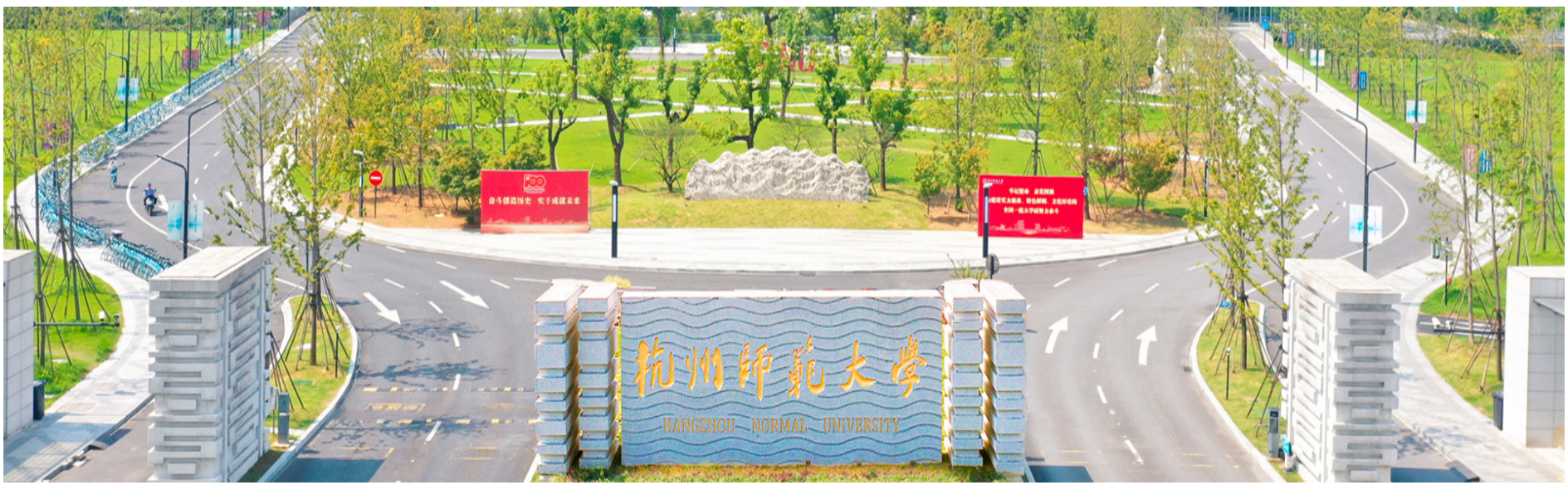
第二讲 - 文本处理基础：N-gram 语言模型

张建章

阿里巴巴商学院
杭州师范大学

2025-02-01



目录

语言模型（Language Model，LM）：是一种机器学习模型，其主要任务是根据已有的上下文来**预测**接下来的单词。该模型为每个可能的下一个**单词**分配一个概率，进而构成一个完整的概率分布；同时，语言模型也能够为整个句子计算联合概率，从而反映出不同词序排列的合理性。

