

Transformer架构与训练笔记

第二版 · 2026-10-17（北京时间）

这一版从你的问题出发讲：一句话进去以后，里面究竟发生了什么？这些计算为什么能用来预测？预测错了，又怎样改变模型？公式保留，但每条公式都接着解释它正在做的事。

怎么读。 可以先顺着第 1—12 章读，不必背符号。第 10 章把一次计算和训练走到底；第 13 章放较长的数学推导，需要核查“为什么成立”时再回来。第 14 章可以按你的原始提问找位置。

文中的分词、注意力比例和小模型参数都是为解释而设置的。提到“华法林”和“黑便”，是沿用你的语言例子，不是实际模型的测量结果，也不用于临床判断。

正文目录

第 1 章：我们究竟要让模型做什么	3
第 2 章：输入明明是一句话，为什么会变成向量	5
第 3 章：Q、K、V 是怎样来的，各自做什么	7
第 4 章：点积怎样变成关注比例	10
第 5 章：算出 z 以后，到底得到了什么	13
第 6 章：整句都输入了，训练为什么没有偷看答案	16
第 7 章：还缺一个把向量变成预测的步骤	19
第 8 章：预测得好不好，怎样变成一个数	22
第 9 章：模型怎样知道该改哪些数字	24
第 10 章：把一次预测和训练真的算完	26
第 11 章：训练完以后，它怎样一句句输出	33
第 12 章：这些计算为什么值得这样组织	35
第 13 章：想追到底时，再看这些数学推导	37
第 14 章：按你的问题回查，以及阅读来源	46

第 1 章：我们究竟要让模型做什么

[返回目录](#)

1.1 先不看架构，只看一个填空

我们沿用你对话里的句子：

患者服用华法林后出现____

先把模型的任务限定得很具体：**看见已经给出的前文，给后面可能出现的文字打分。**

例如，它可能给“黑便”较高的分数，也可能给“头晕”较高的分数。暂时不要问这些分数从哪里来，只记住任务的起点与终点：进去的是前文，出来的是一组候选分数。

这些分数还会变成概率。譬如“黑便”30%、“头晕”20%，其余候选共50%。这组数不是对患者病情的概率判断，而是**模型对下一段文字的预测分配**。

实际模型通常预测的是一个 Token，而非一个完整词语。Token 可以是一个汉字、几个字、词的一部分或其他文本片段。为了把机制讲清楚，本文先假设分词如下：

患者 | 服用 | 华法林 | 后 | 出现 | 黑便 | 。

这是人为指定的教学分词，不声称真实模型一定这样切。

1.2 训练和输出，其实在这里分开

假设训练文本已经写好了整句话“患者服用华法林后出现黑便。”。训练程序知道“出现”后面真实写的是“黑便”。模型给出概率后，我们可以检查它给这个目标分了多少概率，再调整模型。

