Natural Language Processing I

5. 语言模型

WU Xiaokun 吴晓堃

xkun.wu [at] gmail

2022/03/25

文本生成

所有学生的梦想：会写作文的机器人

Thus Spoke Nietzsche 《尼采如是说》

and hand a more good of the same and the former strong that the dreames when the facularіable in a profound therefore, and that is consequence of the strength and whole profound instinct of the german for experience of the former in the fact that it was only be opposing that it is not be not the same and an action and spirit and sense of all the fact to which is as a discipline and the sense of a

– 深度学习21课程练习，训练自《查拉图斯特拉如是说》

所有学生的梦想：会写作文的机器人

Thus Spoke Nietzsche 《尼采如是说》

and hand a more good of the same and the former strong that the dreames when the facularіable in a profound therefore, and that is consequence of the strength and whole profound instinct of the german for experience of the former in the fact that it was only be opposing that it is not be not the same and an action and spirit and sense of all the fact to which is as a discipline and the sense of a

– 深度学习21课程练习，训练自《查拉图斯特拉如是说》

以“我永远不能忘记……”开头完成一篇作文，不少于800字。

预测

文本生成的本质是预测：

他下课后去了□□。操场？南极？天堂？
她把□□递给了老师。作业？香烟？炸弹？

预测

文本生成的本质是预测：

他下课后去了□□。操场？南极？天堂？
她把□□递给了老师。作业？香烟？炸弹？

可能性：**概率 probability**建模

- 语料库中**每个词**出现的可能性：$P(\text{食堂}|\text{他下课后去了})$
- **整个句子**出现的可能性：$P(\text{他下课后去了食堂})$

概率建模，为什么？

语言不是思维的完美映射

- 语音：传递过程信息丢失（歧义、环境噪音）；或网课内容传输
- 文字：拼写、语法错误；文言文中的“通假字”

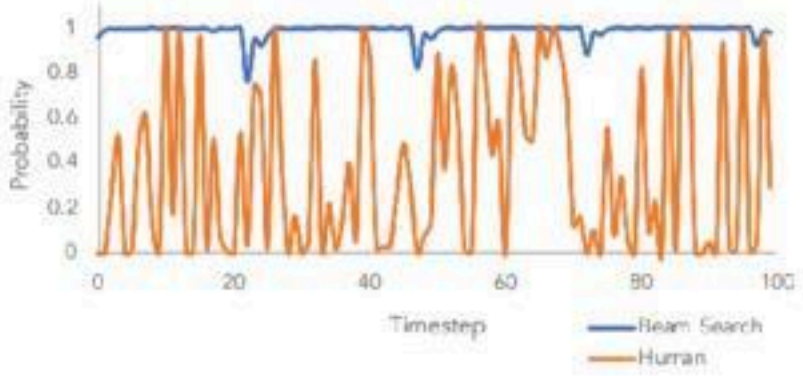
概率建模，为什么？

语言不是思维的完美映射

- 语音：传递过程信息丢失（歧义、环境噪音）；或网课内容传输
- 文字：拼写、语法错误；文言文中的“通假字”

有意思的是，人脑可以**高效地进行推断**。

- 如何验证两人“心有灵犀”？“你说我猜”（但不要随便玩这个游戏）



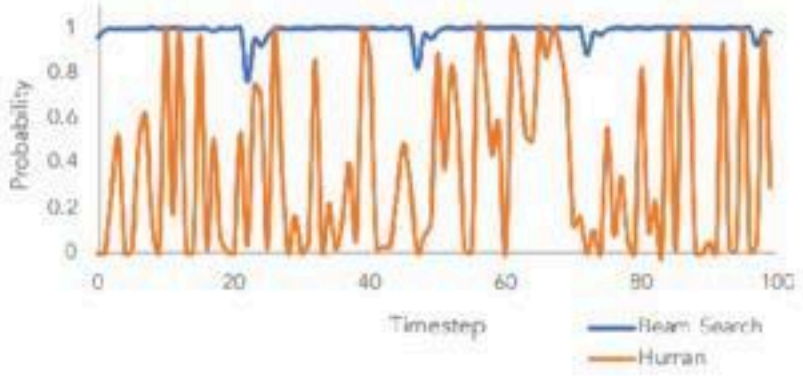
概率建模，为什么？

语言不是思维的完美映射

- 语音：传递过程信息丢失（歧义、环境噪音）；或网课内容传输
- 文字：拼写、语法错误；文言文中的“通假字”

有意思的是，人脑可以**高效地进行推断**。

- 如何验证两人“心有灵犀”？“你说我猜”（但不要随便玩这个游戏）



- 人：**不可能识别对方脑电波**；人工智能：自我感觉良好 ✓

机器翻译

字面直译：生硬，不合语法

他	向	记者	介绍了	主要	内容
He	to	reporters	introduced	main	content

机器翻译

字面直译：生硬，不合语法

他	向	记者	介绍了	主要	内容
He	to	reporters	introduced	main	content

意译是一个不断优化的过程：**趋向最可能**的句子

```
he introduced reporters to the main contents of the statement
he briefed to reporters the main contents of the statement
he briefed reporters on the main contents of the statement
```

