



数据科学与计算机学院  
School of Data and Computer Science

# 自然语言处理

## *Natural Language Processing*

权小军 教授

中山大学数据科学与计算机学院

[quanxj3@mail.sysu.edu.cn](mailto:quanxj3@mail.sysu.edu.cn)

## 四、中文词法分析

# 从字符串到词串

汉语的自然书面文本词与词之间无空格分开，因此，在汉语书面语的处理中（词频统计、结构分析、语义理解等），首先碰到的就是词的切分问题

# 从字符串到词串（续）

中文：中山大学在广州。

English：Sun Yat-sen University is in Guangzhou.



# 中文词法分析的意义

文本分词是各个层次的自然语言处理任务的基础

1. 文语转换Text-to-speech
2. 文本校对 Chinese Text Correction
3. 文本检索 Information Retrieval
4. 词频统计、句法分析、机器翻译、……

# 什么是“词”

□ 语法学定义：能够独立运用的最小的音义结合体

# 分词规范

S 如果是一个W，则：

- S 内部子串粘合度高
- S 外部环境替换度高
- S 本身频度高

# 文本分词中的歧义

- 交集型歧义

例1： 张店区大学生不看重大城市的户口本

张店区	大学生	不	看   重大	城市	的	户口本
张店区	大学生	不	看重   大	城市	的	户口本



# 文本分词中的歧义

- 组合型歧义

例2： 你认为学生会听老师的吗

# 文本分词中的歧义

- 组合型歧义

例2： 你认为学生会听老师的吗

你 认 为	学 生 会	听 老 师 的 吗
你 认 为	学 生   会	听 老 师 的 吗

# 文本分词中的歧义

## □ 真歧义

- 确实能在真实语料中发现多种切分形式
- 比如“应用于”、“地面积”、“解除了”

## □ 伪歧义

- 虽然有多种切分可能性，但在真实语料中往往取其中一种切分形式
- 比如“挨批评”、“市政府”、“太平淡”

# 中文文本分词的基本方法概述

## 中文分词方法

基于词典的分词方法

最大匹配法  
最短路径法  
半词罚分法  
最大概率法

“分”词

基于字序列标注的方法

最大熵模型  
CRF模型  
.....

“合”词

# 分词方法

## 1. 基于词典的分词方法

### a. 最大匹配法

# 最大匹配法分词示例

输入：S1= “计算语言学课程是两个课时”

输出：S2= " "

词典
...
计算语言学
课程
课时
...

# 最大匹配法分词示例

输入：S1= “计算语言学课程是两个课时”

输出：S2= " "

设定最大词长MaxLen = 5

W= 计算语言学

.....

词典
...
计算语言学
课程
课时
...

# 分词方法

## 1. 基于词典的分词方法

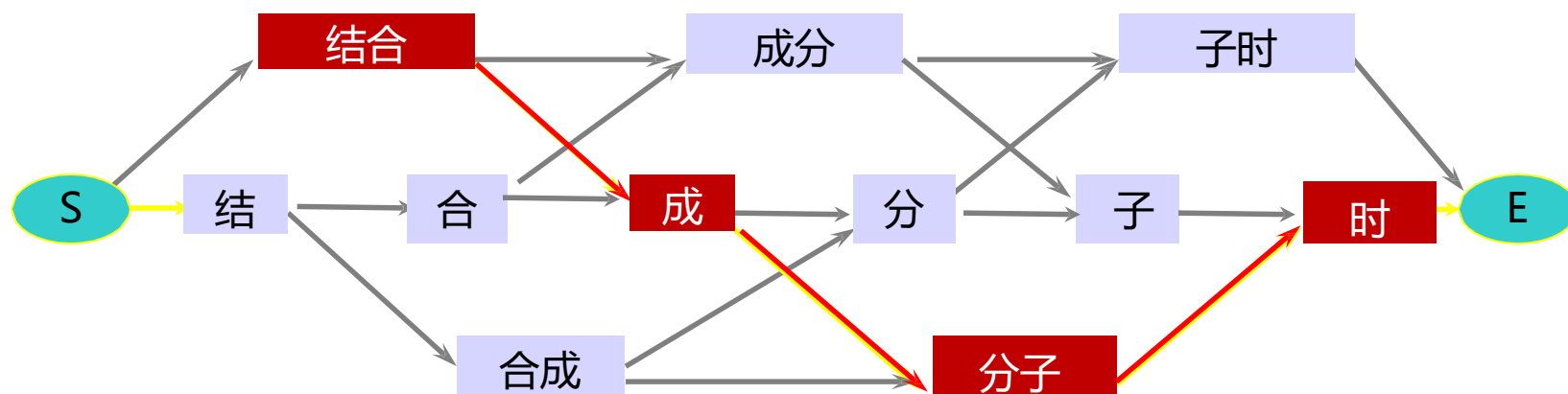
a. 最大匹配法

b. 最优路径法



# 最优路径法

- 看待汉语词语切分问题的新视角：词图上的最优路径求解问题



- 词图给出了一个字符串的全部切分可能性
- 分词任务：寻找一条起点S到终点E的最优路径

# 分词方法

## 1. 基于词典的分词方法

- a. 最大匹配法
  - b. 最优路径法
- 1) 词数最少的路径最优

# 最短路径分词法：词数最少的路径最优

- **基本思想**：在词图上选择一条词数最少的路径

# 分词方法

## 1. 基于词典的分词方法

- a. 最大匹配法
  - b. 最优路径法
- {
- 1) 词数最少的路径最优
  - 2) 半词法

# 半词法分词：词数最少且半词最少

大多数单字在语境里如果能组成合适的词就不倾向于单独使用！

基本概念	半词	如果一个字不单独作为词使用，就是半词。
	整词	如果一个字更倾向于自己成词而不倾向于和别的字组成词，这类“单字词”就称之为“整词”。这类词就是一般说的单字高频成词语素，比如“人、说、我”等。
基本思路	充分利用半词和整词的差别，尽量选择没有半词落单的分词方案。	

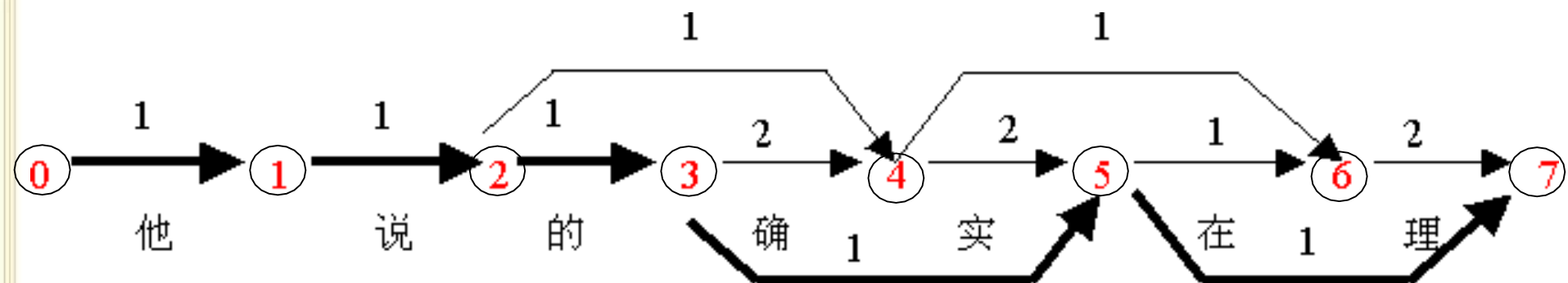
# 半词法分词

■ 在词图的路径优劣评判中引入罚分机制

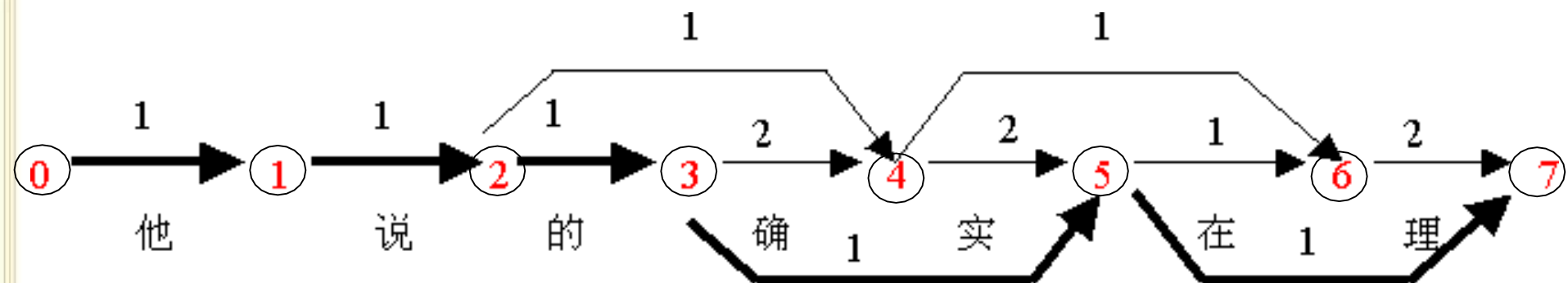
■ 罚分规则：

- 1) 每个词对应的边罚1分。
- 2) 每个半词对应的边加罚1分。
- 3) 一个分词方案的评分为它所对应的路径上所有边的罚分之和。
- 4) 最优路径就是罚分最低的分词路径。

