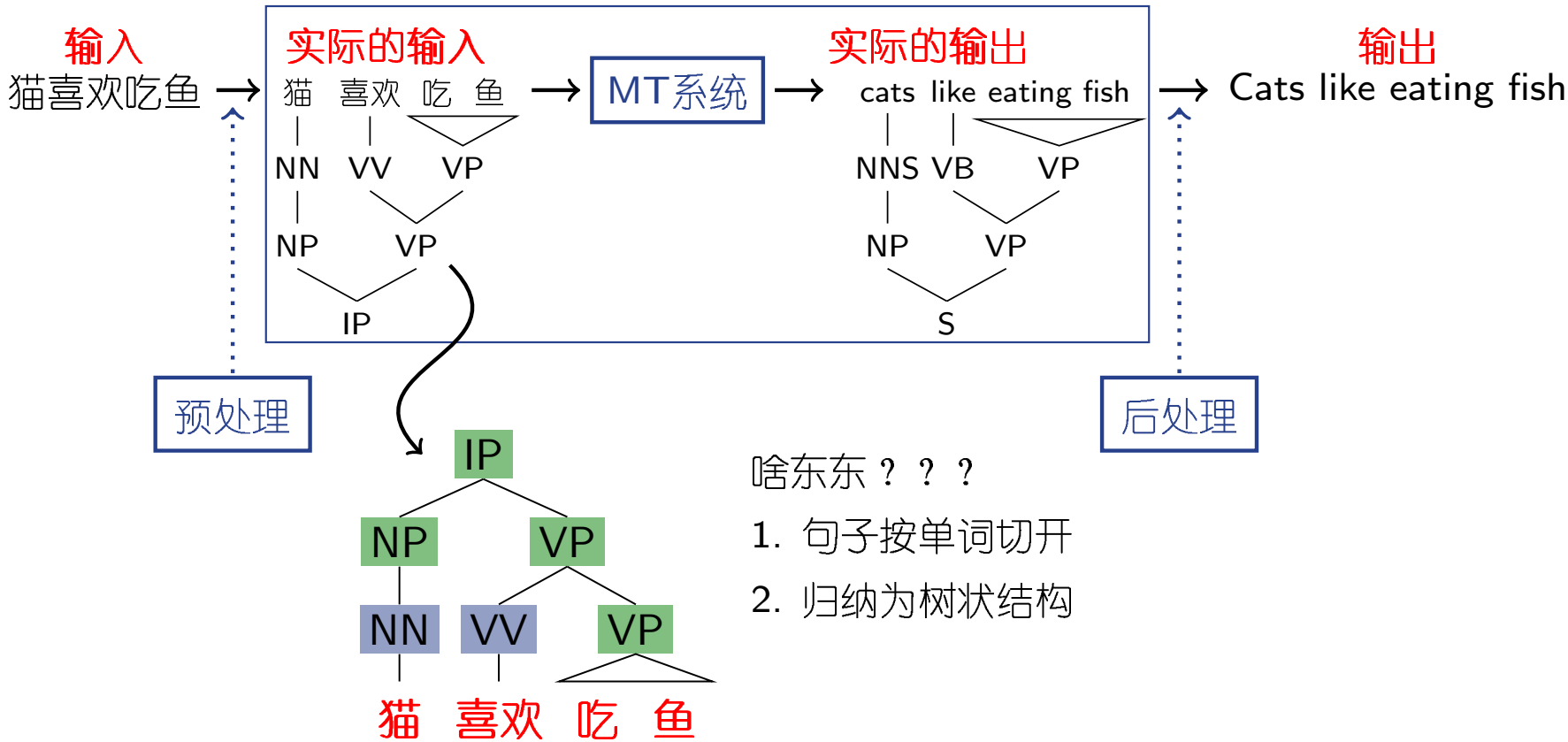


啥东东？？？

1. 句子按单词切开
2. 归纳为树状结构

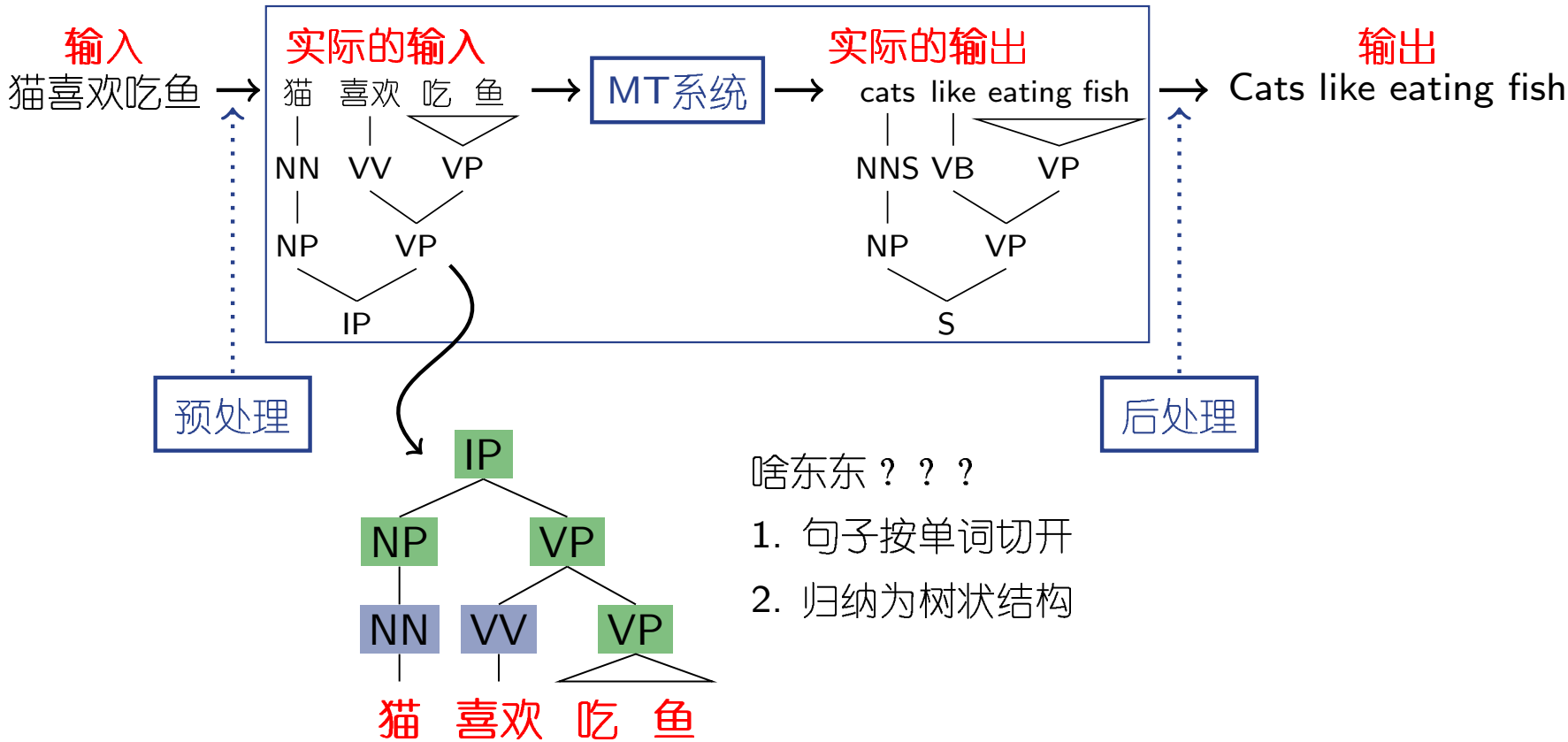
输入和输出是啥样子？

真实的机器翻译(MT)系统 - 词序列？树？



输入和输出是啥样子？

真实的机器翻译(MT)系统 - 词序列？树？



机器翻译系统依赖预处理和后处理系统，以汉英翻译为例：

- 预处理：对输入句子进行单词切分，有时进行专名识别和句法分析
- 后处理：对输出结果进行detoken，有时进行大小写恢复

这一章都讲啥

- 机器翻译系统的输入及输出会依赖很多语言学分析工具，比如，分词、词语形态学分析、专名识别/翻译、词性标注、短语结构句法分析等等

这一章都讲啥

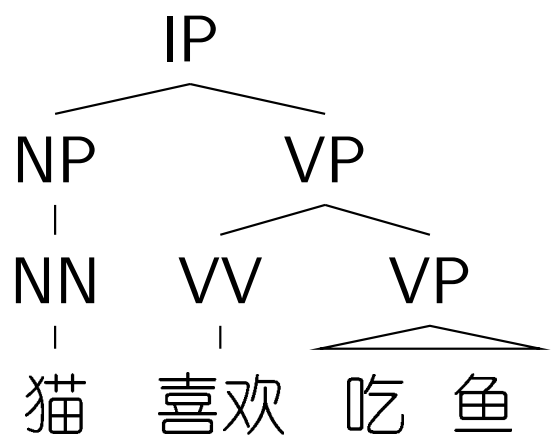
- 机器翻译系统的输入及输出会依赖很多语言学分析工具，比如，分词、词语形态学分析、专名识别/翻译、词性标注、短语结构句法分析等等
- 不同翻译任务所使用的语言学工具不同，我们这里以汉语为例，介绍三方面内容：
  - （中文）分词：将句子按单词进行切割

猫喜欢吃鱼

猫 / 喜欢 / 吃 / 鱼

这一章都讲啥

- 机器翻译系统的输入及输出会依赖很多语言学分析工具，比如，分词、词语形态学分析、专名识别/翻译、词性标注、短语结构句法分析等等
- 不同翻译任务所使用的语言学工具不同，我们这里以汉语为例，介绍三方面内容：
  - （中文）分词：将句子按单词进行切割
  - 短语结构分析：将单词序列表示为短语结构树

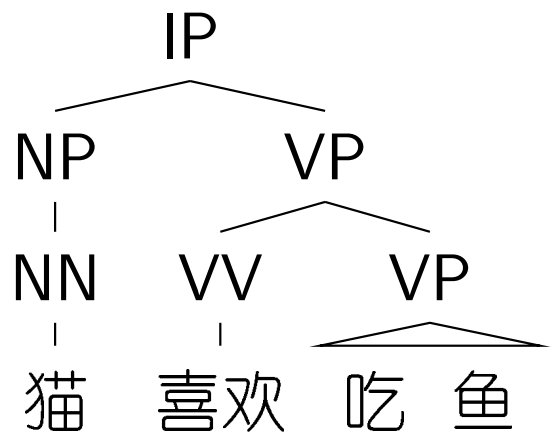


这一章都讲啥

- 机器翻译系统的输入及输出会依赖很多语言学分析工具，比如，分词、词语形态学分析、专名识别/翻译、词性标注、短语结构句法分析等等
- 不同翻译任务所使用的语言学工具不同，我们这里以汉语为例，介绍三方面内容：
  - （中文）分词：将句子按单词进行切割
  - 短语结构分析：将单词序列表示为短语结构树
  - 概率思想介绍：上面两者的概率化描述 - 作为统计机器翻译内容介绍的基础

猫喜欢吃鱼

P(猫 / 喜欢 / 吃 / 鱼)=?

P(

)=?

Outline

1. 词法分析
   - 中文分词 + 统计思想介绍

2. 句法分析
   - 中文短语结构树分析

词法分析

- 对于统计机器翻译系统而言，输入的是已经切分好的单词序列，而不是原始的字符串
  - 我们需要通过词法分析系统来得到这种切分

- 词法分析通常是指将输入的符号序列转化为单词序列的过程，这里我们称之为分词

  Canyouseemywords? → Can you see my words ?

词法分析

- 对于统计机器翻译系统而言，输入的是已经切分好的单词序列，而不是原始的字符串
  - 我们需要通过词法分析系统来得到这种切分

- 词法分析通常是指将输入的符号序列转化为单词序列的过程，这里我们称之为分词

  Canyouseemywords? → Can you see my words ?
- 几乎所有语言的句子都需要经过上述切分才能变成机器翻译系统的输入
  - 比如，中文、日文中单词间没有空格
  - 英文也需要上述切分，主要是处理标点（比如句号）和周围单词的粘连
  - 过程可能非常复杂，如进行复杂的词汇形态学分析

词法分析

- 对于统计机器翻译系统而言，输入的是已经切分好的单词序列，而不是原始的字符串
  - 我们需要通过词法分析系统来得到这种切分

- 词法分析通常是指将输入的符号序列转化为单词序列的过程，这里我们称之为分词

  Canyouseemywords? → Can you see my words ?
- 几乎所有语言的句子都需要经过上述切分才能变成机器翻译系统的输入
  - 比如，中文、日文中单词间没有空格
  - 英文也需要上述切分，主要是处理标点（比如句号）和周围单词的粘连
  - 过程可能非常复杂，如进行复杂的词汇形态学分析

