

# 1 语言模型基础

语言是一套复杂的符号系统。语言符号通常在音韵 (Phonology)、词法 (Morphology)、句法 (Syntax) 的约束下构成, 并承载不同的语义 (Semantics)。语言符号具有不确定性。同样的语义可以由不同的音韵、词法、句法构成的符号来表达; 同样的音韵、词法、句法构成的符号也可以在不同的语境下表达不同的语义。因此, **语言是概率的**。并且, **语言的概率性与认知的概率性也存在着密不可分的关系 [15]**。语言模型 (Language Models, LMs) 旨在**准确预测语言符号的概率**。从语言学的角度, 语言模型可以赋能计算机掌握语法、理解语义, 以完成自然语言处理任务。从认知科学的角度, 准确预测语言符号的概率可以赋能计算机描摹认知、演化智能。从 ELIZA [20] 到 GPT-4 [16], 语言模型经历了从规则模型到统计模型, 再到神经网络模型的发展历程, 逐步从呆板的机械式问答程序成长为具有强大泛化能力的多任务智能模型。本章将按照语言模型发展的顺序依次讲解基于统计方法的 n-grams 语言模型、基于循环神经网络 (Recurrent Neural Network, RNN) 的语言模型, 基于 Transformer 的语言模型。此外, 本章还将介绍如何将语言模型输出概率值解码为目标文本, 以及如何对语言模型的性能进行评估。

\* 本书持续更新, GIT Hub 链接为: <https://github.com/ZJU-LLMs/Foundations-of-LLMs>。

## 1.1 基于统计方法的语言模型

语言模型通过对语料库 (Corpus) 中的语料进行统计或学习来获得预测语言符号概率的能力。通常, 基于统计的语言模型通过直接统计语言符号在语料库中出现的频率来预测语言符号的概率。其中, **n-grams** 是最具代表性的统计语言模型。**n-grams 语言模型基于马尔可夫假设和离散变量的极大似然估计给出语言符号的概率。**本节首先给出 **n-grams** 语言模型的计算方法, 然后讨论 **n-grams** 语言模型如何在马尔可夫假设的基础上应用离散变量极大似然估计给出语言符号出现的概率。

### 1.1.1 n-grams 语言模型

设包含  $N$  个元素的语言符号可以表示为  $w_{1:N} = \{w_1, w_2, w_3, \dots, w_N\}$ 。 $w_{1:N}$  可以代表文本, 也可以代表音频序列等载有语义信息的序列。为了便于理解, 本章令语言符号  $w_{1:N}$  代表文本, 其元素  $w_i \in w_{1:N}$  代表词,  $i = 1, \dots, N$ 。在真实语言模型中,  $w_i$  可以是 Token 等其他形式。关于 Token 的介绍将在第三章中给出。

**n-grams** 语言模型中的 **n-gram** 指的是长度为  $n$  的词序列。**n-grams** 语言模型通过依次统计文本中的 **n-gram** 及其对应的 **(n-1)-gram** 在语料库中出现的相对频率来计算文本  $w_{1:N}$  出现的概率。计算公式如下所示:

$$P_{n\text{-grams}}(w_{1:N}) = \prod_{i=n}^N \frac{C(w_{i-n+1:i})}{C(w_{i-n+1:i-1})}, \quad (1.1)$$

其中,  $C(w_{i-n+1:i})$  为词序列  $\{w_{i-n+1}, \dots, w_i\}$  在语料库中出现的次数,  $C(w_{i-n+1:i-1})$  为词序列  $\{w_{i-n+1}, \dots, w_{i-1}\}$  在语料库中出现的次数。其中,  $n$  为变量, 当  $n = 1$  时, 称之为 **unigram**, 其不考虑文本的上下文关系。此时, 分子  $C(w_{i-n+1:i}) = C(w_i)$ ,  $C(w_i)$  为词  $w_i$  在语料库中出现的次数; 分母  $C(w_{i-n+1:i-1}) = C_{total}$ ,  $C_{total}$  为语料库中包含的词的总数。当  $n = 2$  时, 称之为 **bigrams**, 其对前一个词进行考虑。此时,

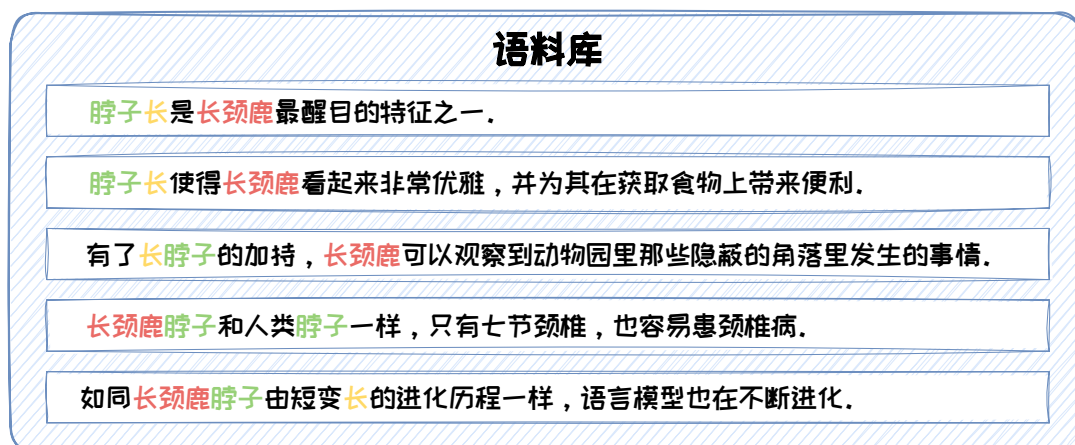


图 1.1: n-grams 示例语料库。

分子  $C(w_{i-n+1}:i) = C(w_{i-1}, w_i)$ ,  $C(w_{i-1}, w_i)$  为词序列  $\{w_{i-1}, w_i\}$  在语料库中出现的次数；分母  $C(w_{i-n+1}:i-1) = C(w_{i-1})$ ,  $C(w_{i-1})$  为词  $w_{i-1}$  在语料库中出现的次数。以此类推，当  $n = 3$  时，称之为 trigrams，其对前两个词进行考虑。当  $n = 4$  时，称之为 4-grams，其对前三个词进行考虑.....

下面通过一个 bigrams 语言模型的例子来展示 n-grams 语言模型对文本出现概率进行计算的具体方式。假设语料库中包含 5 个句子，如图 1.1 所示。基于此语料库，应用 bigrams 对文本“长颈鹿脖子长”（其由 {长颈鹿, 脖子, 长} 三个词构成）出现的概率进行计算，如下式所示：

$$P_{\text{bigrams}}(\text{长颈鹿, 脖子, 长}) = \frac{C(\text{长颈鹿, 脖子})}{C(\text{长颈鹿})} \cdot \frac{C(\text{脖子, 长})}{C(\text{脖子})}。 \quad (1.2)$$

在此语料库中,  $C(\text{长颈鹿}) = 5$ ,  $C(\text{脖子}) = 6$ ,  $C(\text{长颈鹿, 脖子}) = 2$ ,  $C(\text{脖子, 长}) = 2$ , 故有：

$$P_{\text{bigrams}}(\text{长颈鹿, 脖子, 长}) = \frac{2}{5} \cdot \frac{2}{6} = \frac{2}{15}。 \quad (1.3)$$

在此例中，我们可以发现虽然“长颈鹿脖子长”并没有直接出现在语料库中，但是 bigrams 语言模型仍可以预测出“长颈鹿脖子长”出现的概率有  $\frac{2}{15}$ 。由此可见，n-grams 具备对未知文本的泛化能力。这也是其相较于传统基于规则的方法的

优势。但是，这种泛化能力会随着  $n$  的增大而逐渐减弱。应用 trigrams 对文本“长颈鹿脖子长”出现的概率进行计算，将出现以下“零概率”的情况：

$$P_{\text{trigrams}}(\text{长颈鹿, 脖子, 长}) = \frac{C(\text{长颈鹿, 脖子, 长})}{C(\text{长颈鹿, 脖子})} = 0. \quad (1.4)$$

因此，在  $n$ -grams 语言模型中， $n$  代表了拟合语料库的能力与对未知文本的泛化能力之间的权衡。当  $n$  过大时，语料库中难以找到与  $n$ -gram 一模一样的词序列，可能出现大量“零概率”现象；在  $n$  过小时， $n$ -gram 难以承载足够的语言信息，不足以反应语料库的特性。因此，在  $n$ -grams 语言模型中， $n$  的值是影响性能的关键因素。上述的“零概率”现象可以通过平滑（Smoothing）技术进行改善，具体技术可参见文献 [11]。

本小节讲解了  $n$ -grams 语言模型如何计算语言符号出现的概率，但没有分析  $n$ -grams 语言模型的原理。下一小节将从  $n$  阶马尔可夫假设和离散型随机变量的极大似然估计的角度对  $n$ -grams 语言模型背后的统计学原理进行阐述。

### 1.1.2 $n$ -grams 的统计学原理

$n$ -grams 语言模型是在  $n$  阶马尔可夫假设下，对语料库中出现的长度为  $n$  的词序列出现概率的极大似然估计。本节首先给出  $n$  阶马尔可夫假设的定义（见定义 1.1）和离散型随机变量的极大似然估计的定义（见定义 1.2），然后分析  $n$ -grams 如何在马尔可夫假设的基础上应用离散变量极大似然估计给出语言符号出现的概率。