# 半词法分词



# 半词法分词



他 | 说 | 的 | 确实 | 在理 (1+1+1+1+1 = 5分)

他 | 说 | 的确 | 实 | 在理 (1+1+1+2+1 = 6分)

他 | 说 | 的确 | 实在 | 理 (1+1+1+1+2 = 6分)



# 分词方法

## 1. 基于词典的分词方法

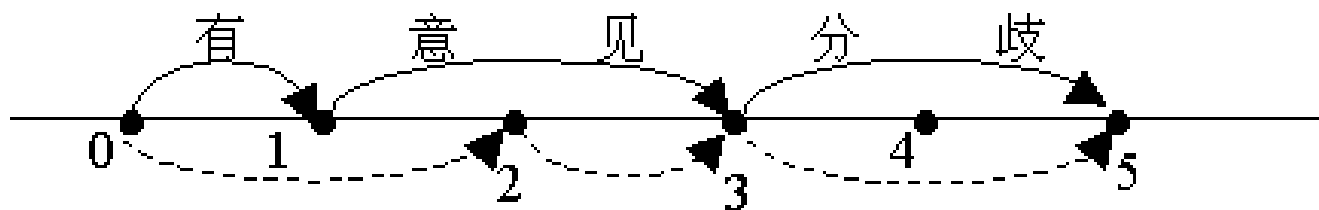
a. 最大匹配法

b. 最优路径法

- 1) 词数最少的路径最优
- 2) 半词法
- 3) 最大概率法分词

# 最大概率法分词：字串成词概率最大的路径最优

**基本思想**：在词图上选择词串概率最大的分词路径作为最优结果



# 最大概率法分词

输入字符串S: 有意见分歧

词串W1: 有/意见/分歧/

词串W2: 有意/见/分歧/

输出: ???

$\text{Max}(P(W1|S), P(W2|S))?$

$$P(W | S) = \frac{P(S | W) \times P(W)}{P(S)} \approx P(W)$$

$$P(W) = P(w_1, w_2, \dots, w_i) \approx P(w_1) \times P(w_2) \times \dots \times P(w_i)$$

$$P(w_i) = \frac{w_i \text{在语料库中的出现次数} n}{\text{语料库中的总词数} N} = \frac{\text{Freq}(w_i)}{N}$$

# 用动态规划算法求解最优路径

- 动态规划算法：最优路径中的第  $i$  个词  $W_i$  的累积概率等于它的左邻词  $W_{i-1}$  的累积概率乘以  $W_i$  自身的概率。

$$P'(w_i) = P'(w_{i-1}) \times P(w_i)$$

- 为方便计算，一般把概率转化为路径代价

$$C = -\log(P)$$

$$C'(w_i) = C'(w_{i-1}) + C(w_i)$$

公式1



最小累积代价    最佳左邻词

# 最大概率法算法流程

- 1) 对一个待分词的字串  $S$ ，按照从左到右的顺序取出全部候选词  $w_1, w_2, \dots, w_i, \dots, w_n$ ；
- 2) 到词典中查出每个候选词的概率值  $P(w_i)$ ，转换为代价  $C(w_i)$ ，并记录每个候选词的全部左邻词；
- 3) 按照公式1计算每个候选词的累计代价，同时比较得到每个候选词的最佳左邻词；
- 4) 如果当前词  $w_n$  是字串  $S$  的尾词，且累计代价  $C'(w_n)$  最大，则  $w_n$  就是  $S$  的终点词；
- 5) 从  $w_n$  开始，按照从右到左顺序，依次将每个词的最佳左邻词输出，即为  $S$  的分词结果。

# 分词方法

## 1. 基于词典的分词方法

- a. 最大匹配法
- b. 最优路径法
  - 1) 词数最少的路径最优
  - 2) 半词法
  - 3) 最大概率法分词

## 2. 字位标注法

# 字位标注法

- 分词可以看做是对字加“词位标记”的过程
- “人”的词位分类示例：

B	E	M	S
词首	词尾	词中	独立词
人 <sub>B</sub> 们	古 <sub>E</sub> 人	小 <sub>M</sub> 人 <sub>M</sub> 国	听 <sub>M</sub> 人 <sub>S</sub> 说

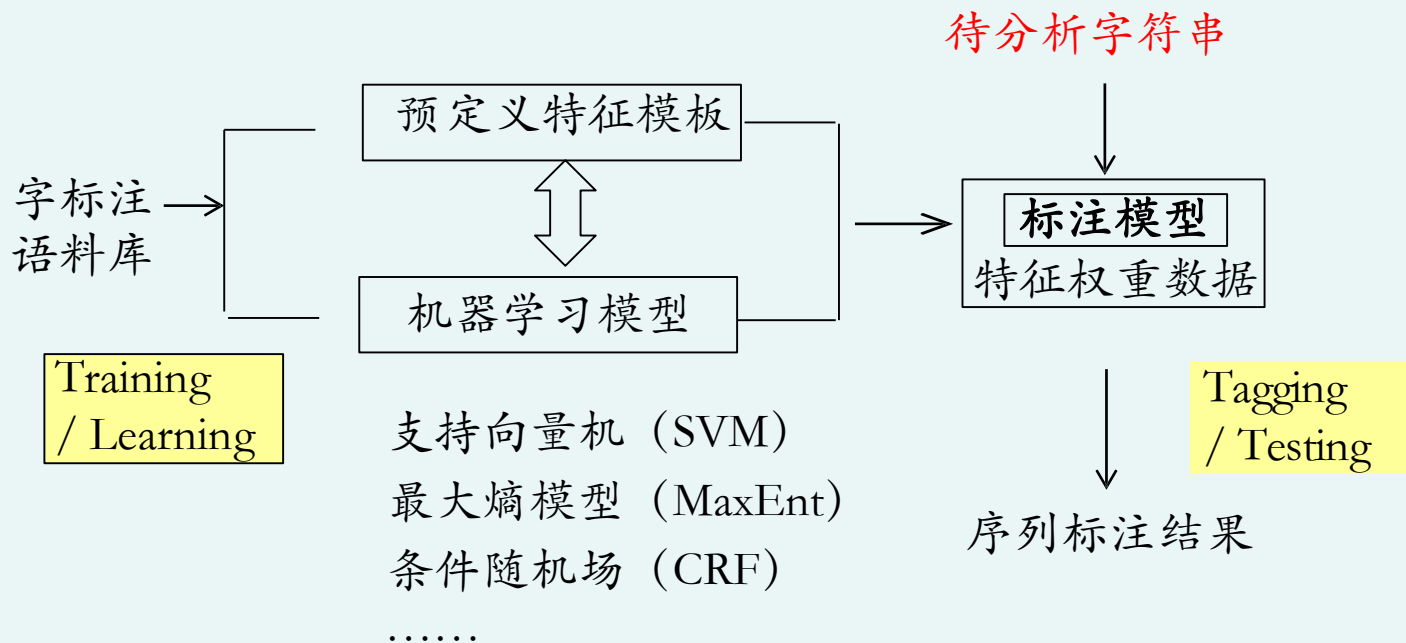
# 基于字序列标注的方法

- **字位标注的原理**：根据字本身及其上下文的特征，来决定当前字的词位标注

特征模板示例	含义
$C_0$	当前字
$C_{-2}, C_{-1}, C_1, C_2$	当前字的左边第二字，第一字，右边第一字，第二字
$C_{-1}C_0, C_0C_1$	当前字跟其左边一个字，当前字跟其右一个字
$C_{-2}C_{-1}, C_1C_2$	当前字的左边两个字，当前字的右边两个字
$C_{-1}C_1$	当前字的左边一个字加右边一个字
$T_{-1}$	左边第一个字的字位标注
$T_{-2}$	左边第二个字的字位标注
Default feature	缺省特征（当上述特征都不适用时）



# 基于字序列标注的方法



# 准确率、召回率、F-Score

## □ 准确率(precision)

$$\text{准确率 (P)} = \frac{\text{切分结果中正确分词数}}{\text{切分结果中所有分词数}} * 100\%$$

## □ 召回率(recall)

$$\text{召回率 (R)} = \frac{\text{切分结果中正确分词数}}{\text{标准答案中所有分词数}} * 100\%$$

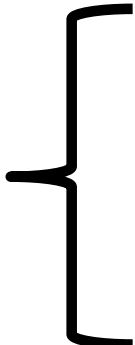
# 准确率、召回率、F-Score

□ F-评价(F-measure 综合准确率和召回率的评价指标)

$$F1 = \frac{2 * P * R}{P + R}$$

# 五、语言模型

# 语言模型

- 
1. 统计语言模型
  2. 神经网络语言模型

# 基本概念

如何计算一段文字(句子)的概率?