语言模型

语言模型 language models (LMs)：对序列（中的词）赋予概率

- $P(W) = P(w_1, w_2, .., w_n)$
- $P(w_n|w_1, w_2, .., w_{n-1})$

有时也称为**语法、文法**

语言模型

语言模型 language models (LMs)：对序列（中的词）赋予概率

- $P(W) = P(w_1, w_2, .., w_n)$
- $P(w_n|w_1, w_2, .., w_{n-1})$

有时也称为**语法、文法**

N元语法 n-gram：连续N个词看作一个单元

- **二元语法 2-gram/bigram**：“your homework”，“我们”

N元语法

简单公式计算

条件概率公式：$P(A|B) = \frac{P(A,B)}{P(B)}$

$$P(\text{食堂}|\text{他下课后去了}) = \frac{\#\text{“他下课后去了食堂”}}{\#\text{“他下课后去了”}}$$

简单公式计算

条件概率公式：$P(A|B) = \frac{P(A,B)}{P(B)}$

$$P(\text{食堂}|\text{他下课后去了}) = \frac{\#\text{“他下课后去了食堂”}}{\#\text{“他下课后去了”}}$$

问题：如何得到统计数字?

简单公式计算

条件概率公式：$P(A|B) = \frac{P(A,B)}{P(B)}$

$$P(\text{食堂}|\text{他下课后去了}) = \frac{\#\text{“他下课后去了食堂”}}{\#\text{“他下课后去了”}}$$

问题：如何得到统计数字?

使用**搜索引擎**（注意引号！）

```
https://www.google.com/search?q="他下课后去了食堂"
```

$$P(\text{食堂}|\text{他下课后去了}) = \frac{\text{2 results}}{\text{About 1,170 results}} = 0.17\%$$

公式计算：准确性

$$P(\text{食堂}|\text{他下课后去了}) = \frac{\text{2 results}}{\text{About 1,170 results}} = 0.17\%$$

问题：10000次下课后总共只去了17次食堂?

公式计算：准确性

$$P(\text{食堂}|\text{他下课后去了}) = \frac{\text{2 results}}{\text{About 1,170 results}} = 0.17\%$$

问题：10000次下课后总共只去了17次食堂?

- 即使是网页搜索**数据量也不够**：语言非常富有创造力
  - 分子：TP 数量不足

公式计算：准确性

$$P(\text{食堂}|\text{他下课后去了}) = \frac{\text{2 results}}{\text{About 1,170 results}} = 0.17\%$$

问题：10000次下课后总共只去了17次食堂?

- 即使是网页搜索**数据量也不够**：语言非常富有创造力
  - 分子：TP 数量不足
- 通用搜索导致大量**无关信息**：主体应该是在校学生
  - 分母：FP 数量太多

公式计算：复杂度

P(他下课后去了食堂)

- #“他下课后去了食堂”：相对容易统计
  - 信息检索：但本身已经是研究问题

公式计算：复杂度

P(他下课后去了食堂)

- # “他下课后去了食堂”：相对容易统计
  - 信息检索：但本身已经是研究问题
- # 所有可能的8字序列
  - 可以认为是**无限多**：无法计算

符号定义

随机变量的概率：$P(X_i = \text{食堂}) = P(\text{食堂})$

- 序列：$w_{1:n} = w_1, w_2, .., w_n$
- **联合概率**：$P(w_1, w_2, .., w_n) = P(X_1 = w_1, X_2 = w_2, .., X_n = w_n)$

符号定义

随机变量的概率：$P(X_i = \text{食堂}) = P(\text{食堂})$

- 序列：$w_{1:n} = w_1, w_2, .., w_n$
- **联合概率**：$P(w_1, w_2, .., w_n) = P(X_1 = w_1, X_2 = w_2, .., X_n = w_n)$

联合概率公式：$P(A, B) = P(A|B)P(B) = P(B|A)P(A)$

- 例如：1班男生的概率，两种计数路径

符号定义

随机变量的概率：$P(X_i = \text{食堂}) = P(\text{食堂})$

- 序列：$w_{1:n} = w_1, w_2, .., w_n$
- **联合概率**：$P(w_1, w_2, .., w_n) = P(X_1 = w_1, X_2 = w_2, .., X_n = w_n)$

联合概率公式：$P(A, B) = P(A|B)P(B) = P(B|A)P(A)$

- 例如：1班男生的概率，两种计数路径
- **条件概率公式**：$P(A|B) = \frac{P(A,B)}{P(B)}, P(B|A) = \frac{P(A,B)}{P(A)}$