#### 定义 1.1 ( $n$ 阶马尔可夫假设)

对序列  $\{w_1, w_2, w_3, \dots, w_N\}$ ，当前状态  $w_N$  出现的概率只与前  $n$  个状态  $\{w_{N-n}, \dots, w_{N-1}\}$  有关，即：

$$P(w_N | w_1, w_2, \dots, w_{N-1}) \approx P(w_N | w_{N-n}, \dots, w_{N-1}). \quad (1.5) \clubsuit$$

**定义 1.2 (离散型随机变量的极大似然估计)**

给定离散型随机变量  $X$  的分布律为  $P\{X = x\} = p(x; \theta)$ ，设  $X_1, \dots, X_N$  为来自  $X$  的样本， $x_1, \dots, x_N$  为对应的观察值， $\theta$  为待估计参数。在参数  $\theta$  下，分布函数随机取到  $x_1, \dots, x_N$  的概率为：

$$p(x|\theta) = \prod_{i=1}^N p(x_i; \theta)。 \quad (1.6)$$

构造似然函数为：

$$L(\theta|x) = p(x|\theta) = \prod_{i=1}^N p(x_i; \theta)。 \quad (1.7)$$

离散型随机变量的极大似然估计旨在找到  $\theta$  使得  $L(\theta|x)$  取最大值。



在上述两个定义的基础上，对 **n-grams** 的统计原理进行讨论。设文本  $w_{1:N}$  出现的概率为  $P(w_{1:N})$ 。根据条件概率的链式法则， $P(w_{1:N})$  可由下式进行计算。

$$\begin{aligned} P(w_{1:N}) &= P(w_1)P(w_2|w_1)P(w_3|w_{1:2})\dots P(w_N|w_{1:N-1}) \\ &= \prod_{i=1}^N P(w_i|w_{1:i-1})。 \end{aligned} \quad (1.8)$$

根据  $n$  阶马尔可夫假设，**n-grams** 语言模型令  $P(w_i|w_{i-n:i-1})$  近似  $P(w_i|w_{1:i-1})$ 。然后，根据离散型随机变量的极大似然估计，令  $\frac{C(w_{i-n:i})}{C(w_{i-n:i-1})}$  近似  $P(w_i|w_{i-n:i-1})$ 。从而，得到 **n-grams** 语言模型的输出  $P_{n\text{-grams}}(w_{1:N})$  是对  $P(w_i|w_{1:i-1})$  的近似。即，

$$P_{n\text{-grams}}(w_{1:N}) \approx P(w_{1:N})。 \quad (1.9)$$

下面，以 **bigrams** 为例，介绍  $\frac{C(w_{i-n:i})}{C(w_{i-n:i-1})}$  与极大似然估计间的关系。假设语料库中共涵盖  $M$  个不同的单词， $\{w_i, w_j\}$  出现的概率为  $P(w_i, w_j)$ ，对应出现的频率为  $C(w_i, w_j)$ ，则其出现的似然函数为：

$$L(\theta) = \prod_{i=1}^M \prod_{j=1}^M P(w_i, w_j)^{C(w_i, w_j)}, \quad (1.10)$$

其中,  $\theta = \{P(w_i, w_j)\}_{i,j=1}^M$ 。根据条件概率公式  $P(w_i, w_j) = P(w_j|w_i)P(w_i)$ , 有

$$L(\theta) = \prod_{i=1}^M \prod_{j=1}^M P(w_j|w_i)^{C(w_i, w_j)} P(w_i)^{C(w_i, w_j)}。 \quad (1.11)$$

其对应的对数似然函数为:

$$L_{\log}(\theta) = \sum_{i=1}^M \sum_{j=1}^M C(w_i, w_j) \log P(w_j|w_i) + \sum_{i=1}^M \sum_{j=1}^M C(w_i, w_j) \log P(w_i)。 \quad (1.12)$$

因为  $\sum_{j=1}^M P(w_j|w_i) = 1$ , 所以最大化对数似然函数可建模为如下的约束优化问题:

$$\begin{aligned} \max \quad & L_{\log}(\theta) \\ \text{s.t.} \quad & \sum_{j=1}^M P(w_j|w_i) = 1 \text{ for } i \in [1, M]。 \end{aligned} \quad (1.13)$$

其拉格朗日对偶为:

$$L(\lambda, L_{\log}) = L_{\log}(\theta) + \sum_{i=1}^M \lambda_i \left( \sum_{j=1}^M P(w_j|w_i) - 1 \right)。 \quad (1.14)$$

对其求关于  $P(w_j|w_i)$  的偏导, 可得:

$$\frac{\partial L(\lambda, L_{\log})}{\partial P(w_j|w_i)} = \sum_{i=1}^M \frac{C(w_i, w_j)}{P(w_j|w_i)} + \sum_{i=1}^M \lambda_i。 \quad (1.15)$$

当导数为 0 时, 有:

$$P(w_j|w_i) = -\frac{C(w_i, w_j)}{\lambda_i}。 \quad (1.16)$$

因  $\sum_{j=1}^M P(w_j|w_i) = 1$ ,  $\lambda_i$  可取值为  $-\sum_{j=1}^M C(w_i, w_j)$ , 即

$$P(w_j|w_i) = \frac{C(w_i, w_j)}{\sum_{j=1}^M C(w_i, w_j)} = \frac{C(w_i, w_j)}{C(w_i)}。 \quad (1.17)$$

上述分析表明 **bigram** 语言模型中的  $\frac{C(w_i, w_j)}{C(w_i)}$  是对语料库中的长度为 2 的词序列的  $P(w_j|w_i)$  的极大似然估计。该结论可扩展到  $n > 2$  的其他 **n-grams** 语言模型模型中。

**n-grams** 语言模型通过统计词序列在语料库中出现的频率来预测语言符号的概率。其对未知序列有一定的泛化性, 但也容易陷入“零概率”的困境。随着神经网络的发展, 基于各类神经网络的语言模型不断被提出, 泛化能力越来越强。基于神



经网络的语言模型不再通过显性的计算公式对语言符号的概率进行计算，而是利用语料库中的样本对神经网络模型进行训练。本章接下来将分别介绍两类最具代表性的基于神经网络的语言模型：基于 RNN 的语言模型和基于 Transformer 的语言模型。

## 1.2 基于 RNN 的语言模型

循环神经网络 (Recurrent Neural Network, RNN) 是一类**网络连接中包含环路的神经网络的总称**。给定一个序列，RNN 的环路用于将历史状态叠加到当前状态上。沿着时间维度，历史状态被循环累积，并作为预测未来状态的依据。因此，RNN 可以基于历史规律，对未来进行预测。基于 RNN 的语言模型，以词序列作为输入，基于**被循环编码的上文和当前词**来预测下一个词出现的概率。本节将先对原始 RNN 的基本原理进行介绍，然后讲解如何利用 RNN 构建语言模型。

### 1.2.1 循环神经网络 RNN

按照推理过程中信号流转的方向，神经网络的正向传播范式可分为两大类：前馈传播范式和循环传播范式。在前馈传播范式中，计算逐层向前，“**不走回头路**”。而在循环传播范式中，某些层的计算结果会通过**环路**被反向引回前面的层中，形成“**螺旋式前进**”的范式。采用前馈传播范式的神经网络可以统称为前馈神经网络 (Feed-forward Neural Network, FNN)，而采用循环传播范式的神经网络被统称为循环神经网络 (Recurrent Neural Network, RNN)。以包含输入层、隐藏层、输出层的神经网络为例。图1.2中展示了最简单的 FNN 和 RNN 的网络结构示意图，可见 FNN 网络结构中仅包含正向通路。而 RNN 的网络结构中除了正向通路，还有一条**环路**将某层的计算结果再次反向连接回前面的层中。



图 1.2: 前馈传播范式与循环传播范式的对比。

设输入序列为  $\{x_1, x_2, x_3, \dots, x_t\}$ ，隐状态为  $\{h_1, h_2, h_3, \dots, h_t\}$ ，对应输出为  $\{o_1, o_2, o_3, \dots, o_t\}$ ，输入层、隐藏层、输出层对应的网络参数分别为  $W_I, W_H, W_O$ 。 $g(\cdot)$  为激活函数， $f(\cdot)$  为输出函数。将输入序列一个元素接着一个元素地串行输入时，对于 FNN，当前的输出只与当前的输入有关，即

$$o_t = f(W_O g(W_I x_t)). \quad (1.18)$$

此处为方便对比，省去了偏置项。独特的环路结构导致 RNN 与 FNN 的推理过程完全不同。RNN 在串行输入的过程中，前面的元素会被循环编码成隐状态，并叠加到当前的输入上面。其在  $t$  时刻的输出如下：

$$\begin{aligned} h_t &= g(W_H h_{t-1} + W_I x_t) = g(W_H g(W_H h_{t-2} + W_I x_{t-1}) + W_I x_t) = \dots \dots \dots \\ o_t &= f(W_O h_t). \end{aligned} \quad (1.19)$$

其中， $t > 0, h_0 = 0$ 。将此过程按照时间维度展开，可得到 RNN 的推理过程，如图 1.3 所示。注意，图中展示的  $t$  时刻以前的神经元，都是过往状态留下的“虚影”并不真实存在，如此展开只是为了解释 RNN 的工作方式。