阳春三月春意盎然，少先队员脸上荡漾着喜悦的笑容，鲜艳的红领巾在他们的胸前迎风飘扬。

- 以一段文字(句子)为单位统计相对频率?
- 根据句子构成单位的概率计算联合概率?

$$p(w_1) \times p(w_2) \times \cdots \times p(w_n)$$

太简单

# 基本概念

语句  $s = w_1 w_2 \dots w_m$  的先验概率：

$$\begin{aligned} p(s) &= p(w_1) \times p(w_2/w_1) \times p(w_3/w_1w_2) \times \dots \\ &\quad \times p(w_m/w_1\dots w_{m-1}) \\ &= \prod_{i=1}^m p(w_i | w_1 \dots w_{i-1}) \end{aligned}$$

当  $i=1$  时,  $p(w_1|w_0) = p(w_1)$ 。

**语言模型！！**

# 基本概念

## □ 问题解决方法

设法减少历史基元的个数，将  $w_1 w_2 \dots w_{i-1}$  映射到等价类  $S(w_1 w_2 \dots w_{i-1})$ ，使等价类的数目远远小于原来不同历史基元的数目。则有：

$$p(w_i | w_1, \dots, w_{i-1}) = p(w_i | S(w_1, \dots, w_{i-1}))$$



# 基本概念

□ 这种情况下的语言模型称为  $n$  元文法( $n$ -gram)模型

□ 通常地,

- 当  $n=1$  时, 即出现在第  $i$  位上的基元  $w_i$  独立于历史。  
一元文法也被写为 uni-gram 或 monogram;
- 当  $n=2$  时, 2-gram (bi-gram) 被称为1阶马尔可夫链;
- 当  $n=3$  时, 3-gram(tri-gram)被称为2阶马尔可夫链,  
依次类推。

# 基本概念

为了保证条件概率在  $i=1$  时有意义，同时为了保证句子内所有字符串的概率和为 1，即  $\sum_s p(s)=1$ ，可以在句子首尾两端增加两个标志：**<BOS>**  $w_1 w_2 \dots w_m$  **<EOS>**。不失一般性，对于  $n>2$  的  $n$ -gram,  $p(s)$  可以分解为：

$$p(s) = \prod_{i=1}^{m+1} p(w_i | w_{i-n+1}^{i-1})$$

其中， $w_i^j$  表示词序列  $w_i \dots w_j$ ， $w_{i-n+1}^{i-1}$  从  $w_0$  开始， $w_0$  为 **<BOS>**， $w_{m+1}$  为 **<EOS>**。

# 基本概念

□ 举例：

给定句子：John read a book

增加标记：<BOS> John read a book <EOS>

# 基本概念

## □ 举例：

给定句子：John read a book

增加标记：<BOS> John read a book <EOS>

**Unigram:** <BOS>, John, read, a, book, <EOS>

# 基本概念

## □ 举例：

给定句子：John read a book

增加标记：<BOS> John read a book <EOS>

Unigram: <BOS>, John, read, a, book, <EOS>

Bigram: (<BOS>John), (John read), (read a), (a book), (book <EOS>)

# 基本概念

## □ 举例：

给定句子：John read a book

增加标记：<BOS> John read a book <EOS>

Unigram: <BOS>, John, read, a, book, <EOS>

Bigram: (<BOS>John), (John read), (read a), (a book), (book <EOS>)

Trigram: (<BOS>John read), (John read a), (read a book), (a book <EOS>)

# 基本概念

<BOS> John read a book <EOS>

基于2元文法的概率为：

# 基本概念

<BOS> John read a book <EOS>

基于2元文法的概率为：

$$\begin{aligned} p(\text{John read a book}) = & p(\text{John}|\text{<BOS>}) \times \\ & p(\text{read}|\text{John}) \times p(\text{a}|\text{read}) \times \\ & p(\text{book}|\text{a}) \times p(\text{<EOS>}|\text{book}) \end{aligned}$$



# 基本概念

如果汉字的总数为： $N$

- 一元语法：1) 样本空间为  $N$
- 2元语法：1) 样本空间为  $N^2$   
2) 效果比一元语法明显提高
- 估计对汉字而言四元语法效果会好一些
- 智能狂拼、微软拼音输入法基于  $n$ -gram.

# 1. 基本概念

## 2. 参数估计

# 参数估计

## □ 两个重要概念:

- 训练语料(training data)

用于建立模型确定模型参数的已知语料

- 最大似然估计(MLE)

用相对频率计算概率的方法

# 参数估计

*<BOS>John read Moby Dick<EOS>*

*<BOS>Mary read a different book<EOS>*

*<BOS>She read a book by Cher<EOS>*

$$p(\text{John} | \text{<BOS>}) = \frac{c(\text{<BOS> John})}{\sum_w c(\text{<BOS> } w)} = \frac{1}{3}$$

$$p(a | \text{read}) = \frac{c(\text{read } a)}{\sum_w c(\text{read } w)} = \frac{2}{3}$$

$$p(\text{read} | \text{John}) = \frac{c(\text{John read})}{\sum_w c(\text{John } w)} = \frac{1}{1}$$

$$p(\text{book} | a) = \frac{c(a \text{ book})}{\sum_w c(a \text{ } w)} = \frac{1}{2}$$

$$p(\text{<EOS>} | \text{book}) = \frac{c(\text{book <EOS>})}{\sum_w c(\text{book } w)} = \frac{1}{2}$$

$$p(\text{John read a book}) = \frac{1}{3} \times 1 \times \frac{2}{3} \times \frac{1}{2} \times \frac{1}{2} \approx 0.06$$

# 参数估计

## 问题：

数据匮乏(稀疏) (*Sparse Data*) 引起零概率问题，如何解决？

# 参数估计

## 问题：

数据匮乏(稀疏) (*Sparse Data*) 引起零概率问题，如何解决？

数据平滑(data smoothing)

# 数据平滑

## □ 数据平滑的基本思想：

调整最大似然估计的概率值,使零概率增值,使非零概率下调,消除零概率,改进模型的整体正确率

## □ 基本目标：

测试样本的语言模型 困惑度(Perplexity)越小越好

## □ 基本约束： $$\sum_{w_i} p(w_i | w_1, w_2, \dots, w_{i-1}) = 1$$

# 数据平滑

## ➤ 回顾—困惑度的定义：

对于一个平滑的  $n$ -gram，其概率为  $p(w_i | w_{i-n+1}^{i-1})$ ，

可以计算句子的概率：
$$p(s) = \prod_{i=1}^{m+1} p(w_i | w_{i-n+1}^{i-1})$$

假定测试语料  $T$  由  $L$  个句子构成  $(t_1, \dots, t_L)$ ，则整个测试集的概率为：

$$p(T) = \prod_{i=1}^L p(t_i)$$



# 数据平滑

## □ 数据平滑方法

### 1) 加1法(Additive smoothing)

基本思想: 每一种情况出现的次数加1。

例如, 对于 *uni-gram*, 设  $w_1, w_2, w_3$  三个词, 概率分别为:  $1/3, 0, 2/3$ , 加1后情况?

# 数据平滑

对于2-gram 有：

$$\begin{aligned} p(w_i | w_{i-1}) &= \frac{1 + c(w_{i-1}w_i)}{\sum_{w_i} [1 + c(w_{i-1}w_i)]} \\ &= \frac{1 + c(w_{i-1}w_i)}{|V| + \sum_{w_i} c(w_{i-1}w_i)} \end{aligned}$$

其中， $V$ 为被考虑语料的词汇量(全部可能的基元数)。

## 2) 减值法/折扣法(Discounting)

基本思想：修改训练样本中事件的实际计数，使样本中(实际出现的)不同事件的概率之和小于1，剩余的 $\epsilon$ 概率量分配给未见概率。

# 数据平滑

## a) Good-Turing 估计

- I. J. Good 引用 Turing 的方法来估计概率分布
- 假设  $N$  是原来训练样本数据的大小,  $n_r$  是在样本中正好出现  $r$  次的事件的数目(此处事件为  $n$ -gram), 即出现 1 次的  $n$ -gram 有  $n_1$  个, 出现 2 次的  $n$ -gram 有  $n_2$  个, ....., 出现  $r$  次的有  $n_r$  个。

# 数据平滑

那么, 
$$N = \sum_{r=1}^{\infty} n_r r$$

由于, 
$$N = \sum_{r=0}^{\infty} n_r r^* = \sum_{r=0}^{\infty} (r+1) n_{r+1} \quad \text{所以, } r^* = (r+1) \frac{n_{r+1}}{n_r}$$

那么, Good-Turing 估计在样本中出现  $r$  次的事件的概率为:

$$p_r = \frac{n_{r^*}}{N}$$

# 数据平滑

这样，原训练样本中所有事件的概率之和为：

$$\sum_{r>0} n_r \times p_r = 1 - \frac{n_0}{N} < 1$$

因此，有  $\frac{n_0}{N}$  的剩余的概率量就可以均分给所有的未见事件 ( $r = 0$ )。

## b) 绝对减值法 (Absolute discounting)

- Hermann Ney 和 U. Essen 1993年提出。
- 基本思想：从每个计数 $r$ 中减去同样的量，剩余的概  
率量由未见事件均分。
- 设 $R$ 为所有可能事件的数目(当事件为 $n$ -gram 时，如  
果统计基元为词，且词汇集的大小为 $L$ ，则 $R=L^n$ )。