如果是生成，输入只有“患者服用华法林后出现”，后面还没提供。模型算出同样类型的概率，然后选出一个 Token，把它接在后面，继续算。

所以两种情况都有“根据前文算预测”这一段。区别主要在预测之后：

- **训练：** 有数据提供的答案，利用预测误差调整模型。
- **生成：** 选择一个候选，把它放进后续前文，模型参数通常保持不变。

“训练”不是直接把“华法林和黑便有关”这一句话塞进某个矩阵。我们先让模型算出预测，再根据预测表现改变其中的数字。怎样改，第 9—10 章会展开。

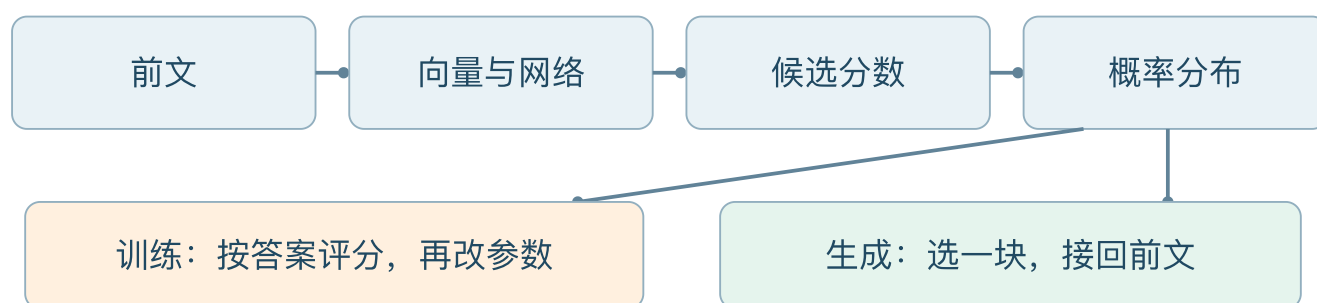


图1：前向预测相同，概率之后的用途不同。

1.3 为什么还要讲向量、矩阵和 Attention

因为计算机不能直接对“患者”两个字做有用的矩阵乘法。我们要先给每个 Token 一串数字，再让这些数字相互影响，最后变成候选文字的分数。

整个过程可以先读成：

把前文变成数字 → 让不同位置交换信息 → 多次加工 → 给候选 Token 打分 → 得到概率。

其中“让不同位置交换信息”主要由注意力机制完成。Attention 是 Transformer 里面的重要模块，但不是整个模型。

这一版先研究 GPT 类模型用于续写的基本计算方式。它能解释基础训练和生成过程，但完整聊天系统还有消息格式、指令训练等部分，后面会说明区别。

第 2 章：输入明明是一句话，为什么会变成向量

[返回目录](#)

2.1 每个 Token 先有一个编号

可以想象有一本固定目录，里面列着模型能处理的所有 Token。这本目录叫**词表**。

Tokenizer 按规则把输入切成 Token，再查出对应编号。

比如，在某个假想词表中，“患者”编号17，“服用”编号92。编号只是用来查找，92减17不表示这两个词之间的语义差距。

编号查到以后，再去一张“向量表”里拿出一行数字。例如：

$$\boldsymbol{x}_{\text{患者}} = (0.2, -0.5, 0.8).$$

这里三个数只是示意。真正模型的每个向量通常有更多坐标。“向量”在这里就是一组有顺序的数，可以进行点积、相加和矩阵变换。

这张可训练的向量表叫 **Embedding**，**嵌入表**。

2.2 为什么要查出一串数字

我们希望模型能把不同 Token 放进同一种可计算的表示中。查出向量以后，模型就能用同一套运算处理“患者”“服用”“华法林”等内容，而不用为每个词单独写一套程序。

这些坐标没有预先规定“第一维是药物，第二维是症状”。它们最初只是一些数，通过训练逐渐变得适合后续计算使用。

所以“词语有了向量”不等于“每一维都能翻译成一个明确的人类概念”。先把它理解成后面各层能读取和加工的数值表示即可。

2.3 一句话有几块，就有几行

现在只输入前五块：

患者 | 服用 | 华法林 | 后 | 出现

按顺序查出五个向量，记为：

$$\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5.$$

这里下标表示在这次输入里的位置： $\boldsymbol{x}_3$ 对应第三块“华法林”，不是某个向量的第三个坐标。

把五个向量按行排好：

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \boldsymbol{x}_3 \\ \boldsymbol{x}_4 \\ \boldsymbol{x}_5 \end{bmatrix}.$$

大写 $\boldsymbol{X}$ 是整句话的矩阵，小写 $\boldsymbol{x}_i$ 是其中一行。如果每行有 d 个数，那么 $\boldsymbol{X}$ 的形状就是“5行、 d 列”，写成 $\boldsymbol{X} \in \mathbb{R}^{5 \times d}$ 。这个写法只表示它是一张5乘 d 的实数表，不额外增加某种神秘含义。

模型处理的是整个 $\boldsymbol{X}$ 。 前面只盯着“华法林”或“出现”讲，是为了放大其中一行的计算，不是说模型一次只收到一个词。

2.4 位置也要告诉模型

“医生治疗患者”和“患者治疗医生”包含相同词语，顺序却改变了意思。模型需要获得位置方面的信息。

一种直接做法是给每个位置一个向量，加到 Token 向量上。如果词向量叫 e_i ，位置向量叫 p_i ，就写：

$$x_i = e_i + p_i.$$

两个向量同样长，按坐标相加。这样，即使同一个 Token 出现在不同位置，输入表示也可以不同。

不是所有模型都采用这种加法。第 13 章会解释另一种常见方式 RoPE。现在只要记住：**内容要表示，顺序也要表示。**

2.5 这些输入向量由谁决定

在通常的全参数训练中，嵌入表里的数也会更新。也就是说，不只是后面的注意力模块在学，输入如何表示，也跟着学习。

但在一次生成过程中，这张表通常不改。你输入不同内容，查到的行会不同；这是使用表，不是重新训练表。

第 3 章：Q、K、V 是怎样来的，各自做什么

[返回目录](#)