统计机器翻译系统的开发依赖性能优良的分词系统！

中文分词任务定义

中文分词就是把中文句子按词切割开

- 小学生都能很容易地完成 - 我们从小的语言学习就是以词为单位的
- 什么是"词"或"单词"？

"词"的定义

词是最小的能够独立运用的语言单位。

— *百度百科*

单词（word），含有语义内容或语用内容，且能被单独念出来的最小单位。

— *维基百科*

单辞亦作"单词"，谓极简短的言词。

— *辞海在线*

中文分词任务定义

中文分词就是把中文句子按词切割开

- 小学生都能很容易地完成 - 我们从小的语言学习就是以词为单位的
- 什么是"词"或"单词"？

"词"的定义

词是最小的能够独立运用的语言单位。

— *百度百科*

单词（word），含有语义内容或语用内容，且能被单独念出来的最小单位。

— *维基百科*

单辞亦作"单词"，谓极简短的言词。

— *辞海在线*

- 问题来了！！以上这种定义计算机是无法理解的，需要其它的方式来让计算机能够进行中文分词

基于词典的分词方法

现在给计算机一本词典（可读），里面定义了所有的单词

- 只需扫描输入句子，查询每个词串是否出现在词典中

基于词典的分词方法

现在给计算机一本词典（可读），里面定义了所有的单词

- 只需扫描输入句子，查询每个词串是否出现在词典中

分词词典

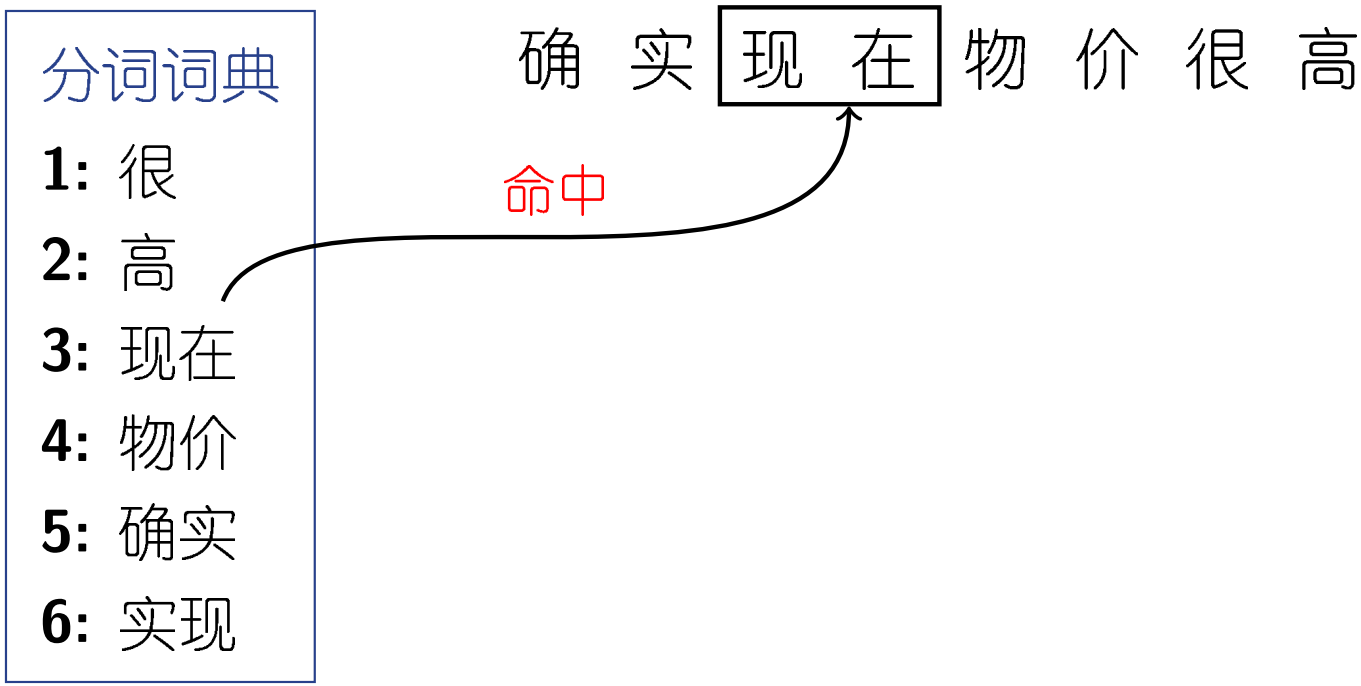
1: 很
2: 高
3: 现在
4: 物价
5: 确实
6: 实现

确 实 现 在 物 价 很 高

基于词典的分词方法

现在给计算机一本词典（可读），里面定义了所有的单词

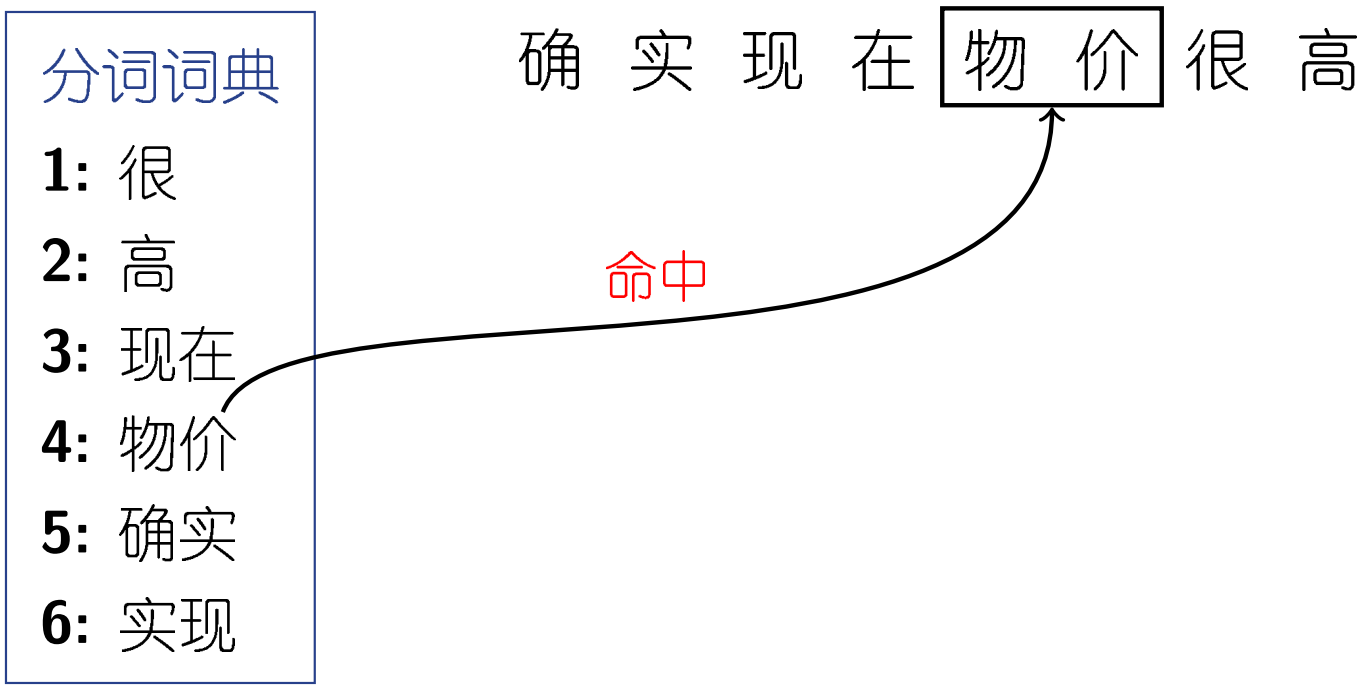
- 只需扫描输入句子，查询每个词串是否出现在词典中



基于词典的分词方法

现在给计算机一本词典（可读），里面定义了所有的单词

- 只需扫描输入句子，查询每个词串是否出现在词典中



基于词典的分词方法

现在给计算机一本词典（可读），里面定义了所有的单词

- 只需扫描输入句子，查询每个词串是否出现在词典中

分词词典

1: 很
2: 高
3: 现在
4: 物价
5: 确实
6: 实现

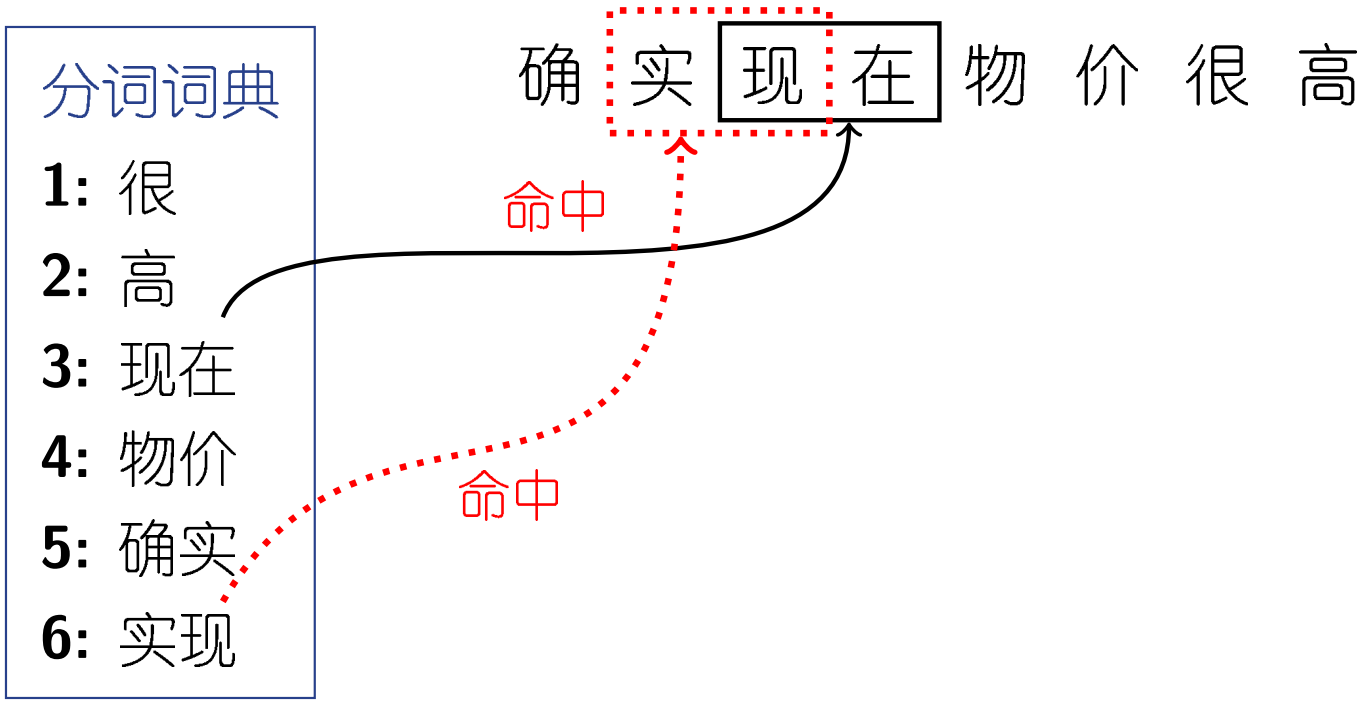
确 实 现 在 物 价 很 高

- 但是基于词典的方法很"硬"，要面对很多歧义
  - 常见问题：合法的单词之间有重叠 - 交叉型歧义

基于词典的分词方法

现在给计算机一本词典（可读），里面定义了所有的单词

- 只需扫描输入句子，查询每个词串是否出现在词典中



- 但是基于词典的方法很"硬"，要面对很多歧义
  - 常见问题：合法的单词之间有重叠 - 交叉型歧义

基于词典的分词方法

现在给计算机一本词典（可读），里面定义了所有的单词

- 只需扫描输入句子，查询每个词串是否出现在词典中

分词词典

1: 很
2: 高
3: 现在
4: 物价
5: 确实
6: 实现

确 实 现 在 物 价 很 高

- 但是基于词典的方法很"硬"，要面对很多歧义
  - 常见问题：合法的单词之间有重叠 - 交叉型歧义
  - 解决方案：自左向右扫描 + 最大匹配

基于词典的分词方法

现在给计算机一本词典（可读），里面定义了所有的单词

- 只需扫描输入句子，查询每个词串是否出现在词典中

起始
↓
确 实 现 在 物 价 很 高

分词词典

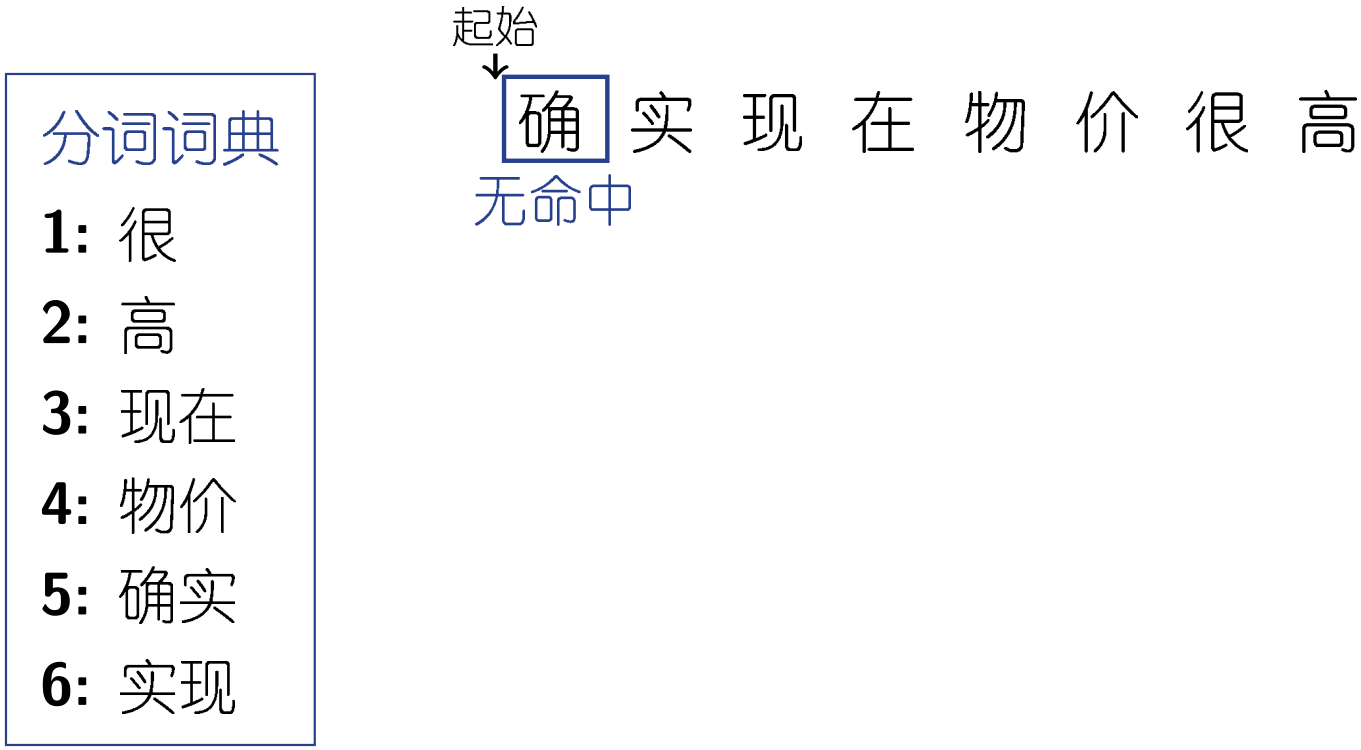
1: 很
2: 高
3: 现在
4: 物价
5: 确实
6: 实现

- 但是基于词典的方法很"硬"，要面对很多歧义
  - 常见问题：合法的单词之间有重叠 - 交叉型歧义
  - 解决方案：自左向右扫描 + 最大匹配

基于词典的分词方法

现在给计算机一本词典（可读），里面定义了所有的单词

- 只需扫描输入句子，查询每个词串是否出现在词典中

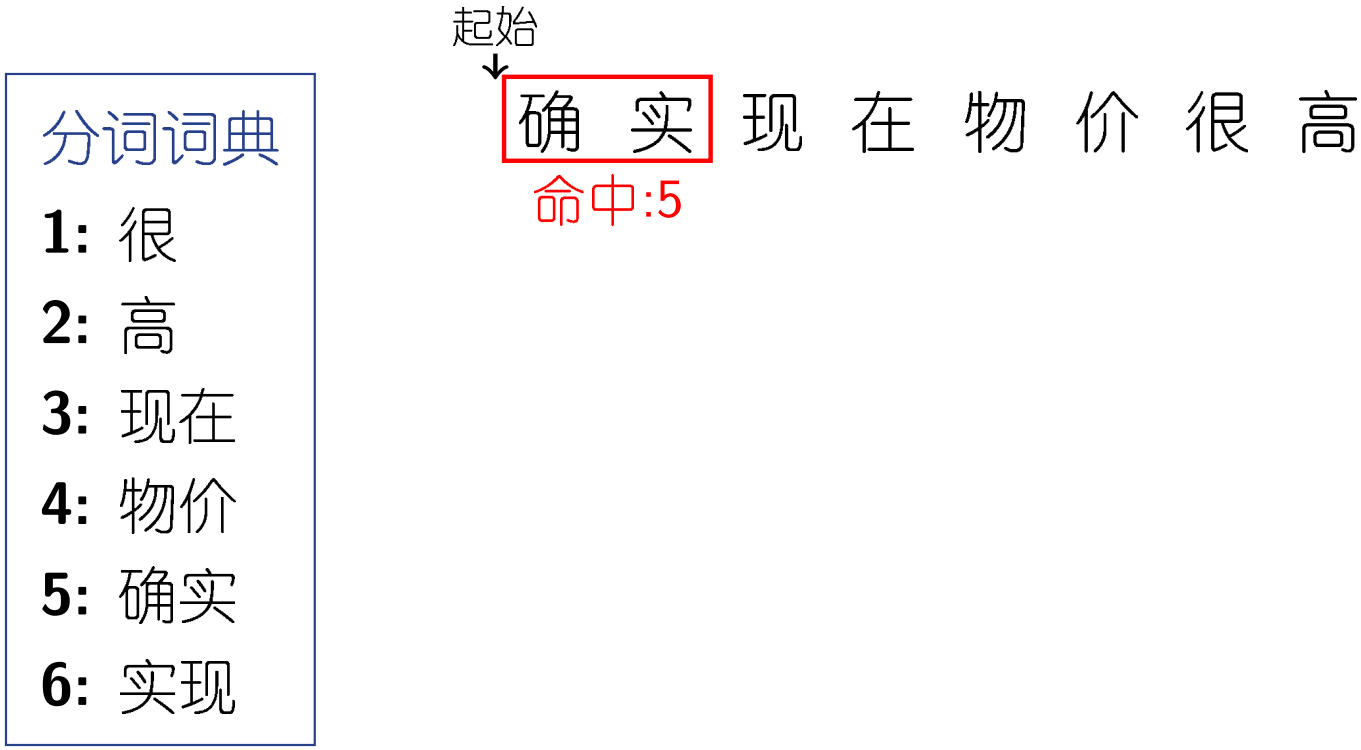


- 但是基于词典的方法很"硬"，要面对很多歧义
  - 常见问题：合法的单词之间有重叠 - 交叉型歧义
  - 解决方案：自左向右扫描 + 最大匹配

基于词典的分词方法

现在给计算机一本词典（可读），里面定义了所有的单词

- 只需扫描输入句子，查询每个词串是否出现在词典中

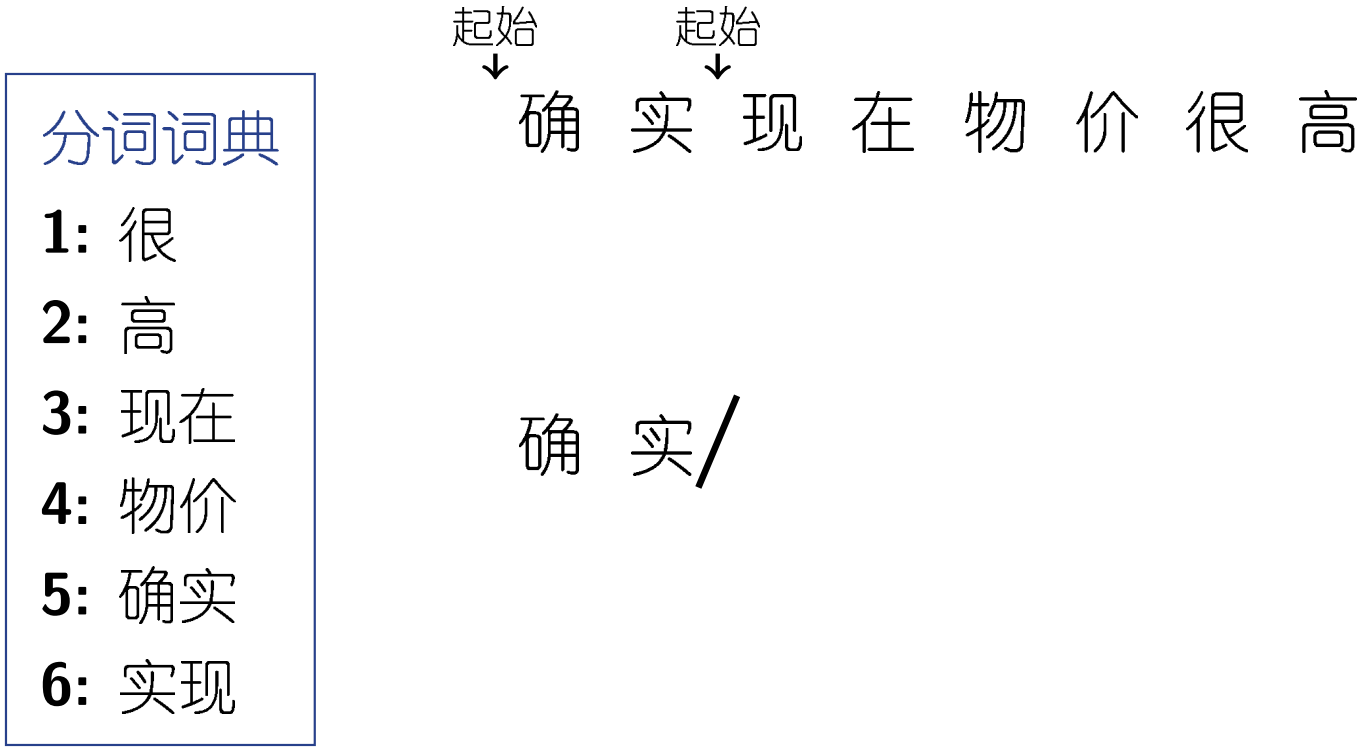


- 但是基于词典的方法很"硬"，要面对很多歧义
  - 常见问题：合法的单词之间有重叠 - 交叉型歧义
  - 解决方案：自左向右扫描 + 最大匹配

基于词典的分词方法

现在给计算机一本词典（可读），里面定义了所有的单词

- 只需扫描输入句子，查询每个词串是否出现在词典中

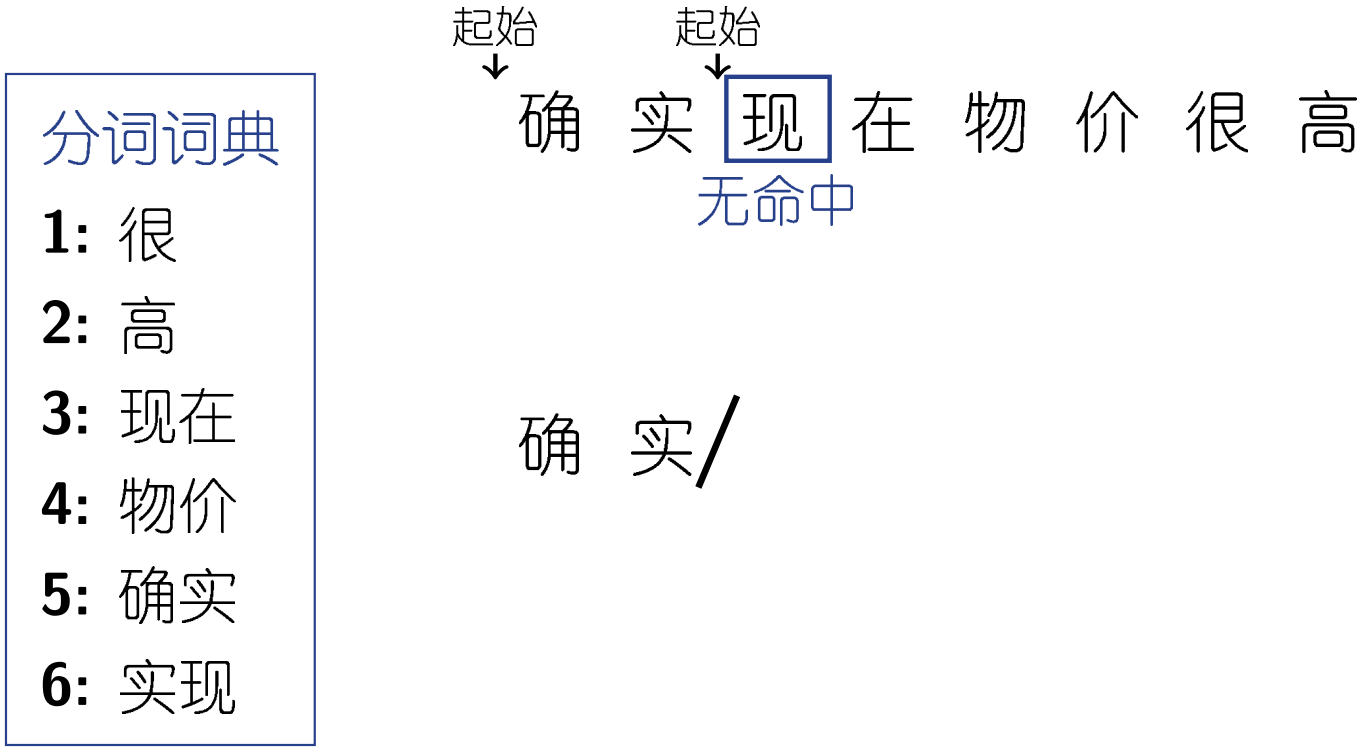


- 但是基于词典的方法很"硬"，要面对很多歧义
  - 常见问题：合法的单词之间有重叠 - 交叉型歧义
  - 解决方案：自左向右扫描 + 最大匹配

基于词典的分词方法

现在给计算机一本词典（可读），里面定义了所有的单词

- 只需扫描输入句子，查询每个词串是否出现在词典中

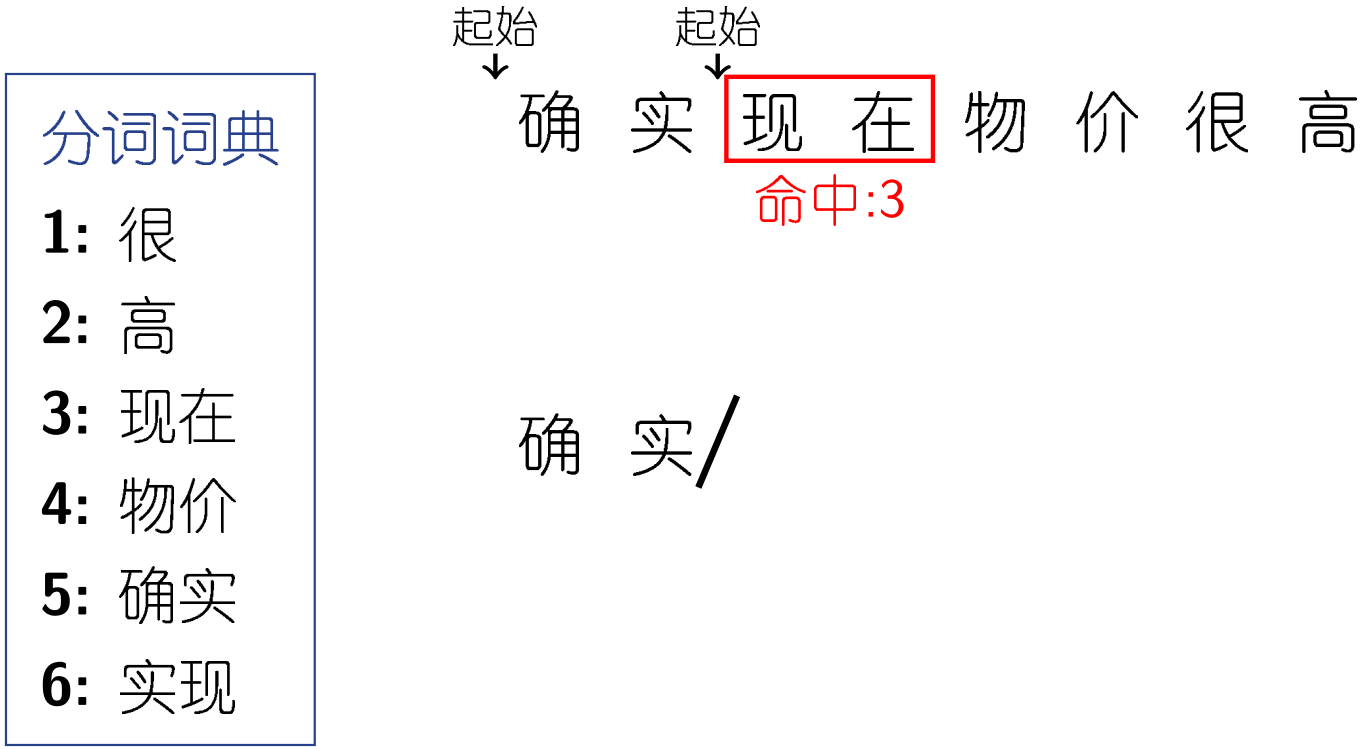


- 但是基于词典的方法很"硬"，要面对很多歧义
  - 常见问题：合法的单词之间有重叠 - 交叉型歧义
  - 解决方案：自左向右扫描 + 最大匹配

基于词典的分词方法

现在给计算机一本词典（可读），里面定义了所有的单词

- 只需扫描输入句子，查询每个词串是否出现在词典中

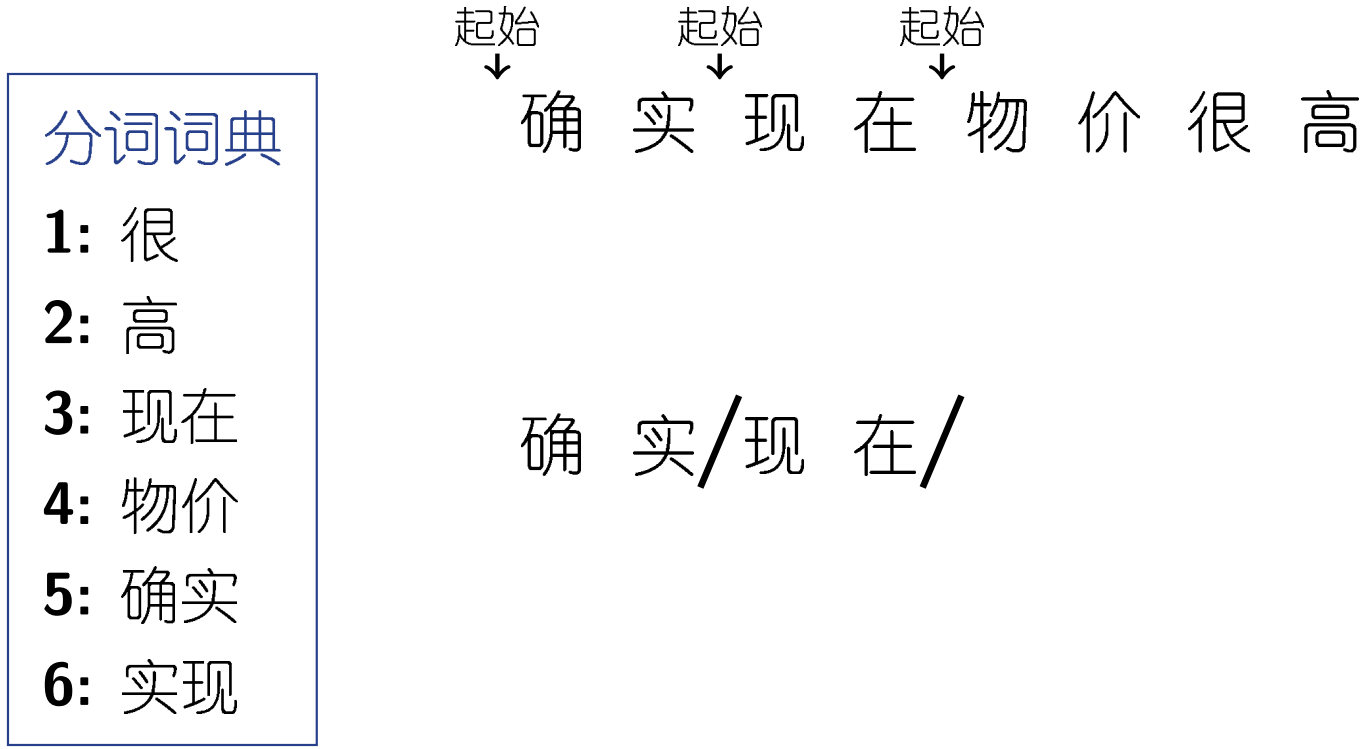


- 但是基于词典的方法很"硬"，要面对很多歧义
  - 常见问题：合法的单词之间有重叠 - 交叉型歧义
  - 解决方案：自左向右扫描 + 最大匹配

基于词典的分词方法

现在给计算机一本词典（可读），里面定义了所有的单词

- 只需扫描输入句子，查询每个词串是否出现在词典中

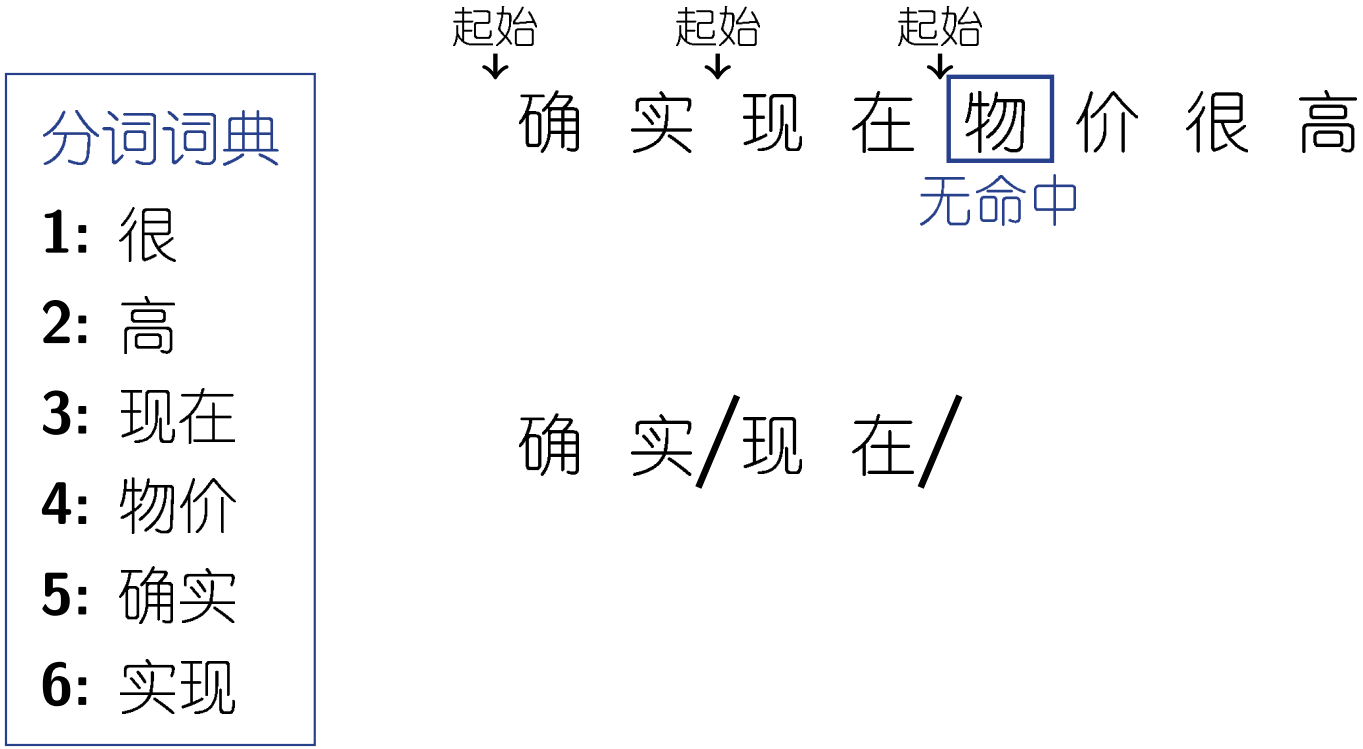


- 但是基于词典的方法很"硬"，要面对很多歧义
  - 常见问题：合法的单词之间有重叠 - 交叉型歧义
  - 解决方案：自左向右扫描 + 最大匹配

基于词典的分词方法

现在给计算机一本词典（可读），里面定义了所有的单词

- 只需扫描输入句子，查询每个词串是否出现在词典中

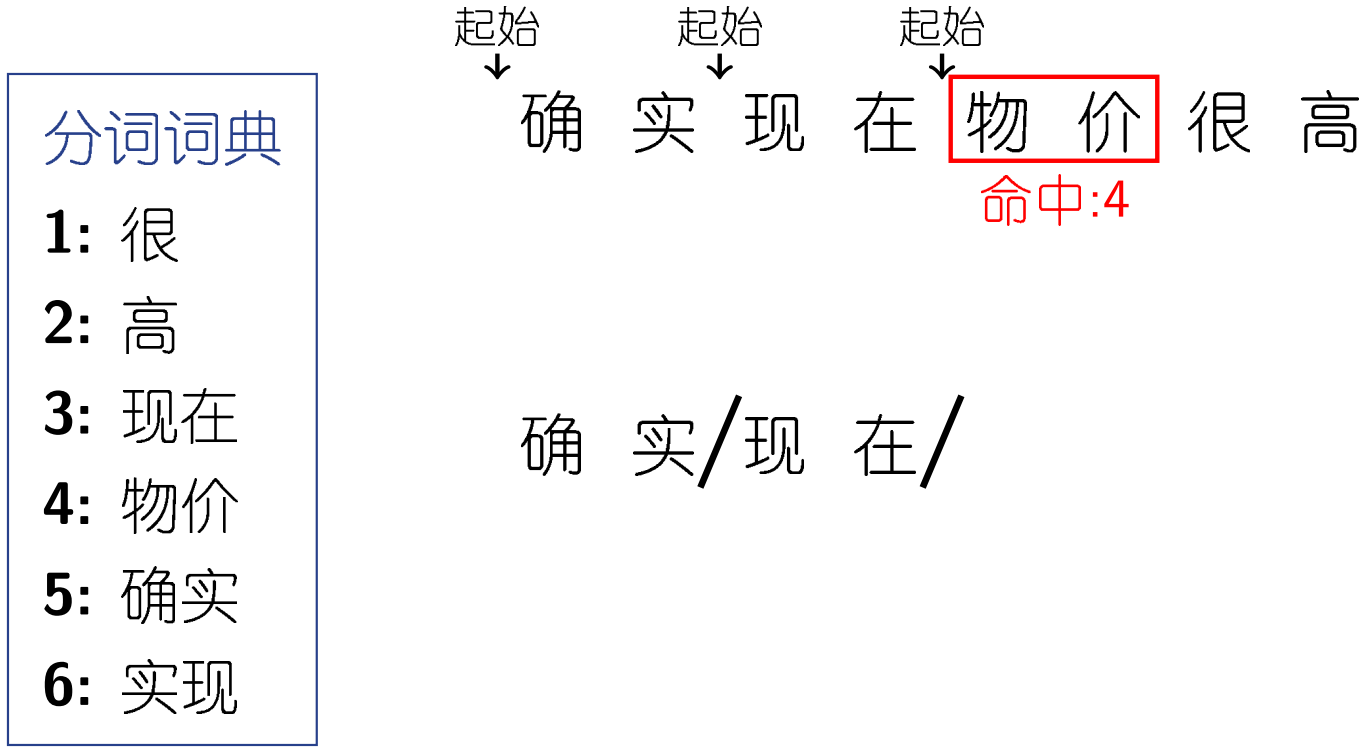


- 但是基于词典的方法很"硬"，要面对很多歧义
  - 常见问题：合法的单词之间有重叠 - 交叉型歧义
  - 解决方案：自左向右扫描 + 最大匹配

基于词典的分词方法

现在给计算机一本词典（可读），里面定义了所有的单词

- 只需扫描输入句子，查询每个词串是否出现在词典中

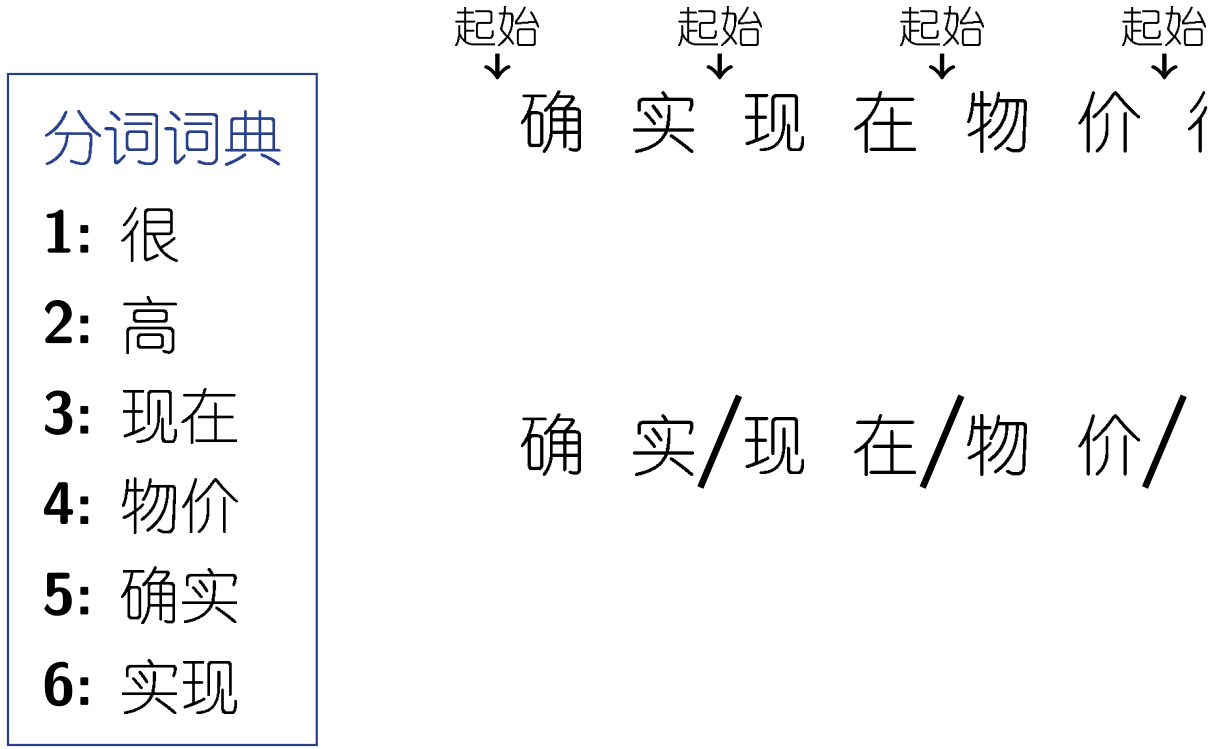


- 但是基于词典的方法很"硬"，要面对很多歧义
  - 常见问题：合法的单词之间有重叠 - 交叉型歧义
  - 解决方案：自左向右扫描 + 最大匹配

基于词典的分词方法

现在给计算机一本词典（可读），里面定义了所有的单词

- 只需扫描输入句子，查询每个词串是否出现在词典中

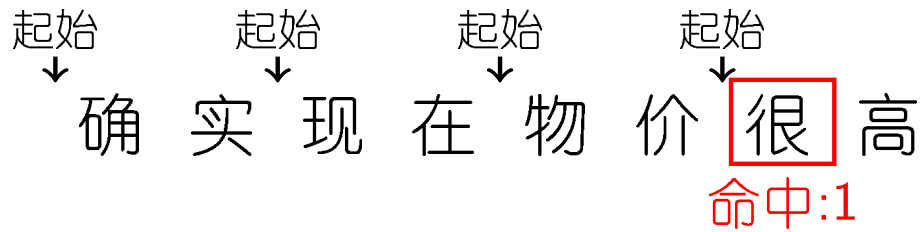


- 但是基于词典的方法很"硬"，要面对很多歧义
  - 常见问题：合法的单词之间有重叠 - 交叉型歧义
  - 解决方案：自左向右扫描 + 最大匹配

基于词典的分词方法

现在给计算机一本词典（可读），里面定义了所有的单词

- 只需扫描输入句子，查询每个词串是否出现在词典中



分词词典

1: 很
2: 高
3: 现在
4: 物价
5: 确实
6: 实现

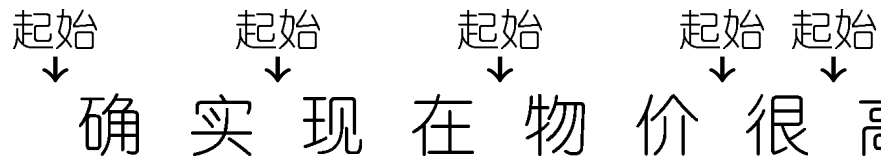
确 实/现 在/物 价/

- 但是基于词典的方法很"硬"，要面对很多歧义
  - 常见问题：合法的单词之间有重叠 - 交叉型歧义
  - 解决方案：自左向右扫描 + 最大匹配

基于词典的分词方法

现在给计算机一本词典（可读），里面定义了所有的单词

- 只需扫描输入句子，查询每个词串是否出现在词典中



分词词典

1: 很
2: 高
3: 现在
4: 物价
5: 确实
6: 实现

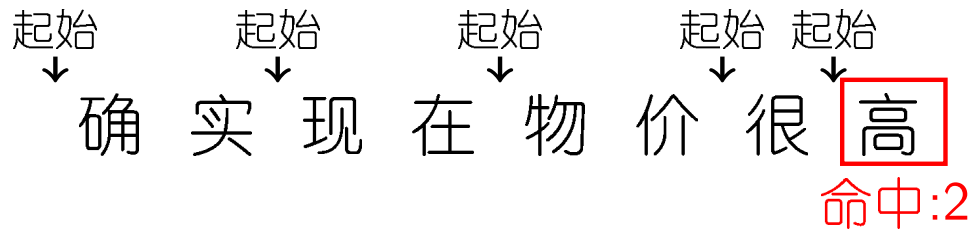
确 实/现 在/物 价/很/

- 但是基于词典的方法很"硬"，要面对很多歧义
  - 常见问题：合法的单词之间有重叠 - 交叉型歧义
  - 解决方案：自左向右扫描 + 最大匹配

基于词典的分词方法

现在给计算机一本词典（可读），里面定义了所有的单词

- 只需扫描输入句子，查询每个词串是否出现在词典中



分词词典

1: 很
2: 高
3: 现在
4: 物价
5: 确实
6: 实现

确 实/现 在/物 价/很/高

- 但是基于词典的方法很"硬"，要面对很多歧义
  - 常见问题：合法的单词之间有重叠 - 交叉型歧义
  - 解决方案：自左向右扫描 + 最大匹配

基于统计的方法

- 基于词典的方法是典型的基于规则的方法
  - 词典需要人工给定，很多时候词典的覆盖度有限
  - 出现歧义的地方需要人工定义消除歧义的规则

基于统计的方法

- 基于词典的方法是典型的基于规则的方法
  - 词典需要人工给定，很多时候词典的覆盖度有限
  - 出现歧义的地方需要人工定义消除歧义的规则
- 另外一种思路是利用"数据"来完成"定义"，让计算机直接从数据中学习知识
  - 也就是我们常说的数据驱动的方法
  - 这个过程也是一个典型的统计学习的过程

基于统计的方法

- 基于词典的方法是典型的基于规则的方法
  - 词典需要人工给定，很多时候词典的覆盖度有限
  - 出现歧义的地方需要人工定义消除歧义的规则
- 另外一种思路是利用"数据"来完成"定义"，让计算机直接从数据中学习知识
  - 也就是我们常说的数据驱动的方法
  - 这个过程也是一个典型的统计学习的过程
- 何为数据驱动？何为统计学习？在分词任务中：
  - **(有标注)数据**：经过人工分词好的句子

学习用数据

1: 这 / 是 / 数据

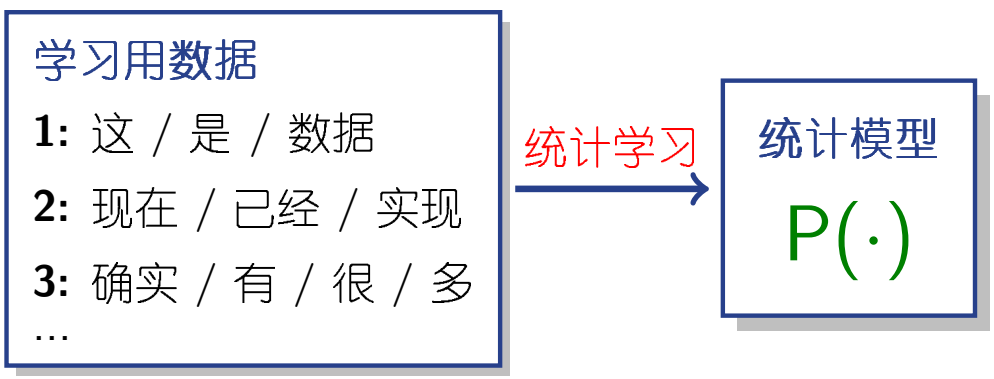
2: 现在 / 已经 / 实现

3: 确实 / 有 / 很 / 多

...

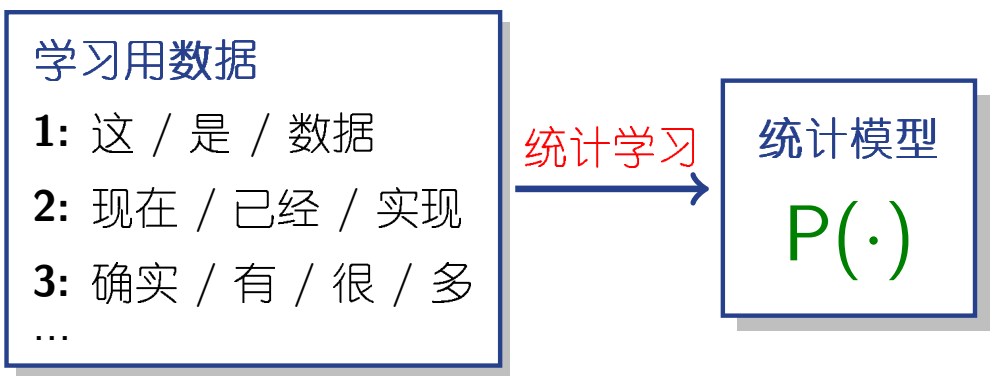
基于统计的方法

- 基于词典的方法是典型的基于规则的方法
  - 词典需要人工给定，很多时候词典的覆盖度有限
  - 出现歧义的地方需要人工定义消除歧义的规则
- 另外一种思路是利用"数据"来完成"定义"，让计算机直接从数据中学习知识
  - 也就是我们常说的数据驱动的方法
  - 这个过程也是一个典型的统计学习的过程
- 何为数据驱动？何为统计学习？在分词任务中：
  - **(有标注)数据**：经过人工分词好的句子
  - **统计学习**：从数据中学习一种对分词现象的统计描述，即概率函数P(·)



基于统计的方法

- 基于词典的方法是典型的基于规则的方法
  - 词典需要人工给定，很多时候词典的覆盖度有限
  - 出现歧义的地方需要人工定义消除歧义的规则
- 另外一种思路是利用"数据"来完成"定义"，让计算机直接从数据中学习知识
  - 也就是我们常说的数据驱动的方法
  - 这个过程也是一个典型的统计学习的过程
- 何为数据驱动？何为统计学习？在分词任务中：
  - **(有标注)数据**：经过人工分词好的句子
  - **统计学习**：从数据中学习一种对分词现象的统计描述，即概率函数P(·)

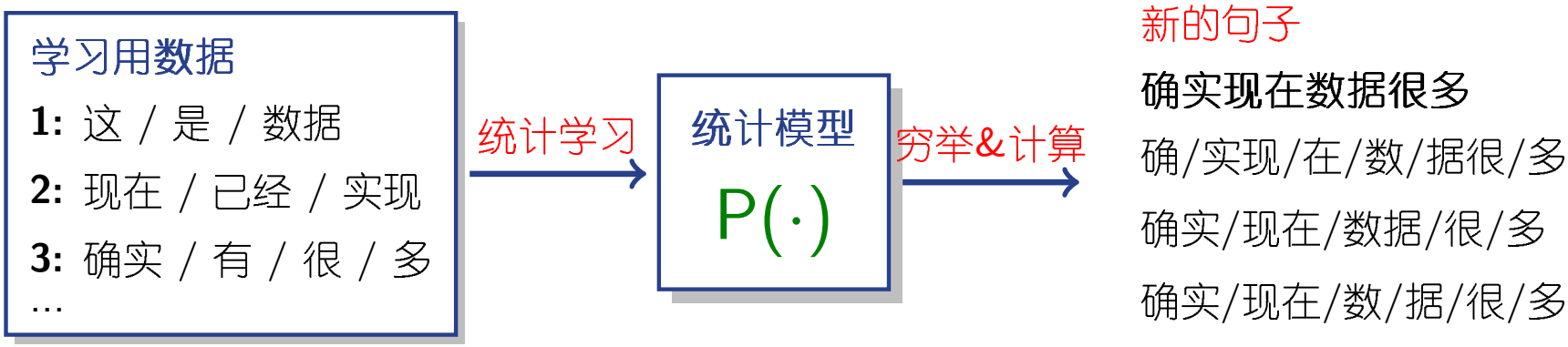


新的句子

确实现在数据很多

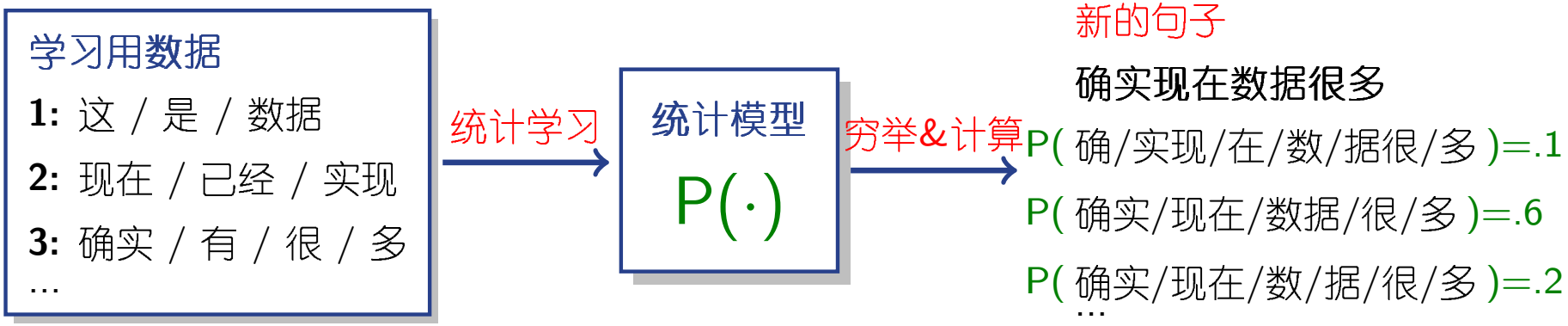
基于统计的方法

- 基于词典的方法是典型的基于规则的方法
  - 词典需要人工给定，很多时候词典的覆盖度有限
  - 出现歧义的地方需要人工定义消除歧义的规则
- 另外一种思路是利用"数据"来完成"定义"，让计算机直接从数据中学习知识
  - 也就是我们常说的数据驱动的方法
  - 这个过程也是一个典型的统计学习的过程
- 何为数据驱动？何为统计学习？在分词任务中：
  - **(有标注)数据**：经过人工分词好的句子
  - **统计学习**：从数据中学习一种对分词现象的统计描述，即概率函数P(·)



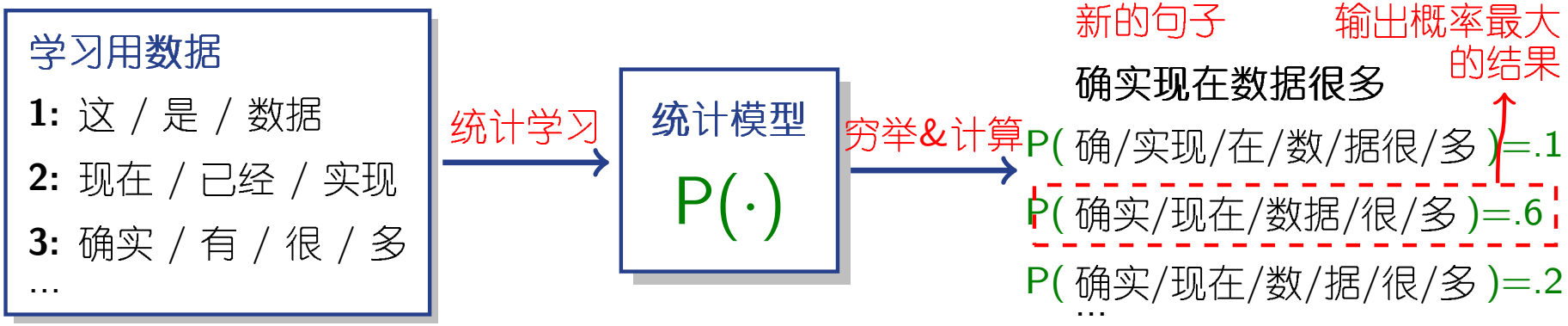
基于统计的方法

- 基于词典的方法是典型的基于规则的方法
  - 词典需要人工给定，很多时候词典的覆盖度有限
  - 出现歧义的地方需要人工定义消除歧义的规则
- 另外一种思路是利用"数据"来完成"定义"，让计算机直接从数据中学习知识
  - 也就是我们常说的数据驱动的方法
  - 这个过程也是一个典型的统计学习的过程
- 何为数据驱动？何为统计学习？在分词任务中：
  - **(有标注)数据**：经过人工分词好的句子
  - **统计学习**：从数据中学习一种对分词现象的统计描述，即概率函数P(·)



基于统计的方法

- 基于词典的方法是典型的基于规则的方法
  - 词典需要人工给定，很多时候词典的覆盖度有限
  - 出现歧义的地方需要人工定义消除歧义的规则
- 另外一种思路是利用"数据"来完成"定义"，让计算机直接从数据中学习知识
  - 也就是我们常说的数据驱动的方法
  - 这个过程也是一个典型的统计学习的过程
- 何为数据驱动？何为统计学习？在分词任务中：
  - **(有标注)数据**：经过人工分词好的句子
  - **统计学习**：从数据中学习一种对分词现象的统计描述，即概率函数P(·)



P(·)是什么？

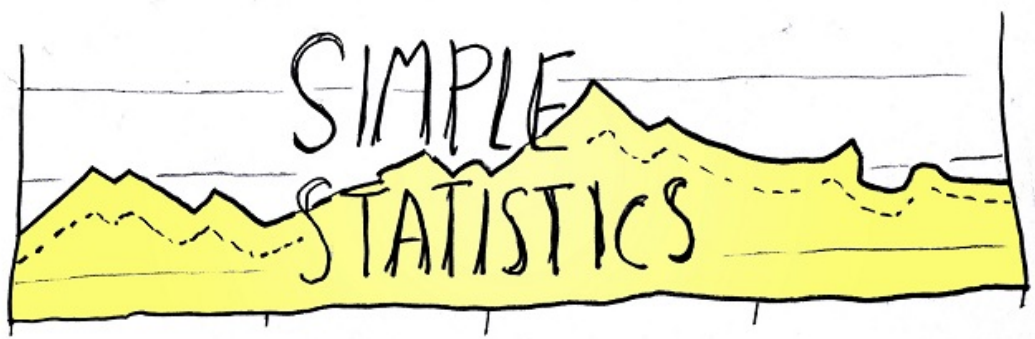
- P(·)表示一个随机事件的可能性，即事件发生的概率
  - P(太阳从东方升起)：”太阳从东方升起”这件事的可能性
  - P($A = B$)：”A=B”这件事的可能性

P(·)是什么？

- P(·)表示一个随机事件的可能性，即事件发生的概率
  - P(太阳从东方升起)：”太阳从东方升起”这件事的可能性
  - P($A = B$)：”A=B”这件事的可能性
- 如果把A看做一个(离散)变量，a看做变量A的一个取值
  - P(A)被称作A的概率函数
  - P($A = a$)被称作$A = a$的概率值，简记为P(a)
  - **非负性**：$\forall x$, P(x) ≥ 0；**归一性**：$\sum_x \mathrm{P}(x) = 1$
  - **联合概率**：两个事件A和B同时出现的概率，记为P(AB)
  - **条件概率**：事件A出现的前提下和B出现的概率，记为P($B|A$)

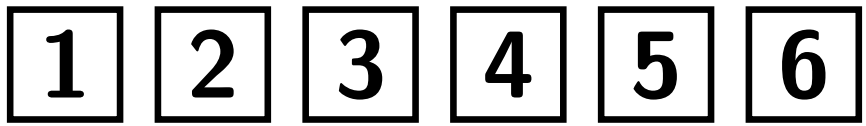
P(·)是什么？

- P(·)表示一个随机事件的可能性，即事件发生的概率
  - P(太阳从东方升起)：”太阳从东方升起”这件事的可能性
  - P($A = B$)：”A=B”这件事的可能性
- 如果把A看做一个(离散)变量，a看做变量A的一个取值
  - P(A)被称作A的概率函数
  - P($A = a$)被称作$A = a$的概率值，简记为P(a)
  - **非负性**：$\forall x$, P(x) ≥ 0；**归一性**：$\sum_x \mathrm{P}(x) = 1$
  - **联合概率**：两个事件A和B同时出现的概率，记为P(AB)
  - **条件概率**：事件A出现的前提下和B出现的概率，记为P($B|A$)
- 所谓统计建模是用概率来描述要解决的问题，就这么简单!!!