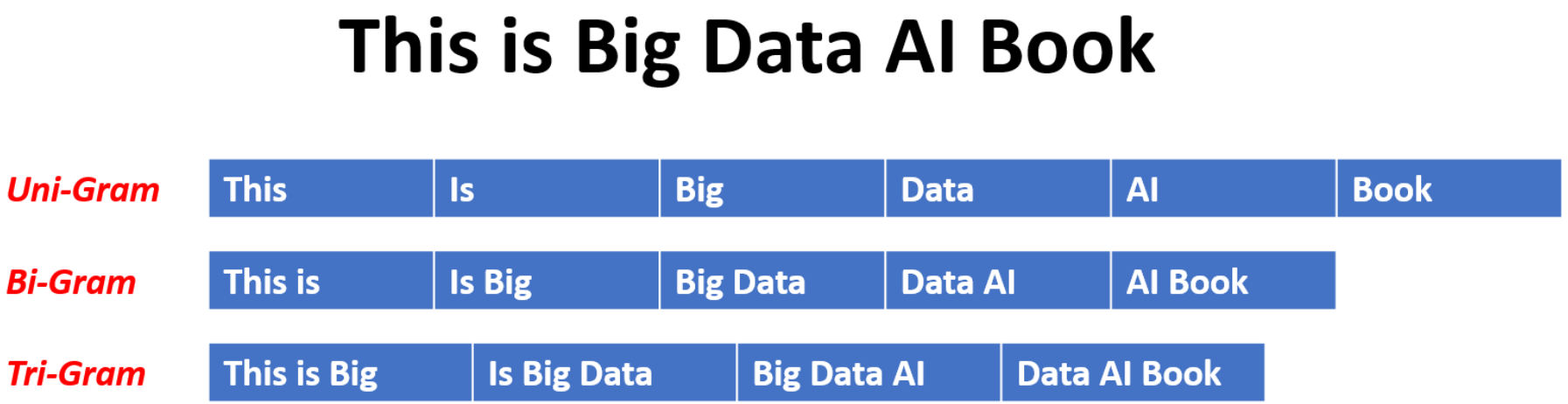
实际功能与应用场景：

- **文本生成与纠错：**通过比较不同词序的概率，语言模型能够帮助纠正语法或拼写错误，如将“Their are”纠正为“There are”，或修正诸如“has improve”之类的不规范表达；
- **语音识别与辅助沟通：**在语音识别系统中，语言模型帮助区分发音相似但意义截然不同的词序；同时，在辅助交流系统（AAC）中，预测功能可以为用户提供上下文相关的候选词，提升输入效率。

单词预测不仅是解决特定任务（如拼写检查、语音识别）的手段，更是训练**大语言模型**的核心任务。当前许多基于神经网络的大语言模型，其训练目标正是通过大量预测任务来学习丰富的语言结构与语义知识。

本讲将以最简单的 **n-gram 语言模型**为例，详细阐述如何利用固定窗口内的上下文（即 n-1 个词）来近似计算下一个词的概率。尽管 n-gram 模型本身在表达能力上较为局限，但其明确的数学形式和直观的统计方法，为理解训练集、测试集、困惑度（Perplexity）、采样以及后续更复杂的插值方法等**大语言模型核心概念**提供了坚实基础。

一个 **n-gram** 就是一个包含 n 个单词的序列，n 常见的取值为 1 (unigram)、2 (bigram)、3 (trigram)，如下图所示：



目录

在 n-gram 模型中，**目标是**通过已知的历史词汇（即前面的 n-1 个单词），**预测下一个单词**。具体来说，n-gram 模型估计 $P(w|h)$ 的条件概率，其中 w 是下一个单词，而 h 是 n-1 个之前的单词。

1. 概率估计的直接方法

为了估计 $P(w|h)$，一种直观的方法是通过**计算相对频率**：统计在一个大规模语料库中，历史序列 h 出现的次数和 h 后跟 w 出现的次数，并通过这些计数来估算概率：

$$P(w|h) = \frac{C(h,w)}{C(h)}$$

其中，$C(h,w)$ 表示序列 h 后接单词 w 的次数，C(h) 表示序列 h 出现的总次数。

2. 链式法则（Chain Rule）

为了估算整个词序列的联合概率 $P(w_1^n) = P(w_1, w_2, ..., w_n)$，可以使用链式法则，将长序列的联合概率转化为一系列条件概率的乘积

$$P(w_1, w_2, ..., w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)...P(w_n|w_1, ..., w_{n-1}) \quad (1)$$

$$= \prod_{k=1}^{n} P(w_k|w_{1:k-1}) \quad (2)$$

这种方法虽然直观，但在实际应用中，由于语言的创造性，某些长词序列的计数可能并不存在，因此需要进行更精巧的概率估计方法。

马尔科夫假设

1. 马尔科夫假设

为了简化问题，n-gram 模型引入了**马尔科夫假设**，即假设下一个单词的概率仅依赖于前 n-1 个单词。例如，在 bigram 模型中 (一阶马尔科夫假设)，预测下一个单词 w 的概率为：