3.1 先弄清为什么需要读取前文

我们要在“患者服用华法林后出现”之后预测下一项。仅看“出现”这两个字，信息不够；前面的“患者”“服用”“华法林”也可能影响后面该写什么。

因此，我们希望“出现”这个位置的表示，能利用前文，而不是永远只表示“出现”自身。

注意力做的事情可以先读成：

在当前允许看的位置里，计算各处应占多少比例，再按这些比例取来内容，形成一个新向量。

这里有两件不同的事：一是定比例，二是取内容。这就是 Q、K、V 分工的起点。

3.2 为什么一个向量要变成三份

同一个输入向量 x_i ，分别乘三个可训练矩阵：

$$q_i = x_i W_Q, \quad k_i = x_i W_K, \quad v_i = x_i W_V.$$

先不要急着记字母。三个 W 是三个不同的数值变换规则；同一份输入经过不同规则，得到三种用途的表示：

- **q, Query**：当前这个位置用什么特征去寻找需要的信息。
- **k, Key**：其他位置用什么特征接受匹配。
- **v, Value**：如果读取这个位置，真正拿来组合的内容向量。

“提问、索引、内容”是帮助理解的类比。q里不是一句文字问题，k也不是一个人工数据库字段；它们仍然是一串数，角色由后续运算决定。

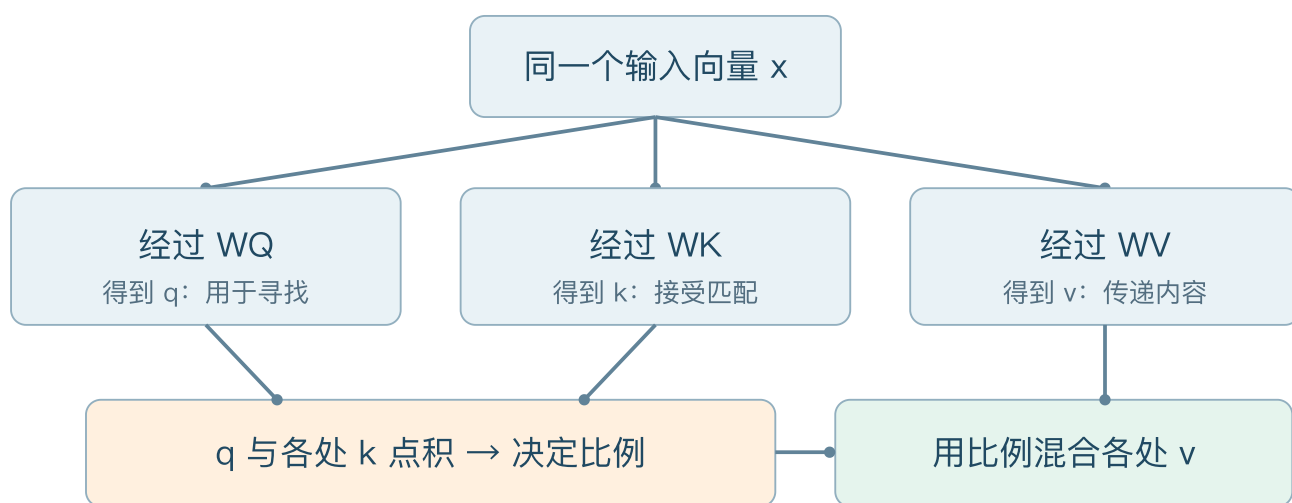


图2：参数矩阵是规则，q/k/v是本次输入经过规则算出的结果。

3.3 WQ 与 q，究竟哪个是模型学的

你原来问过：“这一步到底是在算 WQ 还是 q？”