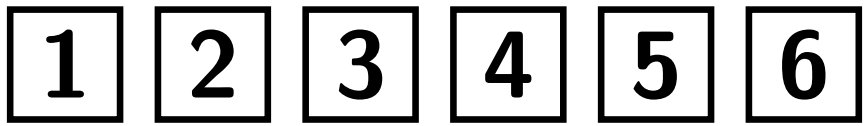
看一个实例：扯远一点儿 - 骰子游戏

- 掷一个骰(tóu)子（1-6），你押一个数字，压中算你赢



看一个实例：扯远一点儿 - 骰子游戏

- 掷一个骰(tóu)子（1-6），你押一个数字，压中算你赢

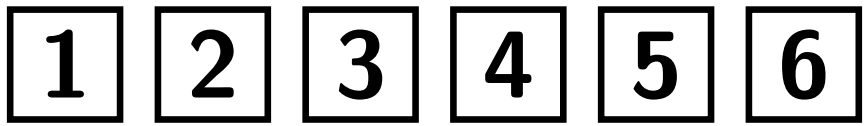


我说：

- 很难赢，六个面，你只能压一个面
- 也能赢，除非你的运气极好

看一个实例：扯远一点儿 - 骰子游戏

- 掷一个骰(tóu)子（1-6），你押一个数字，压中算你赢



我说：

  - 很难赢，六个面，你只能压一个面
  - 也能赢，除非你的运气极好

- 不管了，随便压一个数字，比如：1。掷30次，结果如何

看一个实例：扯远一点儿 - 骰子游戏

- 掷一个骰(tóu)子（1-6），你押一个数字，压中算你赢

1	2	3	4	5	6

我说：

- 很难赢，六个面，你只能压一个面
- 也能赢，除非你的运气极好

- 不管了，随便压一个数字，比如：1。掷30次，结果如何

2	3	1	4	4	1	5	1	4	4
5	6	4	4	3	2	1	4	5	1
4	2	2	3	4	1	5	1	3	4

看一个实例：扯远一点儿 - 骰子游戏

- 掷一个骰(tóu)子（1-6），你押一个数字，压中算你赢

1	2	3	4	5	6

我说：

- 很难赢，六个面，你只能压一个面
- 也能赢，除非你的运气极好

- 不管了，随便压一个数字，比如：1。掷30次，结果如何

2	3	1	4	4	1	5	1	4	4
5	6	4	4	3	2	1	4	5	1
4	2	2	3	4	1	5	1	3	4

命中$7/30 > 1/6$ 还不错

骰子游戏 - 建模

- 似乎你的胜利只能来源于运气
- no! no! no! "随便选一个数字"本身就是一个概率模型，它对骰子的六个面做了均匀分布的假设

$$\mathrm{P}(\text{"1"}) = \mathrm{P}(\text{"2"}) = \ldots = \mathrm{P}(\text{"5"}) = \mathrm{P}(\text{"6"}) = 1/6$$

骰子游戏 - 建模

- 似乎你的胜利只能来源于运气
- no! no! no! "随便选一个数字"本身就是一个概率模型，它对骰子的六个面做了均匀分布的假设

$$\mathrm{P}(\text{"1"}) = \mathrm{P}(\text{"2"}) = \ldots = \mathrm{P}(\text{"5"}) = \mathrm{P}(\text{"6"}) = 1/6$$

- 看出问题来了，我们可以用一种更加"聪明"的方式来定义一个新模型，定义骰子的每个面都以一定的概率出现（不一定是均匀分布）

骰子游戏 - 建模

- 似乎你的胜利只能来源于运气
- no! no! no! "随便选一个数字"本身就是一个概率模型，它对骰子的六个面做了均匀分布的假设

$$\mathrm{P}(\text{"1"}) = \mathrm{P}(\text{"2"}) = \ldots = \mathrm{P}(\text{"5"}) = \mathrm{P}(\text{"6"}) = 1/6$$

- 看出问题来了，我们可以用一种更加"聪明"的方式来定义一个新模型，定义骰子的每个面都以一定的概率出现（不一定是均匀分布）

$$\begin{aligned}
\mathrm{P}(\text{"1"}) &= \theta_1 \\
\mathrm{P}(\text{"2"}) &= \theta_2 \\
\mathrm{P}(\text{"3"}) &= \theta_3 \\
\mathrm{P}(\text{"4"}) &= \theta_4 \\
\mathrm{P}(\text{"5"}) &= \theta_5 \\
\mathrm{P}(\text{"6"}) &= 1 - \sum\nolimits_{1 \le i \le 5} \theta_i \quad \triangleleft \text{ 归一性}
\end{aligned}$$

骰子游戏 - 建模

- 似乎你的胜利只能来源于运气
- no! no! no! "随便选一个数字"本身就是一个概率模型，它对骰子的六个面做了均匀分布的假设

$$\mathrm{P}(\text{"1"}) = \mathrm{P}(\text{"2"}) = \ldots = \mathrm{P}(\text{"5"}) = \mathrm{P}(\text{"6"}) = 1/6$$

- 看出问题来了，我们可以用一种更加"聪明"的方式来定义一个新模型，定义骰子的每个面都以一定的概率出现（不一定是均匀分布）

$$\begin{aligned}
\mathrm{P}(\text{"1"}) &= \theta_1 \\
\mathrm{P}(\text{"2"}) &= \theta_2 \\
\mathrm{P}(\text{"3"}) &= \theta_3 \\
\mathrm{P}(\text{"4"}) &= \theta_4 \\
\mathrm{P}(\text{"5"}) &= \theta_5 \\
\mathrm{P}(\text{"6"}) &= 1 - \sum\nolimits_{1 \le i \le 5} \theta_i \quad \triangleleft \text{归一性}
\end{aligned}$$

问题：如何获得θ_i的值？

骰子游戏 - 模型学习

- 我们可以从大量的实例中学习模型参数。比如，先实验性的掷很多次(X次)，发现"1"出现X_1次、"2"出现X_2次...

骰子游戏 - 模型学习

- 我们可以从大量的实例中学习模型参数。比如，先实验性的掷很多次(X次)，发现"1"出现X_1次、"2"出现X_2次...
- 假设i服从多项式分布，那么各种概率的极大似然估计为

$$\mathrm{P}(\text{"}i\text{"}) = \frac{X_i}{X}$$

注意：X足够大的话，$\frac{X_i}{X}$可以无限逼近$\mathrm{P}(\text{"}i\text{"})$的真实值

何为多项式分布？何为极大似然估计??? - 自己翻翻书

骰子游戏 - 模型学习

- 我们可以从大量的实例中学习模型参数。比如，先实验性的掷很多次(X次)，发现"1"出现X_1次、"2"出现X_2次...
- 假设i服从多项式分布，那么各种概率的极大似然估计为

$$\mathrm{P}(\text{"}i\text{"}) = \frac{X_i}{X}$$

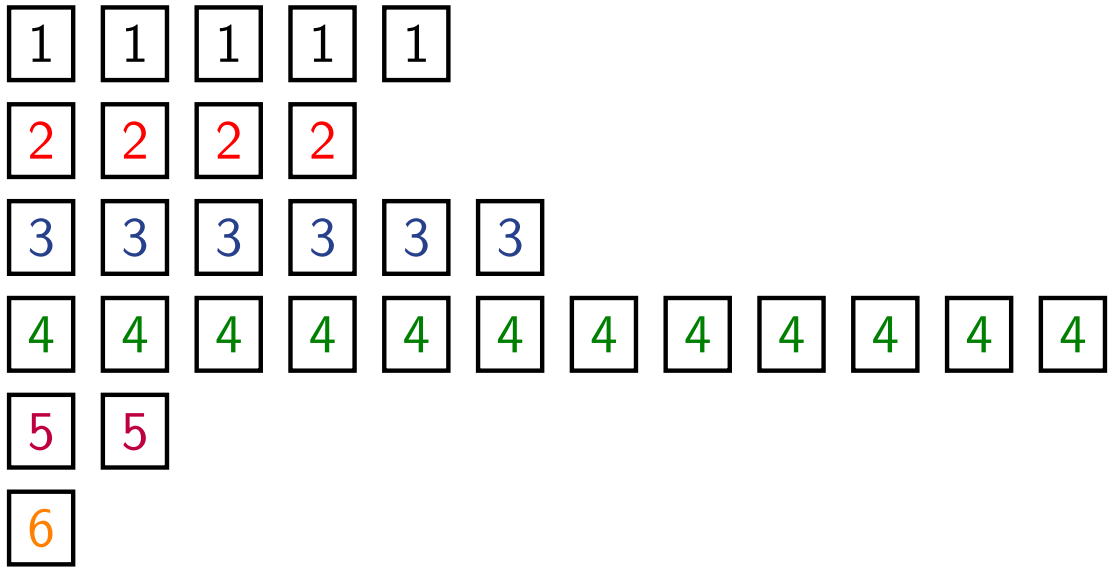
注意：X足够大的话，$\frac{X_i}{X}$可以无限逼近$\mathrm{P}(\text{"}i\text{"})$的真实值
何为多项式分布？何为极大似然估计??? - 自己翻翻书

- 重新来一次，我们在正式开始前先掷30次

3	4	2	3	4	5	1	4	4	3
2	1	4	5	4	4	4	3	1	4
4	3	2	6	1	2	3	4	4	1

骰子游戏：在新一轮游戏上实践

- 我们得到一个有倾向性的模型(a model with a bias)



$$\mathrm{P}(\text{"1"}) = 5/30$$
$$\mathrm{P}(\text{"2"}) = 4/30$$
$$\mathrm{P}(\text{"3"}) = 6/30$$
$$\mathrm{P}(\text{"4"}) = 12/30$$
$$\mathrm{P}(\text{"5"}) = 2/30$$
$$\mathrm{P}(\text{"6"}) = 1/30$$

骰子游戏：在新一轮游戏上实践

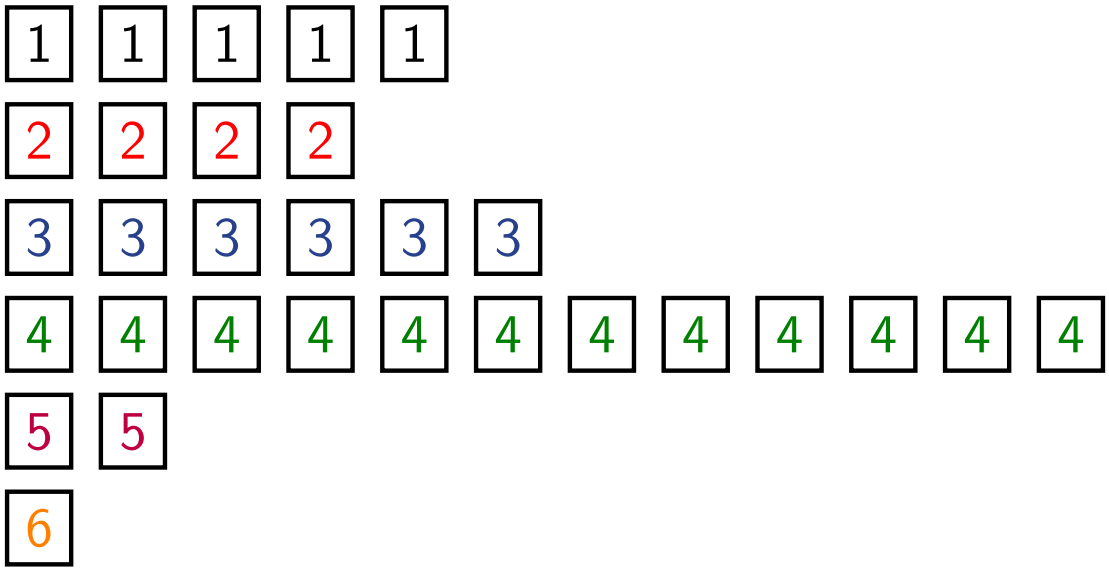
- 我们得到一个有倾向性的模型(a model with a bias)

1 1 1 1 1	$\mathrm{P}(\text{"1"}) = 5/30$
2 2 2 2	$\mathrm{P}(\text{"2"}) = 4/30$
3 3 3 3 3 3	$\mathrm{P}(\text{"3"}) = 6/30$
4 4 4 4 4 4 4 4 4 4 4 4	$\mathrm{P}(\text{"4"}) = 12/30$
5 5	$\mathrm{P}(\text{"5"}) = 2/30$
6	$\mathrm{P}(\text{"6"}) = 1/30$

- 那我们选择一个数字（显然是4），然后开始正式的游戏

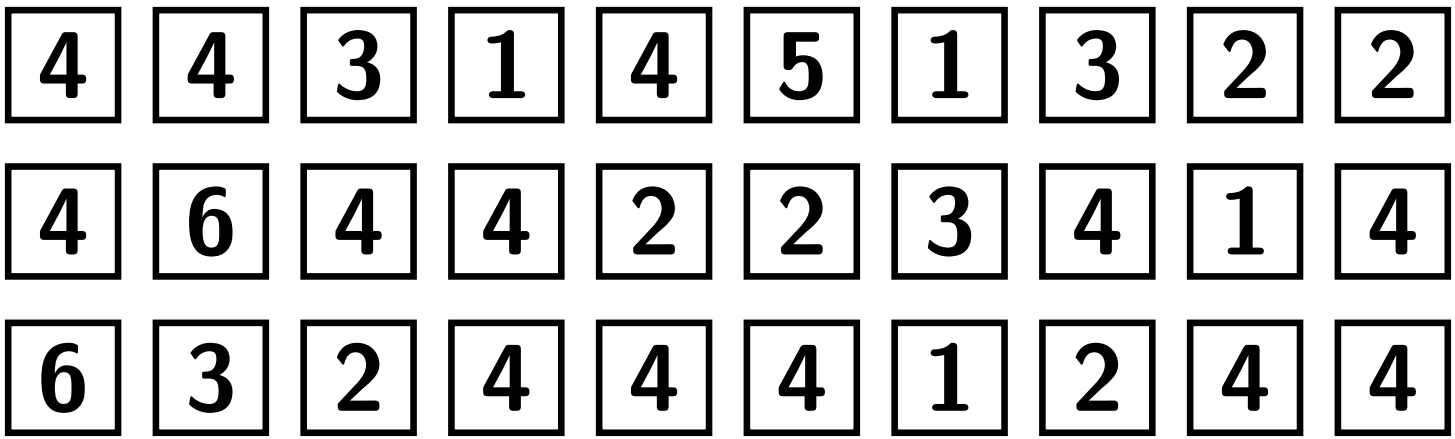
骰子游戏：在新一轮游戏上实践

- 我们得到一个有倾向性的模型(a model with a bias)



$$\mathrm{P}(\text{"1"}) = 5/30$$
$$\mathrm{P}(\text{"2"}) = 4/30$$
$$\mathrm{P}(\text{"3"}) = 6/30$$
$$\mathrm{P}(\text{"4"}) = 12/30$$
$$\mathrm{P}(\text{"5"}) = 2/30$$
$$\mathrm{P}(\text{"6"}) = 1/30$$

- 那我们选择一个数字（显然是4），然后开始正式的游戏



骰子游戏：在新一轮游戏上实践

- 我们得到一个有倾向性的模型(a model with a bias)

1 1 1 1 1	$\mathrm{P}(\text{"1"}) = 5/30$
2 2 2 2	$\mathrm{P}(\text{"2"}) = 4/30$
3 3 3 3 3 3	$\mathrm{P}(\text{"3"}) = 6/30$
4 4 4 4 4 4 4 4 4 4 4 4	$\mathrm{P}(\text{"4"}) = 12/30$
5 5	$\mathrm{P}(\text{"5"}) = 2/30$
6	$\mathrm{P}(\text{"6"}) = 1/30$

- 那我们选择一个数字（显然是4），然后开始正式的游戏

4 4 3 1 4 5 1 3 2 2

4 6 4 4 2 2 3 4 1 4

6 3 2 4 4 4 1 2 4 4

这次得到的结果是 13/30 - 非常不错!

骰子游戏 vs. 分词

假设我们有已经人工分好词的句子（称之为语料或数据），其中每个单词的出现就好比投掷一个巨大骰子

- 骰子有很多个面，每个面代表一个单词
- 骰子是不均匀的，有些面会出现比较多次

骰子游戏 vs. 分词

假设我们有已经人工分好词的句子（称之为语料或数据），其中每个单词的出现就好比投掷一个巨大骰子

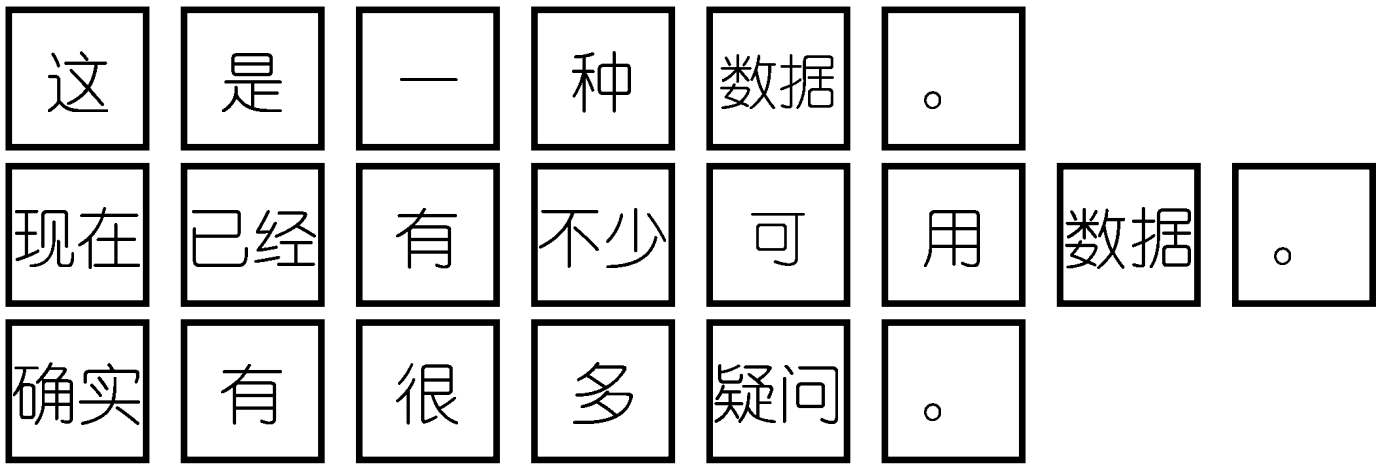
- 骰子有很多个面，每个面代表一个单词
- 骰子是不均匀的，有些面会出现比较多次

88	87	45	47	100	15		
5	230	7	234	500	39	100	15
975	7	234	294	15	15		

骰子游戏 vs. 分词

假设我们有已经人工分好词的句子（称之为语料或数据），其中每个单词的出现就好比投掷一个巨大骰子

- 骰子有很多个面，每个面代表一个单词
- 骰子是不均匀的，有些面会出现比较多次



骰子游戏 vs. 分词

假设我们有已经人工分好词的句子（称之为语料或数据），其中每个单词的出现就好比投掷一个巨大骰子

- 骰子有很多个面，每个面代表一个单词
- 骰子是不均匀的，有些面会出现比较多次

这	是	一	种	数据	。		
现在	已经	有	不少	可	用	数据	。
确实	有	很	多	疑问	。		

总词数：$6 + 8 + 5 = 20$

骰子游戏 vs. 分词

假设我们有已经人工分好词的句子（称之为语料或数据），其中每个单词的出现就好比投掷一个巨大骰子

- 骰子有很多个面，每个面代表一个单词
- 骰子是不均匀的，有些面会出现比较多次

这	是	一	种	数据	。		
现在	已经	有	不少	可	用	数据	。
确实	有	很	多	疑问	。		

总词数：$6 + 8 + 5 = 20$

$\mathrm{P}('很') = 1/20 = 0.05$

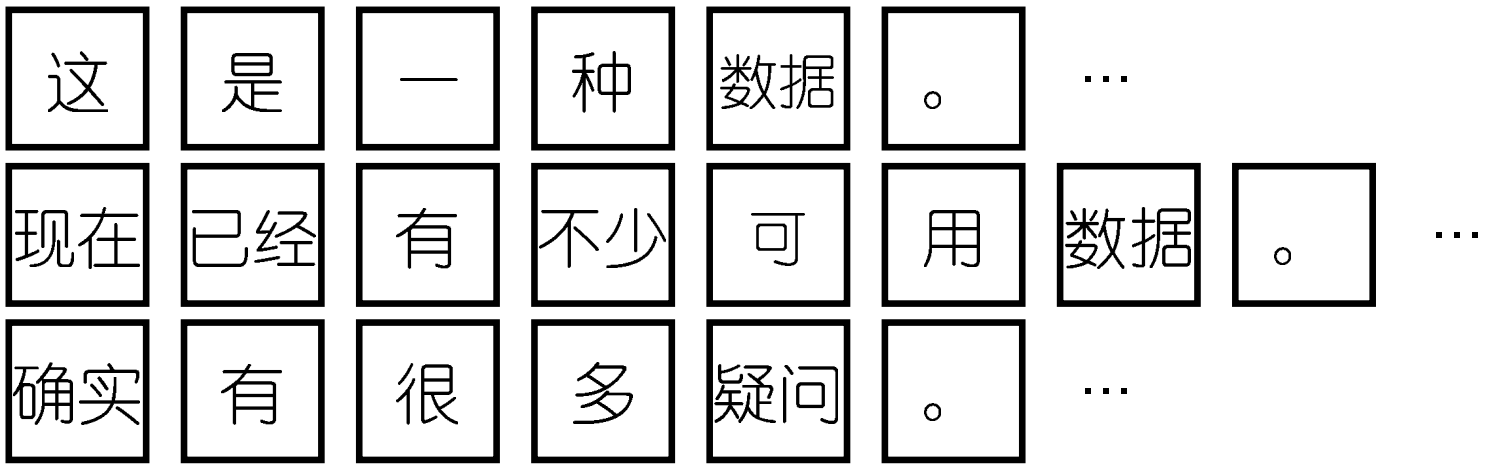
$\mathrm{P}('。') = 3/20 = 0.15$

$\mathrm{P}('确实') = 1/20 = 0.05$

骰子游戏 vs. 分词

假设我们有已经人工分好词的句子（称之为语料或数据），其中每个单词的出现就好比投掷一个巨大骰子

- 骰子有很多个面，每个面代表一个单词
- 骰子是不均匀的，有些面会出现比较多次



总词数：$6 + 8 + 5 = 20$

$\mathrm{P}('很') = 1/20 = 0.05$

$\mathrm{P}('。') = 3/20 = 0.15$

$\mathrm{P}('确实') = 1/20 = 0.05$

更多数据-总词数:100K $\sim$ 1M

$\mathrm{P}('很') = 0.000010$

$\mathrm{P}('。') = 0.001812$

$\mathrm{P}('确实') = 0.000001$

真正的分词过程

- 通过上述的'学习'，我们得到了每个词出现的概率。而现在的问题是如何计算整句分词结果的概率
  P('确实/现在/数据/很/多') =?

真正的分词过程

- 通过上述的'学习'，我们得到了每个词出现的概率。而现在的问题是如何计算整句分词结果的概率
  P('确实/现在/数据/很/多') =?
- 这里我们考虑把整句切分的概率转化为词的概率，怎么办？
  - 独立性假设$\mathrm{P}(AB) = \mathrm{P}(A)\mathrm{P}(B)$：每个词都是相互独立的
  - 设$w_1w_2...w_m$表示一个由单词w_1,w_2,...,w_m组成的切分结果

$$\mathrm{P}(w_1w_2...w_m) = \mathrm{P}(w_1) \cdot \mathrm{P}(w_2) \cdot ... \cdot \mathrm{P}(w_m)$$

真正的分词过程

- 通过上述的'学习'，我们得到了每个词出现的概率。而现在的问题是如何计算整句分词结果的概率
  P('确实/现在/数据/很/多') =?
- 这里我们考虑把整句切分的概率转化为词的概率，怎么办？
  - 独立性假设$\mathrm{P}(AB) = \mathrm{P}(A)\mathrm{P}(B)$：每个词都是相互独立的
  - 设$w_1w_2...w_m$表示一个由单词w_1,w_2,...,w_m组成的切分结果

$$\mathrm{P}(w_1w_2...w_m) = \mathrm{P}(w_1) \cdot \mathrm{P}(w_2) \cdot ... \cdot \mathrm{P}(w_m)$$

$$\mathrm{P}('确实/现在/数据/很/多')$$

真正的分词过程

- 通过上述的'学习'，我们得到了每个词出现的概率。而现在的问题是如何计算整句分词结果的概率
  P('确实/现在/数据/很/多') =?
- 这里我们考虑把整句切分的概率转化为词的概率，怎么办？
  - 独立性假设$\mathrm{P}(AB) = \mathrm{P}(A)\mathrm{P}(B)$：每个词都是相互独立的
  - 设$w_1w_2...w_m$表示一个由单词w_1,w_2,...,w_m组成的切分结果

$$\mathrm{P}(w_1w_2...w_m) = \mathrm{P}(w_1) \cdot \mathrm{P}(w_2) \cdot ... \cdot \mathrm{P}(w_m)$$

$$\begin{aligned} & \mathrm{P}('确实/现在/数据/很/多') \\ = \quad & \mathrm{P}('确实') \cdot \mathrm{P}('现在') \cdot \mathrm{P}('数据') \cdot \mathrm{P}('很') \cdot \mathrm{P}('多') \end{aligned}$$

真正的分词过程

- 通过上述的'学习'，我们得到了每个词出现的概率。而现在的问题是如何计算整句分词结果的概率
  P('确实/现在/数据/很/多') =?
- 这里我们考虑把整句切分的概率转化为词的概率，怎么办？
  - 独立性假设$\mathrm{P}(AB) = \mathrm{P}(A)\mathrm{P}(B)$：每个词都是相互独立的
  - 设$w_1w_2...w_m$表示一个由单词w_1,w_2,...,w_m组成的切分结果

$$\mathrm{P}(w_1w_2...w_m) = \mathrm{P}(w_1) \cdot \mathrm{P}(w_2) \cdot ... \cdot \mathrm{P}(w_m)$$

$$\begin{aligned} & \mathrm{P}(\text{'确实/现在/数据/很/多'}) \\ = \quad & \mathrm{P}(\text{'确实'}) \cdot \mathrm{P}(\text{'现在'}) \cdot \mathrm{P}(\text{'数据'}) \cdot \mathrm{P}(\text{'很'}) \cdot \mathrm{P}(\text{' 多'}) \\ = \quad & 0.000001 \times 0.000022 \times 0.000009 \times 0.000010 \times 0.000078 \\ = \quad & 1.5444 \times 10^{-25} \end{aligned}$$

真正的分词过程

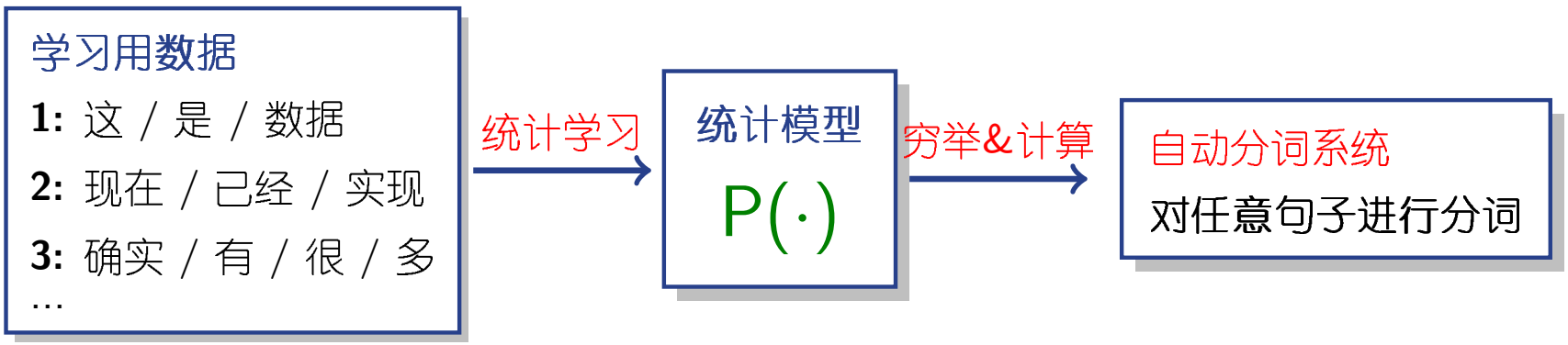
- 通过上述的'学习'，我们得到了每个词出现的概率。而现在的问题是如何计算整句分词结果的概率
  P('确实/现在/数据/很/多') =?
- 这里我们考虑把整句切分的概率转化为词的概率，怎么办？
  - 独立性假设$\mathrm{P}(AB) = \mathrm{P}(A)\mathrm{P}(B)$：每个词都是相互独立的
  - 设$w_1w_2...w_m$表示一个由单词$w_1,w_2,...,w_m$组成的切分结果

$$\mathrm{P}(w_1w_2...w_m) = \mathrm{P}(w_1) \cdot \mathrm{P}(w_2) \cdot ... \cdot \mathrm{P}(w_m)$$

$$\begin{aligned} & \mathrm{P}(\text{'确实/现在/数据/很/多'}) \\ = \quad & \mathrm{P}(\text{'确实'}) \cdot \mathrm{P}(\text{'现在'}) \cdot \mathrm{P}(\text{'数据'}) \cdot \mathrm{P}(\text{'很'}) \cdot \mathrm{P}(\text{' 多'}) \\ = \quad & 0.000001 \times 0.000022 \times 0.000009 \times 0.000010 \times 0.000078 \\ = \quad & 1.5444 \times 10^{-25} \end{aligned}$$

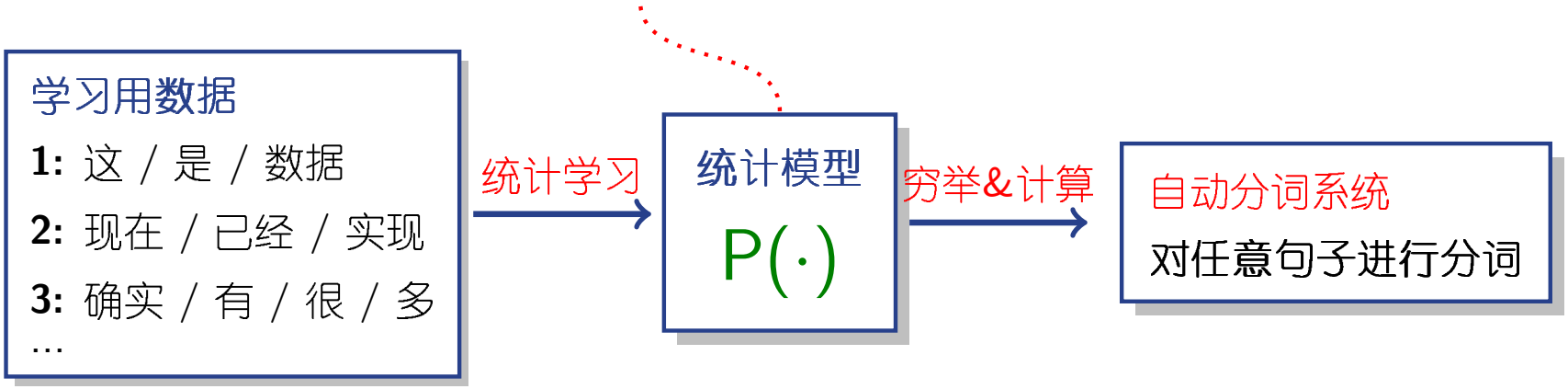
核心思想：通过独立性假设对问题进行'大题小做'