可以发现，在这样一个元素一个元素依次串行输入的设定下，RNN 可以将历史状态以隐变量的形式循环叠加到当前状态上，对历史信息进行考虑，呈现出螺旋式前进的模式。但是，缺乏环路的 FNN 仅对当前状态进行考虑，无法兼顾历史状态。以词序列 {长颈鹿, 脖子, 长} 为例，在给定“脖子”来预测下一个词是什么的时候，FNN 将仅仅考虑“脖子”来进行预测，可能预测出的下一词包含“短”，“疼”等等；而 RNN 将同时考虑“长颈鹿”和“脖子”，其预测出下一词是“长”的概率将更高，历史信息“长颈鹿”的引入，可以有效提升预测性能。



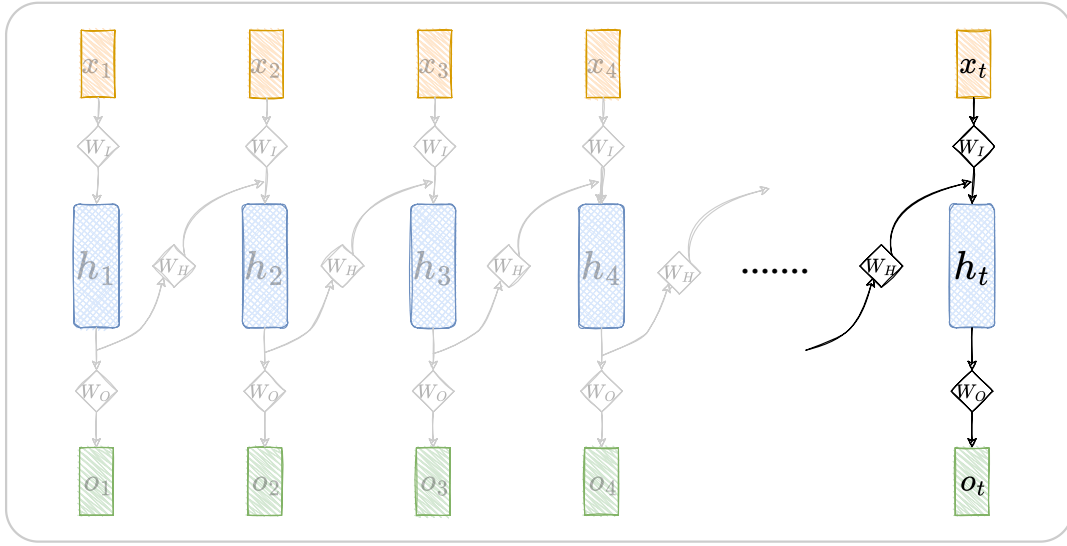


图 1.3: RNN 推理过程从时间维度拆解示意图。

如果 FNN 想要做到对历史信息进行考虑，则需要将所有元素同时输入到模型中去，这将导致模型参数量的激增。虽然，RNN 的结构可以让其在参数量不扩张的情况下实现对历史信息的考虑，但是这样的环路结构给 RNN 的训练带来了挑战。在训练 RNN 时，涉及大量的矩阵联乘操作，容易引发**梯度衰减**或**梯度爆炸**问题。具体分析如下：

设 RNN 语言模型的训练损失为：

$$L = L(x, o, W_I, W_H, W_O) = \sum_{i=1}^t l(o_i, y_i)。 \quad (1.20)$$

其中， $l(\cdot)$  为损失函数， $y_i$  为标签。

损失  $L$  关于参数  $W_H$  的梯度为：

$$\frac{\partial L}{\partial W_H} = \sum_{i=1}^t \frac{\partial l_t}{\partial o_t} \cdot \frac{\partial o_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial h_i} \cdot \frac{\partial h_i}{\partial W_H}。 \quad (1.21)$$

其中，

$$\frac{\partial h_t}{\partial h_i} = \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \cdots \frac{\partial h_{i+1}}{\partial h_i} = \prod_{k=i+1}^t \frac{\partial h_k}{\partial h_{k-1}}。 \quad (1.22)$$

并且,

$$\frac{\partial h_k}{\partial h_{k-1}} = \frac{\partial g(z_k)}{\partial z_k} W_H. \quad (1.23)$$

其中,  $z_k = W_H h_{k-1} + W_I x_k$ 。综上, 有

$$\frac{\partial L}{\partial W_H} = \sum_{i=1}^t \frac{\partial l_t}{\partial o_t} \cdot \frac{\partial o_t}{\partial h_t} \cdot \prod_{k=i}^t \frac{\partial g(z_k)}{\partial z_k} W_H \cdot \frac{\partial h_i}{\partial W_H}. \quad (1.24)$$

从上式中可以看出, 求解  $W_H$  的梯度时涉及大量的矩阵级联相乘。这会导致其数值被级联放大或缩小。文献 [17] 中指出, 当  $W_H$  的最大特征值小于 1 时, 会发生梯度消失; 当  $W_H$  的最大特征值大于 1 时, 会发生梯度爆炸。梯度消失和爆炸导致训练上述 RNN 非常困难。为了解决梯度消失和爆炸问题, GRU [4] 和 LSTM [8] 引入门控结构, 取得了良好效果, 成为主流的 RNN 网络架构。

## 1.2.2 基于 RNN 的语言模型

对词序列  $\{w_1, w_2, w_3, \dots, w_N\}$ , 基于 RNN 的语言模型每次根据当前词  $w_i$  和循环输入的隐藏状态  $h_{i-1}$ , 来预测下一个词  $w_{i+1}$  出现的概率, 即

$$P(w_{i+1}|w_{1:i}) = P(w_{i+1}|w_i, h_{i-1}). \quad (1.25)$$

其中, 当  $i = 1$  时,  $P(w_{i+1}|w_i, h_{i-1}) = P(w_2|w_1)$ 。基于此,  $\{w_1, w_2, w_3, \dots, w_N\}$  整体出现的概率为:

$$P(w_{1:N}) = \prod_{i=1}^{N-1} P(w_{i+1}|w_i, h_{i-1}). \quad (1.26)$$

在基于 RNN 的语言模型中, 输出为一个向量, 其中每一维代表着词典中对应词的概率。设词典  $D$  中共有  $|D|$  个词  $\{\hat{w}_1, \hat{w}_2, \hat{w}_3, \dots, \hat{w}_{|D|}\}$ , 基于 RNN 的语言模型的输出可表示为  $o_i = \{o_i[\hat{w}_d]\}_{d=1}^{|D|}$ , 其中,  $o_i[\hat{w}_d]$  表示词典中的词  $\hat{w}_d$  出现的概率。因此, 对基于 RNN 的语言模型有

$$P(w_{1:N}) = \prod_{i=1}^{N-1} P(w_{i+1}|w_{1:i}) = \prod_{i=1}^N o_i[w_{i+1}]. \quad (1.27)$$

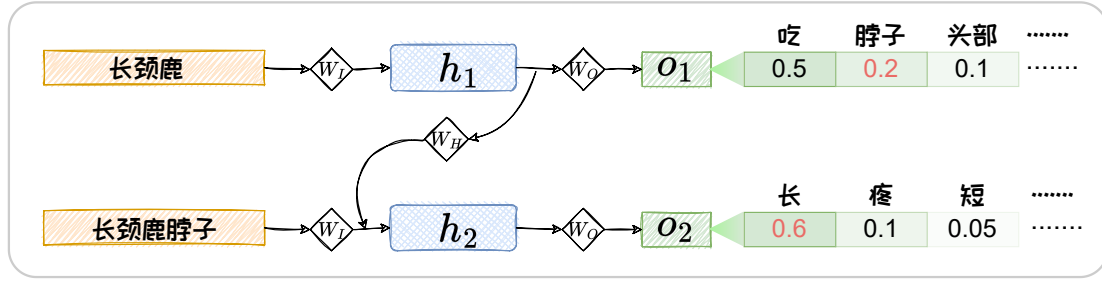


图 1.4: RNN 计算词序列概率示意图。

以下举例对上述过程进行说明。假设词表  $D = \{\text{脖子, 头部, 吃, 长, 疼, 吃, 短}\}$ , 基于 RNN 的语言模型计算“长颈鹿脖子长”的概率的过程如图 1.4 所示。

$$P(\text{长颈鹿脖子长}) = P(\text{脖子}|\text{长颈鹿}) \cdot P(\text{长}|\text{脖子}, h_1) = 0.2 \times 0.6 = 0.12. \quad (1.28)$$

基于以上预训练任务，对 RNN 语言模型进行训练时，可选用如下交叉熵函数作为损失函数。

$$l_{CE}(o_i) = - \sum_{d=1}^{|D|} I(\hat{w}_d = w_{i+1}) \log o_i[w_{i+1}] = - \log o_i[w_{i+1}], \quad (1.29)$$

其中， $I(\cdot)$  为指示函数，当  $\hat{w}_d = w_{i+1}$  时等于 1，当  $\hat{w}_d \neq w_{i+1}$  时等于 0。

设训练集为  $S$ ，RNN 语言模型的损失可以构造为：

$$L(S, W_I, W_H, W_O) = \frac{1}{N|S|} \sum_{s=1}^{|S|} \sum_{i=1}^N l_{CE}(o_{i,s}), \quad (1.30)$$

其中， $o_{i,s}$  为 RNN 语言模型输入第  $s$  个样本的第  $i$  个词时的输出。此处为方便表述，假设每个样本的长度都为  $N$ 。在此损失的基础上，构建计算图，进行反向传播，便可对 RNN 语言模型进行训练。上述训练过程结束之后，我们可以直接利用此模型对序列数据进行特征抽取。抽取的特征可以用于解决下游任务。此外，我们还可以对此语言模型的输出进行解码，在“自回归”的范式下完成文本生成任务。在自回归中，第一轮，我们首先将第一个词输入给 RNN 语言模型，经过解码，得到一个输出词。然后，我们将第一轮输出的词与第一轮输入的词拼接，作为第二轮