用一个特别小的计算区分它们。假设：

$$x_i = (2, 1), \quad W_Q = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}.$$

那么：

$$q_i = x_i W_Q = (2, 3).$$

这一行只是用已有 WQ 计算当前 q。输入变成 (1, 4)，即使 WQ 不变，算出的 q 也会变成 (1, 12)。

所以：**WQ是模型可训练的参数，q是这次输入产生的结果。** 训练时，模型根据最终损失调整 WQ；生成时使用固定的 WQ，针对当前输入重新计算 q。

同样，K/V的参数矩阵与这次算出的k/v也要区分。稍后算注意力比例时，既不在求新的 WQ，也不在重新定义 q，而是在使用已经得到的 q和k。

3.4 q怎样与k匹配

假设当前位置的 q 是 (1, 0)，三个可见位置的 k分别为：

$$k_1 = (1, 0), \quad k_2 = (0, 1), \quad k_3 = (-1, 0).$$

逐个点积得到：

$$q \cdot k_1 = 1, \quad q \cdot k_2 = 0, \quad q \cdot k_3 = -1.$$

现在只有三个分数：(1, 0, -1)。第一项最高，第三项最低。这个步骤尚未读取任何v，也没有得到最终文字概率。

实际模型中，这套匹配是否对预测有帮助，要靠训练。点积本身不会自动知道哪些医学概念有关。它只提供一种可学习的打分方式。

如果写成 $s_{ij} = q_i \cdot k_j$ ， i 表示“谁正在读取”， j 表示“正在考虑从哪里读取”。例如 $s_{5,3}$ 是“出现”位置的 q 与“华法林”位置的 k 之间的匹配分数。

3.5 为什么不直接用原来的 x 点积

可以这样设计，但可调整的余地较少。分别学习 WQ 和 WK ，模型就能学习“按什么特征寻找”和“按什么特征被找到”。这两个方向不必相同。

例如“出现”需要读取“华法林”的信息，不要求反过来“华法林”也以相同比例读取“出现”。不同的投影让匹配具有方向性。更具体的矩阵依据放在第 13 章。

同理，也可以让 K 和 V 使用同一表示，但标准注意力把它们分开，允许“用于匹配的特征”和“传出去的内容”分别学习。这里是设计理由，不是说数学上只能有这一种结构。

第 4 章：点积怎样变成关注比例

[返回目录](#)

4.1 分数还不能直接当比例

上一章得到 $(1, 0, -1)$ 。如果直接把它们拿来加权，就会出现负系数，而且总和为0，无法理解为“这一份占多少、那一份占多少”的比例。

我们希望把这些分数变成非负且总和为1的一组数，同时让高分得到更高比例。常用做法叫 **Softmax**，可以拆成两步。

先分别取指数：

$$(e^1, e^0, e^{-1}) \approx (2.718, 1, 0.368).$$

三个数都正。再分别除以总和4.086：

$$\left(\frac{2.718}{4.086}, \frac{1}{4.086}, \frac{0.368}{4.086} \right) \approx (0.665, 0.245, 0.090).$$

现在三个比例都非负，总和为1，而且顺序仍与原分数一致。也就是约66.5%、24.5%、9.0%。

4.2 一般公式只是把刚才的操作缩写

如果当前候选位置的分数是 $s_1, \dots, s_m$ ，则第j个位置的比例写成：

$$a_j = \frac{e^{s_j}}{\sum_{r=1}^m e^{s_r}}.$$

这里m是当前考虑的位置数量。下方的求和符号表示把m个指数结果全部相加。分子拿第j项，分母拿全部项的总和。

为什么总和为1？因为把所有分子加起来，恰好等于这个公共分母：

$$\sum_{j=1}^m a_j = \frac{\sum_{j=1}^m e^{s_j}}{\sum_{r=1}^m e^{s_r}} = 1.$$

为什么较大的分数比例也较大？指数函数严格递增，而分母对各项相同。更精确地，两项比例之比是 $a_j/a_k = e^{s_j-s_k}$ ，因此Softmax看的是候选之间的分数差。