# 数据平滑

那么，样本出现了  $r$  次的事件的概率可以由如下公式估计：

$$p_r = \begin{cases} \frac{r-b}{N} & \text{当 } r > 0 \\ \frac{b(R-n_0)}{Nn_0} & \text{当 } r = 0 \end{cases}$$

其中， $n_0$  为样本中未出现的事件的数目。 $b$  为减去的常量， $b \leq 1$ 。 $b(R - n_0)/N$ 是由于减值而产生的剩余概率量。

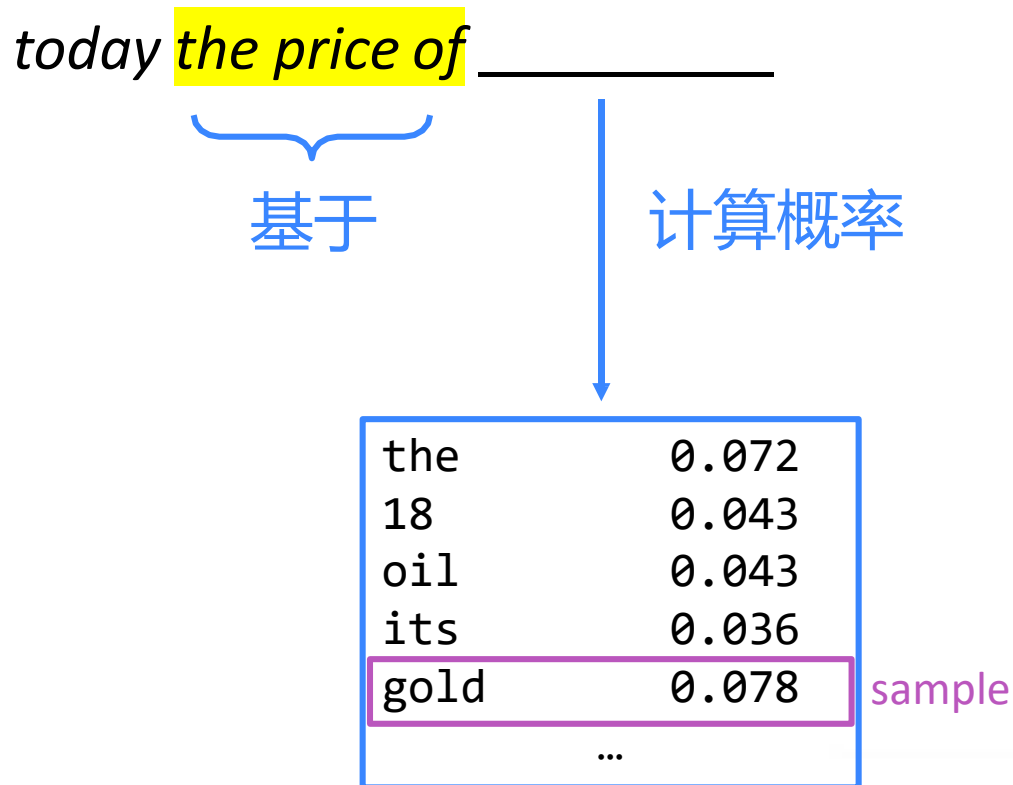


# 语言模型

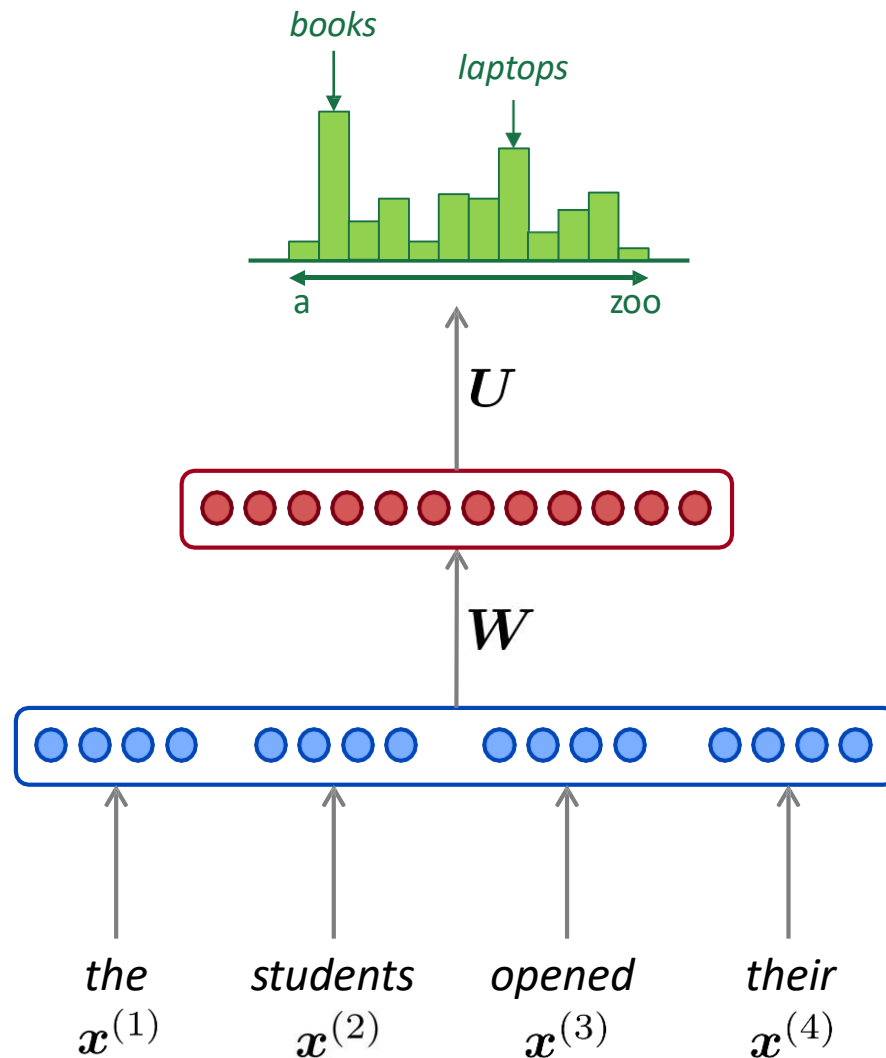
- 1. 统计语言模型
- 2. 神经语言模型

# 基于语言模型的文本生成

- You can also use a Language Model to generate words.



# A fixed-window neural Language Model



**Level 4:** output distribution

$$\hat{y} = \text{softmax}(Uh + b_2) \in \mathbb{R}^{|V|}$$

**Level 3:** hidden layer

$$h = f(We + b_1)$$

**Level 2:** concatenated word embeddings

$$e = [e^{(1)}; e^{(2)}; e^{(3)}; e^{(4)}]$$

**Level 1:** words / one-hot vectors

$$x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}$$

# A fixed-window neural Language Model

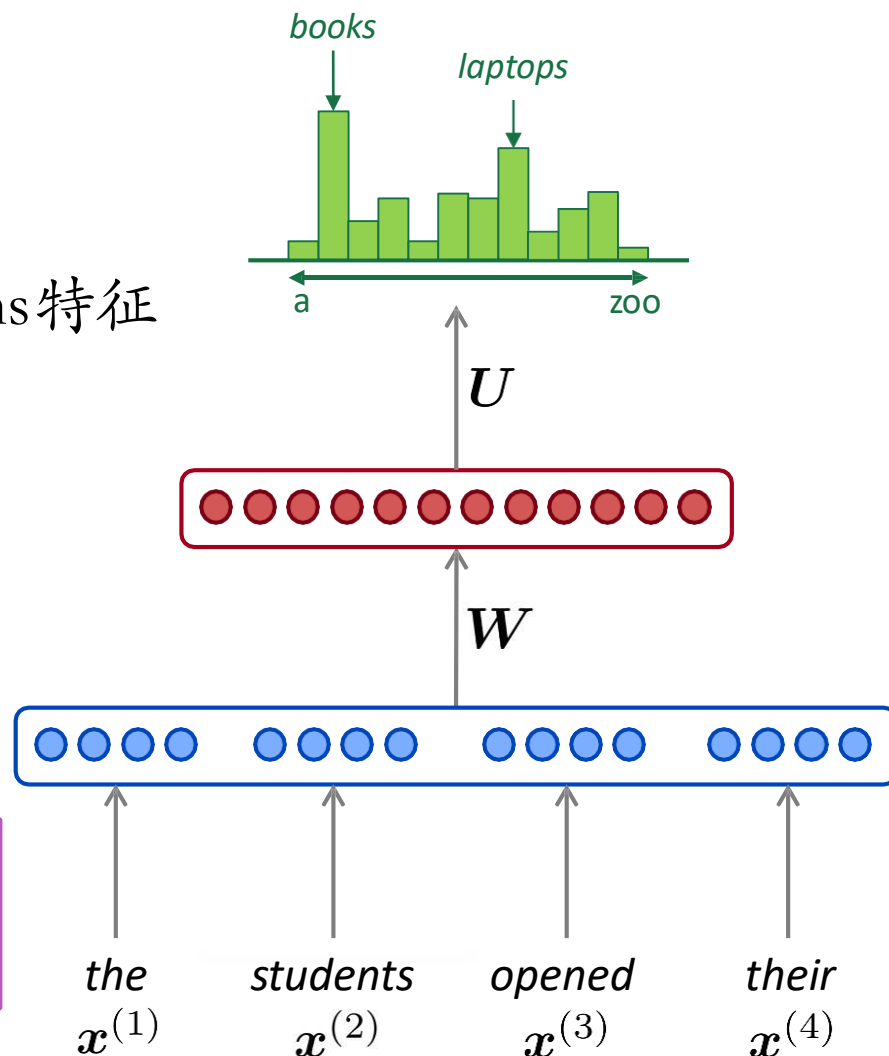
和 $n$ -gram语言模型相比:

- 不存在稀疏性问题
- 不用存储所有已知的 $n$ -grams特征

存在的问题:

- 窗口太小
- 窗口太大,  $W$ 也会变大

需要一种能处理任意  
长度输入的架构

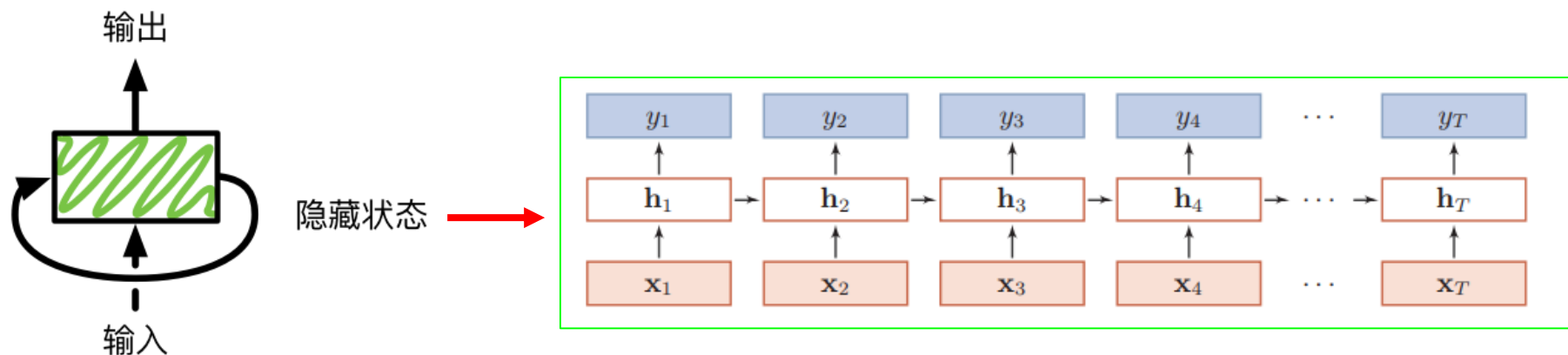


# 循环神经网络

# Recurrent Neural Networks

# 循环神经网络

- 循环神经网络主要用于处理（变长）序列数据

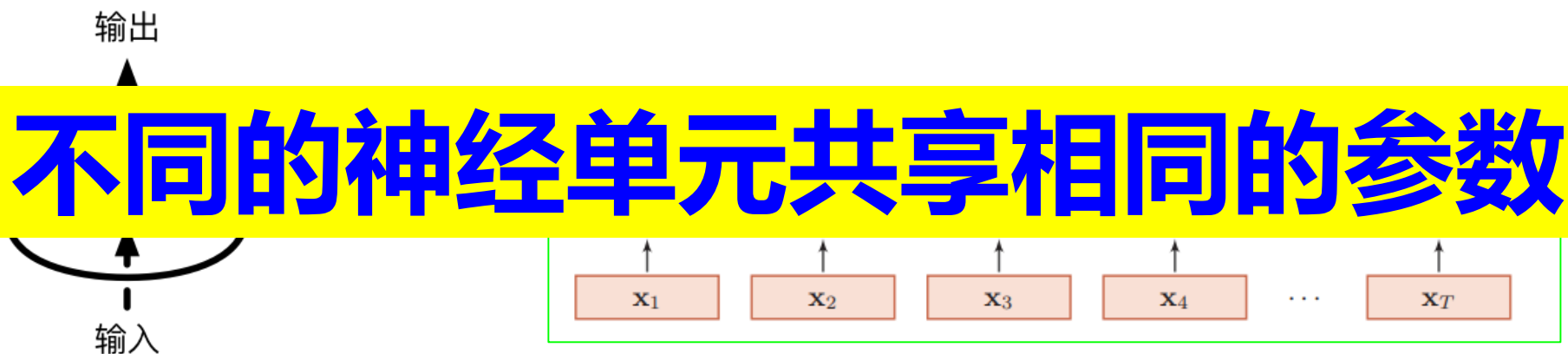


$$\mathbf{h}_t = f(U\mathbf{h}_{t-1} + W\mathbf{x}_t + \mathbf{b}),$$

$$\mathbf{y}_t = V\mathbf{h}_t,$$

# 循环神经网络

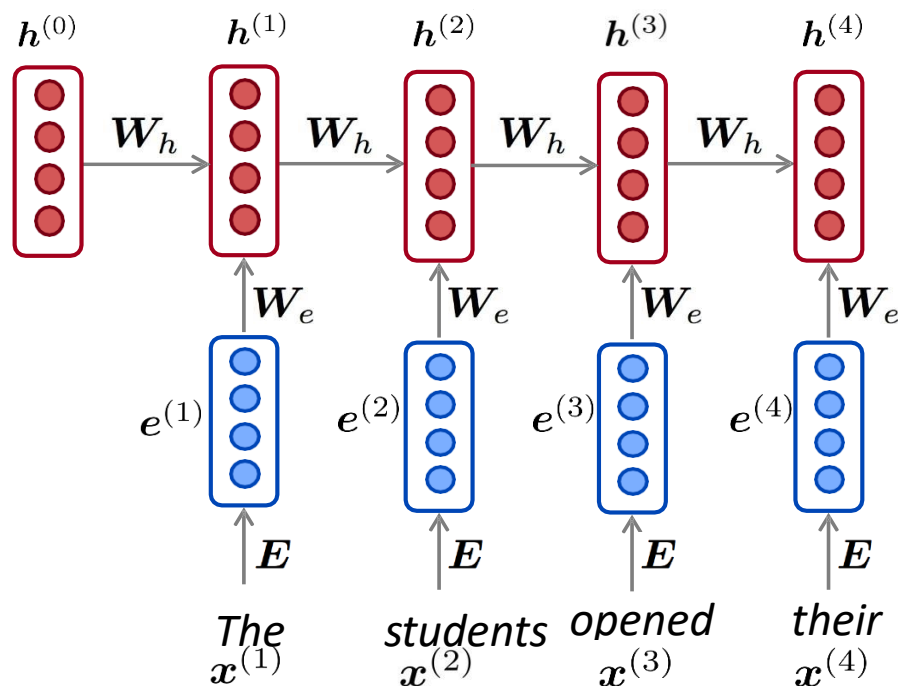
- 循环神经网络主要用于处理（变长）序列数据



$$\mathbf{h}_t = f(U\mathbf{h}_{t-1} + W\mathbf{x}_t + \mathbf{b}),$$

$$\mathbf{y}_t = V\mathbf{h}_t,$$

# RNN语言模型



## Level 4: output distribution

$$\hat{y}^{(t)} = \text{softmax} \left( U h^{(t)} + b_2 \right) \in \mathbb{R}^{|V|}$$

## Level 3: hidden states

$$h^{(t)} = \sigma \left( W_h h^{(t-1)} + W_e e^{(t)} + b_1 \right)$$

$h^{(0)}$  is the initial hidden state

## Level 2: word embeddings

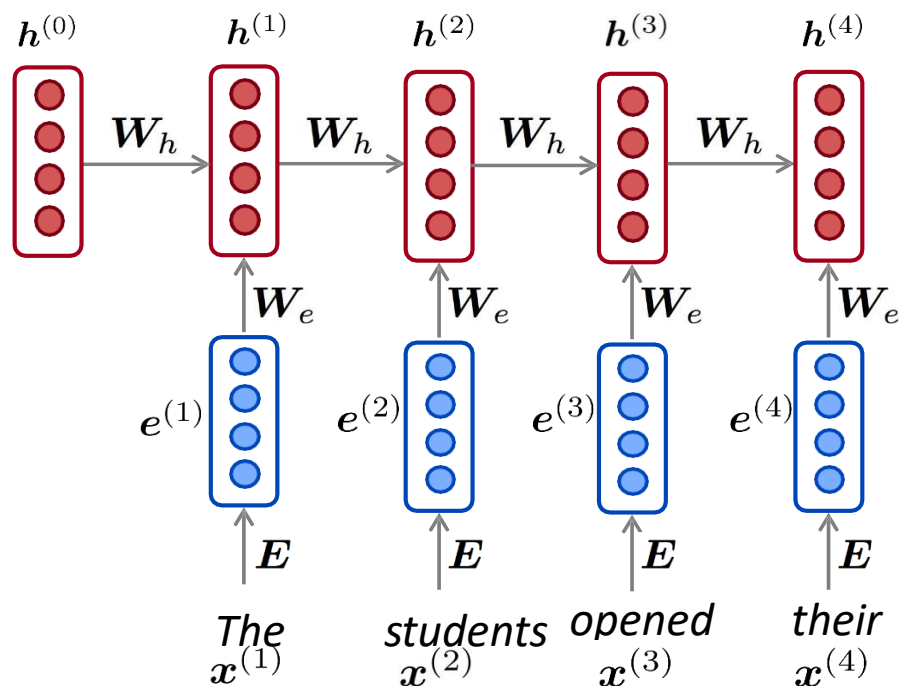
$$e^{(t)} = E x^{(t)}$$

## Level 1: words / one-hot vectors

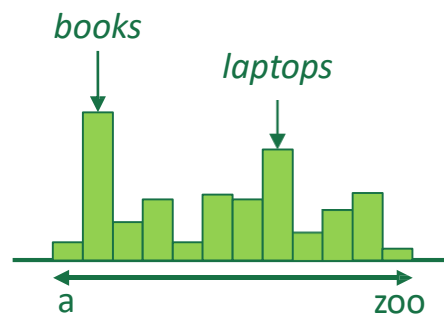
$$x^{(t)} \in \mathbb{R}^{|V|}$$



# RNN语言模型



$$\hat{y}^{(4)} = P(x^{(5)} | \text{the students opened their})$$



# 例子

句子1:

A person that is intelligent has the ability to think, understand, and learn things quickly and well.