符号定义

随机变量的概率：$P(X_i = \text{食堂}) = P(\text{食堂})$

- 序列：$w_{1:n} = w_1, w_2, .., w_n$
- **联合概率**：$P(w_1, w_2, .., w_n) = P(X_1 = w_1, X_2 = w_2, .., X_n = w_n)$

联合概率公式：$P(A, B) = P(A|B)P(B) = P(B|A)P(A)$

- 例如：1班男生的概率，两种计数路径
- **条件概率公式**：$P(A|B) = \frac{P(A,B)}{P(B)}, P(B|A) = \frac{P(A,B)}{P(A)}$
- 多变量：$P(A, B, C, D) = P(A|B, C, D)P(B|C, D)P(C|D)P(D)$

链式法则

两种形式均可：联合概率和条件概率

$$
\begin{aligned}
P(w_1, w_2, .., w_n) &= P(w_1)P(w_2|w_1)P(w_3|w_{1:2})...P(w_n|w_{1:n-1}) \\
&= \prod_{k=1}^{n} P(w_k|w_{1:k-1})
\end{aligned}
$$

链式法则

两种形式均可：联合概率和条件概率

$$
\begin{aligned}
P(w_1, w_2, .., w_n) &= P(w_1)P(w_2|w_1)P(w_3|w_{1:2})...P(w_n|w_{1:n-1}) \\
&= \prod_{k=1}^{n} P(w_k|w_{1:k-1})
\end{aligned}
$$

$$
\begin{aligned}
P(\text{他下课后去了食堂}) &= P(\text{他下课后去了})P(\text{食堂}|\text{他下课后去了}) \\
&= P(\text{他下课后})P(\text{去了}|\text{他下课后})P(\text{食堂}|\text{他下课后去了}) \\
&= P(\text{他})P(\text{下课后}|\text{他})P(\text{去了}|\text{他下课后})P(\text{食堂}|\text{他下课后去了}
\end{aligned}
$$

链式法则

两种形式均可：联合概率和条件概率

$$
\begin{aligned}
P(w_1, w_2, .., w_n) &= P(w_1)P(w_2|w_1)P(w_3|w_{1:2})...P(w_n|w_{1:n-1}) \\
&= \prod_{k=1}^{n} P(w_k|w_{1:k-1})
\end{aligned}
$$

$$
\begin{aligned}
P(\text{他下课后去了食堂}) &= P(\text{他下课后去了})P(\text{食堂}|\text{他下课后去了}) \\
&= P(\text{他下课后})P(\text{去了}|\text{他下课后})P(\text{食堂}|\text{他下课后去了}) \\
&= P(\text{他})P(\text{下课后}|\text{他})P(\text{去了}|\text{他下课后})P(\text{食堂}|\text{他下课后去了}
\end{aligned}
$$

- 但问题仍未解决：两种形式**均无法计算**

N元语法：近似计算

本质是近似计算：

- **用最近的 N 个词替代**观察到的全部字符信息

例如**二元语法 bigram**：$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-1})$

N元语法：近似计算

本质是近似计算：

- **用最近的 N 个词替代**观察到的全部字符信息

例如**二元语法 bigram**：$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-1})$

$$
\begin{aligned}
P(\text{他下课后去了食堂}) &= P(\text{他下课后去了})P(\text{食堂}|\text{他下课后去了}) \\
&\approx P(\text{他下课后去了})P(\text{食堂}|\text{去了}) \\
&\approx P(\text{他})P(\text{下课后}|\text{他})P(\text{去了}|\text{下课后})P(\text{食堂}|\text{去了})
\end{aligned}
$$

N元语法：近似计算

本质是近似计算：

- **用最近的 N 个词替代**观察到的全部字符信息

例如**二元语法 bigram**：$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-1})$

$$
\begin{aligned}
P(\text{他下课后去了食堂}) &= P(\text{他下课后去了})P(\text{食堂}|\text{他下课后去了}) \\
&\approx P(\text{他下课后去了})P(\text{食堂}|\text{去了}) \\
&\approx P(\text{他})P(\text{下课后}|\text{他})P(\text{去了}|\text{下课后})P(\text{食堂}|\text{去了})
\end{aligned}
$$

Markov 假设：当前状态只取决于最近 N 个单元的信息

- **Markov 模型**：预测时不关注过于久远的历史信息

特殊情况

二元语法 bigram：“活在当下”

- 每个词只取决于前面的一个词

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

特殊情况

二元语法 bigram：“活在当下”