这套性质让Softmax很适合这里，但不能据此证明它是唯一可能的设计。

4.3 为什么还要除以根号dk

原对话里最容易把主线打断的就是这一项。先把符号说清： d_k 是每个q和k有多少个坐标。q与k要点积，所以二者必须同样长。

若每个q和k都有64个坐标，点积就是64对数分别相乘，再相加。此时 $d_k = 64$ 。它不是句子有64个词，也不是词表有64项。

原始分数通常要缩放成：

$$s_{ij} = \frac{q_i \cdot k_j}{\sqrt{d_k}}.$$

64维时就是除以8。目的在于让送进Softmax的分数尺度更适当。分数之间的差如果非常大，比例就容易挤成“几乎全给一项”，这种状态可能使注意力分配不容易通过梯度调整。

为什么是8而不是64？在各坐标独立、均值0、方差1的理想化条件下，64项乘积之和的方差是64，标准差是8。缩放针对的是这种典型波动尺度，而不是最大可能值。

这个说法依赖假设，不是说真实训练后的点积一定等于8，也不是除以8后数值一定落在某个固定范围。方差的完整推导和你问过的“为什么可写成平方的期望”，都保留在第13章。

4.4 现在才乘 V

假设某次计算得到三个比例：

$$a = (0.5, 0.25, 0.25).$$

到这里，我们只有“各处占多大比例”，还没有新的内容向量。再给三个位置的内容：

$$v_1 = (0, 0), \quad v_2 = (4, 0), \quad v_3 = (0, 4).$$

按比例取来并相加：

$$z = 0.5v_1 + 0.25v_2 + 0.25v_3 = (1, 1).$$

这就是“为什么又乘V”的答案：前面的计算决定怎样混合，V提供被混合的东西。

假如只有比例 $(0.5, 0.25, 0.25)$ ，我们无法知道结果是 $(1, 1)$ 还是别的向量；必须知道各处v的具体数值。

4.5 这与凸组合有什么关系

你已经学过：系数非负且总和为1的线性组合，就是凸组合。因此这里 z 属于这些 v 组成的凸包。若把三个 v 画成平面上的点， z 就在它们围出的三角形之内。

这个结论严格适用于当前这种归一化后的加权结果。它说的是单个注意力头此刻对Value的组合，不是说整个Transformer永远只能在最初词向量的凸包里移动。

另外，真实训练中可能在注意力系数上加dropout；若随机丢掉部分系数后不重新归一化，当次输出就不能直接沿用“系数和为1”的结论。本文手算部分不使用dropout。

第 5 章：算出 z 以后，到底得到了什么

[返回目录](#)

5.1 z 不是一个词，它是后面要用的表示

你问过：“ x 输出的 z 有什么意义吗？”这比单纯会算公式更关键。

假设“出现”位置读到了前面各处的信息，形成 z 。 z 不会直接写成“黑便”，也不一定存在一维叫“出血”。它仍然只是一串数字，只是这串数字已经受到其他位置内容的影响。

它的用处在于：后面的网络可以利用它，改变对下一Token的打分。

可以比较“我吃了苹果”和“我投资了苹果”。即使“苹果”查出的初始词向量相同，它前面可读取的内容不同，经过注意力后，得到的表示可以不同。后面就有机会针对不同语境产生不同预测。

这里提示词“吃了”“投资了”都放在“苹果”之前。因果语言模型不能让“苹果”位置提前利用后面还未出现的词。

5.2 一整句会得到一整组 z

对五个输入位置，每个位置都可以有自己的读取比例。因此不止有一个 z ，而是：

$$z_1, z_2, z_3, z_4, z_5.$$

把它们排成矩阵 Z ， Z 的第 i 行就是当前位置读回来的内容。不是整句突然只剩一个向量，也不是每个位置都使用同一组比例。

用大写字母压缩整个操作时：

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V,$$

$$A = \text{softmax}_{\text{row}}(QK^T / \sqrt{d_k} + M), \quad Z = AV.$$

这里的 A 是一张比例表：第 i 行表示“第 i 个位置从各处读多少”。 row 表示每行单独做Softmax。 K^T 是把 K 转置，让矩阵乘法一次算出所有 q 与所有 k 的点积。 M 控制哪些位置不允许读取，下一章解释。