$$P(w|h) \approx P(w|w_{n-1}) \tag{3}$$

这意味着，n-gram 模型通过限制上下文的长度，减少了计算复杂度，使得模型更加高效。

2. 扩展到更高阶的 n-gram

当 n 增大时，模型能够利用更多的上下文信息，在 N-gram 模型中 (n-1 阶马尔科夫假设)，预测下一个单词 w 的概率为：

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1}) \tag{4}$$

3. N-gram 模型

将 (4) 式表示的 N-gram 模型带入 (2) 式表示的词序列联合概率，可得如下近似：

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k | w_{n-N+1:n-1})$$

N=2 时，应用 bigram 模型，词序列的联合概率可近似为：

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k | w_{k-1})$$

目录

最大似然估计（MLE）

n-gram 模型的参数（即各个单词序列的概率）可以通过最大似然估计（MLE）来计算。这种方法通过计算语料库中 n-gram 出现的频率，并对频率进行归一化处理来得到概率值。具体而言，大体步骤如下：

1. 频率计算

对于每一个 n-gram，计算其在语料库中的出现次数。

2. 概率归一化

为了将频率转换为概率，需要将每个 n-gram 的频率除以其前一个单词序列的总频率，概率估计公式为：

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

通过这种方法，n-gram 模型根据统计数据估计词汇序列的概率。

Bigram 模型参数最大似然估计示例

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Figure 3.1 Bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Zero counts are in gray. Each cell shows the count of the column label word following the row label word. Thus the cell in row **i** and column **want** means that **want** followed **i** 827 times in the corpus.

Figure 3.2 shows the bigram probabilities after normalization (dividing each cell in Fig. 3.1 by the appropriate unigram for its row, taken from the following set of unigram counts):

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Figure 3.2 Bigram probabilities for eight words in the Berkeley Restaurant Project corpus of 9332 sentences. Zero probabilities are in gray.

Here are a few other useful probabilities:

$$P(\texttt{i}\,|\,\texttt{<s>}) = 0.25 \qquad P(\texttt{english}\,|\,\texttt{want}) = 0.0011$$
$$P(\texttt{food}\,|\,\texttt{english}) = 0.5 \qquad P(\texttt{</s>}\,|\,\texttt{food}) = 0.68$$

Now we can compute the probability of sentences like *I want English food* or *I want Chinese food* by simply multiplying the appropriate bigram probabilities together, as follows:

$$\begin{aligned} P(\texttt{<s> i want english food </s>}) &= P(\texttt{i}\,|\,\texttt{<s>})P(\texttt{want}\,|\,\texttt{i})P(\texttt{english}\,|\,\texttt{want}) \\ &\qquad P(\texttt{food}\,|\,\texttt{english})P(\texttt{</s>}\,|\,\texttt{food}) \\ &= 0.25 \times 0.33 \times 0.0011 \times 0.5 \times 0.68 \\ &= 0.000031 \end{aligned}$$

n-gram 模型的应用与挑战

1. 应用场景

n-gram 语言模型广泛应用于各种自然语言处理任务，如机器翻译、语音识别、文本生成等。通过在大量文本数据上训练，n-gram 模型能够有效地捕捉语言的统计特性，为后续的应用提供支持。

2. 挑战

- **数据稀疏问题**：n-gram 模型的一个主要问题是，当语料库中不存在某个 n-gram 时，模型会面临“零概率”问题，即某些词序列的概率为零。为了解决这个问题，常使用**平滑技术**（如 Laplace 平滑）来调整概率估计。此外，为避免概率乘积数值下溢，语言模型的概率均在计算和存储过程中使用**对数概率**。

- **高阶模型的计算复杂度**：随着 n 的增加，n-gram 模型的参数数量急剧增加，导致计算资源和存储需求的上升，n 通常不超过 3。

目录

评估语言模型的基本目标

1. 评价目标

评估语言模型的最终目的是判断其对未知数据的预测能力，即模型在面对未见过的测试数据时，能否较好地“拟合”数据。较优的模型会为测试数据分配较高的概率，从而体现出更好的泛化性能。

2. 内在与外在评价

外在评价（Extrinsic Evaluation）：将语言模型**嵌入到实际应用中**，评估其在具体任务中的表现。这种评价方式直接反映了模型在特定应用场景中的有效性，但通常需要高昂的计算成本和大量的实际应用数据。

例如，假设我们有两个不同的语言模型，一个使用的是 bigram 模型，另一个使用的是 trigram 模型。在语音识别任务中，我们将这两个模型分别应用到语音转录系统中，比较其转录准确率。