的输入，然后解码得到第二轮的输出。接着，将第二轮的输出和输入拼接，作为第三轮的输入，以此类推。每次将本轮预测到的词拼接到本轮的输入上，输入给语言模型，完成下一轮预测。在循环迭代的“自回归”过程中，我们不断生成新的词，这些词便构成了一段文本。

但上述“自回归”过程存在着两个问题：(1) **错误级联放大**，选用模型自己生成的词作为输入可能会有错误，这样的错误循环输入，将会不断的放大错误，导致模型不能很好拟合训练集；(2) **串行计算效率低**，因为下一个要预测的词依赖上一次的预测，每次预测之间是串行的，难以进行并行加速。为了解决上述两个问题，“**Teacher Forcing**” [21] 在语言模型预训练过程中被广泛应用。在 Teacher Forcing 中，每轮都仅将输出结果与“标准答案”（Ground Truth）进行拼接作为下一轮的输入。在图1.4所示的例子中，第二轮循环中，我们用“长颈鹿脖子”来预测下一个词“长”，而非选用  $o_1$  中概率最高的词“吃”或者其他可能输出的词。

但是，Teacher Forcing 的训练方式将导致**曝光偏差**（Exposure Bias）的问题。曝光偏差是指 Teacher Forcing 训练模型的过程和模型在推理过程存在差异。Teacher Forcing 在训练中，模型将依赖于“标准答案”进行下一次的预测，但是在推理预测中，模型“自回归”的产生文本，没有“标准答案”可参考。所以模型在训练过程中和推理过程中存在偏差，可能推理效果较差。为解决曝光偏差的问题，Bengio 等人提出了针对 RNN 提出了 Scheduled Sampling 方法 [2]。其在 Teacher Forcing 的训练过程中循序渐进的使用一小部分模型自己生成的词代替“标准答案”，在训练过程中对推理中无“标准答案”的情况进行预演。

由于 RNN 模型循环迭代的本质，其不易进行并行计算，导致其在输入序列较长时，训练较慢。下节将对容易并行的基于 Transformer 的语言模型进行介绍。

## 1.3 基于 Transformer 的语言模型

Transformer 是一类基于注意力机制（Attention）的**模块化**构建的神经网络结构。给定一个序列，Transformer 将一定数量的历史状态和当前状态同时输入，然后进行加权相加。对历史状态和当前状态进行“**通盘考虑**”，然后对未来状态进行预测。基于 Transformer 的语言模型，以词序列作为输入，基于一定长度的上文和当前词来预测下一个词出现的概率。本节将先对 Transformer 的基本原理进行介绍，然后讲解如何利用 Transformer 构建语言模型。

### 1.3.1 Transformer

Transformer 是由两种模块组合构建的**模块化网络结构**。两种模块分别为：（1）注意力（Attention）模块；（2）全连接前馈（Fully-connected Feedforward）模块。其中，自注意力模块由自注意力层（Self-Attention Layer）、残差连接（Residual Connections）和层正则化（Layer Normalization）组成。全连接前馈模块由全连接前馈层，残差连接和层正则化组成。两个模块的结构示意图如图1.5所示。以下详细介绍每个层的原理及作用。

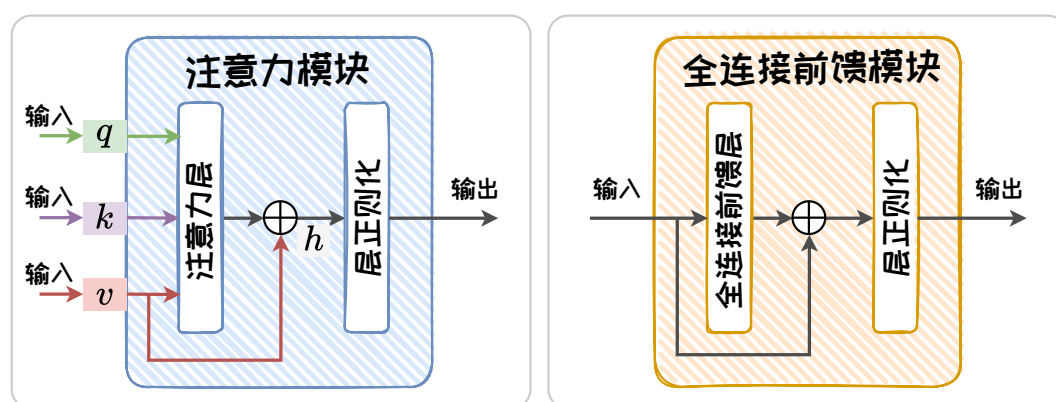


图 1.5: 注意力模块与全连接前馈模块。

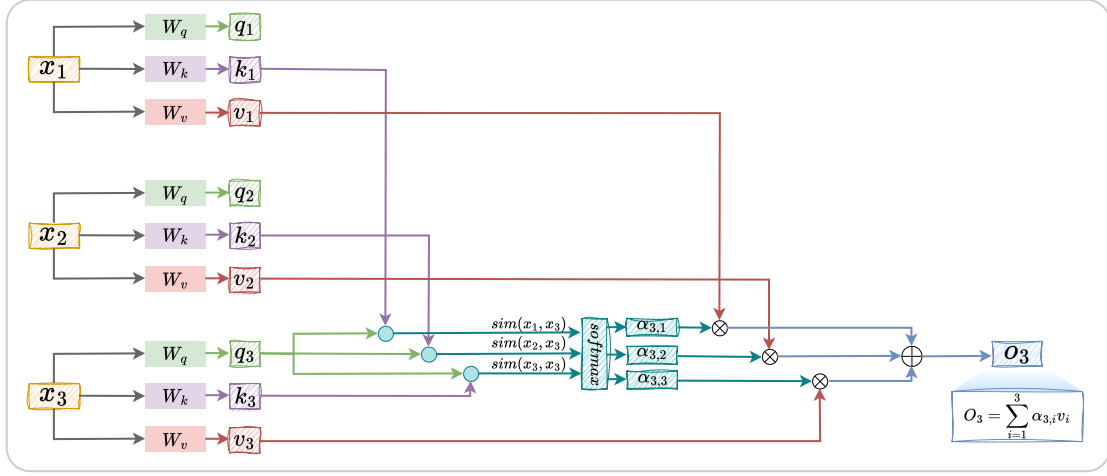


图 1.6: 注意力机制示意图。

### 1. 注意力层 (Attention Layer)

注意力层采用加权平均的思想将前文信息叠加到当前状态上。Transformer 的注意力层将输入编码为 query, key, value 三部分, 即将输入  $\{x_1, x_2, \dots, x_t\}$  编码为  $\{(q_1, k_1, v_1), (q_2, k_2, v_2), \dots, (q_t, k_t, v_t)\}$ 。其中, query 和 key 用于计算自注意力的权重  $\alpha$ , value 是对输入的编码。具体的,

$$Attention(x_t) = \sum_{i=1}^t \alpha_{t,i} v_i. \quad (1.31)$$

其中,

$$\alpha_{t,i} = softmax(sim(x_t, x_i)) = \frac{sim(q_t, k_i)}{\sum_{i=1}^t sim(q_t, k_i)}. \quad (1.32)$$

其中,  $sim(\cdot, \cdot)$  用于度量两个输入之间的相关程度, softmax 函数用于对此相关程度进行归一化。此外,

$$q_i = W_q x_i, \quad k_i = W_k x_i, \quad v_i = W_v x_i, \quad (1.33)$$

其中,  $W_q, W_k, W_v$  分别为 query, key, value 编码器的参数。以包含三个元素的输入  $\{x_1, x_2, x_3\}$  为例, Transformer 自注意力的实现图 1.6 所示。



## 2. 全连接前馈层 (Fully-connected Feedforward Layer)

全连接前馈层占据了 Transformer 近三分之二的参数，掌管着 Transformer 模型的记忆。其可以看作是一种 Key-Value 模式的记忆存储管理模块 [7]。全连接前馈层包含两层，两层之间由 ReLU 作为激活函数。设全连接前馈层的输入为  $v$ ，全连接前馈层可由下式表示：

$$FFN(v) = \max(0, W_1 v + b_1) W_2 + b_2. \quad (1.34)$$

其中， $W_1$  和  $W_2$  分别为第一层和第二层的权重参数， $b_1$  和  $b_2$  分别为第一层和第二层的偏置参数。其中第一层的可看作神经记忆中的 key，而第二层可看作 value。

## 3. 层正则化 (Layer Normalization)

层正则化用以加速神经网络训练过程并取得更好的泛化性能 [1]。设输入到层正则化层的向量为  $v = \{v_i\}_{i=1}^n$ 。层正则化层将在  $v$  的每一维度  $v_i$  上都进行层正则化操作。具体地，层正则化操作可以表示为下列公式：

$$LN(v_i) = \frac{\alpha(v_i - \mu)}{\delta} + \beta. \quad (1.35)$$

其中， $\alpha$  和  $\beta$  为可学习参数。 $\mu$  和  $\delta$  分别是隐藏状态的均值和方差，可由下列公式分别计算。

$$\mu = \frac{1}{n} \sum_{i=1}^n v_i, \quad \delta = \sqrt{\frac{1}{n} \sum_{i=1}^n (v_i - \mu)^2}. \quad (1.36)$$