最后 $Z=AV$ 只是把上一章“按比例乘 v 再相加”的操作，同时对每一行完成。它没有引入新的神秘步骤。

5.3 多头：不只用一张比例表

一个位置可能需要以不同方式组合信息。多头注意力让几组不同的 $Q/K/V$ 投影分别工作，因此同一位置可以同时得到几份不同的加权结果。

例如一份结果偏向某些位置，另一份偏向其他位置。然后把这些结果拼在一起，再用一个矩阵 W_O 混合，变回模型需要的向量长度。

如果两个头各给出 $(1, 2)$ 与 $(3, 4)$ ，拼接就是 $(1, 2, 3, 4)$ 。随后乘 W_O 可以把四个数混合回两个数，或其他指定长度。

不能据此认定真实模型里一定有一个“药物头”和一个“症状头”。多头给了多种处理方式的空间，实际分工由训练产生，也未必这么容易解释。

5.4 为什么要把结果加回原输入

注意力读到的信息，是对当前位置表示的一次补充。常见做法是先投影，再与原输入相加。把投影后的结果暂记为 o ，更新形式就是：

$$x_{\text{new}} = x + o.$$

这叫**残差连接**。比如原来是 $(1, 2)$ ，新读回的信息是 $(0.3, -0.2)$ ，相加成为 $(1.3, 1.8)$ 。

所以 z 并不需要独自承担当前位置的一切信息；还有原输入的直连路径。这里“补充”是帮助理解，不保证每个坐标相加都让信息更完整。如果 o 恰好等于 $-x$ ，相加也会变成零。

5.5 FFN：读完以后，还要加工

把不同位置的信息搬到一起，并不等于已经完成了有用处理。我们还要根据这些特征做进一步变换。每个位置会经过同一套前馈网络，简称FFN。

可以先把它看成：

混合各坐标 → 经过非线性函数 → 再混合各坐标。

写成行向量公式是：

$$\text{FFN}(x) = \sigma(xW_1 + b_1)W_2 + b_2.$$

W_1 、 W_2 是可训练矩阵， b_1 、 b_2 是可训练偏置；偏置就是变换后额外加的一组数。 σ 是逐坐标作用的非线性函数。例如ReLU把负数改成0，保留正数。

为什么要有非线性？如果只有固定矩阵连续相乘，可以合成一个矩阵；加入非线性后，一般就不能这样合并，能表达更复杂的关系。但注意力本身通常已经非线性，不能说“没有FFN整个模型就线性了”。

FFN虽然逐位置运算，它接收的向量已经包含前文信息，所以并不是回到“只认识孤立词语”的状态。

5.6 归一化与重复多层

完整模块里还常有归一化。以LayerNorm为例，它在同一位置的一串坐标里，先计算均值和波动尺度，再做平移、缩放，并允许学习额外的缩放与偏移。

例如 $(1, 2, 3)$ 的均值为2，先减去均值得到 $(-1, 0, 1)$ ，再按它们的波动尺度缩放。目的在于组织数值尺度，帮助后续计算与训练。它不会去取整句未来位置的均值；否则就会破坏我们想要的因果约束。

将“注意力、残差、FFN、归一化”按指定顺序组合，就得到一个Transformer Block。再把这样的Block重复多层。后一层接收的不是最初查表的向量，而是前一层已经加工过的表示。

各层一般有各自的参数；同一层的参数则在不同位置共享。经过最后一层后，我们把当前位置的最终表示叫 h 。于是你见到的 x 、 z 、 h 可以这样区分：

- x ：当前模块收到的表示。
- z ：某个注意力头按比例读回的结果。
- h ：经过整个网络后，用于预测的最终表示。

它们的区别主要在计算中的位置，不是三个天然带有人类语义标签的空间。

第 6 章：整句都输入了，训练为什么没有偷看答案

[返回目录](#)

6.1 先把训练目标对齐

现在训练数据有完整七块：

患者 | 服用 | 华法林 | 后 | 出现 | 黑便 | 。

第1位置用“患者”预测“服用”；第2位置用前两块预测“华法林”；依此类推。每个位置预测的是**后面一块**，不是复制当前位置。

当前用哪个位置的最终表示	允许使用的前文	要预测什么
第1位置	患者	服用
第2位置	患者、服用	华法林
第3位置	到华法林为止	后
第4位置	到后为止	出现
第5位置	到出现为止	黑便
第6位置	到黑便为止	。