# 例子

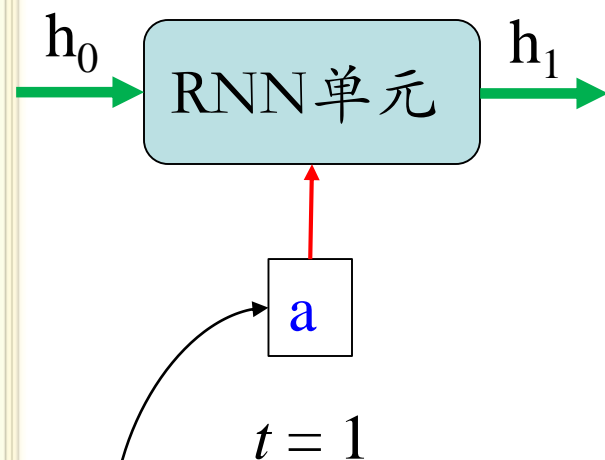
RNN语言模型过程:

RNN单元

A person that is intelligent has the ability to think, understand, and learn things quickly and well.

# 例子

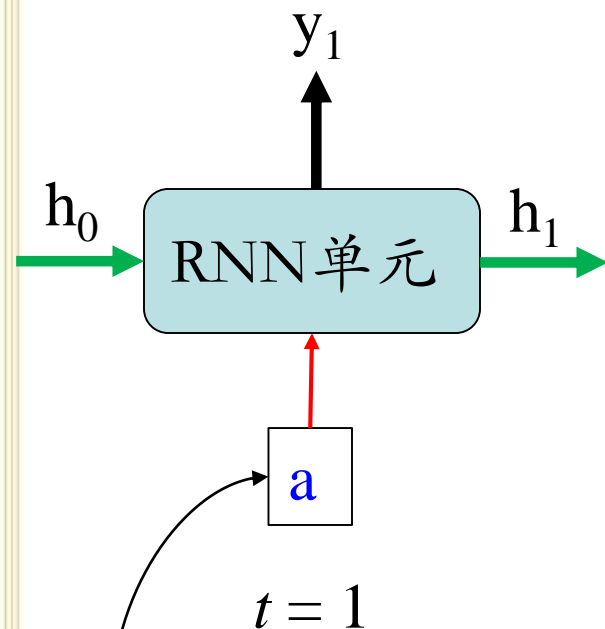
RNN语言模型过程:



A person that is intelligent has the ability to think, understand, and learn things quickly and well.

# 例子

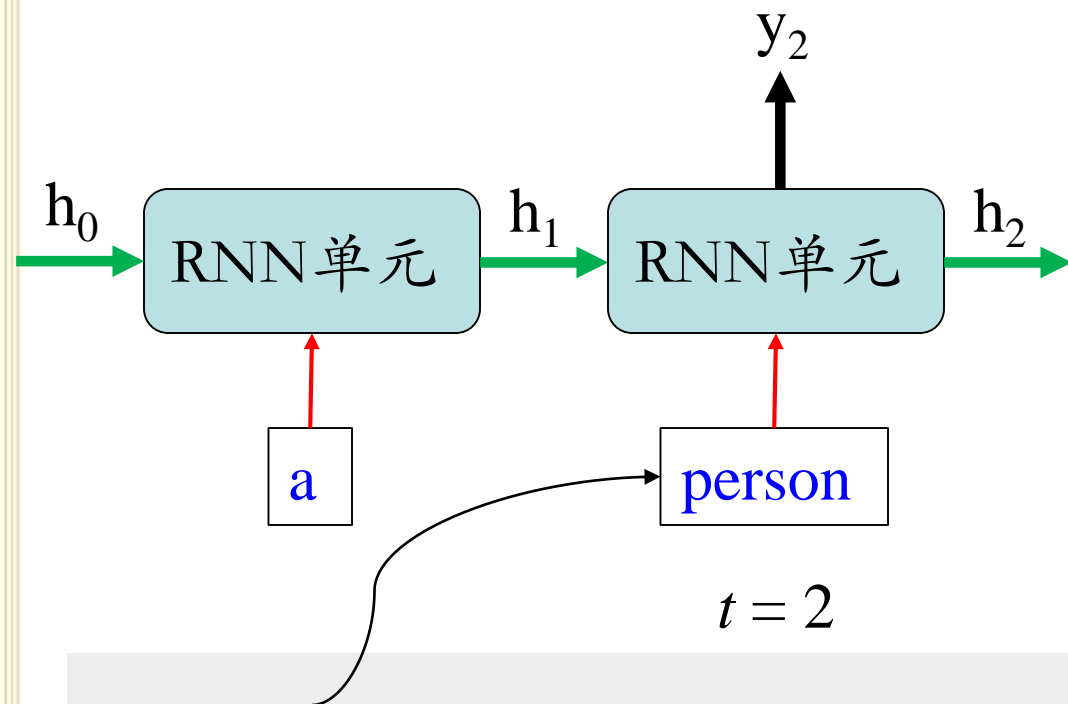
RNN语言模型过程:



A person that is intelligent has the ability to think, understand, and learn things quickly and well.

# 例子

RNN语言模型过程：

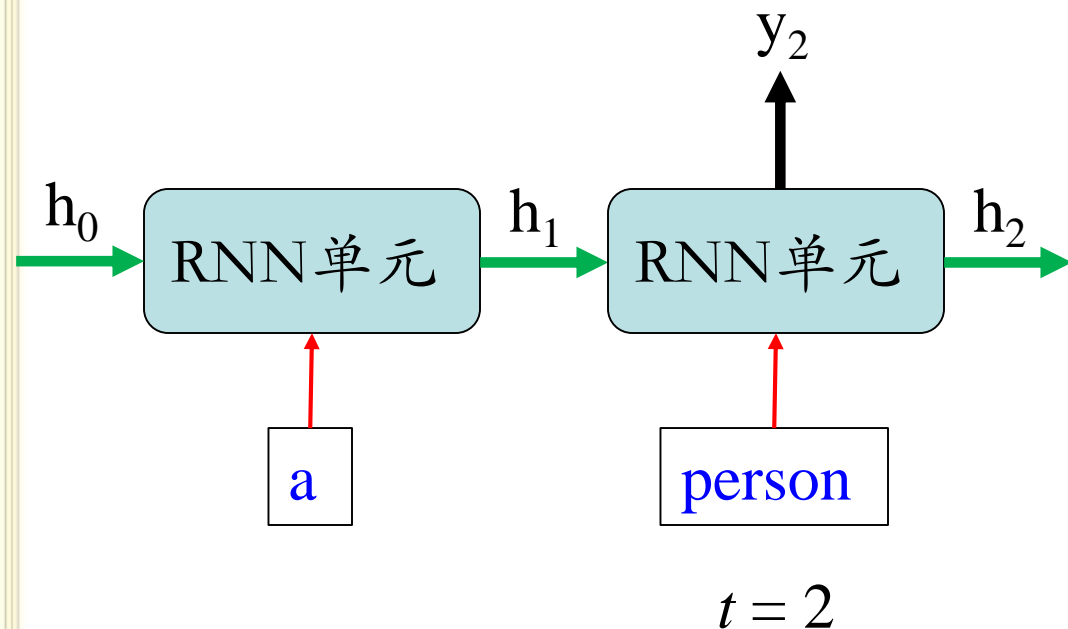


$t = 2$

A **person** that is intelligent has the ability to think, understand, and learn things quickly and well.