自动分词系统


学习用数据
1: 这 / 是 / 数据
2: 现在 / 已经 / 实现
3: 确实 / 有 / 很 / 多
…
统计学习
统计模型
P(·)
穷举&计算
自动分词系统
对任意句子进行分词

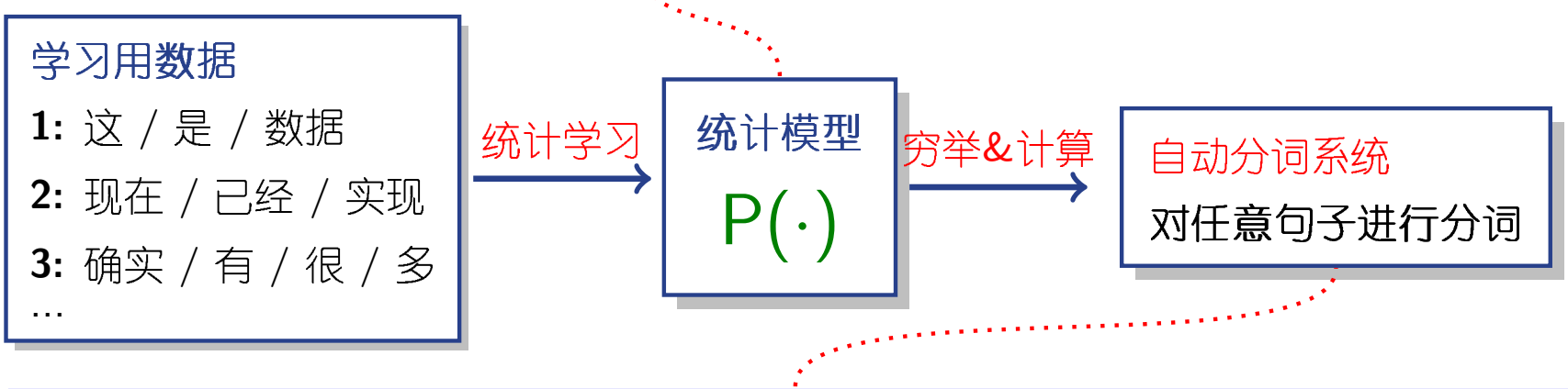
自动分词系统

实际上，通过学习我们得到了一个分词模型P(·)，给定任意的分词结果$W = w_1 w_2 ... w_n$，都能通过$\mathrm{P}(W)=\mathrm{P}(w_1) \cdot \mathrm{P}(w_2) \cdot ... \cdot \mathrm{P}(w_n)$计算这种分词的概率值



自动分词系统

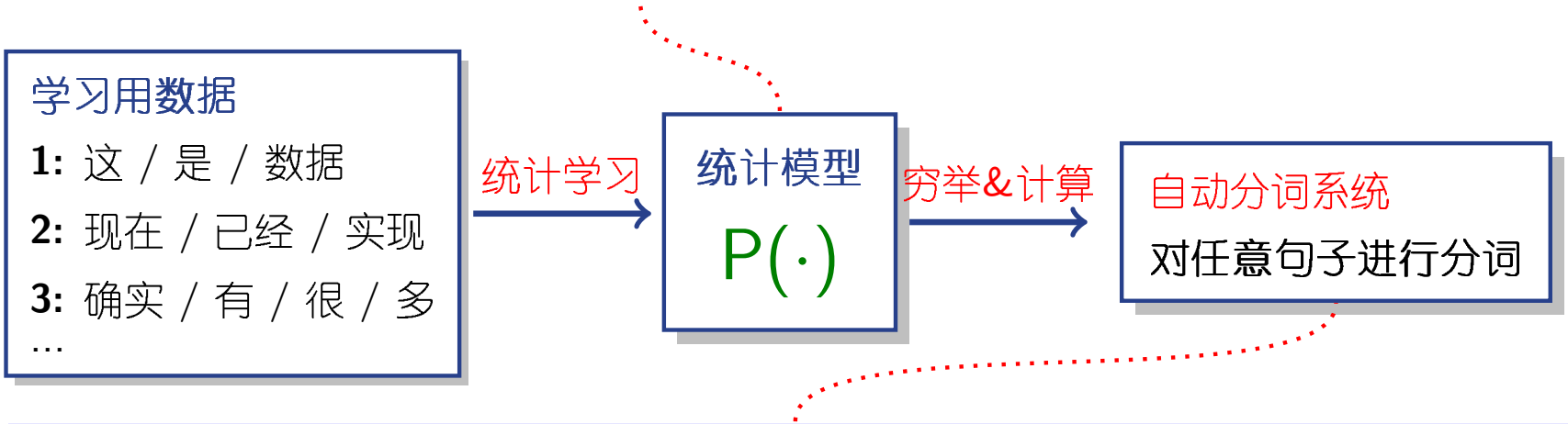
实际上，通过学习我们得到了一个分词模型P(·)，给定任意的分词结果$W = w_1 w_2 ... w_n$，都能通过$\mathrm{P}(W)=\mathrm{P}(w_1) \cdot \mathrm{P}(w_2) \cdot ... \cdot \mathrm{P}(w_n)$计算这种分词的概率值



自动分词系统：对任意的数据句子S，找到最佳的分词结果W^*输出

自动分词系统

实际上，通过学习我们得到了一个分词模型P(·)，给定任意的分词结果$W = w_1 w_2 ... w_n$，都能通过$\mathrm{P}(W)=\mathrm{P}(w_1) \cdot \mathrm{P}(w_2) \cdot ... \cdot \mathrm{P}(w_n)$计算这种分词的概率值

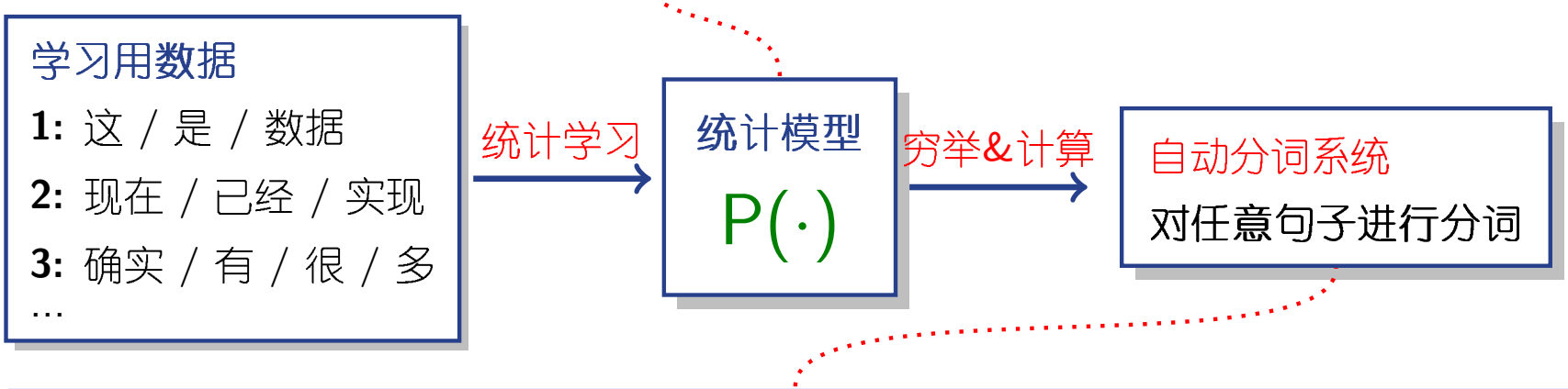


自动分词系统：对任意的数据句子S，找到最佳的分词结果W^*输出

假设输入S='确实现在数据很多'

自动分词系统

实际上，通过学习我们得到了一个分词模型P(·)，给定任意的分词结果$W = w_1 w_2 ... w_n$，都能通过$\mathrm{P}(W)=\mathrm{P}(w_1) \cdot \mathrm{P}(w_2) \cdot ... \cdot \mathrm{P}(w_n)$计算这种分词的概率值



自动分词系统：对任意的数据句子S，找到最佳的分词结果W^*输出

假设输入S='确实现在数据很多'

枚举所有可能的切分

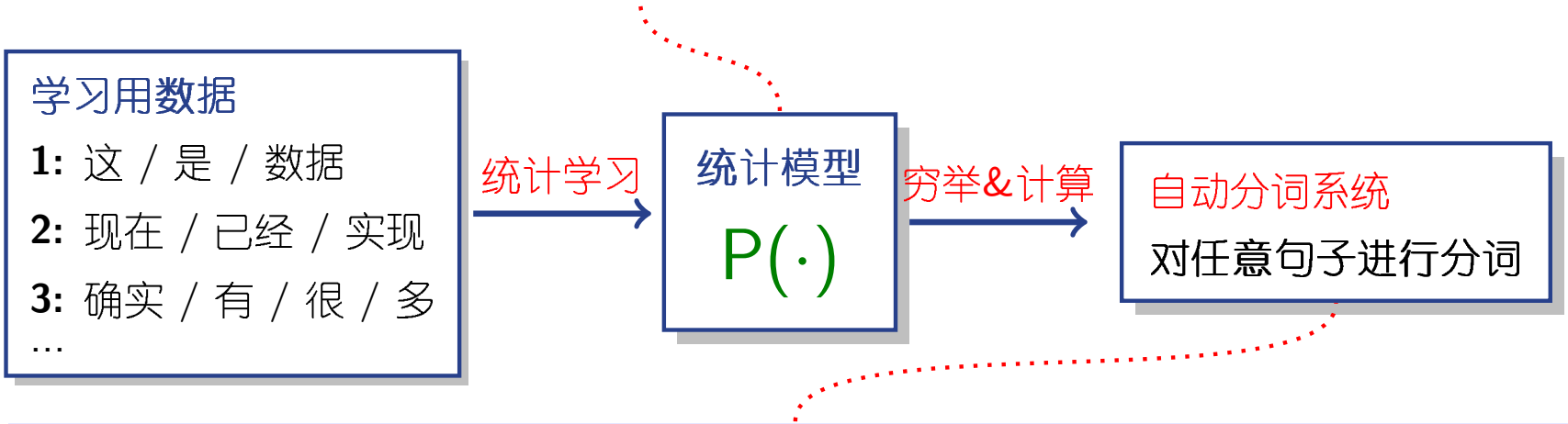
确/实现/在/数/据很/多

确实/现在/数据/很多

...

自动分词系统

实际上，通过学习我们得到了一个分词模型P(·)，给定任意的分词结果$W = w_1 w_2 ... w_n$，都能通过$\mathrm{P}(W)=\mathrm{P}(w_1) \cdot \mathrm{P}(w_2) \cdot ... \cdot \mathrm{P}(w_n)$计算这种分词的概率值



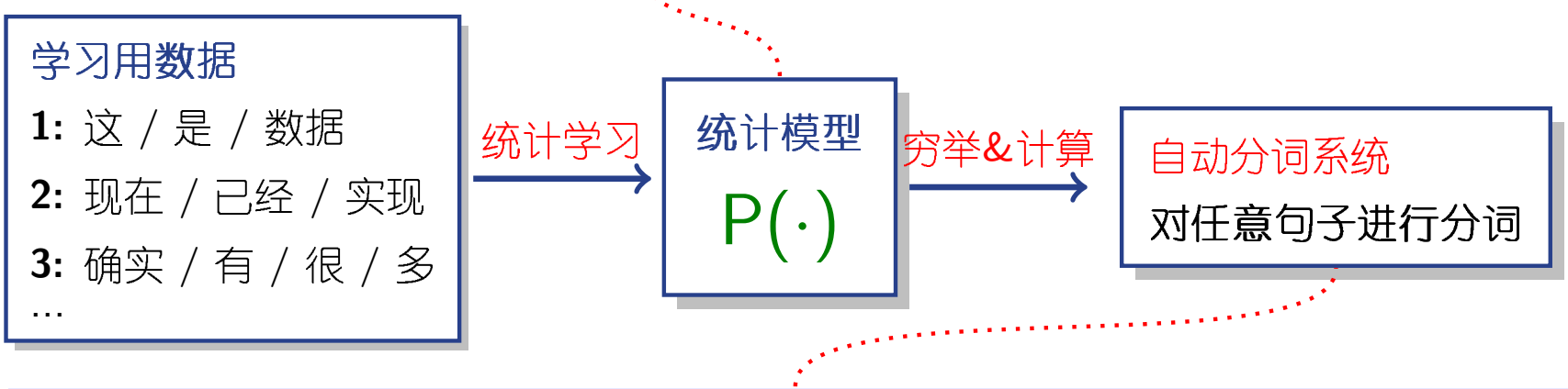
自动分词系统：对任意的数据句子S，找到最佳的分词结果W^*输出

假设输入S='确实现在数据很多'

枚举所有可能的切分	⟶	计算每种切分的概率
确/实现/在/数/据很/多		$\mathrm{P}(\text{'确'}) \cdot \mathrm{P}(\text{'实现'}) \cdot \mathrm{P}(\text{'在'}) \cdot \mathrm{P}(\text{'数'}) \cdot \mathrm{P}(\text{'据很'}) \cdot \mathrm{P}(\text{'多'}) = 2.13 \times 10^{-45}$
确实/现在/数据/很多		$\mathrm{P}(\text{'确实'}) \cdot \mathrm{P}(\text{'现在'}) \cdot \mathrm{P}(\text{'数据'}) \cdot \mathrm{P}(\text{'很'}) \cdot \mathrm{P}(\text{'多'}) = 1.54 \times 10^{-25}$
…		

自动分词系统

实际上，通过学习我们得到了一个分词模型P(·)，给定任意的分词结果$W = w_1 w_2 ... w_n$，都能通过$\mathrm{P}(W)=\mathrm{P}(w_1) \cdot \mathrm{P}(w_2) \cdot ... \cdot \mathrm{P}(w_n)$计算这种分词的概率值



自动分词系统：对任意的数据句子S，找到最佳的分词结果W^*输出

假设输入S='确实现在数据很多'

枚举所有可能的切分	计算每种切分的概率	选择最佳结果
确/实现/在/数/据很/多	$\mathrm{P}('确') \cdot \mathrm{P}('实现') \cdot \mathrm{P}('在') \cdot \mathrm{P}('数') \cdot \mathrm{P}('据很') \cdot \mathrm{P}('多') = 2.13 \times 10^{-45}$	
确实/现在/数据/很多	$\mathrm{P}('确实') \cdot \mathrm{P}('现在') \cdot \mathrm{P}('数据') \cdot \mathrm{P}('很') \cdot \mathrm{P}('多') = 1.54 \times 10^{-25}$	← 输出
…		

进一步扩展：基于n-gram语言模型的方法

- 这种方法也被称作基于1-gram(统计)语言模型的方法
  所谓统计语言模型就是计算$\mathrm{P}(w_1 w_2 ... w_m)$的概率

链式法则	1-gram	2-gram	...	n-gram
$\mathrm{P}(w_1 w_2 ... w_m) =$	$\mathrm{P}(w_1 w_2 ... w_m) =$	$\mathrm{P}(w_1 w_2 ... w_m) =$	...	$\mathrm{P}(w_1 w_2 ... w_m) =$
$\mathrm{P}(w_1)\times$	$\mathrm{P}(w_1)\times$	$\mathrm{P}(w_1)\times$	...	$\mathrm{P}(w_1)\times$
$\mathrm{P}(w_2\|w_1)\times$	$\mathrm{P}(w_2)\times$	$\mathrm{P}(w_2\|w_1)\times$	...	$\mathrm{P}(w_2\|w_1)\times$
$\mathrm{P}(w_3\|w_1 w_2)\times$	$\mathrm{P}(w_3)\times$	$\mathrm{P}(w_3\|w_2)\times$	...	$\mathrm{P}(w_3\|w_1 w_2)\times$
$\mathrm{P}(w_4\|w_1 w_2 w_3)\times$	$\mathrm{P}(w_4)\times$	$\mathrm{P}(w_4\|w_3)\times$	...	$\mathrm{P}(w_4\|w_1 w_2 w_3)\times$
...	...	...	...	...
$\mathrm{P}(w_m\|w_1 ... w_{m-1})$	$\mathrm{P}(w_m)$	$\mathrm{P}(w_m\|w_{m-1})$	...	$\mathrm{P}(w_m\|w_{m-n+1} ... w_{m-1})$

进一步扩展：基于n-gram语言模型的方法

- 这种方法也被称作基于1-gram(统计)语言模型的方法
  所谓统计语言模型就是计算$\mathrm{P}(w_1 w_2 ... w_m)$的概率

链式法则	1-gram	2-gram	...	n-gram
$\mathrm{P}(w_1 w_2 ... w_m) =$	$\mathrm{P}(w_1 w_2 ... w_m) =$	$\mathrm{P}(w_1 w_2 ... w_m) =$	...	$\mathrm{P}(w_1 w_2 ... w_m) =$
$\mathrm{P}(w_1) \times$	$\mathrm{P}(w_1) \times$	$\mathrm{P}(w_1) \times$	...	$\mathrm{P}(w_1) \times$
$\mathrm{P}(w_2 \mid w_1) \times$	$\mathrm{P}(w_2) \times$	$\mathrm{P}(w_2 \mid w_1) \times$	...	$\mathrm{P}(w_2 \mid w_1) \times$
$\mathrm{P}(w_3 \mid w_1 w_2) \times$	$\mathrm{P}(w_3) \times$	$\mathrm{P}(w_3 \mid w_2) \times$	...	$\mathrm{P}(w_3 \mid w_1 w_2) \times$
$\mathrm{P}(w_4 \mid w_1 w_2 w_3) \times$	$\mathrm{P}(w_4) \times$	$\mathrm{P}(w_4 \mid w_3) \times$	...	$\mathrm{P}(w_4 \mid w_1 w_2 w_3) \times$
...	...	...	...	...
$\mathrm{P}(w_m \mid w_1 ... w_{m-1})$	$\mathrm{P}(w_m)$	$\mathrm{P}(w_m \mid w_{m-1})$	...	$\mathrm{P}(w_m \mid w_{m-n+1} ... w_{m-1})$

- **n-gram语言模型**的核心思想是当前词(w_m)出现的概率只依赖于前$n-1$个词$(w_{m-n+1} ... w_{m-1})$

$$\begin{aligned} & \mathrm{P}_{\text{2-gram}}(\text{'确实/现在/数据/很/多'}) \\ = \; & \mathrm{P}(\text{'确实'}) \times \mathrm{P}(\text{'现在'} \mid \text{'确实'}) \times \mathrm{P}(\text{'数据'} \mid \text{'现在'}) \times \\ & \mathrm{P}(\text{'很'} \mid \text{'数据'}) \times \mathrm{P}(\text{'多'} \mid \text{'很'}) \end{aligned}$$

- **训练** - 相对频率估计：$\mathrm{P}(\text{'现在'} \mid \text{'确实'}) = \frac{count(\text{'确实 现在'})}{count(\text{'确实'})}$

理解思想就可以了- 分词/方法/本身/不是/重点

- *上面这个过程就是一个典型的数据驱动的统计学习方法，记住上一页的图就可以了
  - 也称作全概率分词或者基于语言模型的分词
  - 所谓数据驱动就是利用标注好的数据进行学习
  - 最大的优点：整个学习（模型训练）和推导（处理新的句子）过程都全自动进行

理解思想就可以了- 分词/方法/本身/不是/重点

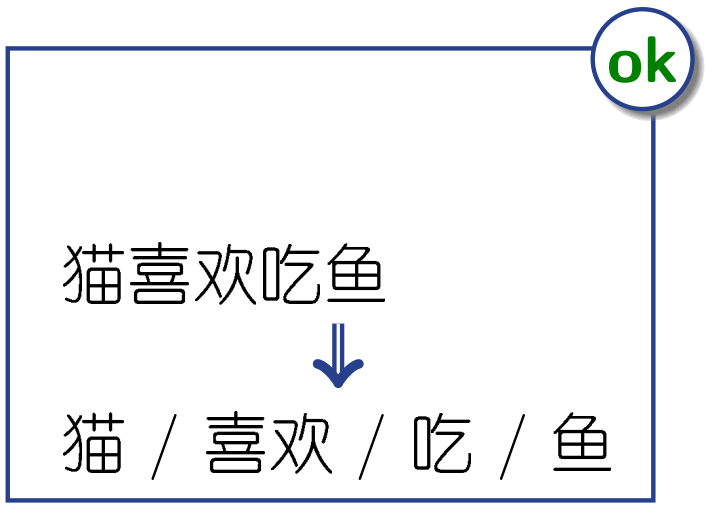
- *上面这个过程就是一个典型的数据驱动的统计学习方法，记住上一页的图就可以了
  - 也称作全概率分词或者基于语言模型的分词
  - 所谓数据驱动就是利用标注好的数据进行学习
  - 最大的优点：整个学习（模型训练）和推导（处理新的句子）过程都全自动进行
- 实现真实的分词系统还需要解决很多问题
  - 如何高效搜索最优解：可以利用动态规划(DP)
    推荐黄老师的”Dynamic programming-based search algorithms in NLP”
  - 未见过的词：概率等于0？赋予一个比较小的缺省值？
  - 如何获取切分好的数据用于模型学习：人工？半自动？

理解思想就可以了- 分词/方法/本身/不是/重点

- *上面这个过程就是一个典型的数据驱动的统计学习方法，记住上一页的图就可以了
  - 也称作全概率分词或者基于语言模型的分词
  - 所谓数据驱动就是利用标注好的数据进行学习
  - 最大的优点：整个学习（模型训练）和推导（处理新的句子）过程都全自动进行
- 实现真实的分词系统还需要解决很多问题
  - 如何高效搜索最优解：可以利用动态规划(DP)
    推荐黄老师的"Dynamic programming-based search algorithms in NLP"
  - 未见过的词：概率等于0？赋予一个比较小的缺省值？
  - 如何获取切分好的数据用于模型学习：人工？半自动？
- 其它可以应用于中文分词的统计方法
  - google一下"中文分词"和"word segmentation"，看看2005年以后的文章
  - google一下CRF，maximum entropy，LSTM+CRF

休息一下，捋捋思路

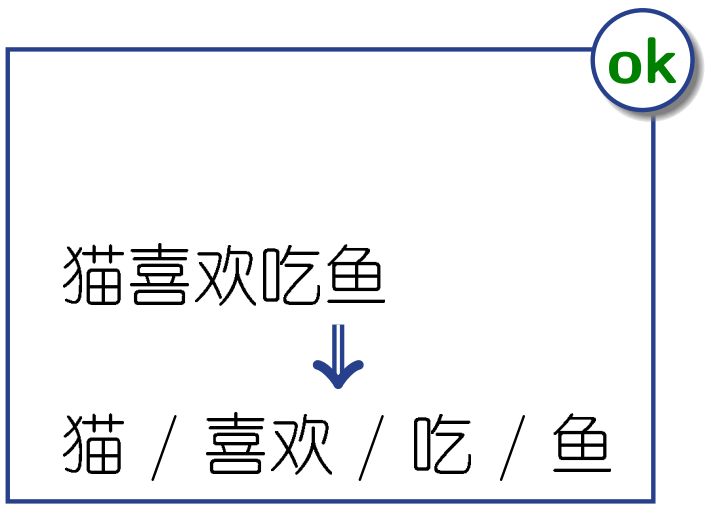
- 已经介绍的内容
  - 句子是由单词构成的
  - 如何定义单词、如何分词
  - 如何利用统计模型完成自动分词系统的构建



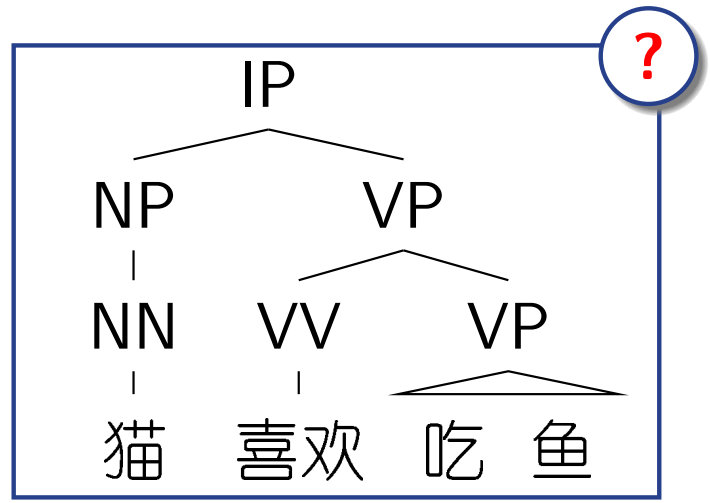
part1: 句子表示为单词串

休息一下，捋捋思路

- 已经介绍的内容
  - 句子是由单词构成的
  - 如何定义单词、如何分词
  - 如何利用统计模型完成自动分词系统的构建
- 即将要介绍的内容
  - 可以用基于树的结构进一步描述句子的句法结构 - 句法树



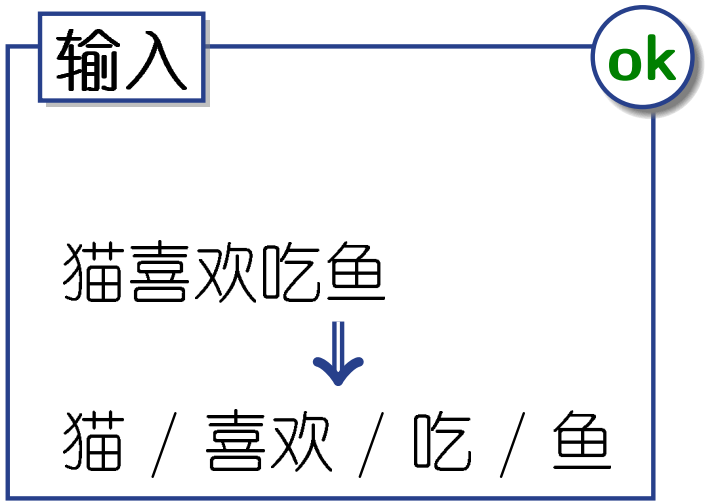
part1: 句子表示为单词串



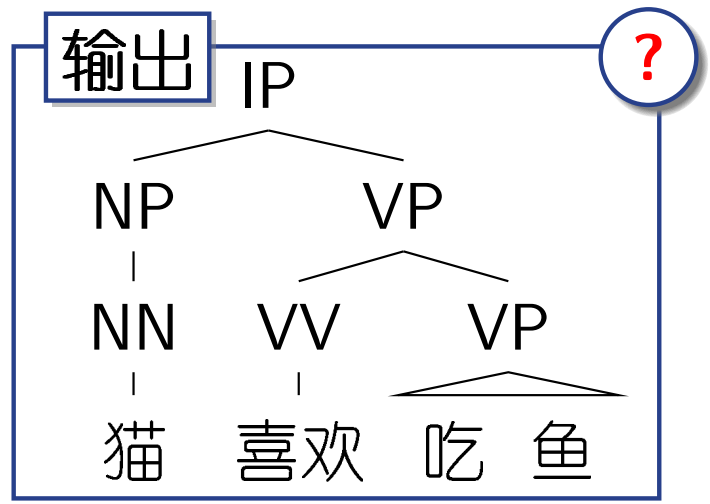
part2: 句子表示为句法树

休息一下，捋捋思路

- 已经介绍的内容
  - 句子是由单词构成的
  - 如何定义单词、如何分词
  - 如何利用统计模型完成自动分词系统的构建
- 即将要介绍的内容
  - 可以用基于树的结构进一步描述句子的句法结构 - 句法树
  - 如何获得一个句子的最佳句法树结构



part1: 句子表示为单词串



part2: 句子表示为句法树

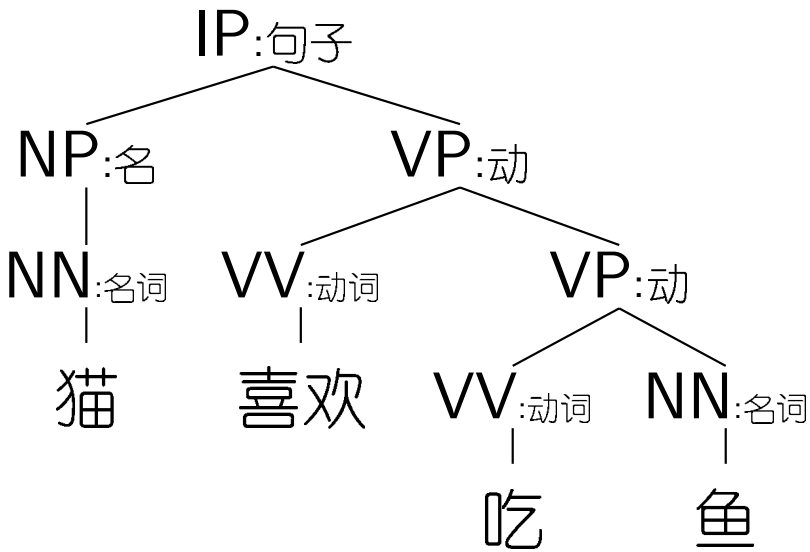
Outline

1. 词法分析
   - 中文分词 + 统计思想介绍

2. 句法分析
   - 中文短语结构树分析

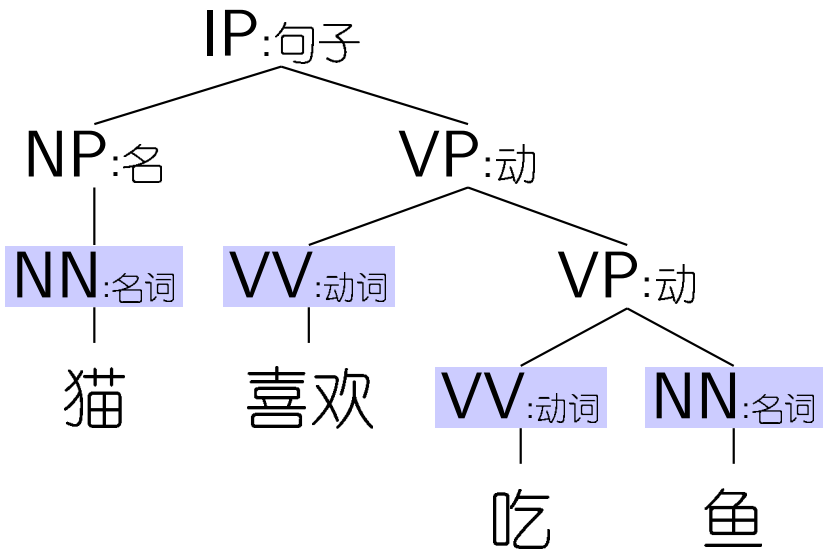
先看两个例子

例子1:短语结构(成分)句法树



先看两个例子

例子1:短语结构(成分)句法树

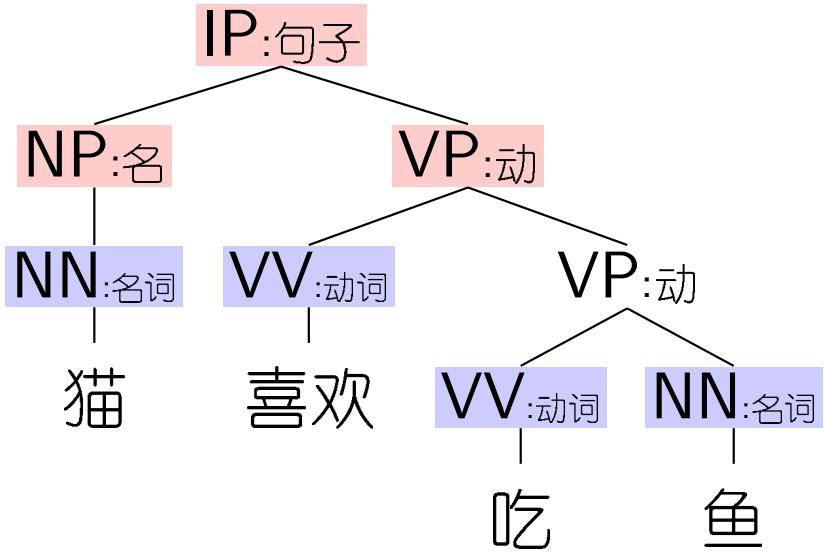


从上图可以了解

- 每个单词都有词性

先看两个例子

例子1:短语结构(成分)句法树

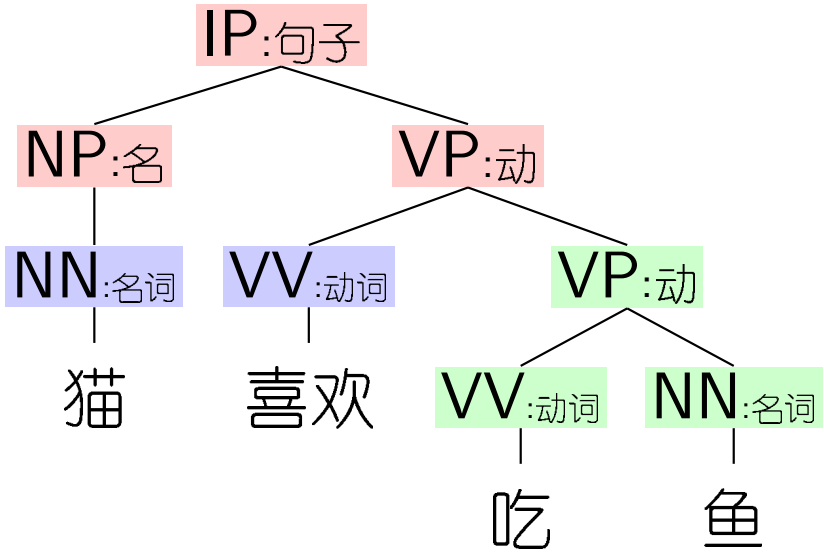


从上图可以了解

- 每个单词都有词性
- 名动结构

先看两个例子

例子1:短语结构(成分)句法树

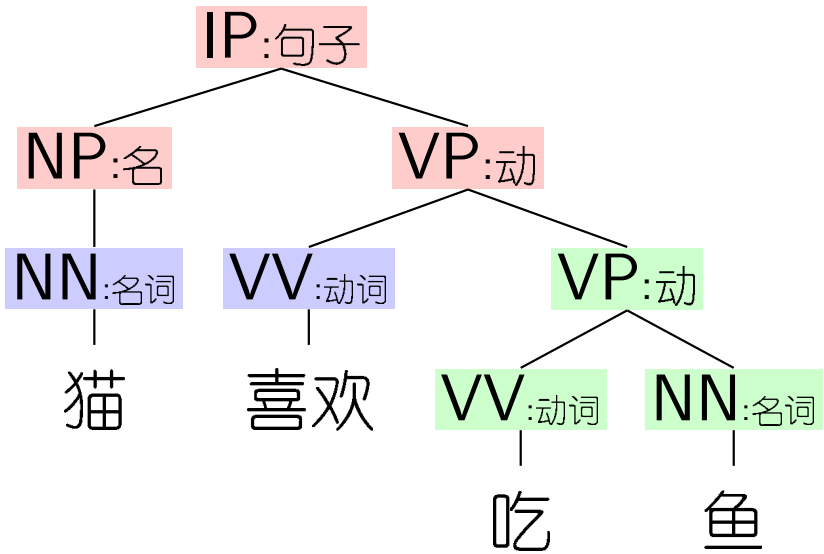


从上图可以了解

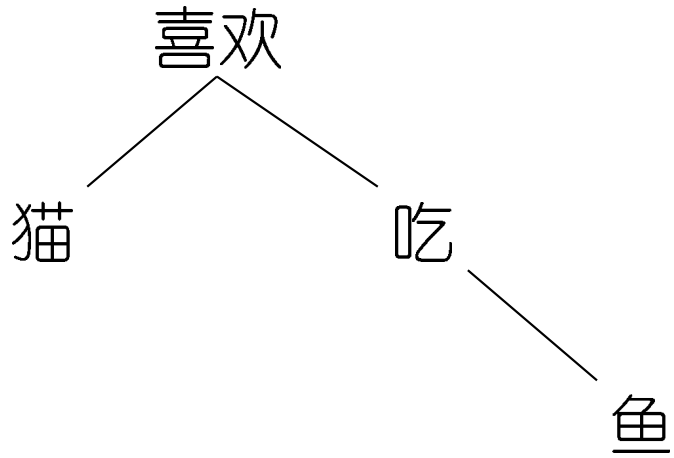
- 每个单词都有词性
- 名动结构
- 动宾结构

先看两个例子

例子1:短语结构(成分)句法树



例子2:依存句法树

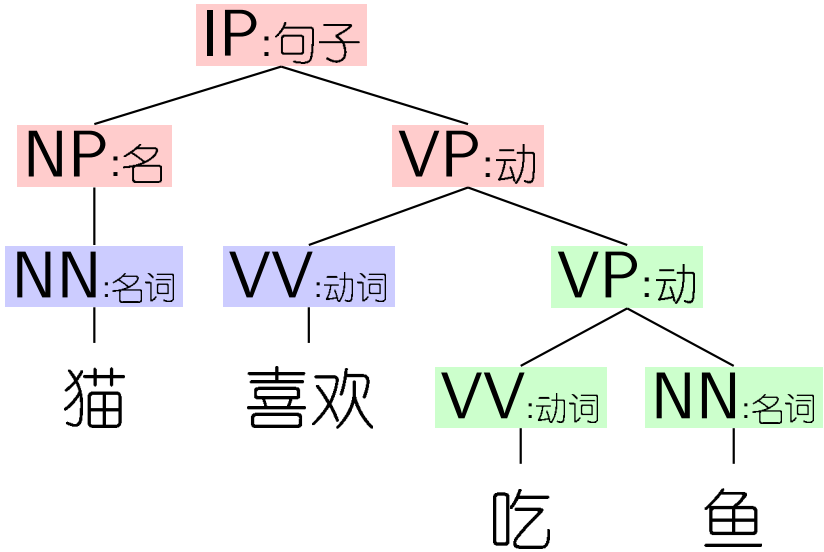


从上图可以了解

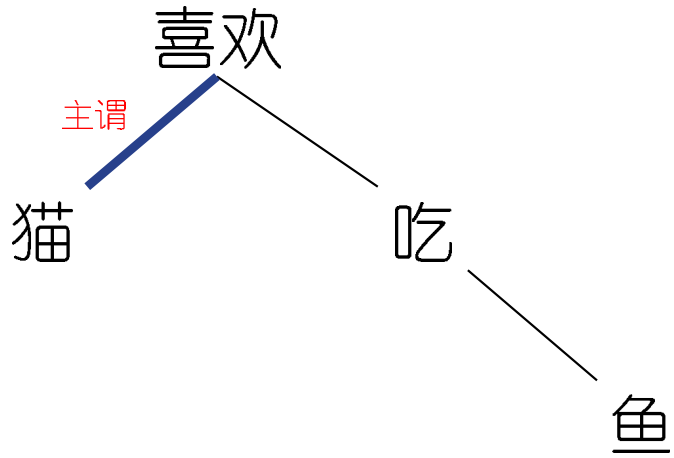
- 每个单词都有词性
- 名动结构
- 动宾结构

先看两个例子

例子1:短语结构(成分)句法树



例子2:依存句法树



从上图可以了解

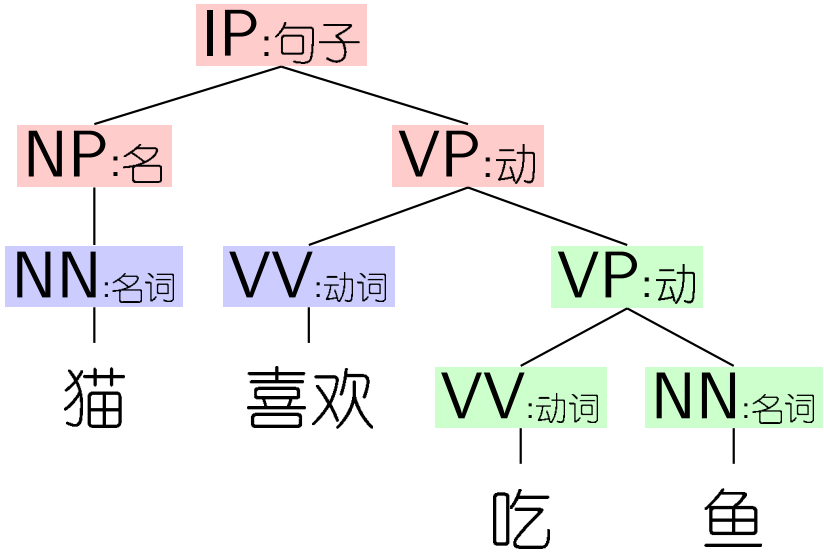
- 每个单词都有词性
- 名动结构
- 动宾结构

从上图可以了解

- '猫'依赖'喜欢'