- 每个词只取决于前面的一个词

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

一元语法 unigram：随机行走

- 没有约束条件，每个词独立计算概率

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k)$$

一般情况：N元语法

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1})$$

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k|w_{k-N+1})$$

一般情况：N元语法

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1})$$

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k|w_{k-N+1})$$

N元语法简单、易理解；但用于语言建模**通常不足**

- 语言中**长距离依赖**对语义理解很重要

一般情况：N元语法

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1})$$

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k|w_{k-N+1})$$

N元语法简单、易理解；但用于语言建模**通常不足**

- 语言中**长距离依赖**对语义理解很重要

N元语法计算

最大似然估计

最大似然估计 Maximum Likelihood Estimation (MLE)

- 解决的问题：事件的可能性有多大?
  - 给定观测结果：统计出现的**频率**即可

最大似然估计

最大似然估计 Maximum Likelihood Estimation (MLE)

- 解决的问题：事件的可能性有多大?
  - 给定观测结果：统计出现的**频率**即可
- 从语料库统计N元词频，然后正则化为**可能性**

例如二元语法：

$$P(w_n|w_{n-1}) = \frac{\Gamma(w_{n-1}w_n)}{\sum_w \Gamma(w_{n-1}w)} = \frac{\Gamma(w_{n-1}w_n)}{\Gamma(w_{n-1})}$$

思考：$\sum_w \Gamma(w_{n-1}w) = \Gamma(w_{n-1})$

最大似然估计

最大似然估计 Maximum Likelihood Estimation (MLE)

- 解决的问题：事件的可能性有多大?
  - 给定观测结果：统计出现的**频率**即可
- 从语料库统计N元词频，然后正则化为**可能性**

例如二元语法：

$$P(w_n|w_{n-1}) = \frac{\Gamma(w_{n-1}w_n)}{\sum_w \Gamma(w_{n-1}w)} = \frac{\Gamma(w_{n-1}w_n)}{\Gamma(w_{n-1})}$$

思考：$\sum_w \Gamma(w_{n-1}w) = \Gamma(w_{n-1})$

- 间接计算：遍历所有邻接词的可能性

Ex. 二元语法概率

```
<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>
```

$$P(w_n|w_{n-1}) = \frac{\Gamma(w_{n-1}w_n)}{\Gamma(w_{n-1})}$$

$P(I\|<s>) = 2/3$	$P(Sam\|<s>) = 1/3$	$P(am\|I) = 2/3$
$P(</s>\|Sam) = 1/2$	$P(Sam\|am) = 1/2$	$P(do\|I) = 1/3$

相对频率

N元语法一般情况：已观测**特定序列**与**前缀序列**的计数之比

$$P(w_n|w_{n-N+1:n-1}) = \frac{\Gamma(w_{n-N+1:n-1}w_n)}{\Gamma(w_{n-N+1:n-1})}$$

相对频率

N元语法一般情况：已观测**特定序列**与**前缀序列**的计数之比

$$P(w_n|w_{n-N+1:n-1}) = \frac{\Gamma(w_{n-N+1:n-1}w_n)}{\Gamma(w_{n-N+1:n-1})}$$

MLE：模型固定，参数未知

1. 统计已知样本，计算所需*分布*
2. **反推模型参数**：最大化样本*分布*的可能性

例如：计算每个句子中词的*分布*，再加权平均

似然的理论问题

似然是一种**经验（而非客观）规律**的描述，无法识别“偶发事件”

$$P(w_n|w_{n-1}) = \frac{\Gamma(w_{n-1}w_n)}{\Gamma(w_{n-1})}$$

$$P(\text{下雨}|\text{吴老师来上课}) = \frac{\Gamma(\text{下雨}, \text{吴老师来上课})}{\Gamma(\text{吴老师来上课})} = \frac{9}{12} = .75$$

- 难道比拜龙王还管用?

似然的理论问题

似然是一种**经验（而非客观）规律**的描述，无法识别“偶发事件”

$$P(w_n|w_{n-1}) = \frac{\Gamma(w_{n-1}w_n)}{\Gamma(w_{n-1})}$$

$$P(\text{下雨}|\text{吴老师来上课}) = \frac{\Gamma(\text{下雨}, \text{吴老师来上课})}{\Gamma(\text{吴老师来上课})} = \frac{9}{12} = .75$$

- 难道比拜龙王还管用?

似然无法表达事件的**因果关系**，无法区分“必然事件”的顺序

$$P(\text{天亮}|\text{公鸡打鸣}) = P(\text{公鸡打鸣}|\text{天亮}) = 1$$

- 统计学信条：“相关性并不意味着因果关系”
  - 因果推断目前是一个“灰色问题”

实验：句子的概率

一些实际问题

计算都是在对数空间

- 防止**下溢 underflow**

$$P(w_1, w_2, .., w_n) = \prod_{k=1}^{n} P(w_k | w_{1:k-1})$$

$$p_1 \times p_2 \times p_3 \times p_4 = \exp(\log p_1 + \log p_2 + \log p_3 + \log p_4)$$