所以要预测“黑便”，用的是“出现”位置最终算出的h5。这里5是输入位置编号，不是要预测“第5个词”。h5负责的是第6块。

6.2 不限制的话，确实会泄漏

你原来问：“没有M的话，理论上每一个词都会联系其他的词吧？”

是的，如果不设任何可见性限制，标准全注意力允许每个位置读取其他所有位置。这样训练时“出现”位置可以直接读到后面的“黑便”，就不再是在练习只凭前文预测。

为了阻止这种情况，每层都只允许位置i读取自己及之前的位置，禁止读取i之后的位置。这叫**因果掩码**。

“因果”在这里指前后依赖方向，不是说注意力已经发现了医学因果关系。

6.3 M怎样把不该看的位置关掉

做Softmax之前，允许位置的分数照常保留，不允许位置的分数视为负无穷。因为指数在负无穷处的极限是0，禁止位置便拿不到权重。

常把这种操作记成加矩阵M：

$$M_{ij} = \begin{cases} 0, & j \leq i, \\ -\infty, & j > i. \end{cases}$$

还是一样：i是“谁在读”，j是“从哪里读”。第5行只能在前5列分比例，第6、7列为0。

严格说，不必把负无穷当成普通实数去运算。直接定义“只在允许集合里做Softmax，其他位置设为0”即可；上述写法是常用简记。第13章给出准确公式。

	患者	服用	华法林	后	出现	黑便	。
患者	可见	屏蔽	屏蔽	屏蔽	屏蔽	屏蔽	屏蔽
服用	可见	可见	屏蔽	屏蔽	屏蔽	屏蔽	屏蔽
华法林	可见	可见	可见	屏蔽	屏蔽	屏蔽	屏蔽
后	可见	可见	可见	可见	屏蔽	屏蔽	屏蔽
出现	可见	可见	可见	可见	可见	屏蔽	屏蔽
黑便	可见	可见	可见	可见	可见	可见	屏蔽
。	可见	可见	可见	可见	可见	可见	可见

图3：每行表示当前位置允许读取哪里。橙框“出现”行不能读取答案“黑便”。这里没有展示注意力大小。

6.4 会不会从别的位置绕路偷看

如果每一层都遵守限制，就不会。

想象第5位置从第3位置读信息。第3位置上一层也只能读前3个位置，因此它不会带来第6位置的内容。再往前追，每条路径都只能停在当前位置或向前，不能绕到未来。

这就是一个逐层归纳：最初各位置只有自己的输入；假设上一层没有未来信息，下一层只读不晚于自己的位置，也不会获得未来信息。逐位置的FFN、残差和LayerNorm不打破这条约束。

所以“训练程序拥有答案”和“生成这个预测的计算能读取答案”是两回事。前者必须知道答案才能评分，后者必须不能提前偷看。

6.5 为什么训练能一次算多个位置

整段文字已经存在，各位置的输入都是已知的。只要用掩码限制各行的可见范围，就可以在同一层里一起计算很多行的预测。

例如一次算出h1到h6，再分别给“服用”“华法林”直到句号打分。不是先真的生成“服用”再算下一行，而是用训练文本提供的真实前文。

因此，即使第5行最看好错误候选，第6行的输入仍然是训练文本中的“黑便”。这种使用真实前文的做法通常叫Teacher Forcing。

训练数组最简单地可以设为：输入前6块，目标为后6块。两边刚好错开一个位置。也可以输入全部7块，忽略没有下一目标的最后位置。若另有EOS结束标记，才可以让句号位置预测EOS；不能默认句号后天然就有一个训练结束目标。

第 7 章：还缺一个把向量变成预测的步骤

[返回目录](#)