先看两个例子

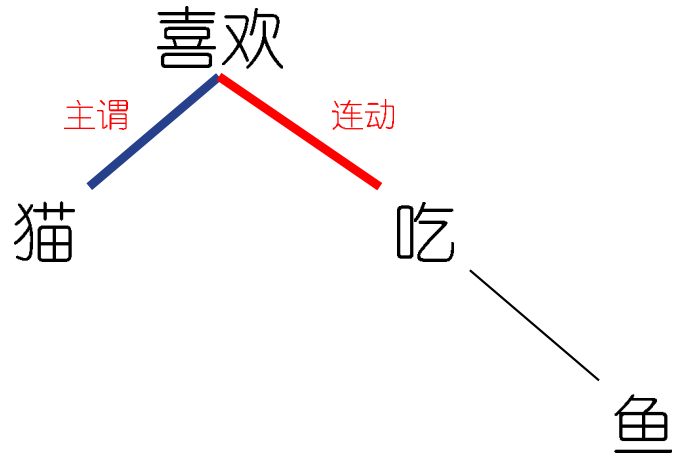
例子1:短语结构(成分)句法树



从上图可以了解

- 每个单词都有词性
- 名动结构
- 动宾结构

例子2:依存句法树

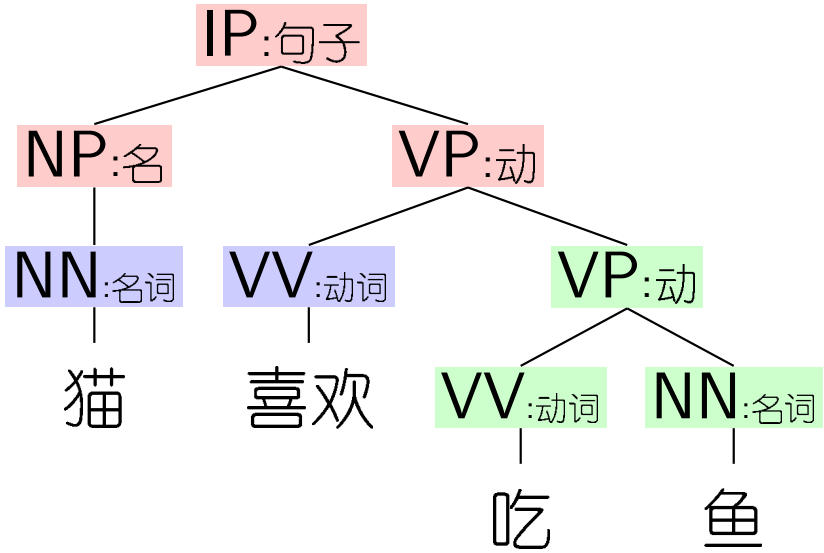


从上图可以了解

- '猫'依赖'喜欢'
- '吃'依赖'喜欢'

先看两个例子

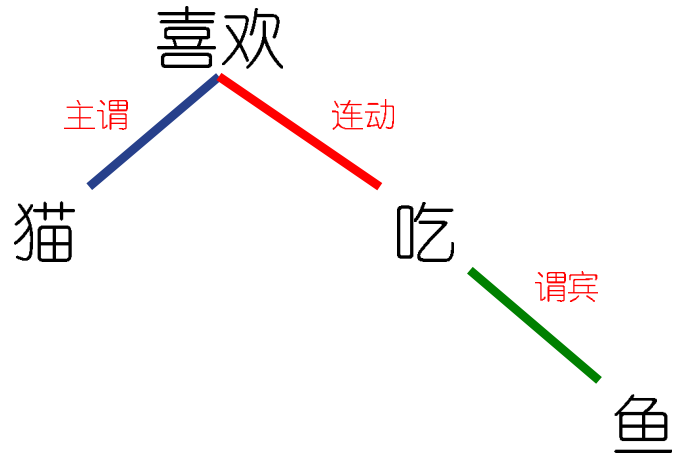
例子1:短语结构(成分)句法树



从上图可以了解

- 每个单词都有词性
- 名动结构
- 动宾结构

例子2:依存句法树

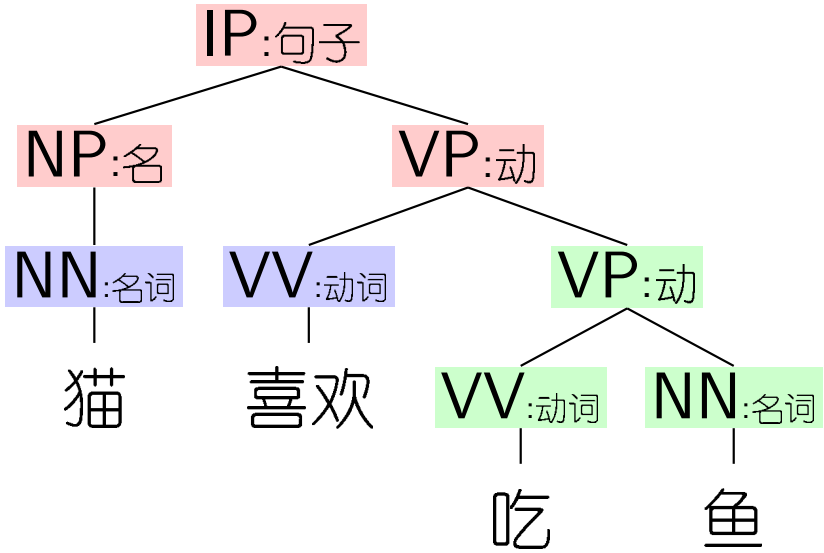


从上图可以了解

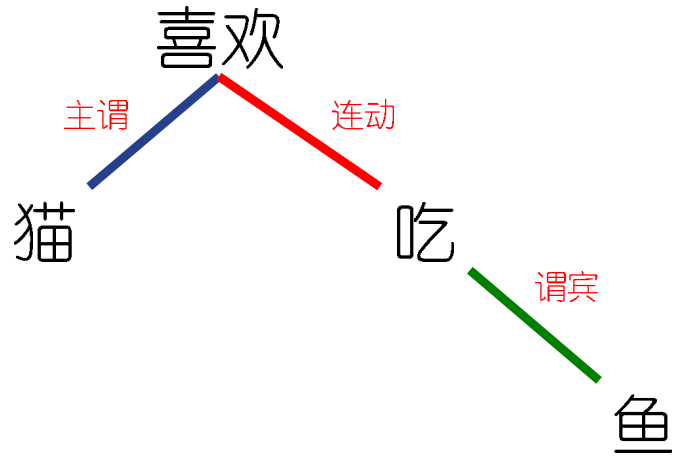
- '猫'依赖'喜欢'
- '吃'依赖'喜欢'
- '鱼'依赖'吃'

先看两个例子

例子1:短语结构(成分)句法树



例子2:依存句法树



从上图可以了解

- 每个单词都有词性
- 名动结构
- 动宾结构

从上图可以了解

- '猫'依赖'喜欢'
- '吃'依赖'喜欢'
- '鱼'依赖'吃'

显然这些句法树都可以帮助机器翻译系统理解句子的结构!

句法分析是什么？

- 简单的实例就是我们上小学时学的句子结构分析
  - 句子的成分：主语、谓语、宾语、状语...
  - 各个成分内部/外部的关系：并列、介宾、连动

小	猫	爬	到 树 上	去 了
定语	主语	谓语	宾语(方位)	补语

句法分析是什么？

- 简单的实例就是我们上小学时学的句子结构分析
  - 句子的成分：主语、谓语、宾语、状语...
  - 各个成分内部/外部的关系：并列、介宾、连动

小	猫	爬	到 树 上	去 了
定语	主语	谓语	宾语(方位)	补语

- 更权威一点儿的定义

句法分析(parsing或syntactic parsing)

在自然语言或者计算机语言中，句法分析是利用形式化的文法规则对一个符号串进行分析的过程

—维基百科(译文)

  - 符号串：可以理解为前面说的单词串
  - 文法：关于语言结构的形式化定义，由规则组成
  - 分析：利用文法解释句子结构

句法分析是什么？

- 简单的实例就是我们上小学时学的句子结构分析
  - 句子的成分：主语、谓语、宾语、状语...
  - 各个成分内部/外部的关系：并列、介宾、连动

小	猫	爬	到 树 上	去 了
定语	主语	谓语	宾语(方位)	补语

- 更权威一点儿的定义

句法分析(parsing或syntactic parsing)

在自然语言或者计算机语言中，句法分析是利用形式化的文法规则对一个符号串进行分析的过程

—维基百科(译文)

  - 符号串：可以理解为前面说的单词串
  - 文法：关于语言结构的形式化定义，由规则组成
  - 分析：利用文法解释句子结构
- 以上这些概念最好能记住，不过似乎和前面画的树也扯不上啊！ 实际上，句法树就是句法分析的一种形象表示

上下文无关文法

- 形式文法是句法分析中的核心内容。但是它本身是非常复杂的一套理论，这里不做深入讨论。推荐一个不错的介绍 http://en.wikipedia.org/wiki/Formal_grammar

上下文无关文法

- 形式文法是句法分析中的核心内容。但是它本身是非常复杂的一套理论，这里不做深入讨论。推荐一个不错的介绍 `http://en.wikipedia.org/wiki/Formal_grammar`
- 作为抛砖引玉这里只介绍一下最常用的上下文无关文法

上下文无关文法(context-free grammar)

一个上下文无关文法可以被视为一个系统 $G =< N, \Sigma, R, S >$，其中

- N为一个非终结符集合
- Σ为一个终结符集合
- R为一个规则(产生式)集合，每条规则 $r \in R$的形式为$X \to Y_1 Y_2 ... Y_n$，其中$X \in N$, $Y_i \in N \cup \Sigma$
- S为一个起始符号集合且$S \subseteq N$

上下文无关文法：实例

- 把非终结符定义为不同的句法标记
  $N = \{\mathrm{NN}, \mathrm{VV}, \mathrm{NP}, \mathrm{VP}, \mathrm{IP}\}$
  注意，词性：NN-名词，VV-动词
  短语结构：NP-名词短语，VP-动词短语，IP-单句
- 把终结符定义为不同的单词
  $\Sigma = \{\text{猫}, \text{喜欢}, \text{吃}, \text{鱼}\}$
- 把起始非终结符定义为整句的开始 $S = \{\mathrm{IP}\}$

上下文无关文法：实例

- 把非终结符定义为不同的句法标记
  $N = \{\mathrm{NN}, \mathrm{VV}, \mathrm{NP}, \mathrm{VP}, \mathrm{IP}\}$
  注意，词性：NN-名词，VV-动词
  短语结构：NP-名词短语，VP-动词短语，IP-单句
- 把终结符定义为不同的单词
  $\Sigma = \{$猫, 喜欢, 吃, 鱼$\}$
- 把起始非终结符定义为整句的开始 $S = \{\mathrm{IP}\}$
- 规则集为对上述句法结构的组装(r_i为规则的编号)

r_1: NN $\rightarrow$ 猫　　r_2: VV $\rightarrow$ 喜欢

r_3: VV $\rightarrow$ 吃　　r_4: NN $\rightarrow$ 鱼

r_5: NP $\rightarrow$ NN　　r_6: VP $\rightarrow$ VV NN

r_7: VP $\rightarrow$ VV VP　　r_8: IP $\rightarrow$ NP VP

上下文无关文法：实例

- 把非终结符定义为不同的句法标记
  $N = \{\mathrm{NN}, \mathrm{VV}, \mathrm{NP}, \mathrm{VP}, \mathrm{IP}\}$
  注意，词性：NN-名词，VV-动词
  短语结构：NP-名词短语，VP-动词短语，IP-单句
- 把终结符定义为不同的单词
  $\Sigma = \{$猫, 喜欢, 吃, 鱼$\}$
- 把起始非终结符定义为整句的开始 $S = \{\mathrm{IP}\}$
- 规则集为对上述句法结构的组装(r_i为规则的编号)

r_1: NN $\rightarrow$ 猫　　r_2: VV $\rightarrow$ 喜欢

r_3: VV $\rightarrow$ 吃　　r_4: NN $\rightarrow$ 鱼

r_5: NP $\rightarrow$ NN　　r_6: VP $\rightarrow$ VV NN

r_7: VP $\rightarrow$ VV VP　　r_8: IP $\rightarrow$ NP VP

r_1, r_2, r_3, r_4 为生成单词词性的规则

上下文无关文法：实例

- 把非终结符定义为不同的句法标记
  $N = \{\mathrm{NN}, \mathrm{VV}, \mathrm{NP}, \mathrm{VP}, \mathrm{IP}\}$
  注意，词性：NN-名词，VV-动词
  短语结构：NP-名词短语，VP-动词短语，IP-单句
- 把终结符定义为不同的单词
  $\Sigma = \{$猫, 喜欢, 吃, 鱼$\}$
- 把起始非终结符定义为整句的开始 $S = \{\mathrm{IP}\}$
- 规则集为对上述句法结构的组装(r_i为规则的编号)

r_1: NN $\rightarrow$ 猫　　r_2: VV $\rightarrow$ 喜欢

r_3: VV $\rightarrow$ 吃　　r_4: NN $\rightarrow$ 鱼

r_5: NP $\rightarrow$ NN　　r_6: VP $\rightarrow$ VV NN

r_7: VP $\rightarrow$ VV VP　　r_8: IP $\rightarrow$ NP VP

r_1, r_2, r_3, r_4 为生成单词词性的规则

r_5 为单变量规则，它将词性NN进一步抽象为名词短语NP

上下文无关文法：实例

- 把非终结符定义为不同的句法标记
  $N = \{\text{NN}, \text{VV}, \text{NP}, \text{VP}, \text{IP}\}$
  注意，词性：NN-名词，VV-动词
  短语结构：NP-名词短语，VP-动词短语，IP-单句
- 把终结符定义为不同的单词
  $\Sigma = \{\text{猫}, \text{喜欢}, \text{吃}, \text{鱼}\}$
- 把起始非终结符定义为整句的开始 $S = \{\text{IP}\}$
- 规则集为对上述句法结构的组装(r_i为规则的编号)

r_1: NN $\rightarrow$ 猫　　r_2: VV $\rightarrow$ 喜欢

r_3: VV $\rightarrow$ 吃　　r_4: NN $\rightarrow$ 鱼

r_5: NP $\rightarrow$ NN　　r_6: VP $\rightarrow$ VV NN

r_7: VP $\rightarrow$ VV VP　　r_8: IP $\rightarrow$ NP VP

r_1, r_2, r_3, r_4 为生成单词词性的规则

r_5 为单变量规则，它将词性NN进一步抽象为名词短语NP

r_6, r_7, r_8 为句法结构规则，比如r_8表示了主(NP)+谓(VP)结构

上下文无关文法：规则使用

- 上下文无关文法规则(下简称CFG规则)是一种产生式规则。对于规则$\alpha \to \beta$，它表示把规则左端的非中介符α替换为规则右端的符号序列β。

上下文无关文法：规则使用

- 上下文无关文法规则(下简称CFG规则)是一种产生式规则。对于规则$\alpha \to \beta$，它表示把规则左端的非中介符α替换为规则右端的符号序列β。

规则的使用

一个符号序列u可以通过使用规则r替换其中的某个非终结符，并得到符号序列v，我们说v是在u上使用r的结果，记为：

$$u \overset{r}{\Rightarrow} v$$

上下文无关文法：规则使用

- 上下文无关文法规则(下简称CFG规则)是一种产生式规则。对于规则$\alpha \to \beta$，它表示把规则左端的非中介符α替换为规则右端的符号序列β。

规则的使用

一个符号序列u可以通过使用规则r替换其中的某个非终结符，并得到符号序列v，我们说v是在u上使用r的结果，记为：

$$u \overset{r}{\Rightarrow} v$$

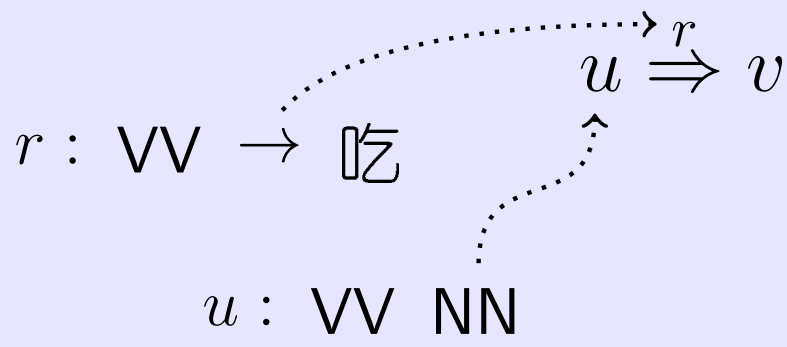
u：VV NN

上下文无关文法：规则使用

- 上下文无关文法规则(下简称CFG规则)是一种产生式规则。对于规则$\alpha \to \beta$，它表示把规则左端的非中介符α替换为规则右端的符号序列β。

规则的使用

一个符号序列u可以通过使用规则r替换其中的某个非终结符，并得到符号序列v，我们说v是在u上使用r的结果，记为：

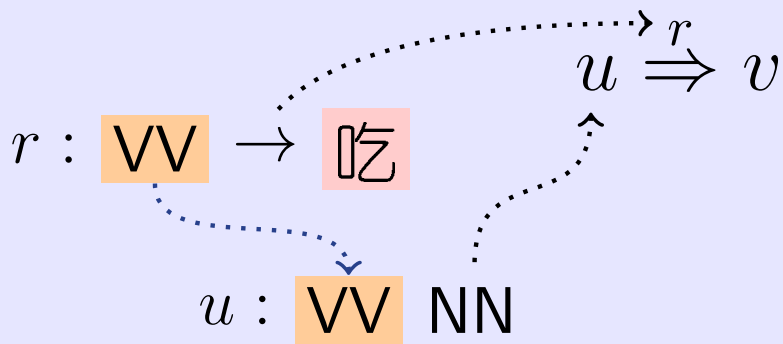


上下文无关文法：规则使用

- 上下文无关文法规则(下简称CFG规则)是一种产生式规则。对于规则$\alpha \to \beta$，它表示把规则左端的非中介符α替换为规则右端的符号序列β。

规则的使用

一个符号序列u可以通过使用规则r替换其中的某个非终结符，并得到符号序列v，我们说v是在u上使用r的结果，记为：

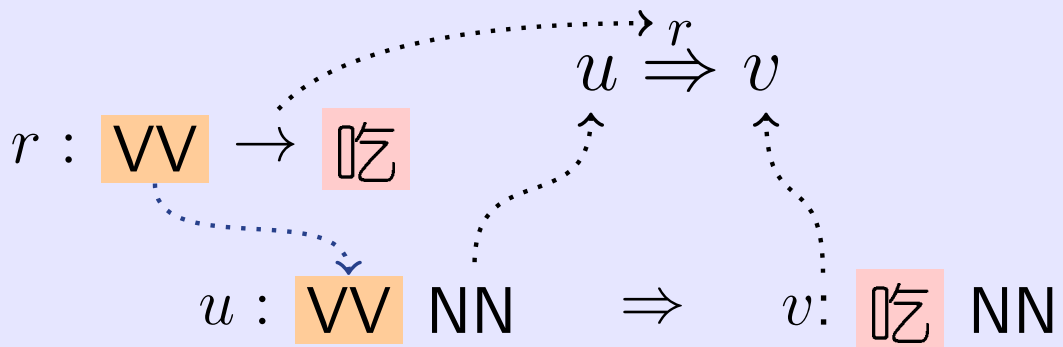


上下文无关文法：规则使用

- 上下文无关文法规则(下简称CFG规则)是一种产生式规则。对于规则$\alpha \to \beta$，它表示把规则左端的非中介符α替换为规则右端的符号序列β。

规则的使用

一个符号序列u可以通过使用规则r替换其中的某个非终结符，并得到符号序列v，我们说v是在u上使用r的结果，记为：

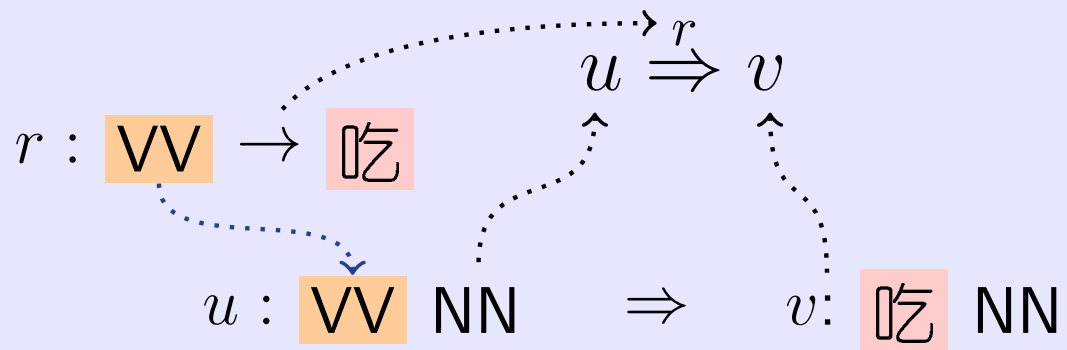


上下文无关文法：规则使用

- 上下文无关文法规则(下简称CFG规则)是一种产生式规则。对于规则$\alpha \to \beta$，它表示把规则左端的非中介符α替换为规则右端的符号序列β。

规则的使用

一个符号序列u可以通过使用规则r替换其中的某个非终结符，并得到符号序列v，我们说v是在u上使用r的结果，记为：



- 如果v是在u上使用多条规则得到的结果，则记为：$u \overset{*}{\Rightarrow} v$

上下文无关文法：推导 vs. 树

实际上，我们可以从一个起始非终结符，不断地使用规则，最终生成一个终结符串。这个过程通常也被成为推导的生成过程。

r_1: NN $\rightarrow$ 猫　　r_2: VV $\rightarrow$ 喜欢

r_3: VV $\rightarrow$ 吃　　r_4: NN $\rightarrow$ 鱼

r_5: NP $\rightarrow$ NN　　r_6: VP $\rightarrow$ VV NN

r_7: VP $\rightarrow$ VV VP　　r_8: IP $\rightarrow$ NP VP

上下文无关文法：推导 vs. 树

实际上，我们可以从一个起始非终结符，不断地使用规则，最终生成一个终结符串。这个过程通常也被成为推导的生成过程。

r_1: NN $\rightarrow$ 猫　　r_2: VV $\rightarrow$ 喜欢

r_3: VV $\rightarrow$ 吃　　r_4: NN $\rightarrow$ 鱼

r_5: NP $\rightarrow$ NN　　r_6: VP $\rightarrow$ VV NN

r_7: VP $\rightarrow$ VV VP　　r_8: IP $\rightarrow$ NP VP

上下文无关文法：推导 vs. 树

实际上，我们可以从一个起始非终结符，不断地使用规则，最终生成一个终结符串。这个过程通常也被成为推导的生成过程。

IP

r_1: NN → 猫　　r_2: VV → 喜欢

r_3: VV → 吃　　r_4: NN → 鱼

r_5: NP → NN　　r_6: VP → VV NN

r_7: VP → VV VP　　r_8: IP → NP VP

IP

上下文无关文法：推导 vs. 树

实际上，我们可以从一个起始非终结符，不断地使用规则，最终生成一个终结符串。这个过程通常也被成为推导的生成过程。

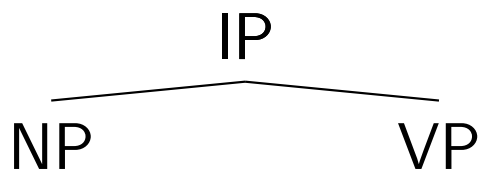
$$\begin{aligned} & \text{IP} \\ \overset{r_8}{\Rightarrow} \quad & \text{NP VP} \end{aligned}$$

r_1: NN → 猫　　r_2: VV → 喜欢

r_3: VV → 吃　　r_4: NN → 鱼

r_5: NP → NN　　r_6: VP → VV NN

r_7: VP → VV VP　　r_8: IP → NP VP



上下文无关文法：推导 vs. 树

实际上，我们可以从一个起始非终结符，不断地使用规则，最终生成一个终结符串。这个过程通常也被成为推导的生成过程。

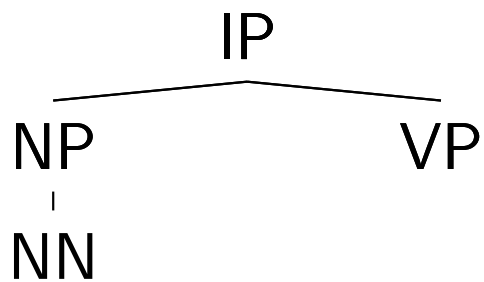
$$
\begin{aligned}
& \text{IP} \\
\overset{r_8}{\Rightarrow} \quad & \text{NP VP} \\
\overset{r_5}{\Rightarrow} \quad & \text{NN VP}
\end{aligned}
$$

r_1: NN $\rightarrow$ 猫　　r_2: VV $\rightarrow$ 喜欢

r_3: VV $\rightarrow$ 吃　　r_4: NN $\rightarrow$ 鱼

r_5: NP $\rightarrow$ NN　　r_6: VP $\rightarrow$ VV NN

r_7: VP $\rightarrow$ VV VP　　r_8: IP $\rightarrow$ NP VP



上下文无关文法：推导 vs. 树

实际上，我们可以从一个起始非终结符，不断地使用规则，最终生成一个终结符串。这个过程通常也被成为推导的生成过程。

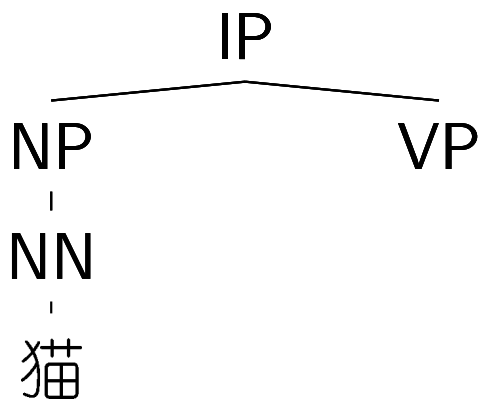
$$
\begin{aligned}
& \text{IP} \\
\overset{r_8}{\Rightarrow} \quad & \text{NP VP} \\
\overset{r_5}{\Rightarrow} \quad & \text{NN VP} \\
\overset{r_1}{\Rightarrow} \quad & \text{猫 VP}
\end{aligned}
$$

r_1: NN → 猫　　r_2: VV → 喜欢

r_3: VV → 吃　　r_4: NN → 鱼

r_5: NP → NN　　r_6: VP → VV NN

r_7: VP → VV VP　　r_8: IP → NP VP



上下文无关文法：推导 vs. 树

实际上，我们可以从一个起始非终结符，不断地使用规则，最终生成一个终结符串。这个过程通常也被成为推导的生成过程。

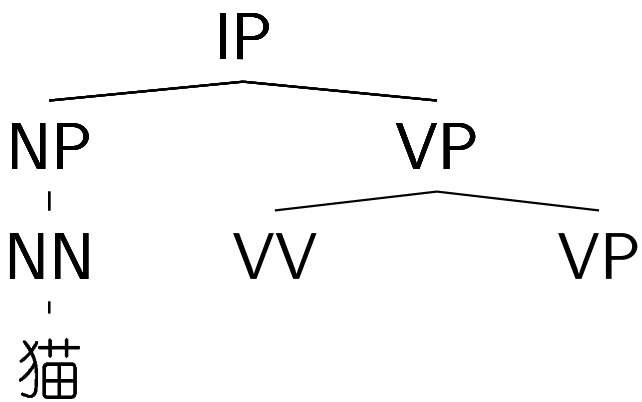
$$
\begin{aligned}
& \text{IP} \\
\overset{r_8}{\Rightarrow} \quad & \text{NP VP} \\
\overset{r_5}{\Rightarrow} \quad & \text{NN VP} \\
\overset{r_1}{\Rightarrow} \quad & \text{猫 VP} \\
\overset{r_7}{\Rightarrow} \quad & \text{猫 VV VP}
\end{aligned}
$$

r_1: NN → 猫　　r_2: VV → 喜欢

r_3: VV → 吃　　r_4: NN → 鱼

r_5: NP → NN　　r_6: VP → VV NN

r_7: VP → VV VP　　r_8: IP → NP VP



上下文无关文法：推导 vs. 树

实际上，我们可以从一个起始非终结符，不断地使用规则，最终生成一个终结符串。这个过程通常也被成为推导的生成过程。

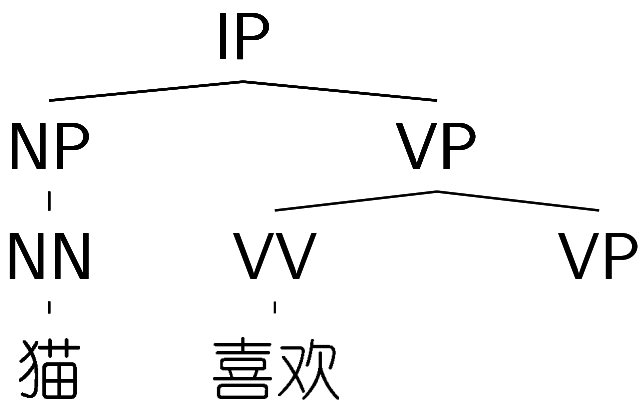
$$
\begin{aligned}
& \text{IP} \\
\overset{r_8}{\Rightarrow} \quad & \text{NP VP} \\
\overset{r_5}{\Rightarrow} \quad & \text{NN VP} \\
\overset{r_1}{\Rightarrow} \quad & \text{猫 VP} \\
\overset{r_7}{\Rightarrow} \quad & \text{猫 VV VP} \\
\overset{r_2}{\Rightarrow} \quad & \text{猫 喜欢 VP}
\end{aligned}
$$