## 4. 残差连接 (Residual Connections)

引入残差连接可以有效解决梯度消失问题。在基本的 Transformer 编码模块中包含两个残差连接。第一个残差连接是将自注意力层的输入由一条旁路叠加到自注意力层的输出上，然后输入给层正则化。第二个残差连接是将全连接前馈层的输入由一条旁路引到全连接前馈层的输出上，然后输入给层正则化。

上述将层正则化置于残差连接之后的网络结构被称为 Post-LN Transformer。与之相对的，还有一种将层正则化置于残差连接之前的网络结构，称之为 Pre-LN

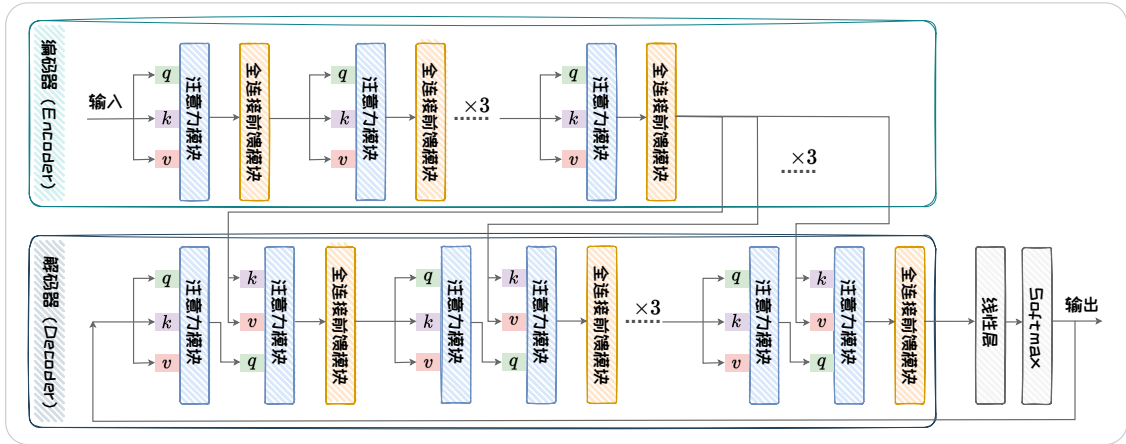


图 1.7: Transformer 结构示意图。

Transformers。对比两者, Post-LN Transformer 应对表征坍塌 (Representation Collapse) 的能力更强, 但处理梯度消失略弱。而 Pre-LN Transformers 可以更好的应对梯度消失, 但处理表征坍塌的能力略弱。具体分析可参考文献 [7, 22]。

原始的 Transformer 采用 Encoder-Decoder 架构, 其包含 Encoder 和 Decoder 两部分。这两部分都是由自注意力模块和全连接前馈模块重复连接构建而成。其整体结构如图 1.7 所示。其中, Encoder 部分由六个级联的 encoder layer 组成, 每个 encoder layer 包含一个注意力模块和一个全连接前馈模块。其中的注意力模块为**自注意力模块** (query, key, value 的输入是相同的)。Decoder 部分由六个级联的 decoder layer 组成, 每个 decoder layer 包含两个注意力模块和一个全连接前馈模块。其中, 第一个注意力模块为**自注意力模块**, 第二个注意力模块为**交叉注意力模块** (query, key, value 的输入不同)。Decoder 中第一个 decoder layer 的自注意力模块的输入为模型的输出。其后的 decoder layer 的自注意力模块的输入为上一个 decoder layer 的输出。Decoder 交叉注意力模块的输入分别是自注意力模块的输出 (query) 和最后一个 encoder layer 的输出 (key, value)。

Transformer 的 Encoder 部分和 Decoder 部分都可以单独用于构造语言模型, 分别对应 Encoder-Only 模型和 Decoder-Only 模型。Encoder-Only 模型和 Decoder-Only 模型的具体结构将在第二章中进行详细介绍。

### 1.3.2 基于 Transformer 的语言模型

在 Transformer 的基础上, 可以设计多种预训练任务来训练语言模型。例如, 我们可以基于 Transformer 的 Encoder 部分, 结合“掩词补全”等任务来训练 Encoder-Only 语言模型, 如 BERT [5]; 我们可以同时应用 Transformer 的 Encoder 和 Decoder 部分, 结合“截断补全”、“顺序恢复”等多个有监督和自监督任务来训练 Encoder-Decoder 语言模型, 如 T5 [18]; 我们可以同时应用 Transformer 的 Decoder 部分, 利用“下一词预测”任务来训练 Decoder-Only 语言模型, 如 GPT-3 [3]。这些语言模型将在第二章中进行详细介绍。下面将以下一词预测任务为例, 简单介绍训练 Transformer 语言模型的流程。

对词序列  $\{w_1, w_2, w_3, \dots, w_N\}$ , 基于 Transformer 的语言模型根据  $\{w_1, w_2, \dots, w_i\}$  预测下一个词  $w_{i+1}$  出现的概率。在基于 Transformer 的语言模型中, 输出为一个向量, 其中每一维代表着词典中对应词的概率。设词典  $D$  中共有  $|D|$  个词  $\{\hat{w}_1, \hat{w}_2, \dots, \hat{w}_{|D|}\}$ 。基于 Transformer 的语言模型的输出可表示为  $o_i = \{o_i[\hat{w}_d]\}_{d=1}^{|D|}$ , 其中,  $o_i[\hat{w}_d]$  表示词典中的词  $\hat{w}_d$  出现的概率。因此, 对 Transformer 的语言模型对词序列  $\{w_1, w_2, w_3, \dots, w_N\}$  整体出现的概率的预测为:

$$P(w_{1:N}) = \prod_{i=1}^{N-1} P(w_{i+1}|w_{1:i}) = \prod_{i=1}^N o_i[w_{i+1}] \quad (1.37)$$

与训练 RNN 语言模型相同, Transformer 语言模型也常用如下交叉熵函数作为损失函数。

$$l_{CE}(o_i) = - \sum_{d=1}^{|D|} I(\hat{w}_d = w_{i+1}) \log o_i[w_{i+1}] = - \log o_i[w_{i+1}]. \quad (1.38)$$

其中,  $I(\cdot)$  为指示函数, 当  $\hat{w}_d = w_{i+1}$  时等于 1, 当  $\hat{w}_d \neq w_{i+1}$  时等于 0。

设训练集为  $S$ , Transformer 语言模型的损失可以构造为:

$$L(S, W) = \frac{1}{N|S|} \sum_{s=1}^{|S|} \sum_{i=1}^N l_{CE}(o_{i,s}) \quad (1.39)$$

其中,  $o_{i,s}$  为 Transformer 语言模型输入样本  $s$  的前  $i$  个词时的输出。在此损失的基础上, 构建计算图, 进行反向传播, 便可对 Transformer 语言模型进行训练。

上述训练过程结束之后, 我们可以将 Encoder 的输出作为特征, 然后应用这些特征解决下游任务。此外, 还可在“自回归”的范式下完成文本生成任务。在自回归中, 第一轮, 我们首先将第一个词输入给 Transformer 语言模型, 经过解码, 得到一个输出词。然后, 我们将第一轮输出的词与第一轮输入的词拼接, 作为第二轮的输入, 然后解码得到第二轮的输出。接着, 将第二轮的输出和输入拼接, 作为第三轮的输入, 以此类推。每次将本轮预测到的词拼接到本轮的输入上, 输入给语言模型, 完成下一轮预测。在循环迭代的“自回归”过程中, 我们不断生成新的词, 这些词便构成了一段文本。与训练 RNN 语言模型一样, Transformer 模型的预训练过程依然采用第 1.2.2 节中提到的“Teacher Forcing”的范式。

相较于 RNN 模型串行的循环迭代模式, Transformer 并行输入的特性, 使其容易进行并行计算。但是, Transformer 并行输入的范式也导致网络模型的规模随输入序列长度的增长而平方次增长。这为应用 Transformer 处理长序列带来挑战。

## 1.4 语言模型的采样方法

语言模型的输出为一个向量, 该向量的每一维代表着词典中对应词的概率。在采用自回归范式的文本生成任务中, 语言模型将依次生成一组向量并将其解码为文本。将这组向量解码为文本的过程被成为语言模型解码。解码过程显著影响着生成文本的质量。当前, 两类主流的解码方法可以总结为 (1). 概率最大化方法; (2). 随机采样方法。两类方法分别在下面章节中进行介绍。

### 1.4.1 概率最大化方法

设词典为  $D$ ，输入文本为  $\{w_1, w_2, w_3, \dots, w_N\}$ ，第  $i$  轮自回归中输出的向量为  $o_i = \{o_i[w_d]\}_{d=1}^{|D|}$ ，模型在  $M$  轮自回归后生成的文本为  $\{w_{N+1}, w_{N+2}, w_{N+3}, \dots, w_{N+M}\}$ 。生成文档的出现的概率可由下式进行计算。

$$P(w_{N+1:N+M}) = \prod_{i=N}^{N+M-1} P(w_{i+1}|w_{1:i}) = \prod_{i=N}^{N+M-1} o_i[w_{i+1}] \quad (1.40)$$

基于概率最大化的解码方法旨在最大化  $P(w_{N+1:N+M})$ ，以生成出可能性最高的文本。该问题的搜索空间大小为  $M^D$ ，是 NP-Hard 问题。现有概率最大化方法通常采用启发式搜索方法。本节将介绍两种常用的基于概率最大化的解码方法。