一些实际问题

计算都是在对数空间

- 防止下溢 underflow

$$P(w_1, w_2, .., w_n) = \prod_{k=1}^{n} P(w_k | w_{1:k-1})$$

$$p_1 \times p_2 \times p_3 \times p_4 = \exp(\log p_1 + \log p_2 + \log p_3 + \log p_4)$$

- 加法计算更快

Google Ngram 视图

1800-2019 年间书籍中出现的词汇及其词频

- 支持通配符 *

Google Ngram Viewer: https://books.google.com/ngrams

搜索示例：

- *
- 毛泽东，马克思，鲁迅，孔子, * 主义
- University of *, * 大学

评测语言模型

评测

显式评测 extrinsic evaluation：嵌入到应用中，评测应用的性能

- 唯一可以实测出任务性能的方法
  - 类比电脑评测：软件跑分

评测

显式评测 extrinsic evaluation：嵌入到应用中，评测应用的性能

- 唯一可以实测出任务性能的方法
  - 类比电脑评测：软件跑分
- 过于昂贵；需要综合比较多个应用

评测

显式评测 extrinsic evaluation：嵌入到应用中，评测应用的性能

- 唯一可以实测出任务性能的方法
  - 类比电脑评测：软件跑分
- 过于昂贵；需要综合比较多个应用

例如语音识别任务：数脚本中正确的词数。

评测

显式评测 extrinsic evaluation：嵌入到应用中，评测应用的性能

- 唯一可以实测出任务性能的方法
  - 类比电脑评测：软件跑分
- 过于昂贵；需要综合比较多个应用

例如语音识别任务：数脚本中正确的词数。

隐式评测 intrinsic evaluation：评测模型与应用无关的内在性能

- 快速、简单：只度量几个内在指标
  - 类比电脑评测：直接比较主频，其他硬件一般是适配的

测试数据集

评测不同模型的性能：在同一数据集上训练；在同一数据集上评测

训练集 training set：用于训练模型

- 模型学到的必然是训练集中数据的分布

测试数据集

评测不同模型的性能：在同一数据集上训练；在同一数据集上评测

训练集 training set：用于训练模型

- 模型学到的必然是训练集中数据的分布

测试集 test set：用于评测性能

- 采样偏倚：数据集需要同分布
  - 把全部数据划分成均质的两部分

测试数据集

评测不同模型的性能：在同一数据集上训练；在同一数据集上评测

训练集 training set：用于训练模型

- 模型学到的必然是训练集中数据的分布

测试集 test set：用于评测性能

- 采样偏倚：数据集需要同分布
  - 把全部数据划分成均质的两部分
- 评测作弊：训练集中不能出现测试集中的数据

测试数据集

评测不同模型的性能：在同一数据集上训练；在同一数据集上评测

训练集 training set：用于训练模型

- 模型学到的必然是训练集中数据的分布

测试集 test set：用于评测性能

- 采样偏倚：数据集需要同分布
  - 把全部数据划分成均质的两部分
- 评测作弊：训练集中不能出现测试集中的数据

思考：以上两点几乎不可能同时满足；**泛化**问题

Shannon 猜词游戏

两人交替猜下一个词

```
我经常点_
我经常点炸鸡腿_
我经常点炸鸡腿和薯条_
```

Shannon 猜词游戏

两人交替猜下一个词

```
我经常点_
我经常点炸鸡腿_
我经常点炸鸡腿和薯条_
```

目标：使对方无法继续接话（如何用一个词把对方噎死？）

```
我经常点炮_
```

Shannon 猜词游戏

两人交替猜下一个词

```
我经常点_
我经常点炸鸡腿_
我经常点炸鸡腿和薯条_
```

目标：使对方无法继续接话（如何用一个词把对方噎死？）

```
我经常点炮_
```

给每个可以填的词一个**概率值**

- 猜字母：`bedr_`
  - 仍是潜在正确的单词
- 成语接龙

Shannon 猜词游戏

两人交替猜下一个词

```
我经常点_
我经常点炸鸡腿_
我经常点炸鸡腿和薯条_
```

目标：使对方无法继续接话（如何用一个词把对方噎死？）

```
我经常点炮_
```

给每个可以填的词一个**概率值**

- 猜字母：`bedr_`
  - 仍是潜在正确的单词
- 成语接龙

好模型：给**正确**、**难续接**的词更高的概率。如何量化？

困惑度

困惑度 perplexity (PP)：模型在测试集上算得概率的倒数，再由词数归一化

$$
\begin{aligned}
PP(W) &= P(w_1 w_2 .. w_N)^{-\frac{1}{N}} \\
&= \sqrt[N]{\frac{1}{P(w_1 w_2 .. w_N)}} = \sqrt[N]{\prod_{k=1}^{N} \frac{1}{P(w_k | w_{1:k-1})}}
\end{aligned}
$$