r_1: NN → 猫　　r_2: VV → 喜欢

r_3: VV → 吃　　r_4: NN → 鱼

r_5: NP → NN　　r_6: VP → VV NN

r_7: VP → VV VP　　r_8: IP → NP VP



上下文无关文法：推导 vs. 树

实际上，我们可以从一个起始非终结符，不断地使用规则，最终生成一个终结符串。这个过程通常也被成为推导的生成过程。

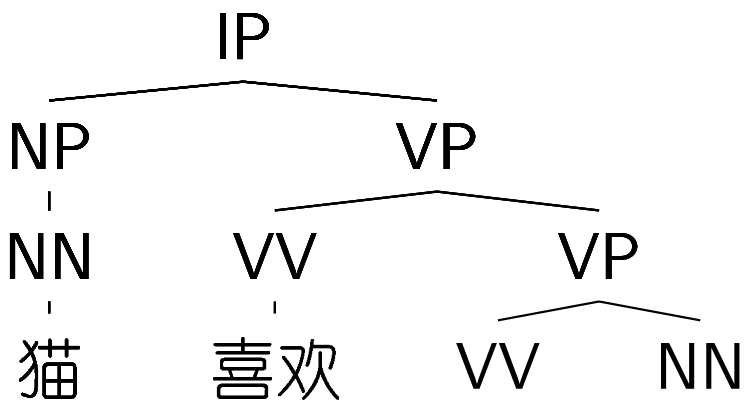
$$
\begin{aligned}
& \text{IP} \\
\overset{r_8}{\Rightarrow} \quad & \text{NP VP} \\
\overset{r_5}{\Rightarrow} \quad & \text{NN VP} \\
\overset{r_1}{\Rightarrow} \quad & \text{猫 VP} \\
\overset{r_7}{\Rightarrow} \quad & \text{猫 VV VP} \\
\overset{r_2}{\Rightarrow} \quad & \text{猫 喜欢 VP} \\
\overset{r_6}{\Rightarrow} \quad & \text{猫 喜欢 VV NN}
\end{aligned}
$$

r_1: NN → 猫　　r_2: VV → 喜欢

r_3: VV → 吃　　r_4: NN → 鱼

r_5: NP → NN　　r_6: VP → VV NN

r_7: VP → VV VP　　r_8: IP → NP VP



上下文无关文法：推导 vs. 树

实际上，我们可以从一个起始非终结符，不断地使用规则，最终生成一个终结符串。这个过程通常也被成为推导的生成过程。

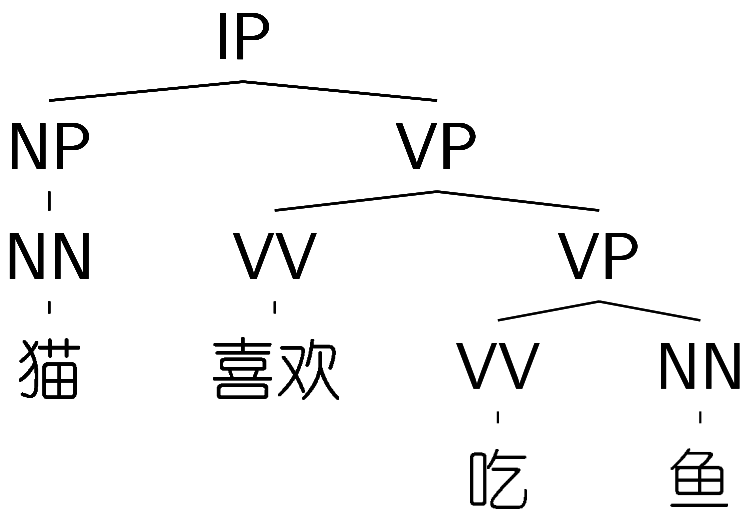
$$
\begin{aligned}
& \text{IP} \\
\overset{r_8}{\Rightarrow} \quad & \text{NP VP} \\
\overset{r_5}{\Rightarrow} \quad & \text{NN VP} \\
\overset{r_1}{\Rightarrow} \quad & \text{猫 VP} \\
\overset{r_7}{\Rightarrow} \quad & \text{猫 VV VP} \\
\overset{r_2}{\Rightarrow} \quad & \text{猫 喜欢 VP} \\
\overset{r_6}{\Rightarrow} \quad & \text{猫 喜欢 VV NN} \\
\overset{r_3}{\Rightarrow} \quad & \text{猫 喜欢 吃 NN} \\
\overset{r_4}{\Rightarrow} \quad & \text{猫 喜欢 吃 鱼}
\end{aligned}
$$

r_1: NN → 猫　　r_2: VV → 喜欢

r_3: VV → 吃　　r_4: NN → 鱼

r_5: NP → NN　　r_6: VP → VV NN

r_7: VP → VV VP　　r_8: IP → NP VP



上下文无关文法：推导 vs. 树

实际上，我们可以从一个起始非终结符，不断地使用规则，最终生成一个终结符串。这个过程通常也被成为推导的生成过程。

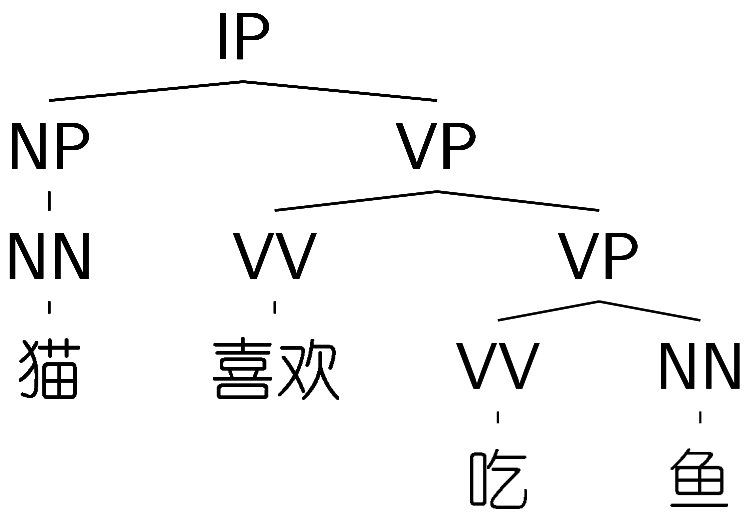
$$
\begin{aligned}
& \text{IP} \\
\overset{r_8}{\Rightarrow} \quad & \text{NP VP} \\
\overset{r_5}{\Rightarrow} \quad & \text{NN VP} \\
\overset{r_1}{\Rightarrow} \quad & \text{猫 VP} \\
\overset{r_7}{\Rightarrow} \quad & \text{猫 VV VP} \\
\overset{r_2}{\Rightarrow} \quad & \text{猫 喜欢 VP} \\
\overset{r_6}{\Rightarrow} \quad & \text{猫 喜欢 VV NN} \\
\overset{r_3}{\Rightarrow} \quad & \text{猫 喜欢 吃 NN} \\
\overset{r_4}{\Rightarrow} \quad & \text{猫 喜欢 吃 鱼}
\end{aligned}
$$

r_1: NN → 猫　　r_2: VV → 喜欢

r_3: VV → 吃　　r_4: NN → 鱼

r_5: NP → NN　　r_6: VP → VV NN

r_7: VP → VV VP　　r_8: IP → NP VP



这个规则的使用序列本质上就对应了句法树的生成过程

上下文无关文法：推导 - 没有图的一页

推导

给定一个文法$G =< N, \Sigma, R, S >$，对于一个字符串序列$s_0, s_1, ..., s_n$和规则序列$r_1, r_2, ..., r_n$，满足

$$s_0 \overset{r_1}{\Rightarrow} s_1 \overset{r_2}{\Rightarrow} s_2 \overset{r_3}{\Rightarrow} ... \overset{r_n}{\Rightarrow} s_n$$

且

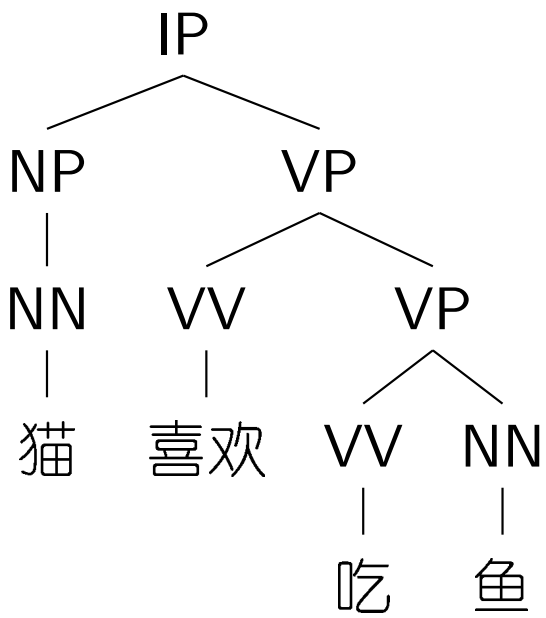
- $\forall i \in [0, n], s_i \in (N \cup \Sigma)^*$ $\lhd$ s_i为合法的字符串
- $\forall j \in [1, n], r_j \in R$ $\lhd$ r_j为G的规则
- $s_0 \in S$ $\lhd$ s_0为起始非终结符
- $s_n \in \Sigma^*$ $\lhd$ s_n为终结符序列

则$s_0 \overset{r_1}{\Rightarrow} s_1 \overset{r_2}{\Rightarrow} s_2 \overset{r_3}{\Rightarrow} ... \overset{r_n}{\Rightarrow} s_n$为一个推导(derivation)

通常我们也把推导简记为$d = r_1 \circ r_2 \circ ... \circ r_n$，其中$\circ$表示规则的组合

歧义1：相同的树对应不同的推导

生成同一棵句法树，我们可以使用不同的规则序列，即，不同的推导最终会得到相同的句法树



推导1

$$
\begin{aligned}
& \text{IP} \\
\xRightarrow{r_8} \ & \text{NP VP} \\
\xRightarrow{r_5} \ & \text{NN VP} \\
\xRightarrow{r_1} \ & \text{猫 VP} \\
\xRightarrow{r_7} \ & \text{猫 VV VP} \\
\xRightarrow{r_2} \ & \text{猫 喜欢 VP} \\
\xRightarrow{r_6} \ & \text{猫 喜欢 VV NN} \\
\xRightarrow{r_3} \ & \text{猫 喜欢 吃 NN} \\
\xRightarrow{r_4} \ & \text{猫 喜欢 吃 鱼}
\end{aligned}
$$

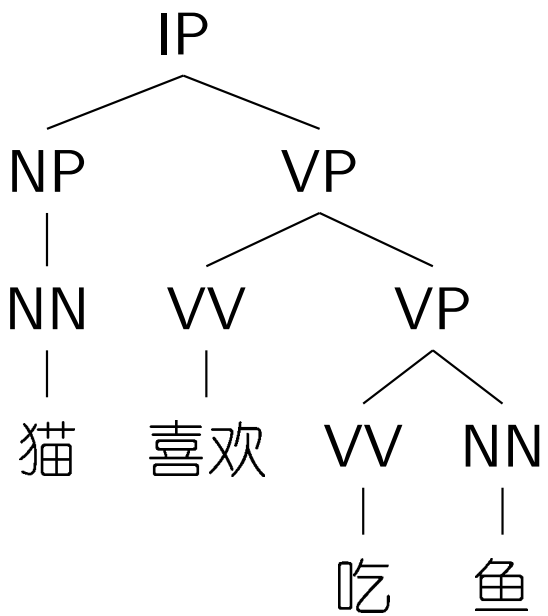
推导2

$$
\begin{aligned}
& \text{IP} \\
\xRightarrow{r_8} \ & \text{NP VP} \\
\xRightarrow{r_7} \ & \text{NP VV VP} \\
\xRightarrow{r_2} \ & \text{NP 喜欢 VP} \\
\xRightarrow{r_6} \ & \text{NP 喜欢 VV NN} \\
\xRightarrow{r_4} \ & \text{NP 喜欢 VV 鱼} \\
\xRightarrow{r_5} \ & \text{NN 喜欢 VV 鱼} \\
\xRightarrow{r_3} \ & \text{NN 喜欢 吃 鱼} \\
\xRightarrow{r_1} \ & \text{猫 喜欢 吃 鱼}
\end{aligned}
$$

歧义1：相同的树对应不同的推导

生成同一棵句法树，我们可以使用不同的规则序列，即，不同的推导最终会得到相同的句法树

- 解决方法：规则使用都服从最左优先原则



推导1

IP

$\overset{r_8}{\Rightarrow}$ NP VP

$\overset{r_5}{\Rightarrow}$ NN VP

$\overset{r_1}{\Rightarrow}$ 猫 VP

$\overset{r_7}{\Rightarrow}$ 猫 VV VP

$\overset{r_2}{\Rightarrow}$ 猫 喜欢 VP

$\overset{r_6}{\Rightarrow}$ 猫 喜欢 VV NN

$\overset{r_3}{\Rightarrow}$ 猫 喜欢 吃 NN

$\overset{r_4}{\Rightarrow}$ 猫 喜欢 吃 鱼

推导2

IP

$\overset{r_8}{\Rightarrow}$ NP VP

$\overset{r_7}{\Rightarrow}$ NP VV VP

$\overset{r_2}{\Rightarrow}$ NP 喜欢 VP

$\overset{r_6}{\Rightarrow}$ NP 喜欢 VV NN

$\overset{r_4}{\Rightarrow}$ NP 喜欢 VV 鱼

$\overset{r_5}{\Rightarrow}$ NN 喜欢 VV 鱼

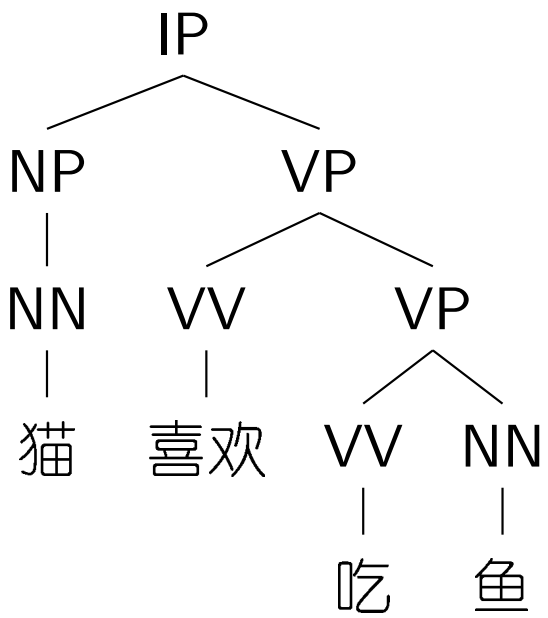
$\overset{r_3}{\Rightarrow}$ NN 喜欢 吃 鱼

$\overset{r_1}{\Rightarrow}$ 猫 喜欢 吃 鱼

歧义1：相同的树对应不同的推导

生成同一棵句法树，我们可以使用不同的规则序列，即，不同的推导最终会得到相同的句法树

- 解决方法：规则使用都服从最左优先原则
- 这样得到的推导被称为最左优先推导，无歧义



推导1 - YES

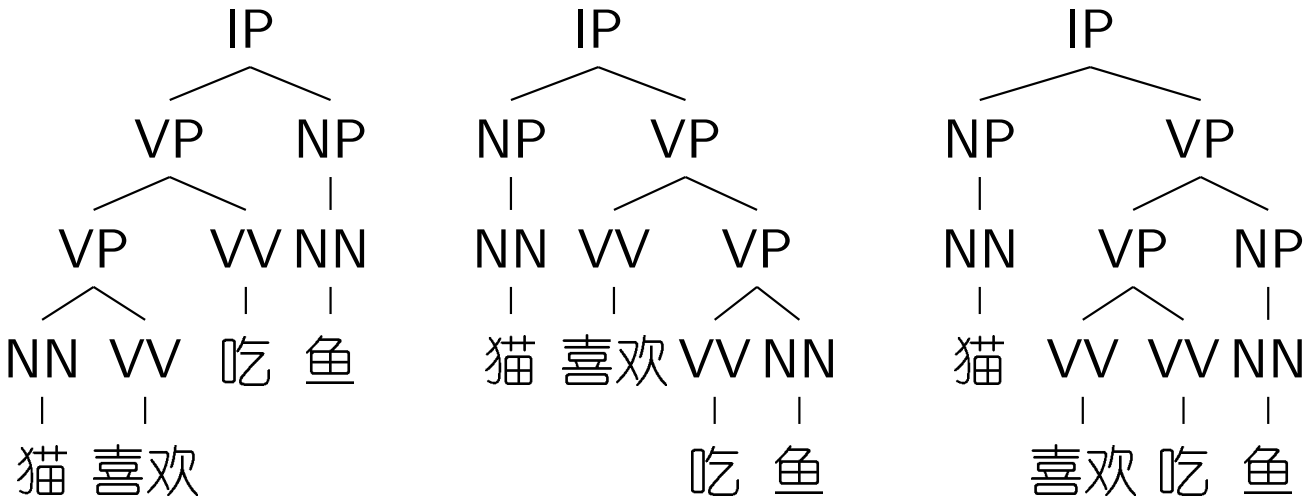
$$
\begin{aligned}
&\text{IP}\\
\overset{r_8}{\Rightarrow}\ &\text{NP VP}\\
\overset{r_5}{\Rightarrow}\ &\text{NN VP}\\
\overset{r_1}{\Rightarrow}\ &\text{猫 VP}\\
\overset{r_7}{\Rightarrow}\ &\text{猫 VV VP}\\
\overset{r_2}{\Rightarrow}\ &\text{猫 喜欢 VP}\\
\overset{r_6}{\Rightarrow}\ &\text{猫 喜欢 VV NN}\\
\overset{r_3}{\Rightarrow}\ &\text{猫 喜欢 吃 NN}\\
\overset{r_4}{\Rightarrow}\ &\text{猫 喜欢 吃 鱼}
\end{aligned}
$$

推导2 - NO

$$
\begin{aligned}
&\text{IP}\\
\overset{r_8}{\Rightarrow}\ &\text{NP VP}\\
\overset{r_7}{\Rightarrow}\ &\text{NP VV VP}\\
\overset{r_2}{\Rightarrow}\ &\text{NP 喜欢 VP}\\
\overset{r_6}{\Rightarrow}\ &\text{NP 喜欢 VV NN}\\
\overset{r_4}{\Rightarrow}\ &\text{NP 喜欢 VV 鱼}\\
\overset{r_5}{\Rightarrow}\ &\text{NN 喜欢 VV 鱼}\\
\overset{r_3}{\Rightarrow}\ &\text{NN 喜欢 吃 鱼}\\
\overset{r_1}{\Rightarrow}\ &\text{猫 喜欢 吃 鱼}
\end{aligned}
$$

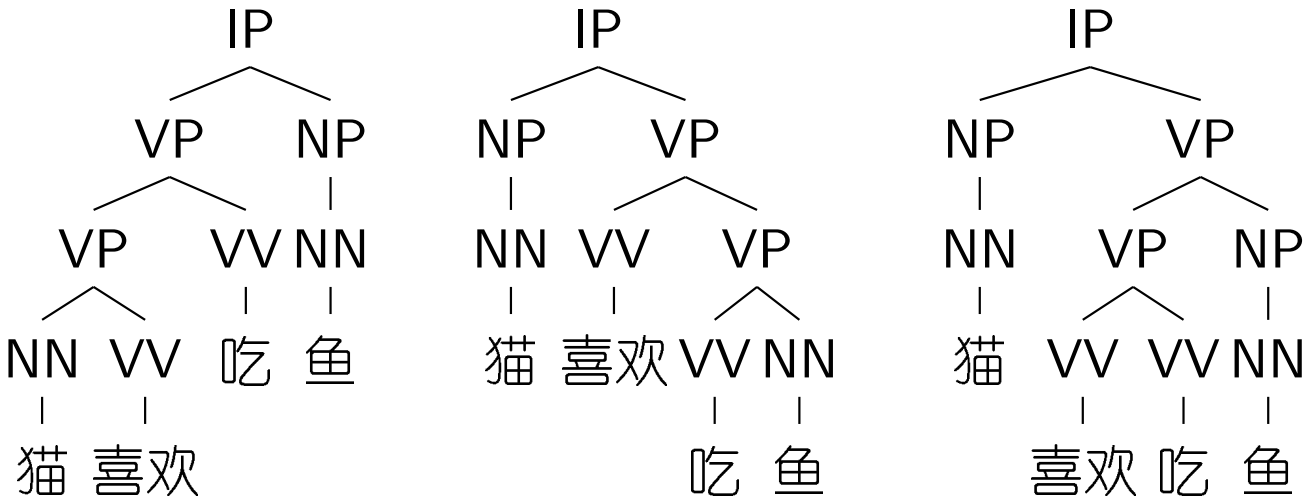
歧义2：相同的句子对应不同的句法分析树

生成一个单词序列，我们可以使用不同的最左推导，即，同一个句子可以有不同的树与之对应



歧义2：相同的句子对应不同的句法分析树

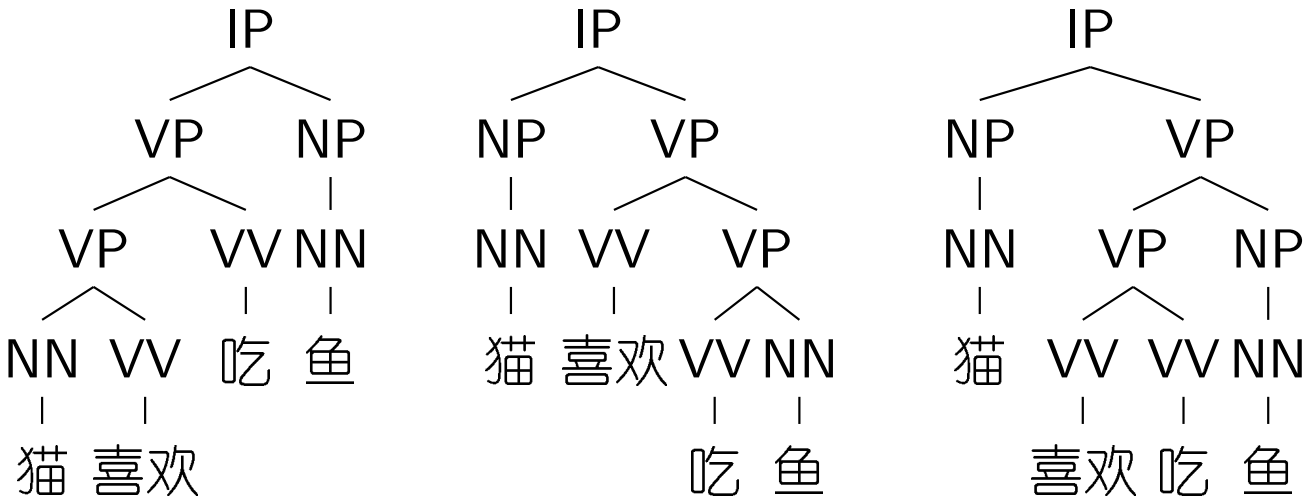
生成一个单词序列，我们可以使用不同的最左推导，即，同一个句子可以有不同的树与之对应



语言学家: 不对 对 不对

歧义2：相同的句子对应不同的句法分析树

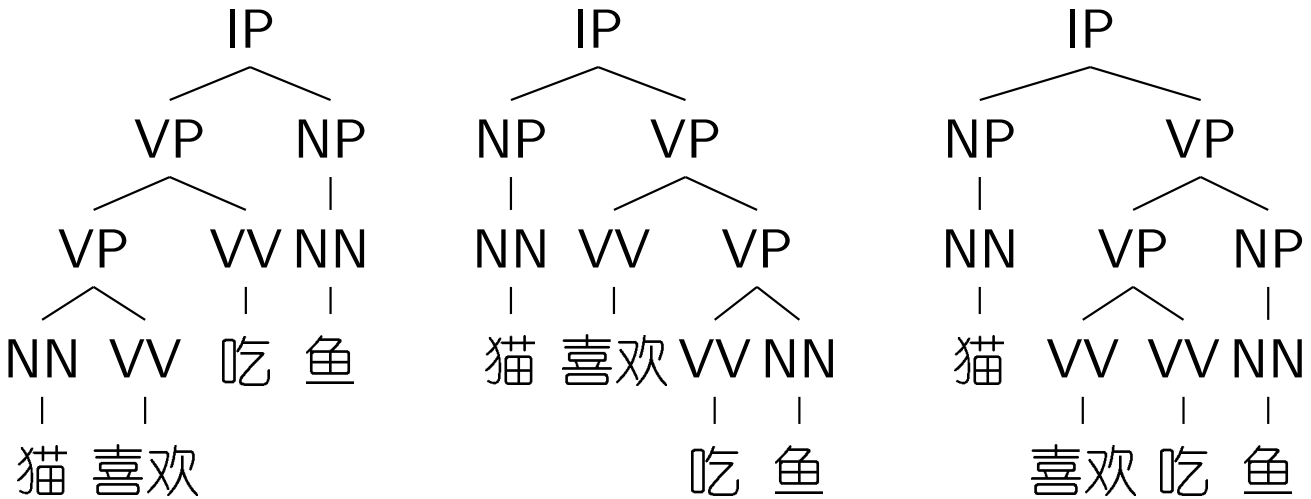
生成一个单词序列，我们可以使用不同的最左推导，即，同一个句子可以有不同的树与之对应



语言学家:	不对	对	不对
我们:	似乎对了	比较肯定	不太可能

歧义2：相同的句子对应不同的句法分析树

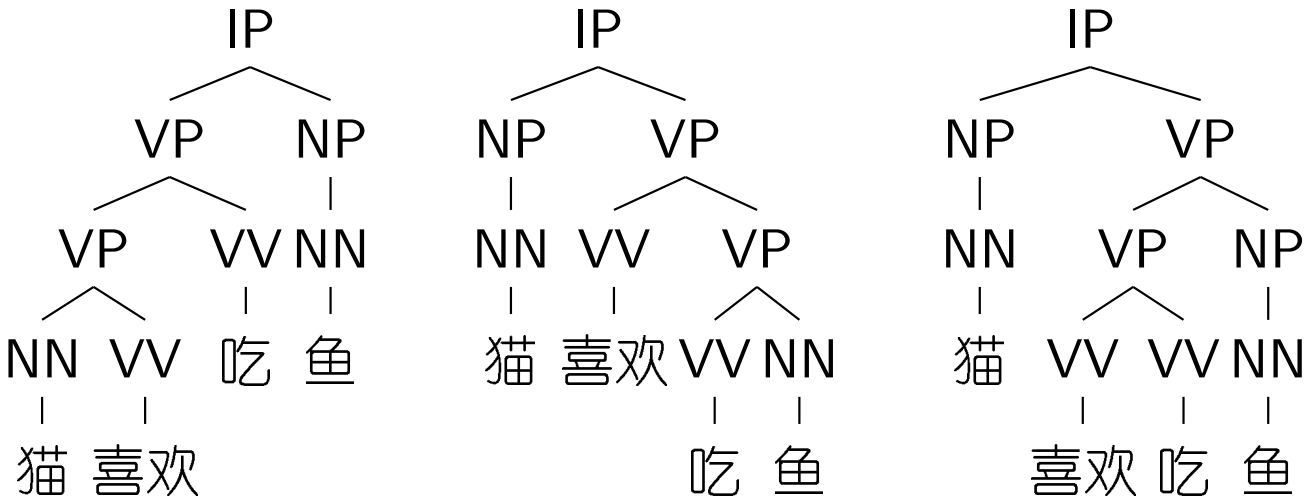
生成一个单词序列，我们可以使用不同的最左推导，即，同一个句子可以有不同的树与之对应



语言学家:	不对	对	不对
我们:	似乎对了	比较肯定	不太可能
分析器:	P=0.2	P=0.6	P=0.1

歧义2：相同的句子对应不同的句法分析树

生成一个单词序列，我们可以使用不同的最左推导，即，同一个句子可以有不同的树与之对应



语言学家:	不对	对	不对
我们:	似乎对了	比较肯定	不太可能
分析器:	P=0.2	P=0.6	P=0.1

- 句法分析任务本质上就是从所有可能的推导（句法树）中选择最优结果。
  - 人：确定的结果 - 对 or 错
  - 统计句法分析系统：所有结果都有可能，只是出现的概率不同

统计句法分析

- 我们现在知道，在给定上下文无关文法(CFG)的情况下:
  - 每个(最左)推导都对应一棵句法树 (反之不一定成立)

$$d = \text{CFG句法分析树}$$

  - 每个句子可能对应多个翻译推导

统计句法分析

- 我们现在知道，在给定上下文无关文法(CFG)的情况下:
  - 每个(最左)推导都对应一棵句法树 (反之不一定成立)

$$d = \text{CFG句法分析树}$$

  - 每个句子可能对应多个翻译推导
- 统计句法分析：对于任意的句子，找到概率最大的句法树(推导)输出

统计句法分析

- 我们现在知道，在给定上下文无关文法(CFG)的情况下:
  - 每个(最左)推导都对应一棵句法树 (反之不一定成立)

$$d = \text{CFG句法分析树}$$

  - 每个句子可能对应多个翻译推导
- 统计句法分析：对于任意的句子，找到概率最大的句法树(推导)输出

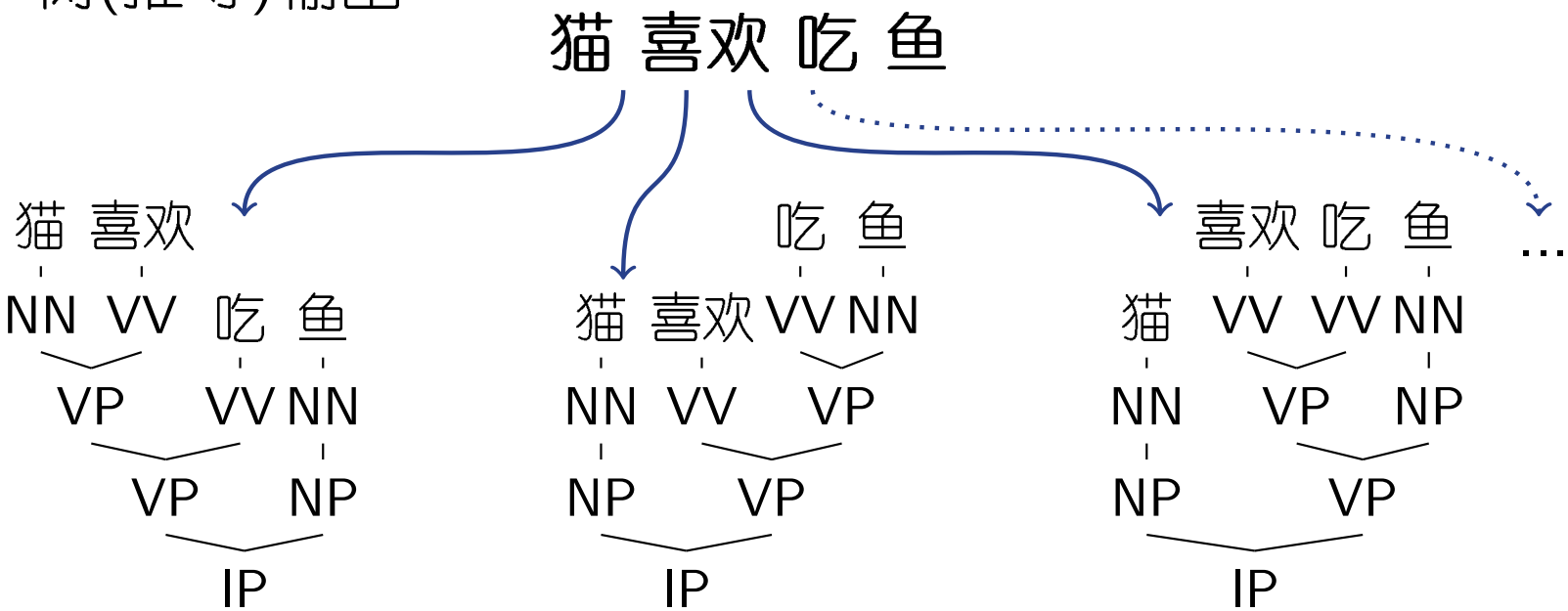
猫 喜欢 吃 鱼

统计句法分析

- 我们现在知道，在给定上下文无关文法(CFG)的情况下:
  - 每个(最左)推导都对应一棵句法树 (反之不一定成立)

$$d = \text{CFG句法分析树}$$

  - 每个句子可能对应多个翻译推导
- 统计句法分析：对于任意的句子，找到概率最大的句法树(推导)输出

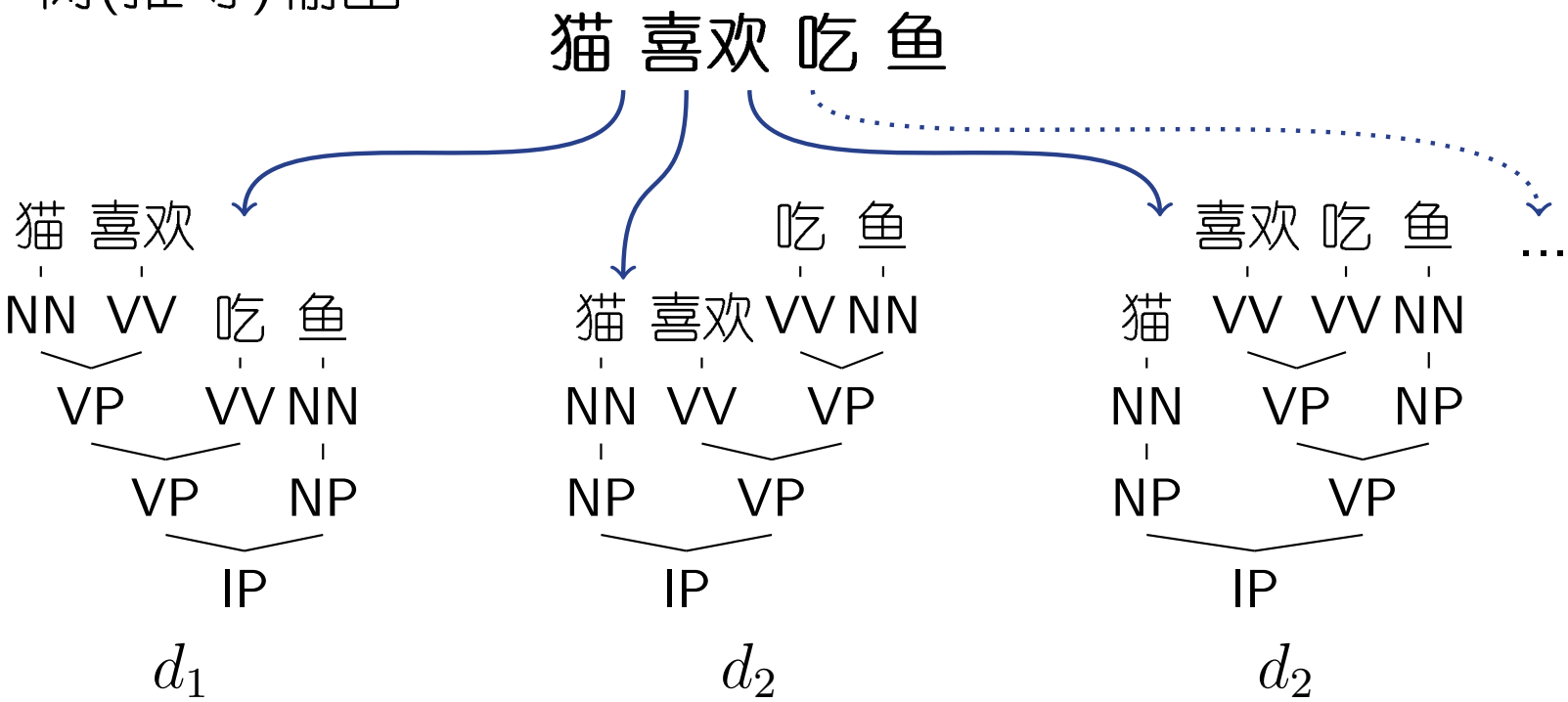


统计句法分析

- 我们现在知道，在给定上下文无关文法(CFG)的情况下:
  - 每个(最左)推导都对应一棵句法树 (反之不一定成立)