内在评价（Intrinsic Evaluation）：指在不依赖于具体应用的情况下，通过一些**统计指标**或数学方法直接评估语言模型本身的性能。这类评价通常使用一些标准的评估指标，如困惑度（Perplexity）等。

训练集、测试集、开发集

训练集（Training Set）：用于学习模型参数的语料库。例如，在 n-gram 语言模型中，使用最大似然估计，利用训练集中的 n-gram 计数，经过归一化处理得到模型的概率分布。

测试集（Test Set）：一组与训练集不重叠的、独立的语料，用于评估模型在新数据上的泛化能力。如果一个模型在训练集上表现极佳，但在测试集上效果很差，则说明模型可能过拟合训练数据，无法适用于未知数据。

开发集（Dev Set）：模型在训练过程中即使不直接将测试数据用于训练，多次反复在测试集上调试模型，也可能使模型隐式地适应测试集分布。因此，通常建议在模型开发阶段使用一个独立的开发集，待模型调试完毕后再**在测试集上进行一次性评估**。

数据集划分的原则

领域选择：测试集应能真实反映目标应用领域的语言特征。例如：若模型用于语音识别化学讲座，其测试集应来自化学讲座文本。若构建通用语言模型，则测试集应涵盖多样化的文本来源，避免单一作者或单一文档对评价结果的偏倚。

数据量的权衡：理想情况下，测试集应足够大，以确保统计上的代表性和评价的稳定性；但又不能过大以至于严重影响训练数据的量。

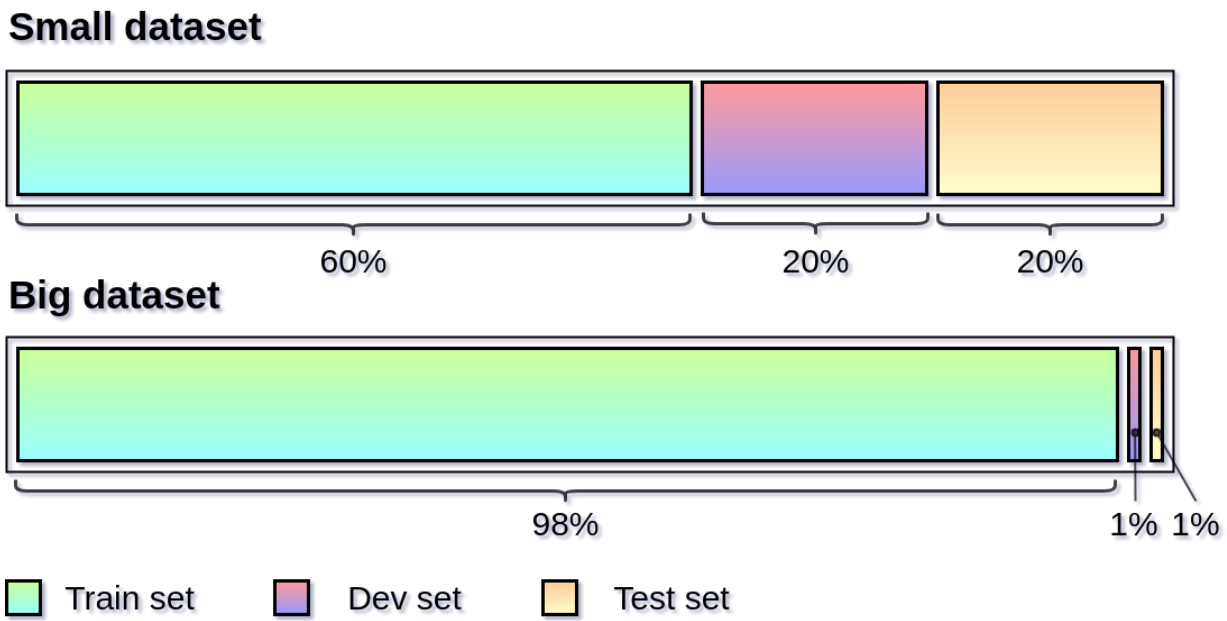


图 1: 数据集的划分比例

未完待续