#### 1. 贪心搜索 (Greedy Search)

贪心搜索在在每轮预测中都选择概率最大的词，即

$$w_{i+1} = \arg \max_{w \in D} o_i[w] \quad (1.41)$$

贪心搜索只顾“眼前利益”，忽略了“远期效益”。当前概率大的词有可能导致后续的词概率都很小。贪心搜索容易陷入局部最优，难以达到全局最优解。以图1.8为例，当输入为“生成一个以长颈鹿开头的故事：长颈鹿”时，预测第一个词为“是”的概率最高，为 0.3。但选定“是”之后，其他的词的概率都偏低。如果按照贪心搜索的方式，我们最终得到的输出为“是草食”。其概率仅为 0.03。而如果我们在第一个词选择了概率第二的“脖子”，然后第二个词选到了“长”，最终的概率可以达到 0.1。通过此例，可以看出贪心搜索在求解概率最大的时候容易陷入局部最优。为缓解此问题，可以采用波束搜索 (Beam Search) 方法进行解码。

#### 2. 波束搜索 (Beam Search)

波束搜索在每轮预测中都先保留  $b$  个可能性最高的词  $B_i = \{w_{i+1}^1, w_{i+1}^2, \dots, w_{i+1}^b\}$ ，即：

$$\min\{o_i[w] \text{ for } w \in B_i\} > \max\{o_i[w] \text{ for } w \in D - B_i\}. \quad (1.42)$$



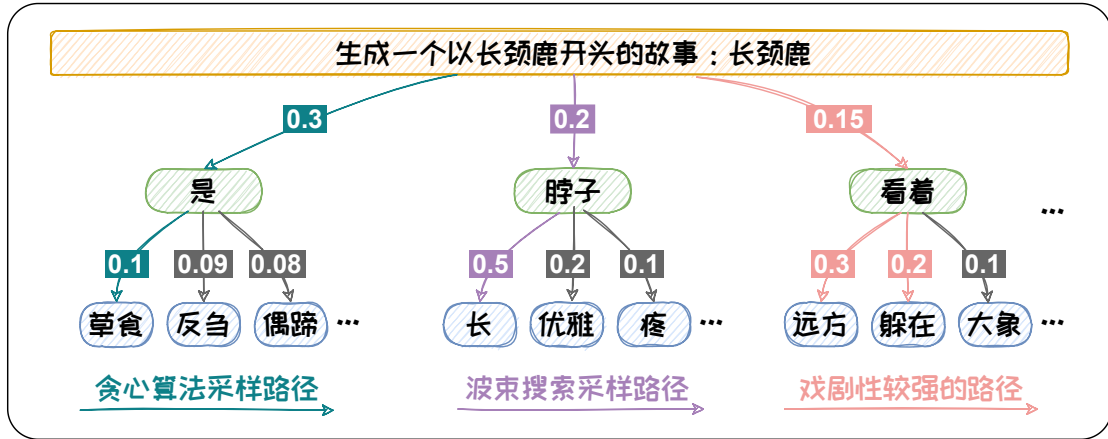


图 1.8: 贪心搜索与波束搜索对比以及概率最大化解码的潜在问题。

在结束搜索时，得到  $M$  个集合，即  $\{B_i\}_{i=1}^M$ 。找出最优组合使得联合概率最大，即：

$$\{w_{N+1}, \dots, w_{N+M}\} = \arg \max_{\{w^i \in B_i \text{ for } 1 \leq i \leq M\}} \prod_{i=1}^M o_{N+i}[w^i]. \quad (1.43)$$

继续以上面的“生成一个以长颈鹿开头的故事”为例，从图 1.8 中可以看出如果我们采用  $b = 2$  的波束搜索方法，我们可以得到“是草食”，“是反刍”，“脖子长”，“脖子优雅”四个候选组合，对应的概率分别为：0.03，0.027，0.1，0.04。我们容易选择到概率最高的“脖子长”。

但是，概率最大的文本**通常是最为常见的文本**。这些文本会略显平庸。在开放式文本生成中，无论是贪心搜索还是波束搜索都容易生成一些“废话文学”——重复且平庸的文本。其所生成的文本缺乏多样性 [19]。如在图 1.8 中的例子所示，概率最大的方法会生成“脖子长”。“长颈鹿脖子长”这样的文本新颖性较低。为了提升生成文本的新颖度，我们可以在解码过程中加入一些随机元素。这样的话就可以解码到一些不常见的组合，从而使得生成的文本更具创意，更适合开放式文本任务。在解码过程中加入随机性的方法，成为随机采样方法。下节将对随机采样方法进行介绍。



## 1.4.2 随机采样方法

为了增加生成文本的多样性，随机采样的方法在预测时增加了随机性。在每轮预测时，其先选出一组可能性高的候选词，然后按照其概率分布进行随机采样，采样出的词作为本轮的预测结果。当前，主流的 Top-K 采样和 Top-P 采样方法分别通过指定候选词数量和划定候选词概率阈值的方法对候选词进行选择。在采样方法中加入 Temperature 机制可以对候选词的概率分布进行调整。接下来将对 Top-K 采样、Top-P 采样和 Temperature 机制分别展开介绍。

### 1. Top-K 采样

Top-K 采样在每轮预测中都选取  $K$  个概率最高的词  $\{w_{i+1}^1, w_{i+1}^2, \dots, w_{i+1}^K\}$  作为本轮的候选词集合，然后对这些词的概率用 softmax 函数进行归一化，得到如下分布函数

$$p(w_{i+1}^1, \dots, w_{i+1}^K) = \left\{ \frac{\exp(o_i[w_{i+1}^1])}{\sum_{j=1}^K \exp(o_i[w_{i+1}^j])}, \dots, \frac{\exp(o_i[w_{i+1}^K])}{\sum_{j=1}^K \exp(o_i[w_{i+1}^j])} \right\}. \quad (1.44)$$

然后根据该分布采样出本轮的预测的结果，即

$$w_{i+1} \sim p(w_{i+1}^1, \dots, w_{i+1}^K). \quad (1.45)$$

Top-K 采样可以有效的增加生成文本的新颖度，例如在上述图1.8所示的例子中选用 Top-3 采样的策略，则有可能会选择到“看着躲在”。“长颈鹿看着躲在”可能是一个极具戏剧性的悬疑故事的开头。

但是，将候选集设置为固定的大小  $K$  将导致上述分布在不同轮次的预测中存在很大差异。当候选词的分布的方差较大的时候，可能会导致本轮预测选到概率较小、不符合常理的词，从而产生“胡言乱语”。例如，在如图1.9 (a) 所示的例子中，Top-2 采样有可能采样出“长颈鹿有四条裤子”。而当候选词的分布的方差较小的时候，甚至趋于均匀分布时，固定尺寸的候选集中无法容纳更多的具有相近概率的词，导致候选集不够丰富，从而导致所选词缺乏新颖性而产生“枯燥无趣”的

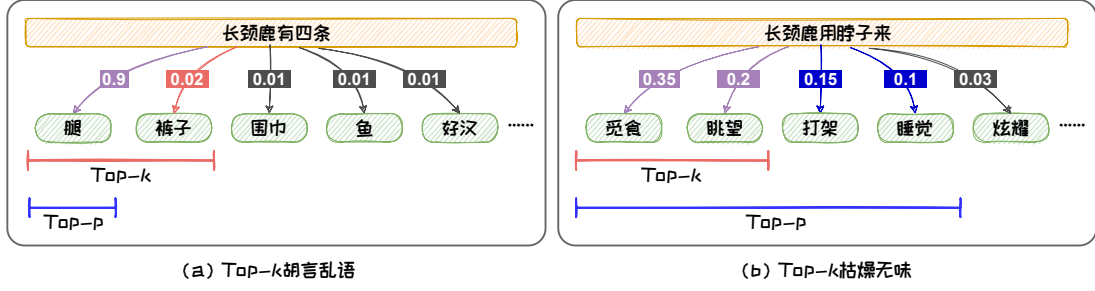


图 1.9: Top-K 采样的潜在问题及其与 Top-P 方法的对比。

文本。例如，在如下图 1.9 (b) 所示的例子中，通过 Top-2 采样，我们只能得到“长颈鹿用脖子来觅食”或者“长颈鹿用脖子来眺望”，这些都是人们熟知的长颈鹿脖子的用途，缺乏新意。但是其实长颈鹿的脖子还可以用于打架或睡觉，Top-2 采样的方式容易将这些新颖的不常见的知识排除。为了解决上述问题，我们可以使用 Top-P 采样，也称 Nucleus 采样。

## 2. Top-P 采样

为了解决固定候选集所带来的问题，Top-P 采样（即 Nucleus 采样）被提出 [9]。其设定阈值  $p$  来对候选集进行选取。其候选集可表示为  $S_p = \{w_{i+1}^1, w_{i+1}^2, \dots, w_{i+1}^{|S_p|}\}$ ，其中，对  $S_p$  有， $\sum_{w \in S_p} o_i[w] \geq p$ 。候选集中元素的分布服从

$$p(w_{i+1}^1, \dots, w_{i+1}^{|S_p|}) = \left\{ \frac{\exp(o_i[w_{i+1}^1])}{\sum_{j=1}^{|S_p|} \exp(o_i[w_{i+1}^j])}, \dots, \frac{\exp(o_i[w_{i+1}^{|S_p|}])}{\sum_{j=1}^{|S_p|} \exp(o_i[w_{i+1}^j])} \right\}. \quad (1.46)$$

然后根据该分布采样出本轮的预测的结果，即

$$w_{i+1} \sim p(w_{i+1}^1, \dots, w_{i+1}^{|S_p|}). \quad (1.47)$$

应用阈值作为候选集选取的标准之后，Top-P 采样可以避免选到概率较小、不符合常理的词，从而减少“胡言乱语”。例如在图 1.9 (a) 所示例子中，我们若以 0.9 作为阈值，则就可以很好的避免“长颈鹿有四条裤子”的问题。并且，其还可以容