# 例子

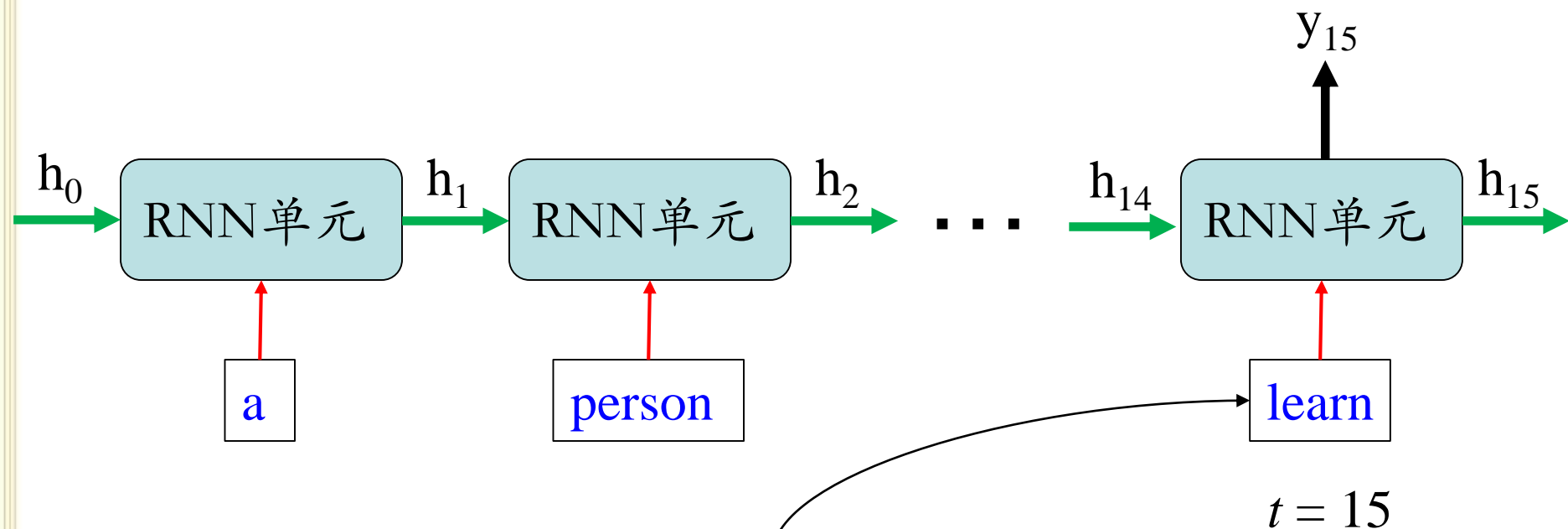
RNN语言模型过程：



A person that is intelligent has the ability to think, understand, and learn things quickly and well.

# 例子

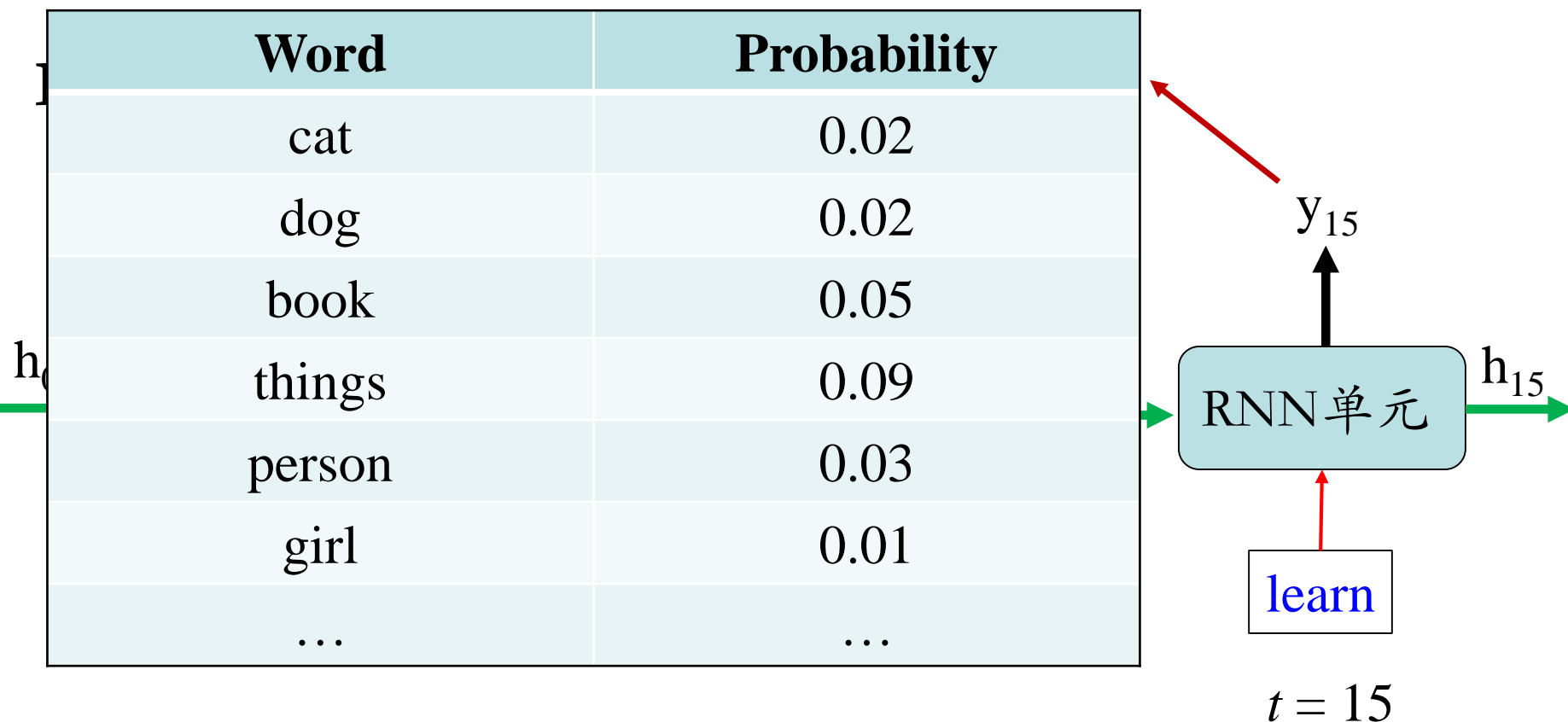
RNN语言模型过程:



A person that is intelligent has the ability to think, understand,  
and **learn**\_\_\_\_\_

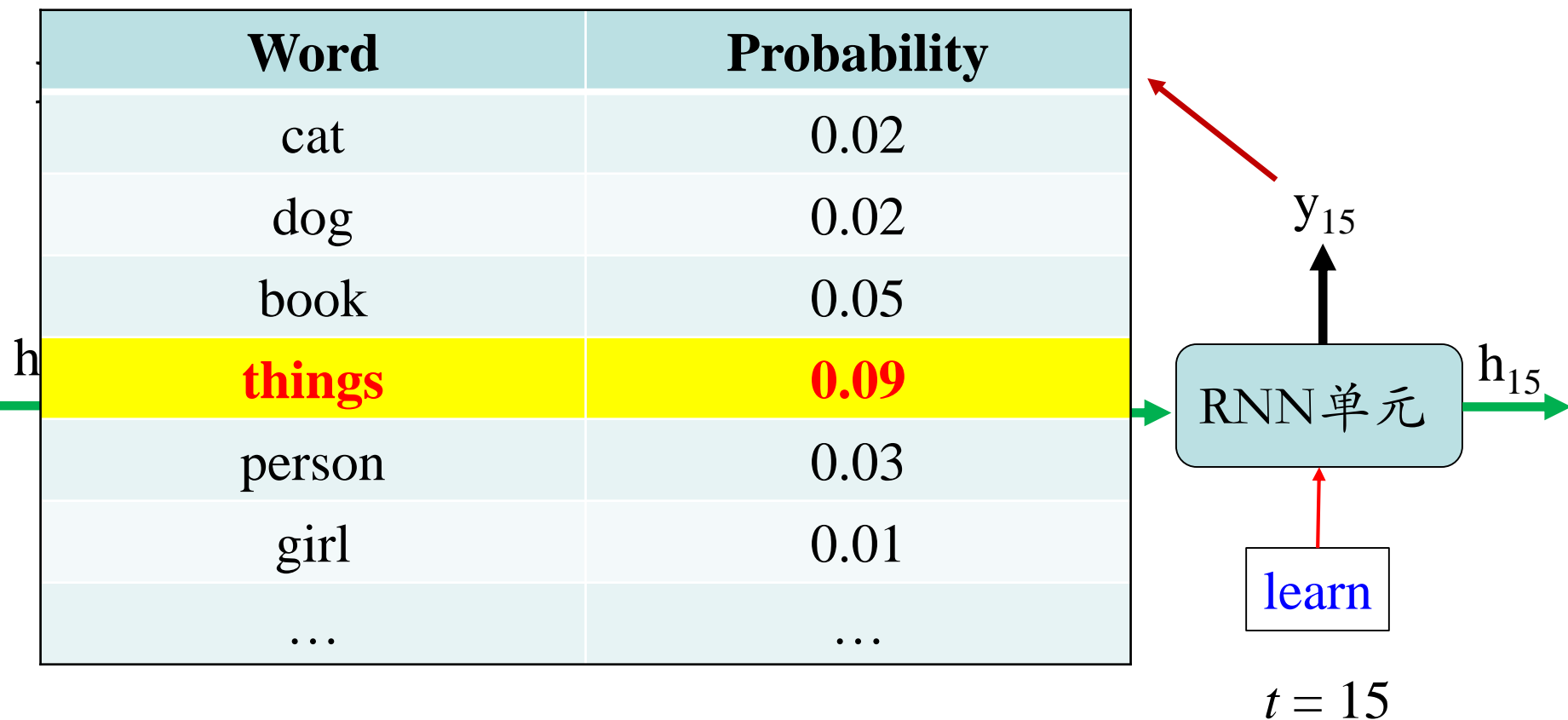


# 例子



A person that is intelligent has the ability to think, understand,  
and **learn**\_\_\_\_\_