困惑度

困惑度 perplexity (PP)：模型在测试集上算得概率的倒数，再由词数归一化

$$
\begin{aligned}
PP(W) &= P(w_1 w_2 .. w_N)^{-\frac{1}{N}} \\
&= \sqrt[N]{\frac{1}{P(w_1 w_2 .. w_N)}} = \sqrt[N]{\prod_{k=1}^{N} \frac{1}{P(w_k | w_{1:k-1})}}
\end{aligned}
$$

- 概率的倒数的几何平均
- 序列的概率高：困惑度低

困惑度

困惑度 perplexity (PP)：模型在测试集上算得概率的倒数，再由词数归一化

$$
\begin{aligned}
PP(W) &= P(w_1 w_2 .. w_N)^{-\frac{1}{N}} \\
&= \sqrt[N]{\frac{1}{P(w_1 w_2 .. w_N)}} = \sqrt[N]{\prod_{k=1}^{N} \frac{1}{P(w_k | w_{1:k-1})}}
\end{aligned}
$$

- 概率的倒数的几何平均
- 序列的概率高：困惑度低

对于二元语法：$PP(W) = \sqrt[N]{\prod_{k=1}^{N} \frac{1}{P(w_k | w_{k-1})}}$

困惑度

困惑度 perplexity (PP)：模型在测试集上算得概率的倒数，再由词数归一化

$$PP(W) = P(w_1 w_2 .. w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 .. w_N)}} = \sqrt[N]{\prod_{k=1}^{N} \frac{1}{P(w_k | w_{1:k-1})}}$$

- 概率的倒数的几何平均
- 序列的概率高：困惑度低

对于二元语法：$PP(W) = \sqrt[N]{\prod_{k=1}^{N} \frac{1}{P(w_k | w_{k-1})}}$

目标：最小化困惑度；**最大化测试集的概率。**

Shannon 猜词游戏与困惑度

给每个可以填的词一个概率值

- 不同情况的可选范围差异巨大
  - 猜词目标：限制对方可选的范围

分支因子 branching factor：下一位置可能出现的词数

Shannon 猜词游戏与困惑度

给每个可以填的词一个概率值

- 不同情况的可选范围差异巨大
  - 猜词目标：限制对方可选的范围

分支因子 branching factor：下一位置可能出现的词数

例如：使用等概率一元模型对随机数字串建模

- 每个词出现的概率相同，**无法有规律地预测**：困惑度比较高
- $PP(W) = P(w_1 w_2 .. w_N)^{-\frac{1}{N}} = (\frac{1}{10}^N)^{-\frac{1}{N}} = 10$
  - 与分支因子相同

加权分支因子

反之，如果训练集中的数字不是均匀分布

- 假设**模型的构造**：0 出现的概率是 91%；1-9 分别都是 1%
- 分支因子仍然是 10

加权分支因子

反之，如果训练集中的数字不是均匀分布

- 假设**模型的构造**：0 出现的概率是 91%；1-9 分别都是 1%
- 分支因子仍然是 10

困惑度反映模型对数据的**解释能力**

- $P(0123456789) = (0.91 \times 0.01^9)^{-0.1} = 63.6936$
  - 小概率事件频发，难以预测：如果观测没错，**模型必定出错**

加权分支因子

反之，如果训练集中的数字不是均匀分布

- 假设**模型的构造**：0 出现的概率是 91%；1-9 分别都是 1%
- 分支因子仍然是 10

困惑度反映模型对数据的**解释能力**

- $P(0123456789) = (0.91 \times 0.01^9)^{-0.1} = 63.6936$
  - 小概率事件频发，难以预测：如果观测没错，**模型必定出错**
- $P(0001000000) = (0.91^9 \times 0.01)^{-0.1} = 1.7253$
  - 更容易预测，即概率更高：模型性能很好

加权分支因子

反之，如果训练集中的数字不是均匀分布

- 假设**模型的构造**：0 出现的概率是 91%；1-9 分别都是 1%
- 分支因子仍然是 10

困惑度反映模型对数据的**解释能力**

- $P(0123456789) = (0.91 \times 0.01^9)^{-0.1} = 63.6936$
  - 小概率事件频发，难以预测：如果观测没错，**模型必定出错**
- $P(0001000000) = (0.91^9 \times 0.01)^{-0.1} = 1.7253$
  - 更容易预测，即概率更高：模型性能很好