纳更多的具有相近概率的词，增加文本的丰富度，改善“枯燥无趣”。例如在图1.9 (b) 所示的例子中，我们若以 0.9 作为阈值，则就可以包含打架、睡觉等长颈鹿脖子鲜为人知的用途。

### 3. Temperature 机制

Top-K 采样和 Top-P 采样的随机性由语言模型输出的概率决定，不可自由调整。但在不同场景中，我们对于随机性的要求可能不一样。比如在开放文本生成中，我们更倾向于生成更具创造力的文本，所以我们需要采样具有更强的随机性。而在代码生成中，我们希望生成的代码更为保守，所以我们需要较弱的随机性。引入 Temperature 机制可以对解码随机性进行调节。Temperature 机制通过对 Softmax 函数中的自变量进行尺度变换，然后利用 Softmax 函数的非线性实现对分布的控制。设 Temperature 尺度变换的变量为  $T$ 。

引入 Temperature 后，Top-K 采样的候选集的分布如下所示：

$$p(w_{i+1}^1, \dots, w_{i+1}^K) = \left\{ \frac{\exp(\frac{o_i[w_{i+1}^1]}{T})}{\sum_{j=1}^K \exp(\frac{o_i[w_{i+1}^j]}{T})}, \dots, \frac{\exp(\frac{o_i[w_{i+1}^K]}{T})}{\sum_{j=1}^K \exp(\frac{o_i[w_{i+1}^j]}{T})} \right\}. \quad (1.48)$$

引入 Temperature 后，Top-P 采样的候选集的分布如下所示：

$$p(w_{i+1}^1, \dots, w_{i+1}^{|S_p|}) = \left\{ \frac{\exp(\frac{o_i[w_{i+1}^1]}{T})}{\sum_{j=1}^{|S_p|} \exp(\frac{o_i[w_{i+1}^j]}{T})}, \dots, \frac{\exp(\frac{o_i[w_{i+1}^{|S_p|}]}{T})}{\sum_{j=1}^{|S_p|} \exp(\frac{o_i[w_{i+1}^j]}{T})} \right\}. \quad (1.49)$$

容易看出，当  $T > 1$  时，Temperature 机制会使得候选集中的词的概率差距减小，分布变得更平坦，从而增加随机性。当  $0 < T < 1$  时，Temperature 机制会使得候选集中的元素概率差距加大，强者越强，弱者越弱，概率高的候选词会容易被选到，从而随机性变弱。Temperature 机制可以有效的对随机性进行调节来满足不同的需求。

## 1.5 语言模型的评测

得到一个语言模型后，我们需要对其生成能力进行评测，以判断其优劣。评测语言模型生成能力的方法可以分为两类。第一类方法不依赖具体任务，直接通过语言模型的输出来评测模型的生成能力，称之为内在评测（Intrinsic Evaluation）。第二类方法通过某些具体任务，如机器翻译、摘要生成等，来评测语言模型处理这些具体生成任务的能力，称之为外在评测（Extrinsic Evaluation）。

### 1.5.1 内在评测

在内在评测中，测试文本通常由与预训练中所用的文本独立同分布的文本构成，**不依赖于具体任务**。最为常用的内部评测指标是困惑度（Perplexity）[10]。其度量了语言模型对测试文本感到“困惑”的程度。设测试文本为  $s_{test} = w_{1:N}$ 。语言模型在测试文本  $s_{test}$  上的困惑度  $PPL$  可由下式计算：

$$PPL(s_{test}) = P(w_{1:N})^{-\frac{1}{N}} = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i|w_{<i})}}。 \quad (1.50)$$

由上式可以看出，如果语言模型对测试文本越“肯定”（即生成测试文本的概率越高），则困惑度的值越小。而语言模型对测试文本越“不确定”（即生成测试文本的概率越低），则困惑度的值越大。由于测试文本和预训练文本同分布，预训练文本代表了我们要让语言模型学会生成的文本，如果语言模型在这些测试文本上越不“困惑”，则说明语言模型越符合我们对其训练的初衷。因此，困惑度可以一定程度上衡量语言模型的生成能力。

对困惑度进行改写，其可以改写成如下等价形式。

$$PPL(s_{test}) = \exp\left(-\frac{1}{N} \sum_{i=1}^N \log P(w_i|w_{<i})\right)。 \quad (1.51)$$

其中， $-\frac{1}{N} \sum_{i=1}^N \log P(w_i|w_{<i})$  可以看作是生成模型生成的词分布与测试样本真实

的词分布间的交叉熵，即  $-\frac{1}{N} \sum_{i=1}^N \sum_{d=1}^{|D|} I(\hat{w}_d = w_i) \log o_{i-1}[w_i]$ ，其中  $D$  为语言模型所采用的词典。因为  $P(w_i|w_{<i}) \leq 1$ ，所以此交叉熵是生成模型生成的词分布的信息熵的上界，即

$$-\frac{1}{N} \sum_{i=1}^N P(w_i|w_{<i}) \log P(w_i|w_{<i}) \leq -\frac{1}{N} \sum_{i=1}^N \log P(w_i|w_{<i}). \quad (1.52)$$

因此，困惑度减小也意味着熵减，意味着模型“胡言乱语”的可能性降低。

## 1.5.2 外在评测

在外在评测中，测试文本通常包括该任务上的问题和对应的标准答案，其依赖于具体任务。通过外在评测，我们可以评判语言模型处理特定任务的能力。外在评测方法通常可以分为基于统计指标的评测方法和基于语言模型的评测方法两类。以下对此两类方法中的经典方法进行介绍。

### 1. 基于统计指标的评测

基于统计指标的方法构造统计指标来评测语言模型的输出与标准答案间的契合程度，并以此作为评测语言模型生成能力的依据。BLEU (BiLingual Evaluation Understudy) 和 ROUGE (Recall-Oriented Understudy for Gisting Evaluation) 是应用最为广泛的两种统计指标。其中，BLEU 是精度导向的指标，而 ROUGE 是召回导向的指标。以下分别对这两个指标展开介绍。

BLEU 被提出用于评价模型在机器翻译 (Machine Translation, MT) 任务上的效果 [6]。其在词级别上计算生成的翻译与参考翻译间的重合程度。具体地，BLEU 计算多层次  $n$ -gram 精度的几何平均。设生成的翻译文本的集合为  $S_{gen} = \{S_{gen}^i\}_{i=1}^{|S_{gen}|}$ ，对应的参考翻译集合为  $S_{ref} = \{S_{ref}^i\}_{i=1}^{|S_{ref}|}$ ，其中， $S_{gen}^i$  与  $S_{ref}^i$  一一对应，且  $|S_{ref}| = |S_{gen}|$ 。原始的  $n$ -gram 精度的定义如下：

$$Pr(g_n) = \frac{\sum_{i=1}^{|S_{gen}|} \sum_{g_n \in S_{gen}^i} Count_{match}(g_n, S_{ref}^i)}{\sum_{i=1}^{|S_{gen}|} \sum_{g_n \in S_{gen}^i} Count(g_n)}. \quad (1.53)$$



其中,  $g_n$  代表  $n$ -gram。上式的分子计算了生成的翻译与参考翻译的重合的  $n$ -gram 的个数, 分母计算了生成的翻译中包含的  $n$ -gram 的总数。例如, MT 模型将“大语言模型”翻译成英文, 生成的翻译为“big language models”, 而参考文本为“large language models”。当  $n = 1$  时,  $Pr(g_1) = \frac{2}{3}$ 。当  $n = 2$  时,  $Pr(g_2) = \frac{1}{2}$ 。

基于  $n$ -gram 精度, BLEU 取  $N$  个  $n$ -gram 精度的几何平均作为评测结果:

$$BLEU = \sqrt[N]{\prod_{i=1}^N Pr(g_n)} = \exp \left( \frac{1}{N} \sum_{n=1}^N \log Pr(g_n) \right) \quad (1.54)$$

例如, 当  $N = 3$  时, BLEU 是 unigram 精度, bigram 精度, trigram 精度的几何平均。在以上原始 BLEU 的基础上, 我们还可以通过不同的  $n$ -gram 精度进行加权或对不同的文本长度设置惩罚项来对 BLEU 进行调整, 从而得到更为贴近人类评测的结果。

ROUGE 被提出用于评价模型在摘要生成 (Summarization) 任务上的效果 [13]。常用的 ROUGE 评测包含 ROUGE-N, ROUGE-L, ROUGE-W, 和 ROUGE-S 四种。其中, ROUGE-N 是基于  $n$ -gram 的召回指标, ROUGE-L 是基于最长公共子序列 (Longest Common Subsequence, LCS) 的召回指标。ROUGE-W 是在 ROUGE-L 的基础上, 引入对 LCS 的加权操作后的召回指标。ROUGE-S 是基于 Skip-bigram 的召回指标。下面给出 ROUGE-N, ROUGE-L 的定义。ROUGE-W 和 ROUGE-S 的具体计算方法可在 [13] 中找到。

ROUGE-N 的定义如下:

$$ROUGE-N = \frac{\sum_{s \in S_{ref}} \sum_{g_n \in s} Count_{match}(g_n, s_{gen})}{\sum_{s \in S_{ref}} \sum_{g_n \in s} Count(g_n)} \quad (1.55)$$