# 例子



A person that is intelligent has the ability to think, understand,  
and **learn**\_\_\_\_\_

# RNN Language Model

## RNN 的优点:

- 能够处理任意长度的输入;
- $t$ 时刻可以访问之前任意时刻的信息;
- 对于较长的输入, 模型的大小不变;
- 权重共享

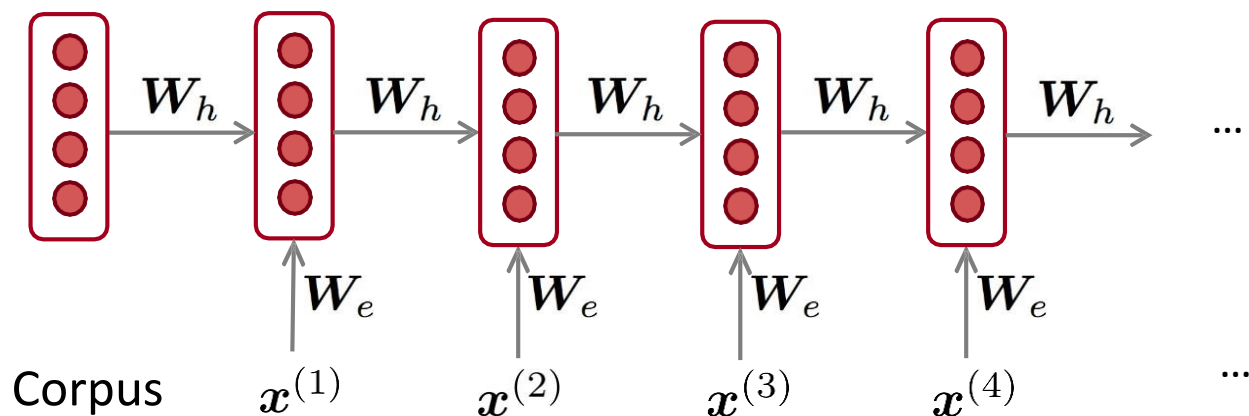
# RNN Language Model

## RNN 的不足:

- 循环计算过程较慢;
- 实际运用中, 很难访问距当前时刻较远的信息;

# RNN语言模型的训练

- Step 1: 输入文本数据集，每个文本表示为： $x^{(1)}, \dots, x^{(T)}$ ；

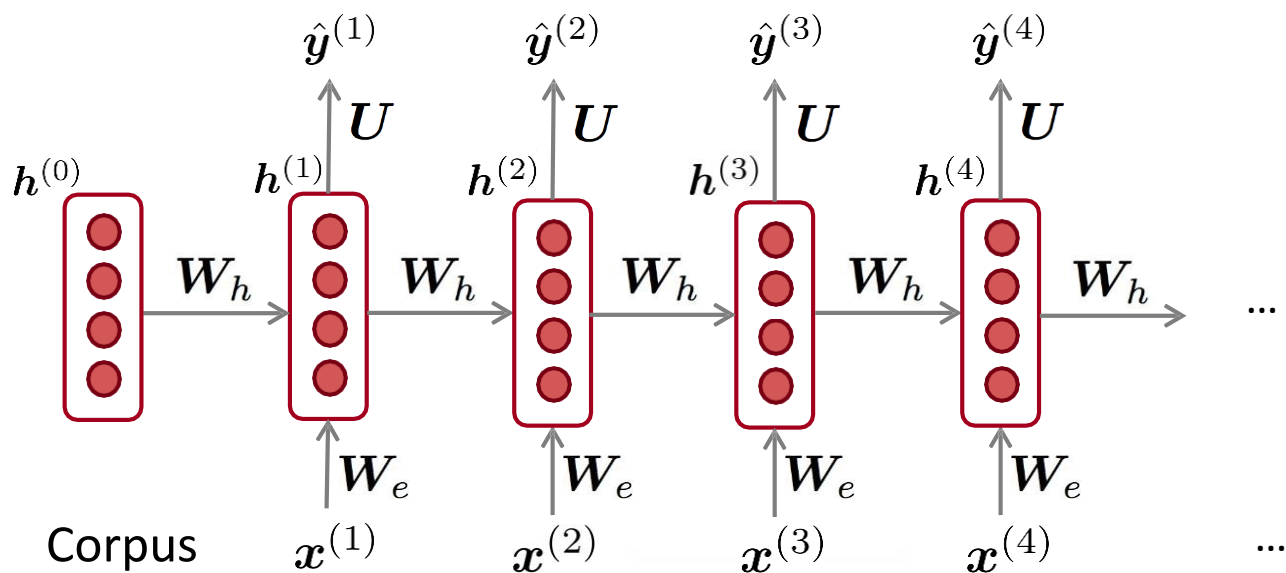


# RNN语言模型的训练

- Step 1: 输入文本数据集，每个文本表示为： $x^{(1)}, \dots, x^{(T)}$ ；
- Step 2: 逐词把每条文本输入给RNN语言模型，计算每一时刻的输出概率  $\hat{y}^{(t)}$ ；

# RNN语言模型的训练

- Step 1: 输入文本数据集，每个文本表示为： $x^{(1)}, \dots, x^{(T)}$ ；
- Step 2: 逐词把每条文本输入给RNN语言模型，计算每一时刻的输出概率  $\hat{y}^{(t)}$ ；



# RNN语言模型的训练

- Step 1: 输入文本数据集，每个文本表示为： $x^{(1)}, \dots, x^{(T)}$ ；
- Step 2: 逐词把每条文本输入给RNN语言模型，计算每一时刻的输出概率  $\hat{y}^{(t)}$ ；
- Step 3: 用交叉熵(**cross-entropy**)计算每一时刻的损失(loss)



# RNN语言模型的训练

- Step 1: 输入文本数据集，每个文本表示为： $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)}$ ；
- Step 2: 逐词把每条文本输入给RNN语言模型，计算每一时刻的输出概率  $\hat{\mathbf{y}}^{(t)}$ ；
- Step 3: 用交叉熵(**cross-entropy**)计算每一时刻的损失(loss)
  - 方法：使用时刻  $t$  的预测分布  $\hat{\mathbf{y}}^{(t)}$  和下一个真实词  $\mathbf{x}^{(t+1)}$  的表示  $\mathbf{y}^{(t)}$ ：

$$J^{(t)}(\theta) = CE(\mathbf{y}^{(t)}, \hat{\mathbf{y}}^{(t)}) = - \sum_{w \in V} \mathbf{y}_w^{(t)} \log \hat{\mathbf{y}}_w^{(t)} = - \log \hat{\mathbf{y}}_{\mathbf{x}_{t+1}}^{(t)}$$

# RNN语言模型的训练

- Step 1: 输入文本数据集，每个文本表示为： $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)}$ ；
- Step 2: 逐词把每条文本输入给RNN语言模型，计算每一时刻的输出概率  $\hat{\mathbf{y}}^{(t)}$ ；
- Step 3: 用交叉熵(**cross-entropy**)计算每一时刻的损失(loss)
  - 方法：使用时刻  $t$  的预测分布  $\hat{\mathbf{y}}^{(t)}$  和下一个真实词  $\mathbf{x}^{(t+1)}$  的表示  $\mathbf{y}^{(t)}$ ：

$$J^{(t)}(\theta) = CE(\mathbf{y}^{(t)}, \hat{\mathbf{y}}^{(t)}) = - \sum_{w \in V} \mathbf{y}_w^{(t)} \log \hat{\mathbf{y}}_w^{(t)} = - \log \hat{\mathbf{y}}_{\mathbf{x}_{t+1}}^{(t)}$$

- Step 4: 计算所有文本的**平均损失**：

$$J(\theta) = \frac{1}{T} \sum_{t=1}^T J^{(t)}(\theta) = \frac{1}{T} \sum_{t=1}^T - \log \hat{\mathbf{y}}_{\mathbf{x}_{t+1}}^{(t)}$$

# RNN语言模型的训练

- 通常，使用随机梯度下降(Stochastic Gradient Descent) 计算一小部分随机数据(mini-batch)的损失；
- 根据得到的损失，计算梯度并更新参数；

# 注意

- 语言模型: 预测下一个词的系统
- RNN: 一种神经网络:
  - 输入是任意长度的文本;
  - 共享权重;
  - 每一步都可以产生输出;
- Recurrent Neural Network  $\neq$  Language Model
- RNN可以用来构建语言模型, 但RNN还有更多用途;

# 注意!

- 常见的RNN模型
  - LSTM
  - GRU
  - Bidirectional
  - Multi-layer

# Thank you!

权小军 中山大学数据科学与计算机学院