困惑度可以看作**加权的分支因子**

- 权重：模型从训练集中学得；可以看成模型的定义

评测N元模型

训练集： *Wall Street Journal*

- 38 M 个词, 19,979 个词类

测试集： 1.5 M 个词

	一元	二元	三元
困惑度	962	170	109

评测N元模型

训练集：*Wall Street Journal*

- 38 M 个词, 19,979 个词类

测试集：1.5 M 个词

	一元	二元	三元
困惑度	962	170	109

结论：模型对序列信息**建模越好，困惑度就越低**

评测N元模型

训练集：*Wall Street Journal*

- 38 M 个词, 19,979 个词类

测试集：1.5 M 个词

	一元	二元	三元
困惑度	962	170	109

结论：模型对序列信息**建模越好，困惑度就越低**

注意：困惑度是**隐式**评测度量

- 低困惑度不保证具体任务上性能好
  - 模型改进一定要**在实际任务上显式评测**
- 通常作为快速检查算法的指标

模型采样

Shannon 可视化算法

（从概率分布）采样：根据概率对随机点取值

- 语言模型：代表句子或下一个词的概率分布
- 模型质量**取决于训练集、模型种类**
  - 与作者的写作风格相似
  - N较大时建模更好

Shannon 可视化算法

（从概率分布）采样：根据概率对随机点取值

- 语言模型：代表句子或下一个词的概率分布
- 模型质量**取决于训练集、模型种类**
  - 与作者的写作风格相似
  - N较大时建模更好

[Shannon 1951] 可视化二元语言模型：

1. 写入起始符<s>
2. $v =$ <s>
3. 直到遇见终止符</s>：
   1. 根据概率分布写入随机词 $w = \arg\max_w P(w|v)$
   2. $v = w$

课程项目：N元语言模型作文

可选作课程项目：

1. 训练一个N元语言模型（或其他高级模型），
2. 使用模型以“我永远不能忘记……”开头完成一篇作文，字数控制在800字左右。

课程项目：N元语言模型作文

可选作课程项目：

1. 训练一个N元语言模型（或其他高级模型），
2. 使用模型以“我永远不能忘记……”开头完成一篇作文，字数控制在800字左右。

要求：

- 内容积极、健康，**严禁极端言论**
  - “我做的人工智障胡说八道”：不要试图糊弄专家
    - 你的训练数据集是怎么采集的?

课程项目：N元语言模型作文

可选作课程项目：

1. 训练一个N元语言模型（或其他高级模型），
2. 使用模型以“我永远不能忘记……”开头完成一篇作文，字数控制在800字左右。

要求：

- 内容积极、健康，**严禁极端言论**
  - “我做的人工智障胡说八道”：不要试图糊弄专家
    - 你的训练数据集是怎么采集的?

提示：不同训练文本对应不同的写作风格

- 《红楼梦》版：我永远不能忘记林妹妹刚来贾府的那一天……
- 《资治通鉴》版：可以先把开头翻译成文言文

模仿 Shakespeare I

1-gram

```
— To him swallowed confess hear
both. Which. Of save on trail for
are ay device and rote life have
— Hill he late speaks; or! a more to
leg less first you enter
```

2-gram

```
— Why dost stand forth thy canopy,
forsooth; he is this palpable hit
the King Henry. Live king. Follow.
— What means, sir. I confess she?
then all sorts, he is trim, captain.
```

模仿 Shakespeare II

3-gram

```
— Fly, and will rid me these news of
price. Therefore the sadness of
parting, as they say, 'tis done.
— This shall forbid it should be
branded, if renown made it empty.
```

4-gram

```
— King Henry. What! I will go seek
the traitor Gloucester. Exeunt some
of the watch. A great banquet serv'd
in;
— It cannot be but so.
```

- 感觉很像原文?

Shakespeare 语料库

对于文本生成任务来说，通常语料库**规模不足**

- $N = 884,647, V = 29,066$
- $300,000$ 个二元词汇，很多吗?

Shakespeare 语料库

对于文本生成任务来说，通常语料库**规模不足**

- $N = 884,647, V = 29,066$
- $300,000$ 个二元词汇，很多吗?
- 可能的总数：$V^2 = 8.44$ 亿
  - 99.96% 的二元词汇从来没出现过

Shakespeare 语料库

对于文本生成任务来说，通常语料库**规模不足**

- $N = 884,647, V = 29,066$
- $300,000$ 个二元词汇，很多吗?
- 可能的总数：$V^2 = 8.44$ 亿
  - 99.96% 的二元词汇从来没出现过

- 例如4-gram：**泛化性太差**，生成文本不够丰富多彩
  - `It cannot be but so.`：看起来像? 其实就是原文
  - `It cannot be but _`：语料库中只有5中情况
    - `{that, I, he, thou, so}`

Wall Street Journal 语料库 I

1-gram

```
- Months the my and issue of year foreign new exchange's september
- were recession exchange new endorsed a acquire to six executives
```

2-gram

```
- Last December through the way to preserve the Hudson corporation
N.
- B. E. C. Taylor would seem to complete the major central
planners one
- point five percent of U. S. E. has already old M. X. corporation
of living
- on information such as more frequently fishing to keep her
```

Wall Street Journal 语料库 II

3-gram

```
- They also point to ninety nine point six billion dollars from
two hundred
- four oh six three percent of the rates of interest stores as
Mexico and
- Brazil on market conditions
```