ROUGE-L 的定义如下:

$$ROUGE-L = \frac{(1 + \beta^2) R_r^g R_g^r}{R_r^g + \beta^2 R_g^r} \quad (1.56)$$



其中,

$$R_r^g = \frac{LCS(s_{ref}, s_{gen})}{|s_{ref}|}, \quad (1.57)$$

$$R_g^r = \frac{LCS(s_{ref}, s_{gen})}{|s_{gen}|}, \quad (1.58)$$

$LCS(s_{ref}, s_{gen})$  是模型生成的摘要  $s_{gen}$  与参考摘要  $s_{ref}$  间的最大公共子序列的长度,  $\beta = R_g^r / R_r^g$ 。

基于统计指标的评测方法通过对语言模型生成的答案和标准答案间的重叠程度进行评分。这样的评分无法完全适应生成任务中表达的多样性, 与人类的评测相差甚远, 尤其是在生成的样本具有较强的创造性和多样性的时候。为解决此问题, 可以在评测中引入一个其他语言模型作为“裁判”, 利用此“裁判”在预训练阶段掌握的能力对生成的文本进行评测。下面对这种引入“裁判”语言模型的评测方法进行介绍。

## 2. 基于语言模型的评测

目前基于语言模型的评测方法主要分为两类: (1) 基于上下文词嵌入 (Contextual Embeddings) 的评测方法; (2) 基于生成模型的评测方法。典型的基于上下文词嵌入的评测方法是 BERTScore [24]。典型的基于生成模型的评测方法是 G-EVAL [14]。与 BERTScore 相比, G-EVAL 无需人类标注的参考答案。这使其可以更好的适应到缺乏人类标注的任务中。

BERTScore 在 BERT 的上下文词嵌入向量的基础上, 计算生成文本  $s_{gen}$  和参考文本  $s_{ref}$  间的相似度来对生成样本进行评测。BERT 将在第二章给出详细介绍。设生成文本包含  $|s_{gen}|$  个词, 即  $s_{gen} = \{w_g^i\}_{i=1}^{|s_{gen}|}$ 。设参考文本包含  $|s_{ref}|$  个词, 即  $s_{ref} = \{w_r^i\}_{i=1}^{|s_{ref}|}$ 。利用 BERT 分别得到  $s_{gen}$  和  $s_{ref}$  中每个词的上下文词嵌入向量, 即  $v_g^i = BERT(w_g^i | s_{gen})$ ,  $v_r^i = BERT(w_r^i | s_{ref})$ 。利用生成文本和参考文本的词嵌入向量集合  $v_{gen} = \{v_g^i\}_{i=1}^{|v_{gen}|}$  和  $v_{ref} = \{v_r^i\}_{i=1}^{|v_{ref}|}$  便可计算 BERTScore。BERTScore

从精度 (Precision), 召回 (Recall) 和 F1 量度三个方面对生成文档进行评测。其定义分别如下:

$$P_{BERT} = \frac{1}{|v_{gen}|} \sum_{i=1}^{|v_{gen}|} \max_{v_r \in v_{ref}} v_g^i \top v_r^i, \quad (1.59)$$

$$R_{BERT} = \frac{1}{|v_{ref}|} \sum_{i=1}^{|v_{ref}|} \max_{v_g \in v_{gen}} v_r^i \top v_g^i, \quad (1.60)$$

$$F_{BERT} = \frac{2P_{BERT} \cdot R_{BERT}}{P_{BERT} + R_{BERT}}. \quad (1.61)$$

相较于统计评测指标, BERTScore 更接近人类评测结果。但是, BERTScore 依赖于人类给出的参考文本。这使其无法应用于缺乏人类标注样本的场景中。得益于生成式大语言模型的发展, G-EVAL 利用 GPT-4 在没有参考文本的情况下对生成文本进行评分。G-EVAL 通过提示工程 (Prompt Engineering) 引导 GPT-4 输出评测分数。Prompt Engineering 将在本书第三章进行详细讲解。

如下图所示, G-EVAL 的 Prompt 分为三部分: (1) 任务描述与评分标准; (2) 评测步骤; (3) 输入文本与生成的文本。在第一部分中, 任务描述指明需要的评测的任务式什么 (如摘要生成), 评分标准给出评分需要的范围, 评分需要考虑的因素等内容。第二部分的评测步骤是在第一部分内容的基础上由 GPT-4 自己生成的思维链 (Chain-of-Thoughts, CoT)。本书的第三章将对思维链进行详细讲解。第三部分的输入文本与生成的文本是源文本和待评测模型生成的文本。例如摘要生成任务中的输入文本是原文, 而生成的文本就是生成摘要。将上述三部分组合在一个 prompt 里面然后输入给 GPT-4, GPT-4 便可给出对应的评分。直接将 GPT-4 给出的得分作为评分会出现区分度不够的问题, 因此, G-EVAL 还引入了对所有可能得分进行加权平均的机制来进行改进 [14]。

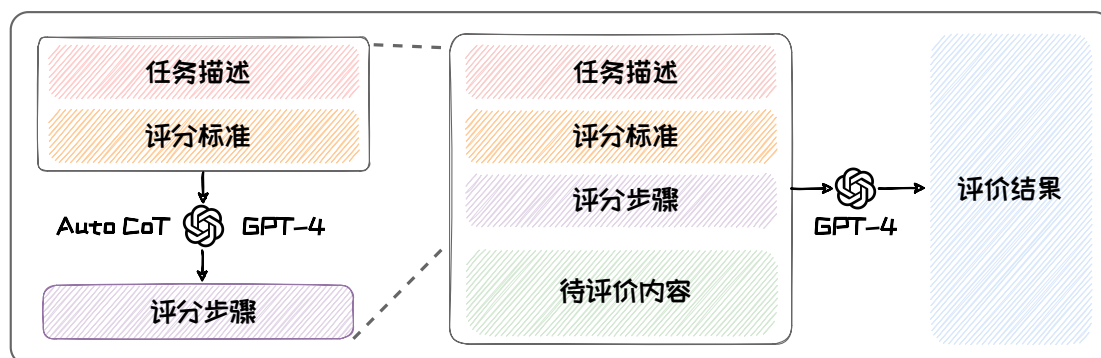


图 1.10: G-EVAL 评测流程。

除 G-EVAL 外，近期还有多种基于生成模型的评测方法被提出 [12]。其中典型的有 InstructScore [23]，其除了给出数值的评分，还可以给出对该得分的解释。基于生成模型的评测方法相较于基于统计指标的方法和基于上下文词嵌入的评测方法而言，在准确性、灵活性、可解释性等方面都具有独到的优势。可以预见，未来基于生成模型的评测方法将得到更为广泛的关注和应用。

## 参考文献

- [1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450.
- [2] Samy Bengio et al. “Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks”. In: *NeurIPS*. 2015.
- [3] Tom B. Brown et al. “Language Models are Few-Shot Learners”. In: *NeurIPS*. 2020.
- [4] Junyoung Chung et al. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. 2014. arXiv: 1412.3555.
- [5] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *NAACL-HLT*. 2019.
- [6] Markus Freitag, David Grangier, and Isaac Caswell. “BLEU might be Guilty but References are not Innocent”. In: *EMNLP*. 2020.
- [7] Mor Geva et al. “Transformer Feed-Forward Layers Are Key-Value Memories”. In: *EMNLP*. 2021.

- [8] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computing* 9.8 (1997), pp. 1735–1780.
- [9] Ari Holtzman et al. “The Curious Case of Neural Text Degeneration”. In: *ICLR*. 2020.
- [10] F. Jelinek et al. “Perplexity—a measure of the difficulty of speech recognition tasks”. In: *The Journal of the Acoustical Society of America* 62 (1997), S63–S63.
- [11] Dan Jurafsky and James H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd Edition*. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International, 2009. ISBN: 9780135041963.
- [12] Zhen Li et al. *Leveraging Large Language Models for NLG Evaluation: Advances and Challenges*. 2024. arXiv: [2401.07103](#).
- [13] Chin-Yew Lin. “Rouge: A package for automatic evaluation of summaries”. In: *ACL*. 2004.
- [14] Yang Liu et al. “GpTeval: Nlg evaluation using gpt-4 with better human alignment”. In: *EMNLP*. 2023.
- [15] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, 2001. ISBN: 978-0-262-13360-9.
- [16] OpenAI. *GPT-4 Technical Report*. 2024. arXiv: [2303.08774](#).
- [17] Razvan Pascanu, Tomás Mikolov, and Yoshua Bengio. “On the difficulty of training recurrent neural networks”. In: *ICML*. 2013.
- [18] Colin Raffel et al. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *Journal of Machine Learning Research* 21 (2020), 140:1–140:67.
- [19] Ashwin K. Vijayakumar et al. “Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models”. In: *AAAI*. 2018.
- [20] Joseph Weizenbaum. “ELIZA - a computer program for the study of natural language communication between man and machine”. In: *Communications Of The ACM* 9.1 (1966), pp. 36–45.
- [21] Ronald J. Williams and David Zipser. “A Learning Algorithm for Continually Running Fully Recurrent Neural Networks”. In: *Neural Computing* 1.2 (1989), pp. 270–280.

- [22] Shufang Xie et al. *ResiDual: Transformer with Dual Residual Connections*. 2023. arXiv: [2304.14802](#).
- [23] Wenda Xu et al. “INSTRUCTSCORE: Towards Explainable Text Generation Evaluation with Automatic Feedback”. In: *EMNLP*. 2023.
- [24] Tianyi Zhang et al. “BERTScore: Evaluating Text Generation with BERT”. In: *ICLR*. 2020.