$$d = \text{CFG句法分析树}$$

  - 每个句子可能对应多个翻译推导
- 统计句法分析：对于任意的句子，找到概率最大的句法树(推导)输出

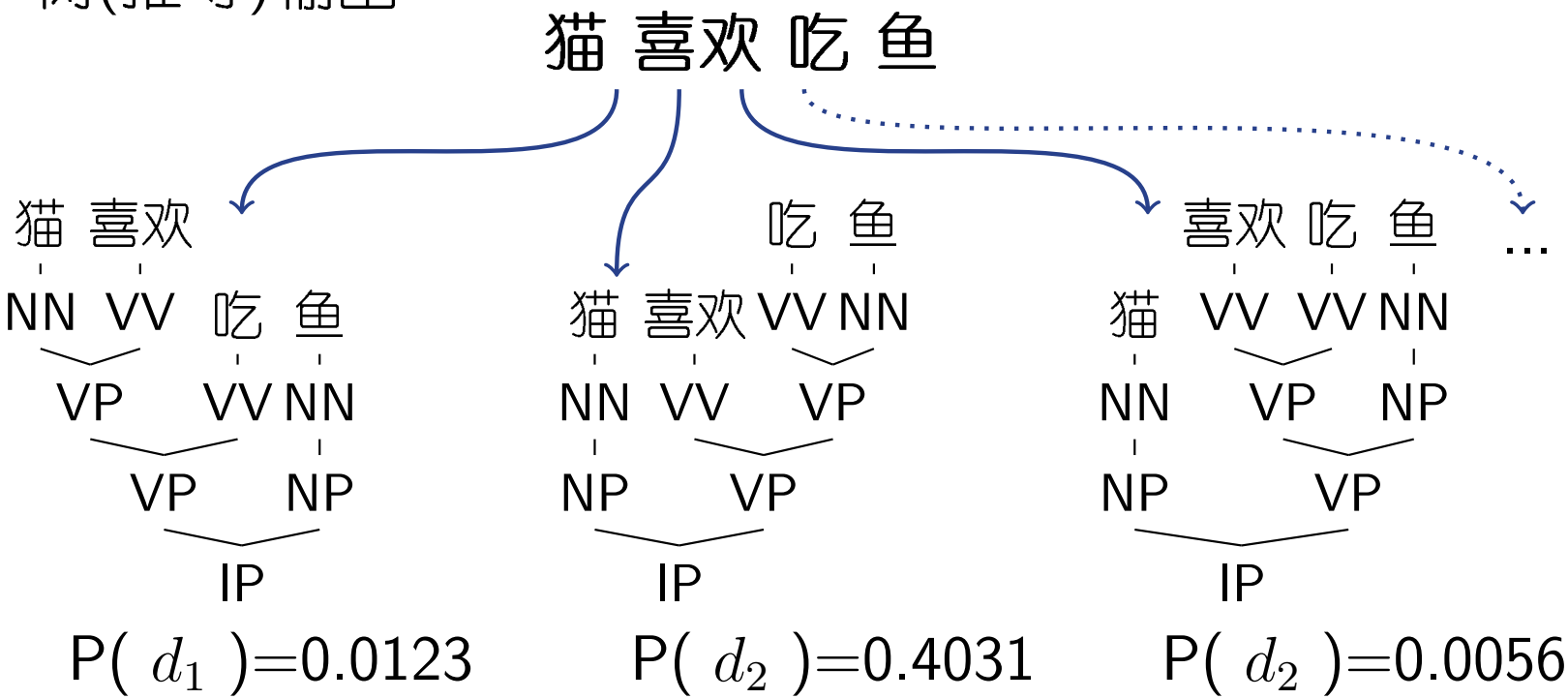


统计句法分析

- 我们现在知道，在给定上下文无关文法(CFG)的情况下:
  - 每个(最左)推导都对应一棵句法树 (反之不一定成立)

$$d = \text{CFG句法分析树}$$

  - 每个句子可能对应多个翻译推导
- 统计句法分析：对于任意的句子，找到概率最大的句法树(推导)输出

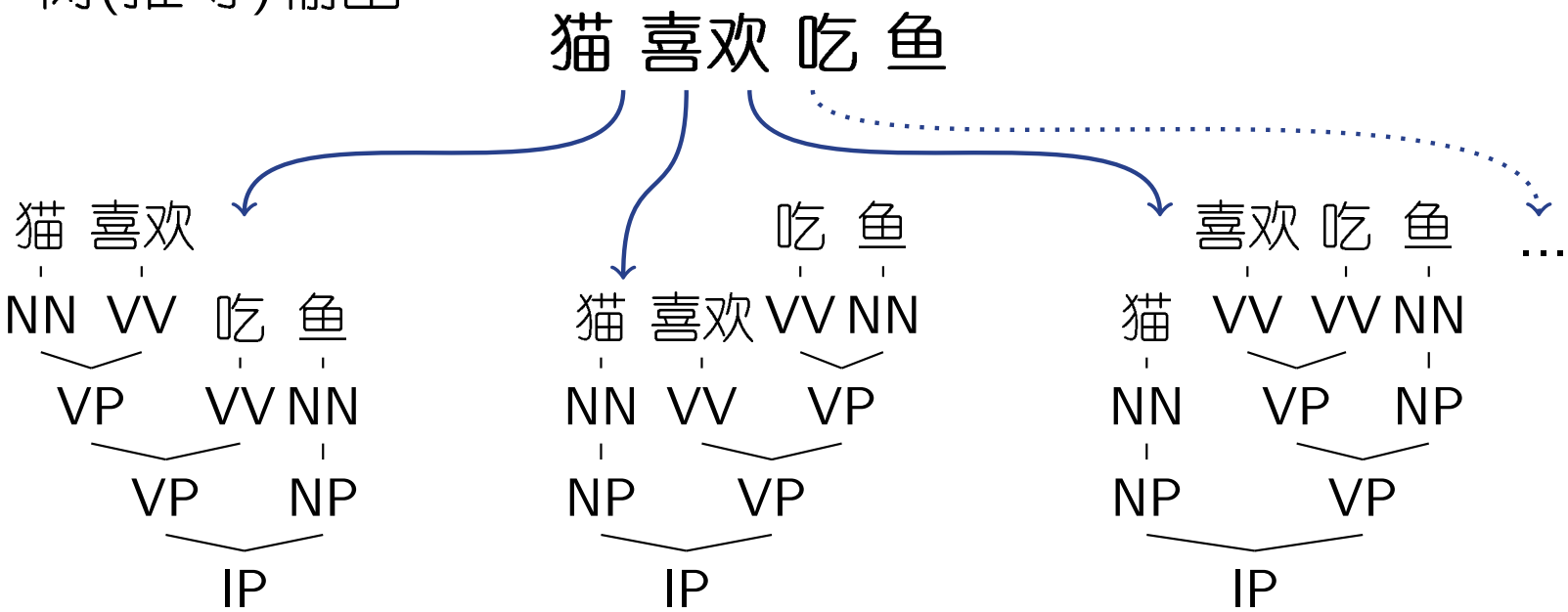


统计句法分析

- 我们现在知道，在给定上下文无关文法(CFG)的情况下:
  - 每个(最左)推导都对应一棵句法树 (反之不一定成立)

$$d = \text{CFG句法分析树}$$

  - 每个句子可能对应多个翻译推导
- 统计句法分析：对于任意的句子，找到概率最大的句法树(推导)输出



如何计算每棵句法树的概率，即推导的概率P(d)???

P(d)=?

- 在上下文无关文法中，每条规则之间的使用是相互独立的，因此可以把P(d)分解为规则概率的乘积

$$\begin{aligned} \mathrm{P}(d) &= \mathrm{P}(r_1 \circ r_2 \circ \ldots \circ r_n) \\ &\equiv \mathrm{P}(r_1) \cdot \mathrm{P}(r_2) \cdot \ldots \cdot \mathrm{P}(r_n) \end{aligned}$$

注意：上下文无关文法本身就包含规则使用的独立性假设，因此上式在CFG框架下并不是近似

P(d)=?

- 在上下文无关文法中，每条规则之间的使用是相互独立的，因此可以把P(d)分解为规则概率的乘积

$$\begin{aligned} \mathrm{P}(d) &= \mathrm{P}(r_1 \circ r_2 \circ \ldots \circ r_n) \\ &\equiv \mathrm{P}(r_1) \cdot \mathrm{P}(r_2) \cdot \ldots \cdot \mathrm{P}(r_n) \end{aligned}$$

  注意：上下文无关文法本身就包含规则使用的独立性假设，因此上式在CFG框架下并不是近似
- 不太简单的问题：如何得到规则的生成概率？
  简单的方法：假设我们有人工标注的数据，其中包括很多句子的人工标注的句法树，称之为树库。然后，对于规则$r : \alpha \rightarrow \beta$

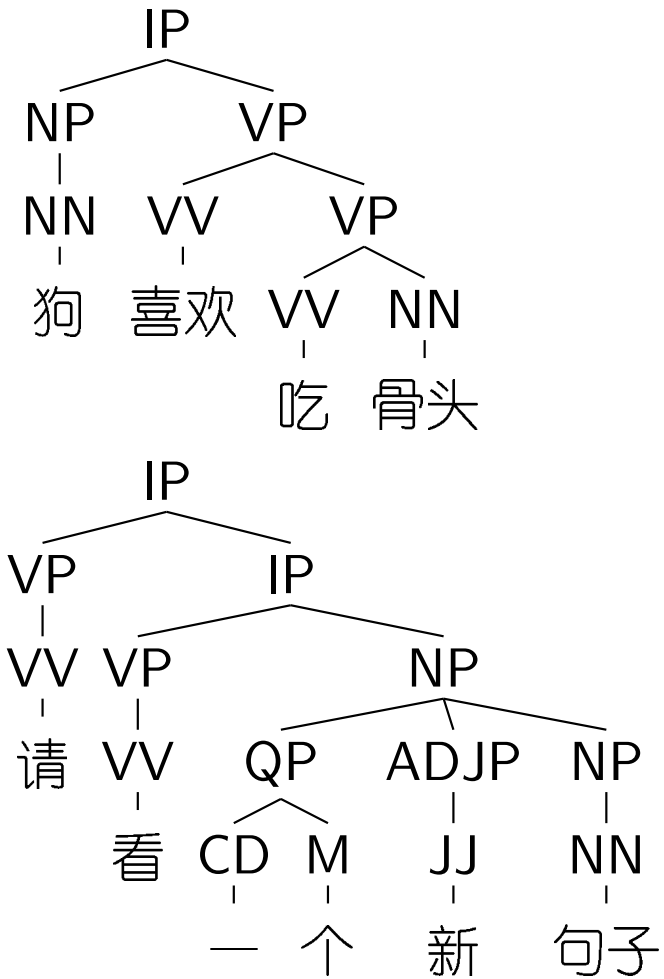
$$\mathrm{P}(r) = \frac{\text{规则}r\text{在树库中出现的次数}}{\alpha\text{在树库中出现的次数}}$$

  这里$\mathrm{P}(r) \equiv \mathrm{P}(\beta|\alpha)$，即规则的概率就是给定左手端生成右手端的概率。

P(r)=?

不用看前面的公式，来个例子

数据：树库

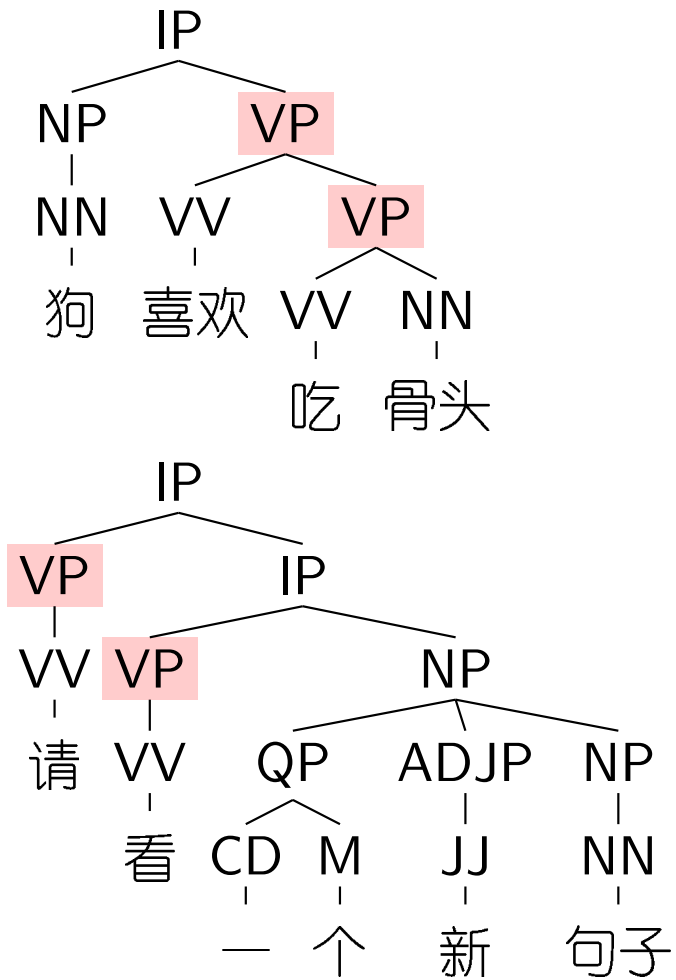


$$P(\text{'VP} \rightarrow \text{VV NN'}) = \frac{\text{'VP'和'VV NN'同时出现的次数}}{\text{'VP'出现的次数}}$$

P(r)=?

不用看前面的公式，来个例子

数据：树库

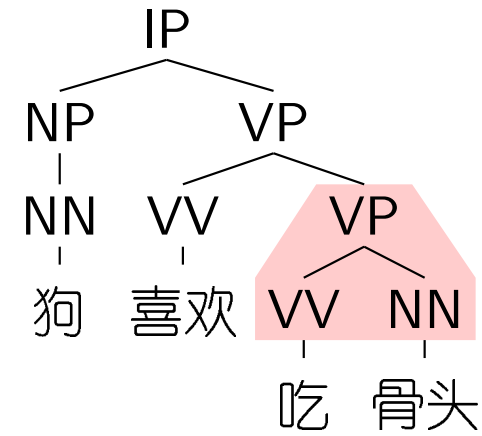


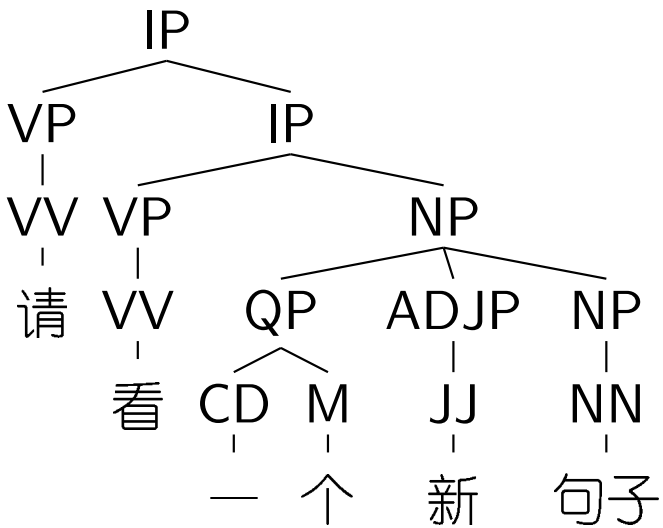
$$\text{P('VP} \rightarrow \text{VV NN')} = \frac{\text{'VP'和'VV NN'同时出现的次数}}{\text{'VP'出现的次数=4}}$$

P(r)=?

不用看前面的公式，来个例子

数据：树库



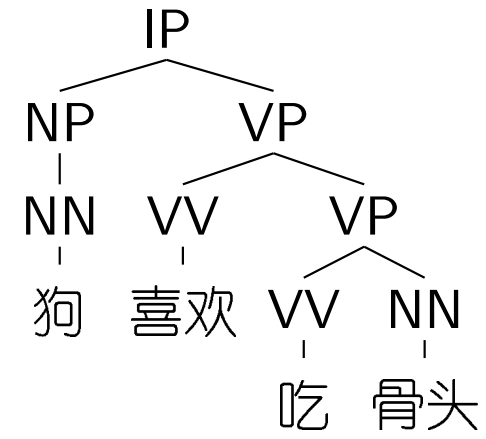


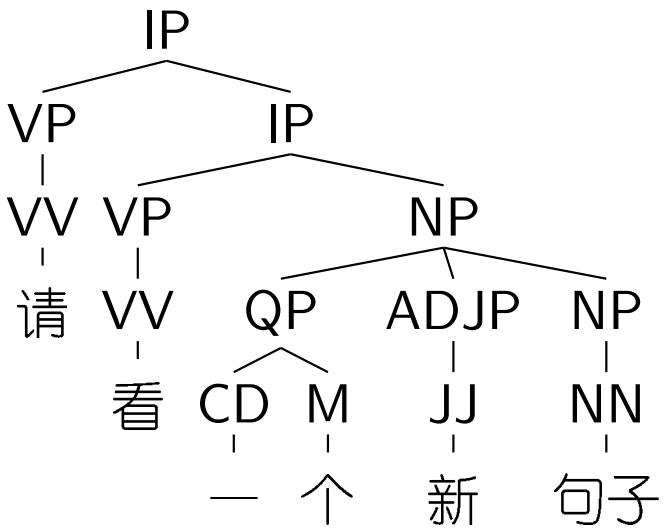
$$P(\text{'VP} \rightarrow \text{VV NN'}) = \frac{\text{'VP'和'VV NN'同时出现的次数=1}}{\text{'VP'出现的次数=4}}$$

P(r)=?

不用看前面的公式，来个例子

数据：树库



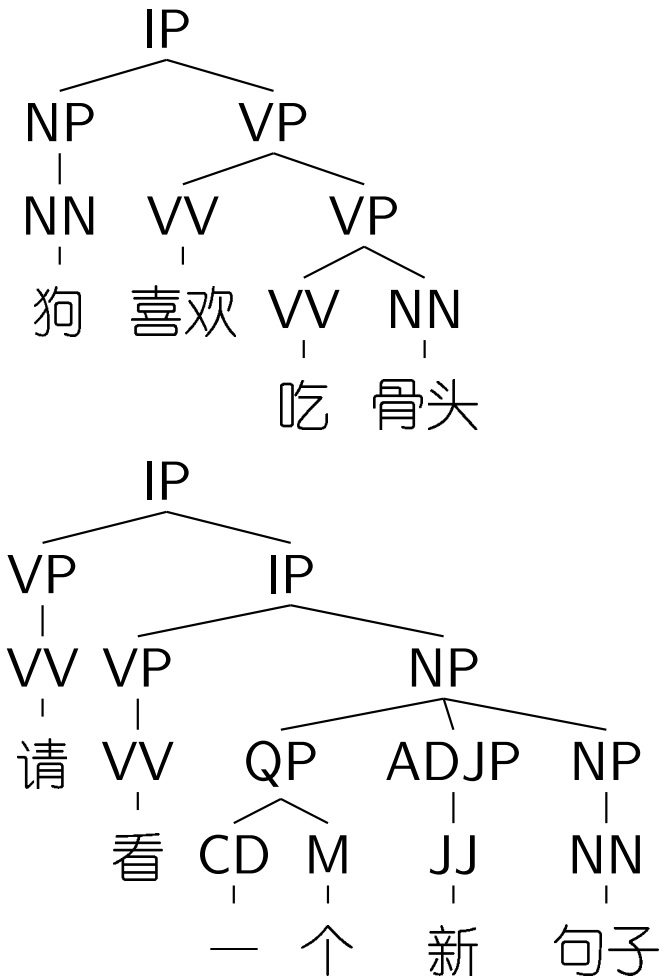


$$P(\text{'VP} \rightarrow \text{VV NN'}) = \frac{\text{'VP'和'VV NN'同时出现的次数=1}}{\text{'VP'出现的次数=4}} = \frac{1}{4}$$

P(r)=?

不用看前面的公式，来个例子

数据：树库



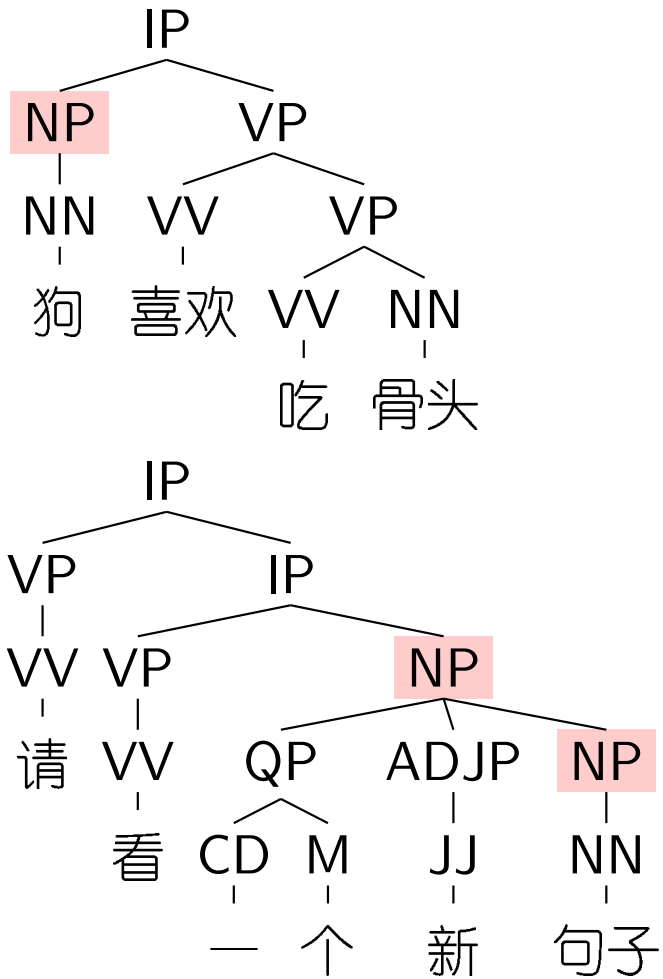
$$P(\text{'VP} \rightarrow \text{VV NN'}) = \frac{\text{'VP'和'VV NN'同时出现的次数=1}}{\text{'VP'出现的次数=4}} = \frac{1}{4}$$

$$P(\text{'NP} \rightarrow \text{NN'}) = \frac{\text{'NP'和'NN'同时出现的次数}}{\text{'NP'出现的次数}}$$

P(r)=?

不用看前面的公式，来个例子

数据：树库



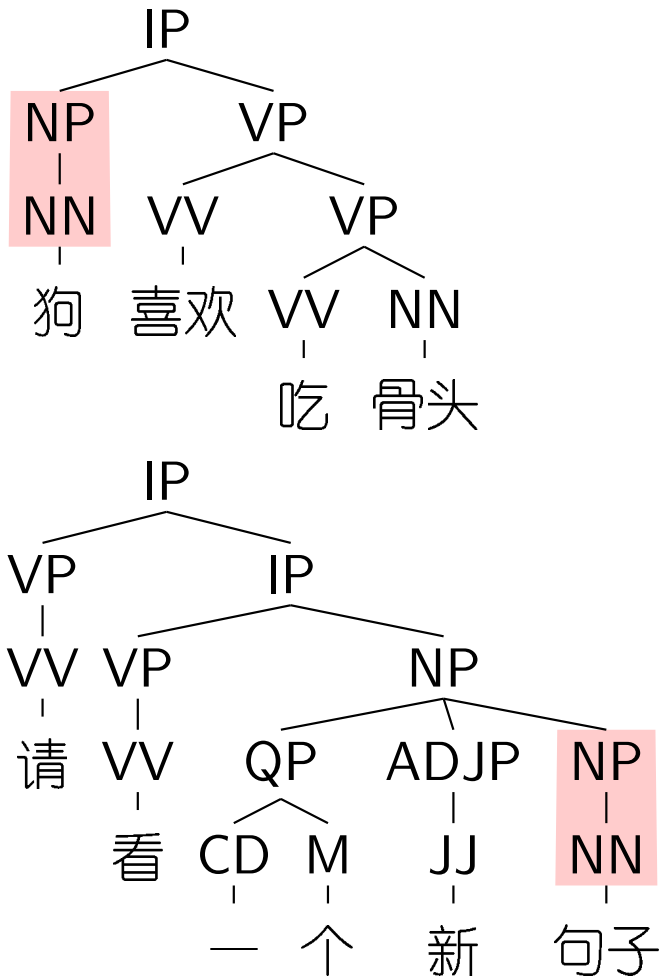
$$P(\text{'VP} \rightarrow \text{VV NN'}) = \frac{\text{'VP'和'VV NN'同时出现的次数=1}}{\text{'VP'出现的次数=4}} = \frac{1}{4}$$

$$P(\text{'NP} \rightarrow \text{NN'}) = \frac{\text{'NP'和'NN'同时出现的次数}}{\text{'NP'出现的次数=3}}$$

P(r)=?

不用看前面的公式，来个例子

数据：树库



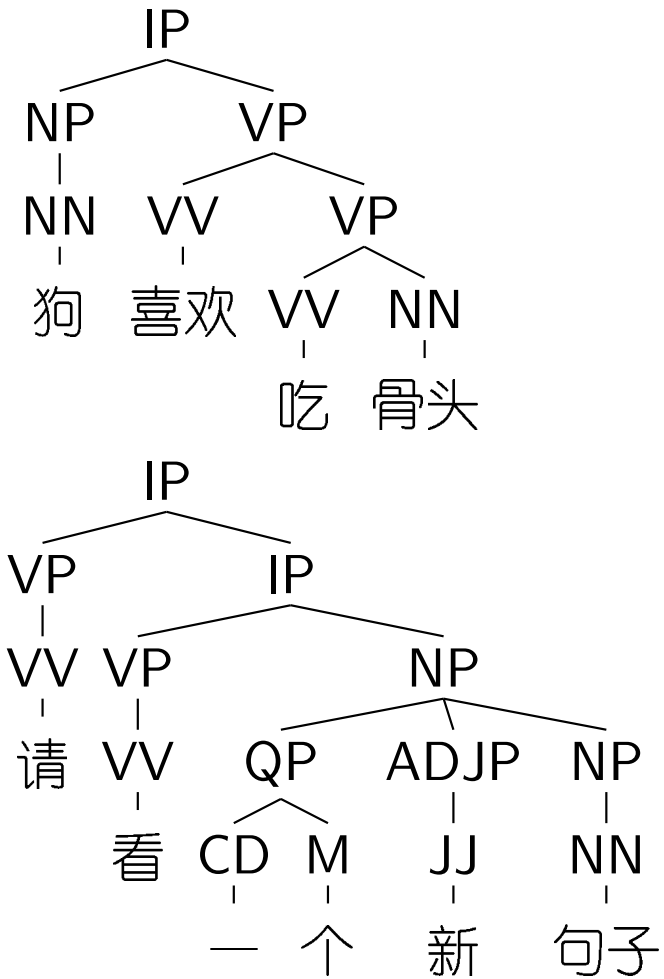
$$P(\text{'VP} \rightarrow \text{VV NN'}) = \frac{\text{'VP'和'VV NN'同时出现的次数=1}}{\text{'VP'出现的次数=4}} = \frac{1}{4}$$

$$P(\text{'NP} \rightarrow \text{NN'}) = \frac{\text{'NP'和'NN'同时出现的次数=2}}{\text{'NP'出现的次数=3}}$$

P(r)=?

不用看前面的公式，来个例子

数据：树库



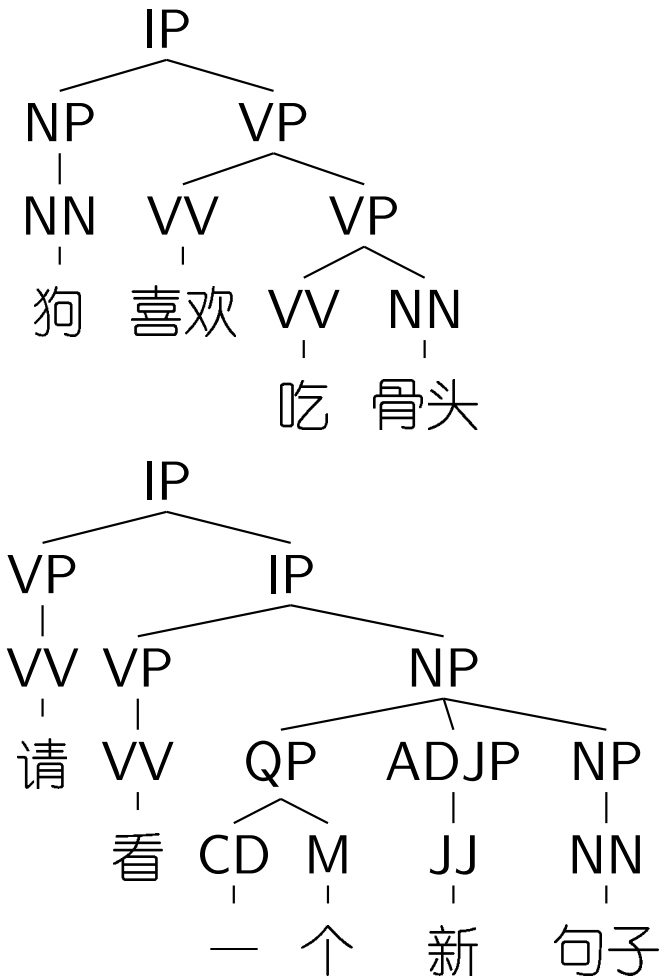
$$P(\text{'VP} \rightarrow \text{VV NN'}) = \frac{\text{'VP'和'VV NN'同时出现的次数=1}}{\text{'VP'出现的次数=4}} = \frac{1}{4}$$

$$P(\text{'NP} \rightarrow \text{NN'}) = \frac{\text{'NP'和'NN'同时出现的次数=2}}{\text{'NP'出现的次数=3}} = \frac{2}{3}$$

P(r)=?

不用看前面的公式，来个例子

数据：树库



$$P(\text{'VP} \rightarrow \text{VV NN'}) = \frac{\text{'VP'和'VV NN'同时出现的次数=1}}{\text{'VP'出现的次数=4}} = \frac{1}{4}$$

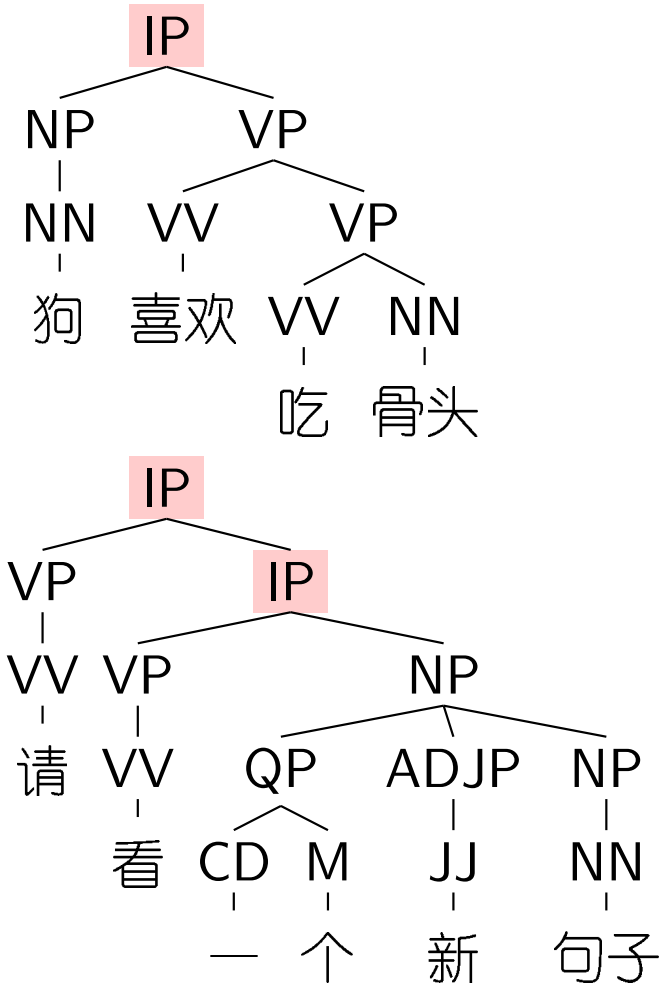
$$P(\text{'NP} \rightarrow \text{NN'}) = \frac{\text{'NP'和'NN'同时出现的次数=2}}{\text{'NP'出现的次数=3}} = \frac{2}{3}$$

$$P(\text{'IP} \rightarrow \text{NP NP'}) = \frac{\text{'IP'和'NP NP'同时出现的次数}}{\text{'IP'出现的次数}}$$

P(r)=?

不用看前面的公式，来个例子

数据：树库



$$P(\text{'VP} \rightarrow \text{VV NN'}) = \frac{\text{'VP'和'VV NN'同时出现的次数=1}}{\text{'VP'出现的次数=4}} = \frac{1}{4}$$

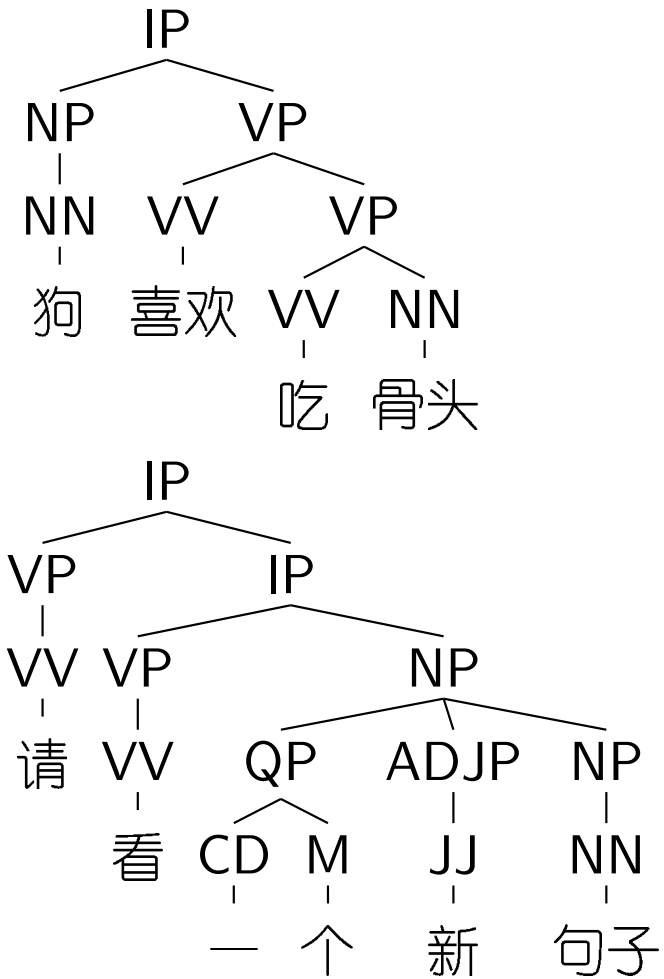
$$P(\text{'NP} \rightarrow \text{NN'}) = \frac{\text{'NP'和'NN'同时出现的次数=2}}{\text{'NP'出现的次数=3}} = \frac{2}{3}$$

$$P(\text{'IP} \rightarrow \text{NP NP'}) = \frac{\text{'IP'和'NP NP'同时出现的次数}}{\text{'IP'出现的次数=3}}$$

P(r)=?