Wall Street Journal 语料库 II

3-gram

```
- They also point to ninety nine point six billion dollars from
two hundred
- four oh six three percent of the rates of interest stores as
Mexico and
- Brazil on market conditions
```

如果文本风格不一样，统计模型是无法正确预测的

- 本质问题：**过拟合**，泛化能力差

泛化：未知词汇

词汇表本身是确定、有限的

- 问题：如何生成词汇表从未出现的词汇？

泛化：未知词汇

词汇表本身是确定、有限的

- 问题：如何生成词汇表从未出现的词汇？

封闭词汇表 closed vocabulary：词汇表固定，不考虑未知词

- 例如：语言识别、机器翻译

开放词汇表 open vocabulary：添加伪码 <UNK>，指代未知词

泛化：未知词汇

词汇表本身是确定、有限的

- 问题：如何生成词汇表从未出现的词汇？

封闭词汇表 closed vocabulary：词汇表固定，不考虑未知词

- 例如：语言识别、机器翻译

开放词汇表 open vocabulary：添加伪码 <UNK>，指代未知词

训练模型时构建词汇表；评测模型时必须使用**相同的词汇表**

- 标准词汇表：将训练集中不在词汇表里的词设为<UNK>
- 自建词汇表：频率较低的词设为<UNK>；超过预设词数的词设为<UNK>

泛化：未知词汇

词汇表本身是确定、有限的

- 问题：如何生成词汇表从未出现的词汇？

封闭词汇表 closed vocabulary：词汇表固定，不考虑未知词

- 例如：语言识别、机器翻译

开放词汇表 open vocabulary：添加伪码 <UNK>，指代未知词

训练模型时构建词汇表；评测模型时必须使用**相同的词汇表**

- 标准词汇表：将训练集中不在词汇表里的词设为<UNK>
- 自建词汇表：频率较低的词设为<UNK>；超过预设词数的词设为<UNK>

[Buck 2014] 给未知词汇较高的概率可以**降低困惑度**

- 可以看成谨慎的预测：把握不大就不判断

泛化：稀疏性

回顾：任何语料库都是**规模相对不足**的

- 词汇序列只占所有可能的一小部分
  - Shakespeare：99.96% 的二元词汇从来没出现过
- 问题：如何生成模型从未见过的词汇序列?

泛化：稀疏性

回顾：任何语料库都是**规模相对不足**的

- 词汇序列只占所有可能的一小部分
  - Shakespeare：99.96% 的二元词汇从来没出现过
- 问题：如何生成模型从未见过的词汇序列?

$P(\text{学生宿舍}|\text{去了}) = 0$，但测试集出现了

- 模型**过拟合**：相对（数据）复杂，死记硬背
- **无法计算困惑度**：测试集的概率为0，并且在分母上

泛化：稀疏性

回顾：任何语料库都是**规模相对不足**的

- 词汇序列只占所有可能的一小部分
  - Shakespeare：99.96% 的二元词汇从来没出现过
- 问题：如何生成模型从未见过的词汇序列?

$P(\text{学生宿舍}|\text{去了}) = 0$，但测试集出现了

- 模型**过拟合**：相对（数据）复杂，死记硬背
- **无法计算困惑度**：测试集的概率为0，并且在分母上

解决策略：1. 增加训练数据；2. 改进模型；3. 人为指定一个非0概率

- 从“烧钱的暴力美学”到“无成本的小技巧”

Review

本章内容

语言模型、N元语法。最大似然估计。困惑度、分支因子。模型采样。平滑处理。

重点：语言模型、N元语法；最大似然估计；困惑度、分支因子；平滑处理。

难点：N元词汇的稀疏性；Shannon可视化算法。

学习目标

- 理解语言模型的公式计算，N元语法的假设。
- 理解N元语法的最大似然估计算法。
- 理解困惑度、分支因子对评测的用途。
- 了解文本生成的Shannon可视化算法。
- 理解N元词汇的稀疏性特点、导致的问题，以及解决方案。
- 理解平滑处理的基本原理，以及几种常用方案。

问题

简述语言模型的两种计算公式与含义。

简述N元语法的假设与条件概率公式。

举例使用最大似然估计MLE计算N元语法。

简述N元词汇的稀疏性特点、导致的问题，以及解决方案。

简述平滑处理的基本原理，以及几种常用方案。

课程项目

可选作课程项目：

1. 训练一个N元语言模型（或其他高级模型），
2. 使用模型以“我永远不能忘记……”开头完成一篇作文，字数控制在800字左右。

要求：

- 内容积极、健康，**严禁极端言论**
  - “我做的人工智障胡说八道”：不要试图糊弄专家
    - 你的训练数据集是怎么采集的?

提示：不同训练文本对应不同的写作风格

- 《红楼梦》版：我永远不能忘记林妹妹刚来贾府的那一天……
- 《资治通鉴》版：可以先把开头翻译成文言文