不用看前面的公式，来个例子

数据：树库



$$P(\text{'VP} \rightarrow \text{VV NN'}) = \frac{\text{'VP'和'VV NN'同时出现的次数=1}}{\text{'VP'出现的次数=4}} = \frac{1}{4}$$

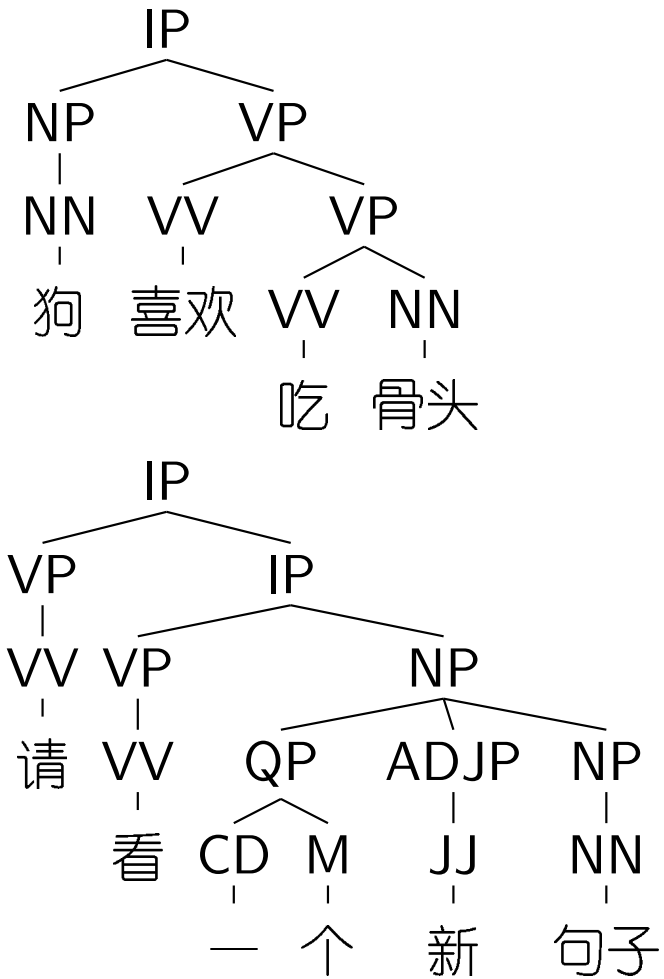
$$P(\text{'NP} \rightarrow \text{NN'}) = \frac{\text{'NP'和'NN'同时出现的次数=2}}{\text{'NP'出现的次数=3}} = \frac{2}{3}$$

$$P(\text{'IP} \rightarrow \text{NP NP'}) = \frac{\text{'IP'和'NP NP'同时出现的次数=0}}{\text{'IP'出现的次数=3}}$$

P(r)=?

不用看前面的公式，来个例子

数据：树库

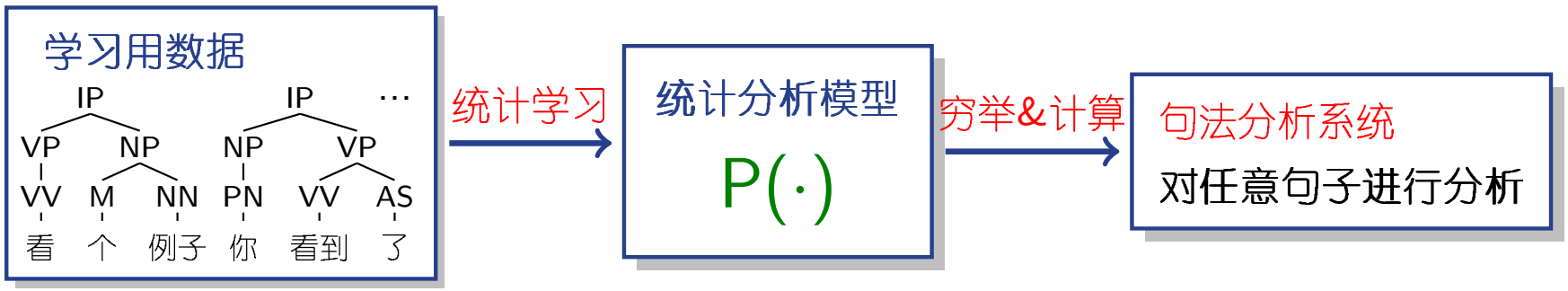


$$P(\text{'VP} \rightarrow \text{VV NN'}) = \frac{\text{'VP'和'VV NN'同时出现的次数=1}}{\text{'VP'出现的次数=4}} = \frac{1}{4}$$

$$P(\text{'NP} \rightarrow \text{NN'}) = \frac{\text{'NP'和'NN'同时出现的次数=2}}{\text{'NP'出现的次数=3}} = \frac{2}{3}$$

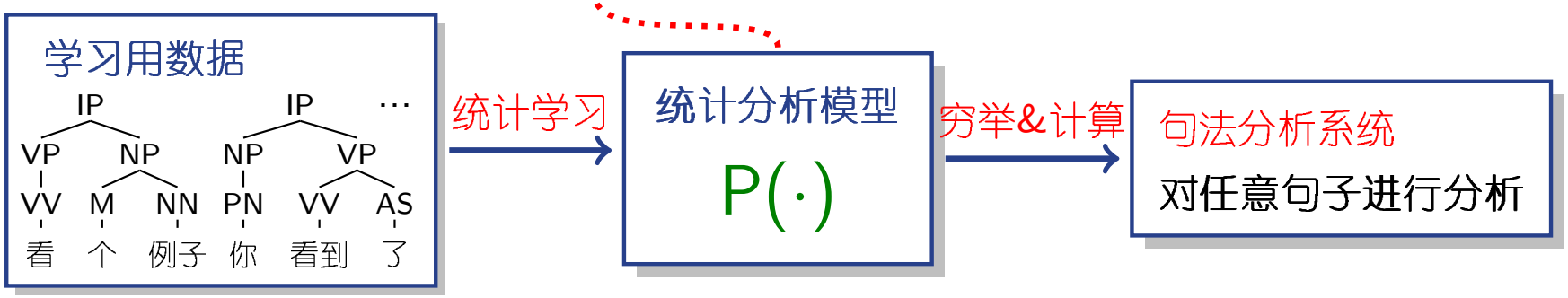
$$P(\text{'IP} \rightarrow \text{NP NP'}) = \frac{\text{'IP'和'NP NP'同时出现的次数=0}}{\text{'IP'出现的次数=3}} = 0 \text{ or 一个很小的值(平滑?)}$$

句法分析过程


学习用数据
IP
VP
NP
VV
M
NN
看
个
例子
IP
NP
VP
PN
VV
AS
你
看到
了
…
统计学习
统计分析模型
P(·)
穷举&计算
句法分析系统
对任意句子进行分析

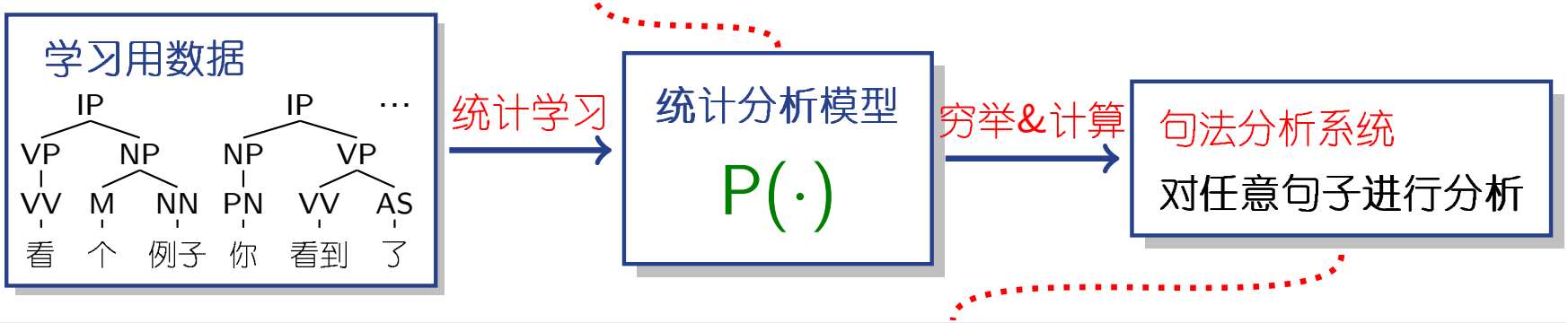
句法分析过程

我们学习得到了一个(CFG)句法分析模型P(·)，对任意的句法分析结果$d = r_1 \circ r_2 \circ ... \circ r_n$，都能通过P($d$)=$\prod_{i=1}^{n} \mathrm{P}(r_i)$计算其概率值



句法分析过程

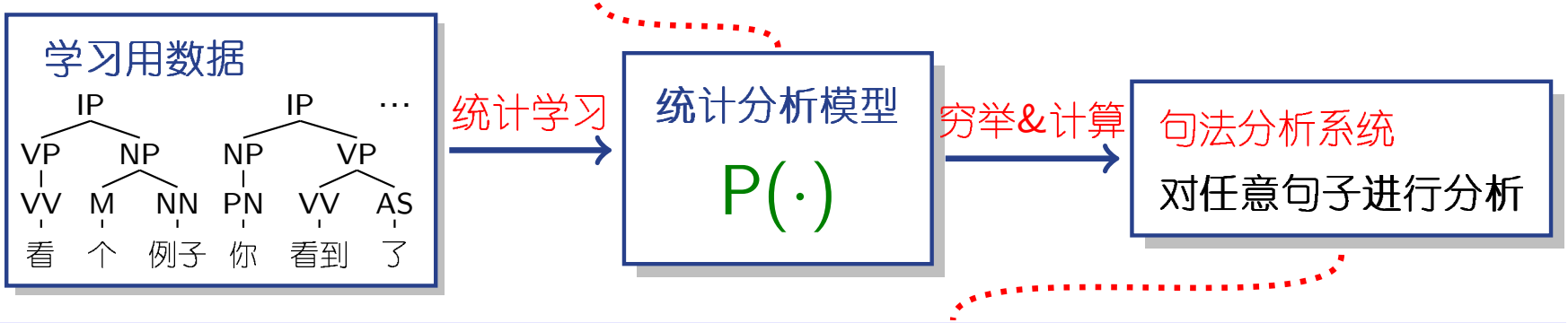
我们学习得到了一个(CFG)句法分析模型P(·)，对任意的句法分析结果$d = r_1 \circ r_2 \circ ... \circ r_n$，都能通过P($d$)=$\prod_{i=1}^{n} \mathrm{P}(r_i)$计算其概率值

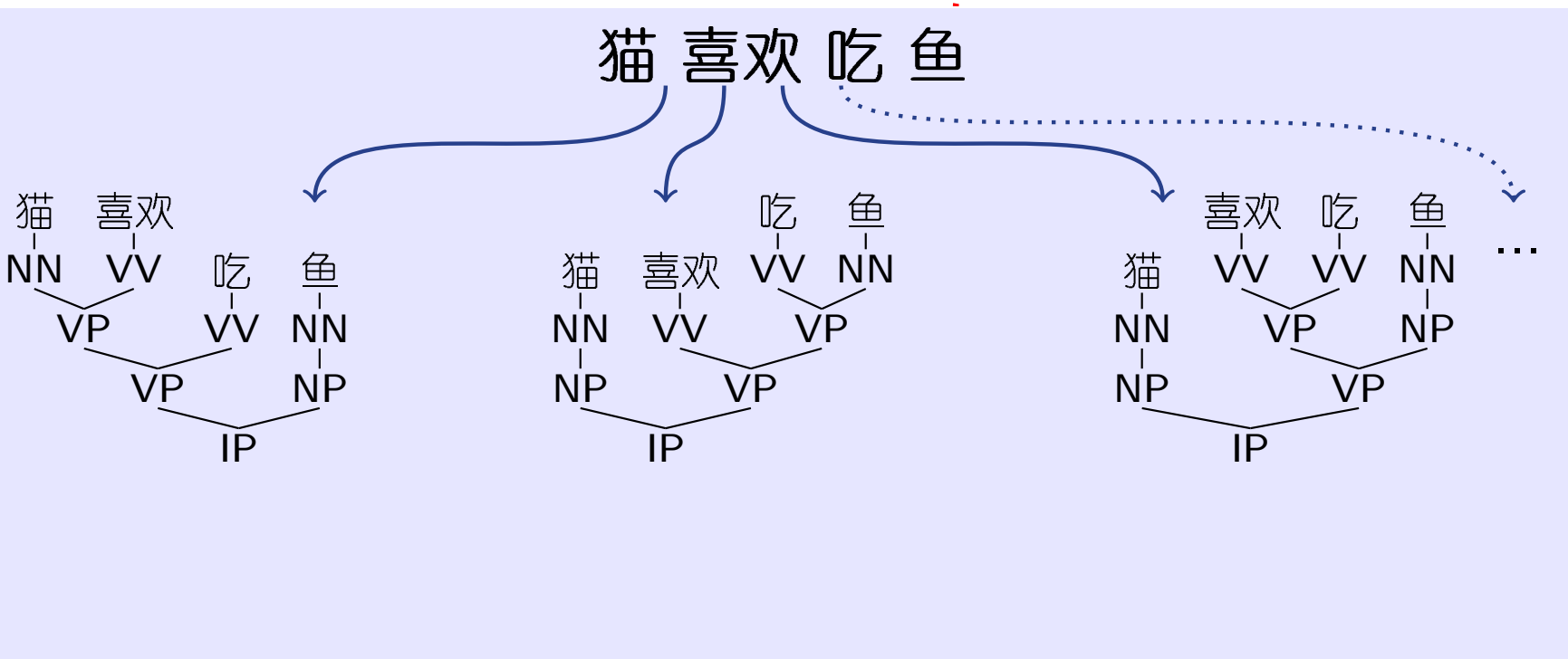


猫 喜欢 吃 鱼

句法分析过程

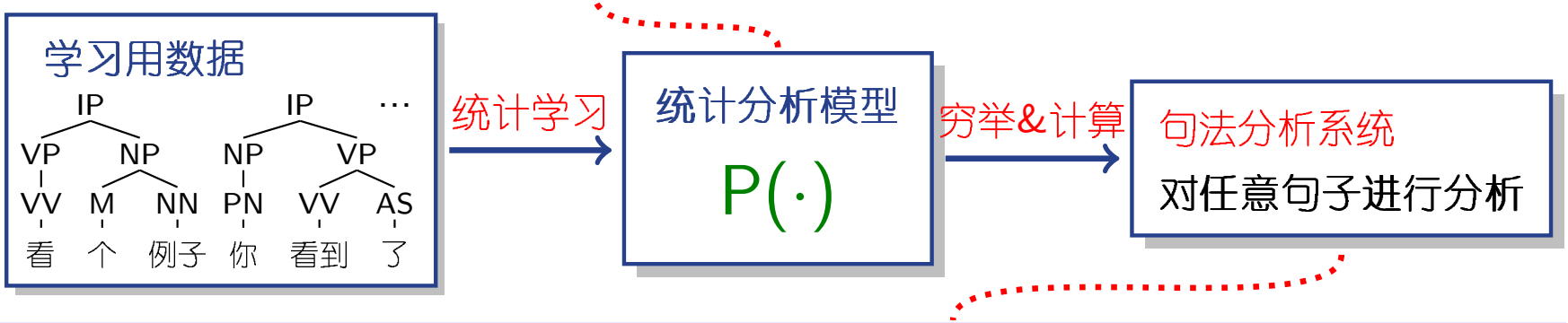
我们学习得到了一个(CFG)句法分析模型P(·)，对任意的句法分析结果$d = r_1 \circ r_2 \circ ... \circ r_n$，都能通过P($d$)=$\prod_{i=1}^{n} \mathrm{P}(r_i)$计算其概率值

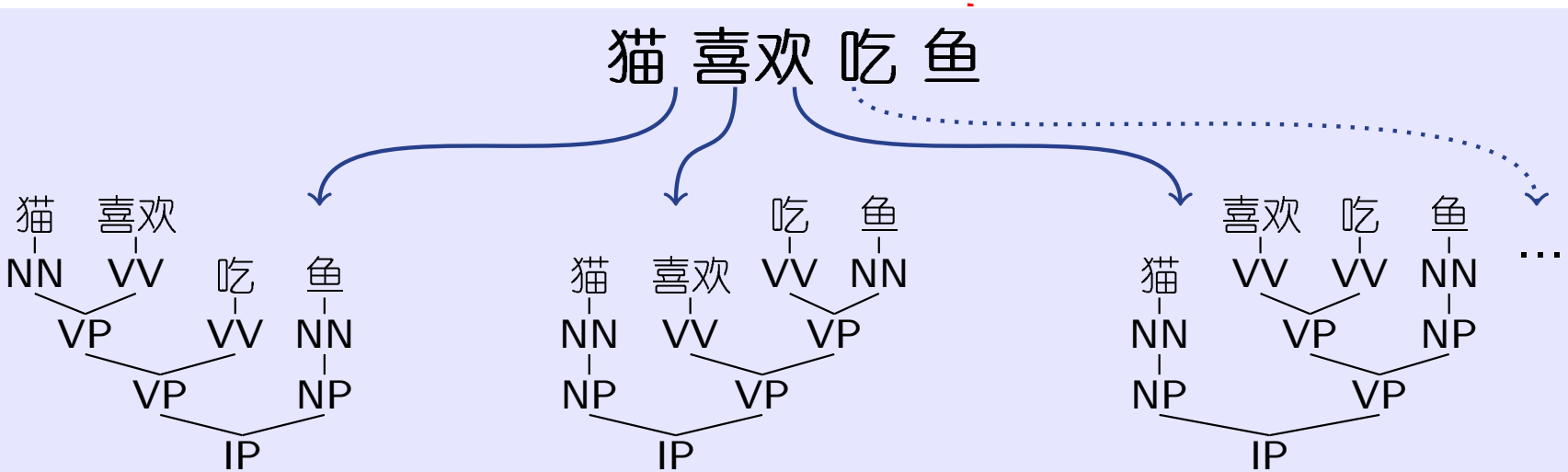




句法分析过程

我们学习得到了一个(CFG)句法分析模型P(·)，对任意的句法分析结果$d = r_1 \circ r_2 \circ ... \circ r_n$，都能通过P(d)=$\prod_{i=1}^{n} \mathrm{P}(r_i)$计算其概率值

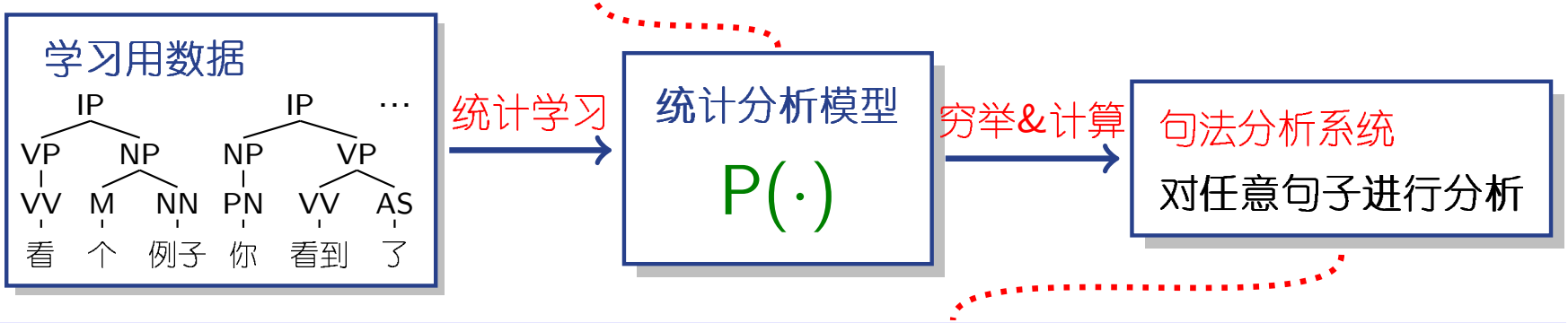




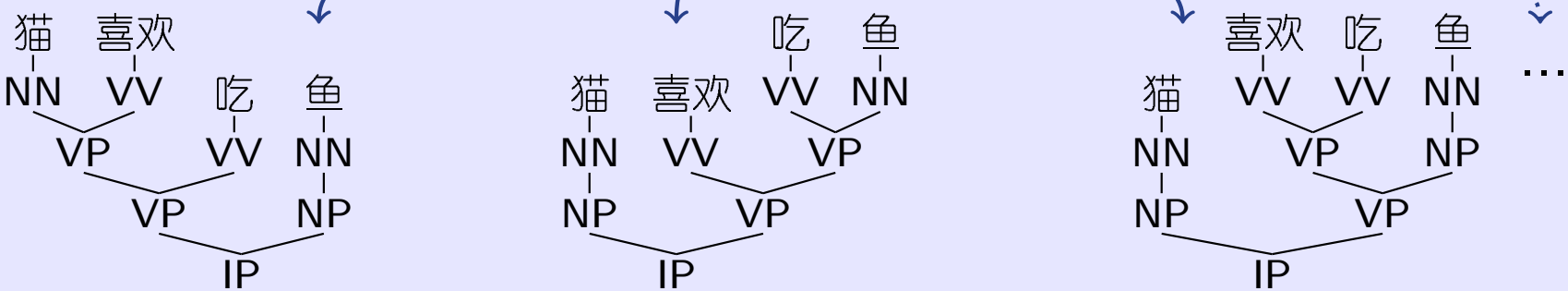
$$\mathrm{P}(d_1) = \mathrm{P}(\mathrm{IP} \rightarrow \mathrm{NP\ VP}) \cdot \mathrm{P}(\mathrm{NP} \rightarrow \mathrm{NN}) \cdot \mathrm{P}(\mathrm{NN} \rightarrow 猫) \cdot \mathrm{P}(\mathrm{VP} \rightarrow \mathrm{VP\ VV}) \cdot$$
$$\mathrm{P}(\mathrm{VV} \rightarrow 喜欢) \cdot \mathrm{P}(\mathrm{VP} \rightarrow \mathrm{NN\ VV}) \cdot \mathrm{P}(\mathrm{VV} \rightarrow 吃) \cdot \mathrm{P}(\mathrm{NN} \rightarrow 鱼)$$
$$=0.0004$$

句法分析过程

我们学习得到了一个(CFG)句法分析模型P(·)，对任意的句法分析结果$d = r_1 \circ r_2 \circ ... \circ r_n$，都能通过P(d)=$\prod_{i=1}^{n} \mathrm{P}(r_i)$计算其概率值



猫 喜欢 吃 鱼



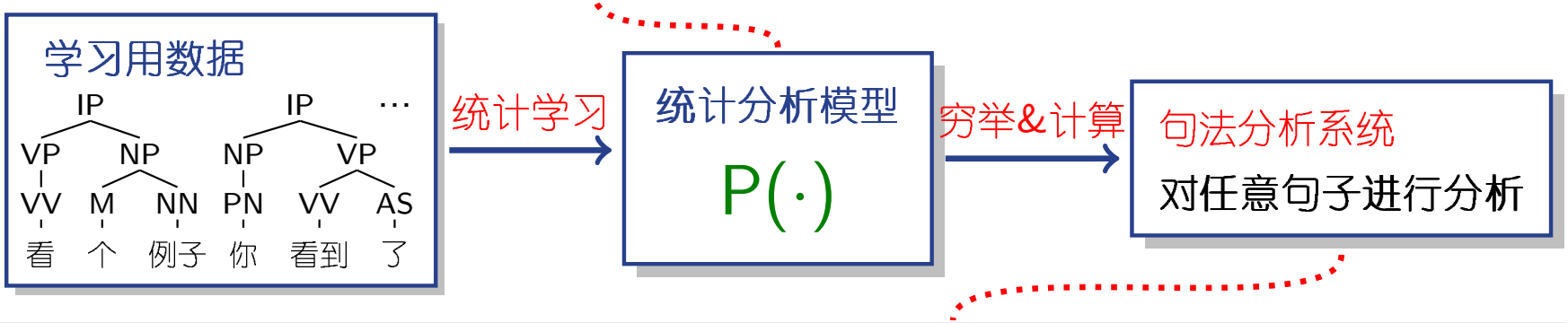
$\mathrm{P}(d_1) = \prod_{r_i \in d_1} \mathrm{P}(r_i)$ =0.0004

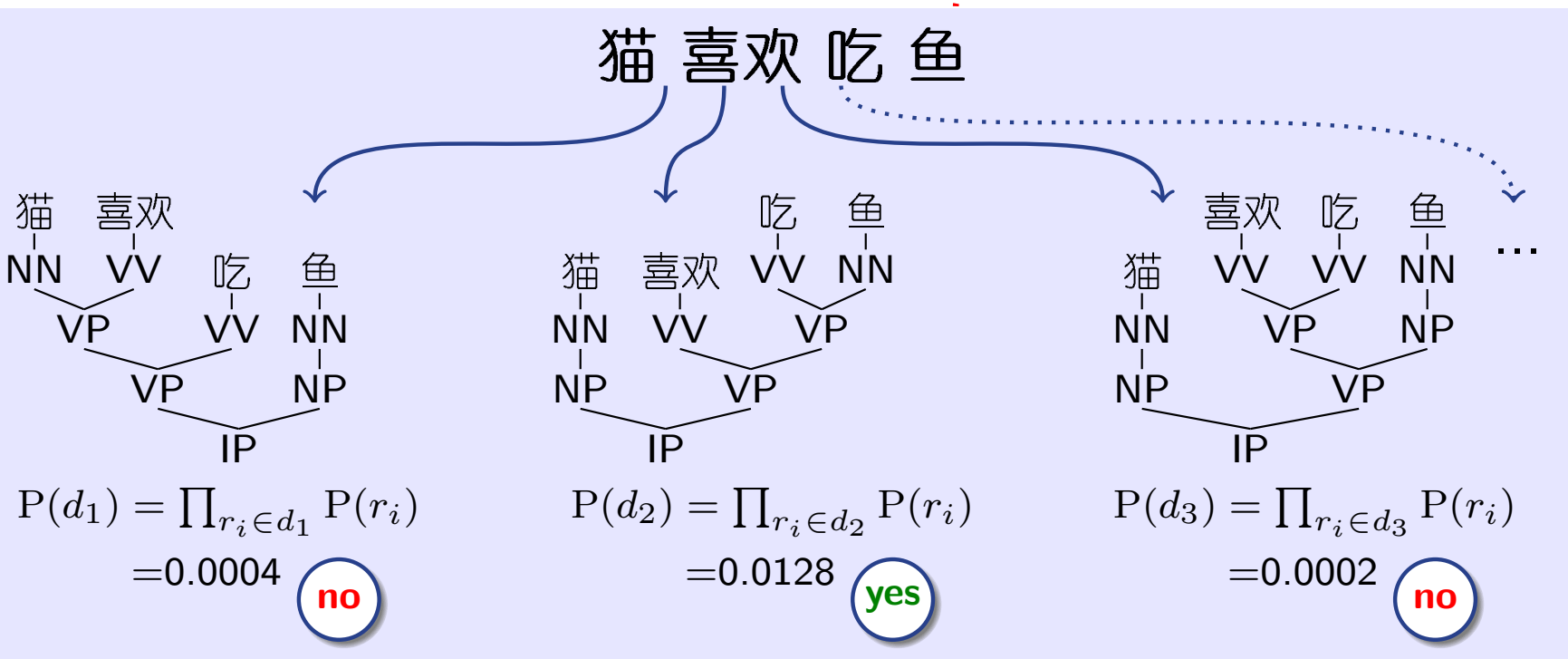
$\mathrm{P}(d_2) = \prod_{r_i \in d_2} \mathrm{P}(r_i)$ =0.0128

$\mathrm{P}(d_3) = \prod_{r_i \in d_3} \mathrm{P}(r_i)$ =0.0002

句法分析过程

我们学习得到了一个(CFG)句法分析模型P(·)，对任意的句法分析结果$d = r_1 \circ r_2 \circ ... \circ r_n$，都能通过P(d)=$\prod_{i=1}^{n} \mathrm{P}(r_i)$计算其概率值





句法分析并不简单

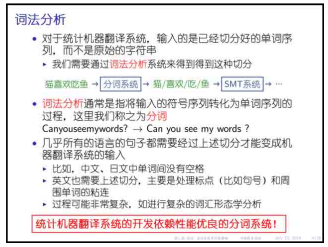
句法分析是自然语言处理的难题之一，这里只介绍了皮毛

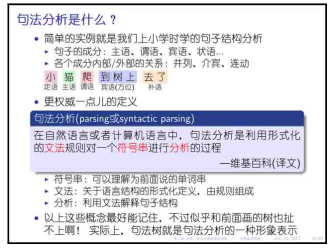
- **文法形式**：除了上下文无关文法，还有不下20种其它的文法
  - TAG(树邻接文法)、TSG(树替换文法)等 - 不是本课程重点
- **句法分析算法**：如何高效地搜索最优解 - 这里没说
  - 构建句法分析器最重要的部分，与机器翻译解码问题相关
  - 非常重要，在后面统计机器翻译解码中还有相关算法介绍
  - 自底向上、A*等算法
- **统计学习**：如何学习句法分析模型
  - 机器学习中的基础问题，句法分析可以看做是一种应用
  - google一下parsing supervised unsupervised learning
- **句法(树)表现形式**：短语结构树 vs. 依存树
  - 本课程以短语结构树为基础 - 但是这两种结构在统计机器翻译中都有使用
  - 关于依存分析，google一下dependency parsing
- ...

内容已经介绍完了，总结一下

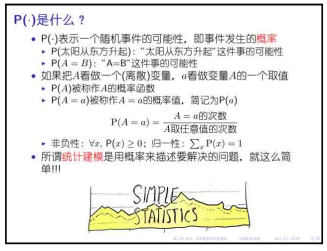
说了很多，记住三方面主要内容就可以了：

1. 任务：啥是分词、啥是句法分析？
   见第5页和第22页

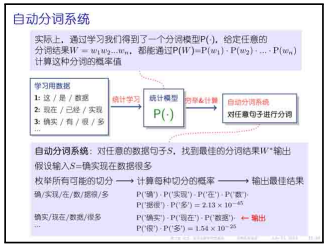


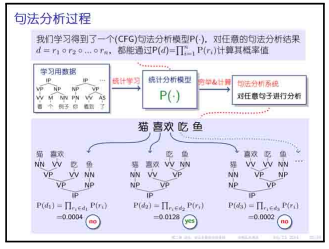


2. 手段：啥是P(·)?
   见第9页



3. 方法：如何用P(·)解决分词和句法分析问题？
   见第16页和第33页

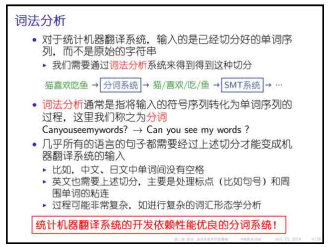


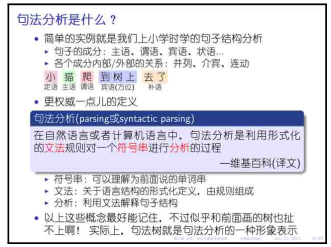


内容已经介绍完了，总结一下

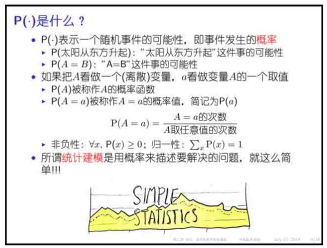
说了很多，记住三方面主要内容就可以了：

1. 任务：啥是分词、啥是句法分析？
   见第5页和第22页

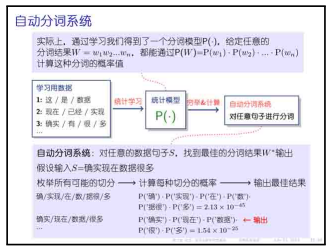


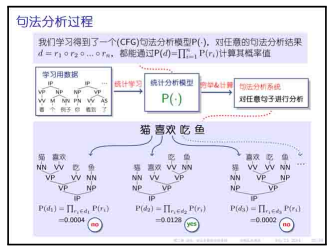


2. 手段：啥是P(·)?
   见第9页



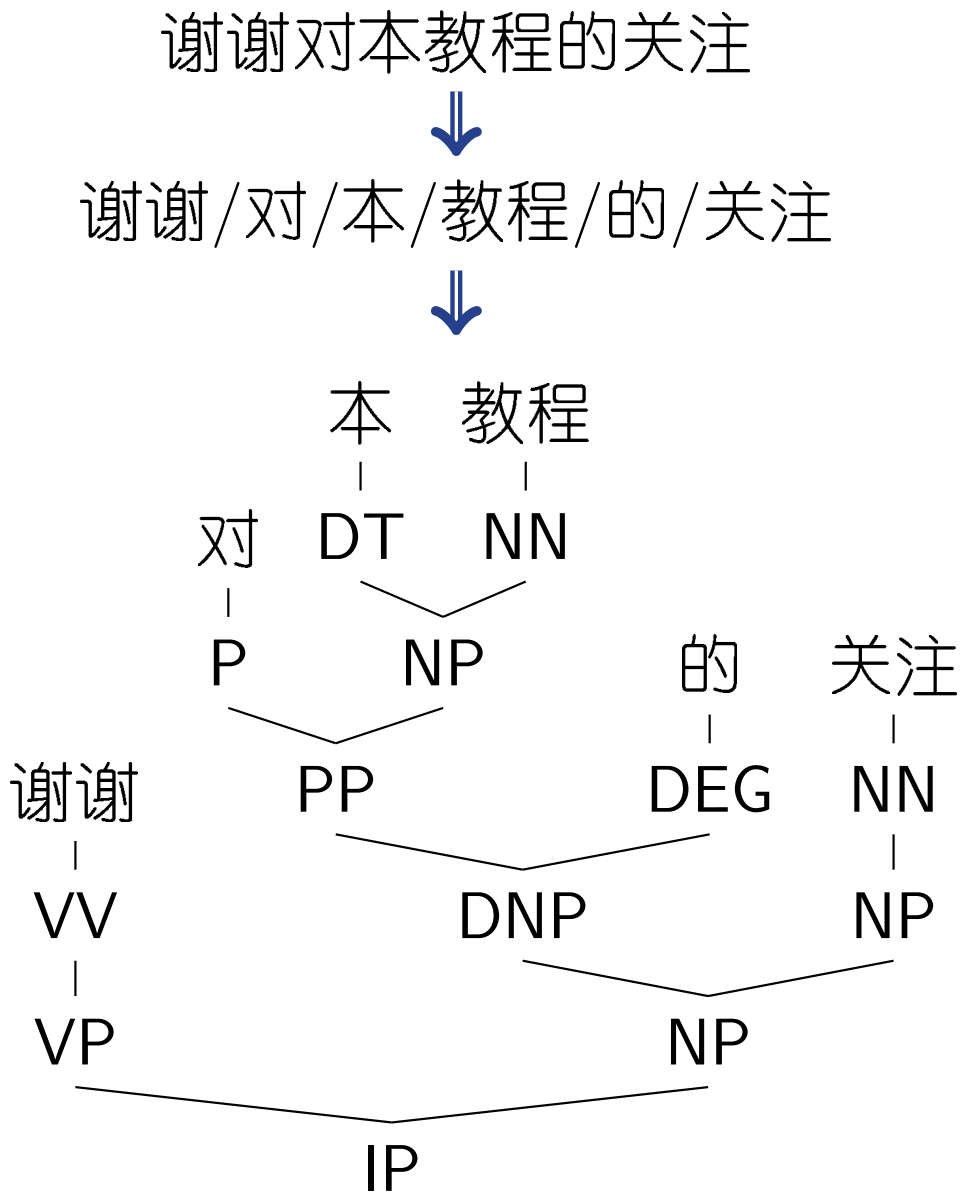
3. 方法：如何用P(·)解决分词和句法分析问题？
   见第16页和第33页





还是那句话：理解思想最最重要，实现方法可以后面再消化

第二章内容也结束喽！


谢谢对本教程的关注
谢谢/对/本/教程/的/关注
本
教程
对
DT
NN
P
NP
的
关注
谢谢
PP
DEG
NN
VV
DNP
NP
VP
NP
IP