7.1 你追问的预测函数，就在这里

到目前为止，我们得到的是h5，一串经过前文影响和多层加工的数。它还不是“黑便”的概率。

你问“那还得有一个什么预测函数吧”，正好指出了这一步。我们需要一个**输出层**，把h5变成词表中每个候选的分数。

把输出矩阵记成 U 。为了避免“ W ”到处都是，这里用不同字母，但 U 同样是可训练参数。若暂不加偏置：

$$\ell = h_5 U.$$

ℓ 读作ell，是一行候选分数，常叫logits。它不是损失 L ；两者字形相近，含义不同。

7.2 一个向量怎样同时给很多候选打分

先假设 h 只有两个坐标，词表也只有三个候选：黑便、呕血、头晕。把三个候选各自的打分向量作为 U 的三列：

$$h = (1, 2), \quad U = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

相乘得到：

$$\ell = (2, 1, 0).$$

第一项是 $(1, 2)$ 与第一列 $(2, 0)$ 的点积，等于2；第二项等于1；第三项等于0。

于是每个候选都有一个分数。真实模型只不过隐藏向量更长、候选更多。若 h 长 d 、词表有 N 项， U 就有 d 行 N 列，结果有 N 个分数。

这里没有规定 h 第一维永远代表某种疾病。 h 的表示方式与 U 的读取方式共同训练，只要二者配合能改善预测，就可能有用。

7.3 分数再变成概率

对这三个输出分数做Softmax：

$$p = \text{softmax}(2, 1, 0) \approx (0.665, 0.245, 0.090).$$

于是“黑便”的模型预测概率约66.5%。“呕血”和“头晕”则各占相应份额。

这里又出现Softmax，但别与注意力里的比例混淆：

在哪里做Softmax	候选是谁	得到什么
Attention内部	已经输入、允许读取的位置	从各处取内容的比例
最终输出层	词表中可能生成的Token	下一Token的预测概率

词表中某个候选没有出现在输入里，也照样可以被打分。否则模型就只能抄前文，无法续写新内容。

7.4 完整公式，现在才把它合起来

对第*i*个位置，若加上输出偏置*b*：

$$\ell_i = h_i U + b, \quad p_i = \text{softmax}(\ell_i).$$

下标*i*表示这是第*i*位置作出的下一Token预测。 p_i 是一整组概率，不是单个数；要问“黑便”的概率，需要取这一组中属于“黑便”的那一项。

有时会写：

$$p_{\theta}(\text{黑便} \mid t_1, \dots, t_5).$$

这里*t*₁到*t*₅是已有五块前文，竖线读作“在这些前文已知的条件下”。 θ 代表模型当前全部参数；把它写在*p*的下标，是提醒你换一套参数，预测概率也可能改变。整个式子表示一个数：这套模型给“黑便”分了多少概率。

公式并没有额外执行一步操作，只是把前面讲过的整个预测过程压缩记下来。

第 8 章：预测得好不好，怎样变成一个数

[返回目录](#)

8.1 训练时，正确目标是从数据里读的

假设这条训练文本在“出现”后面写的是“黑便”，模型却只给“黑便”30%的概率。

训练程序并不需要请另一个模型判断“应该改哪里”。它先从预测分布里取出正确目标的概率，再用一个函数把这次预测表现变成一个损失数值。

所谓“正确”，是指训练样本实际给出的后续，不是证明这句文字在现实中必然正确。错误文本也能被放进训练数据，这一点后面讨论能力边界时很重要。

8.2 为什么写成负对数

把正确目标的概率暂记为 p ，定义这次损失：

$$L = -\ln p.$$

这里 $\ln$ 是自然对数， L 是loss，损失。举几个数就能看出它怎么评分：

给正确目标的概率 p	损失 L ，约为
0.1	2.303
0.3	1.204
0.9	0.105
1	0

给正确目标的概率越高，损失越小；把它压到接近0，损失就很大。对概率求导也能验证： $dL/dp = -1/p < 0$ ，所以在0到1之间提高 p 会降低 L 。

不过“满足这些要求”还不能唯一证明非用对数不可。更系统的依据，是希望训练文本整体得到较高概率。

8.3 一段文字的概率怎样连起来

例如连续三个预测，分别对真实下一项给出 p_1 、 p_2 、 p_3 。这里 p_2 是在前面真实内容已知的条件下计算的， p_3 也是如此。

它们对应整段真实续写的条件概率是乘积 $p_1p_2p_3$ 。这来自概率链式法则，不要求三个Token互相独立。

要让这个乘积大，等价于让它的负对数小：

$$-\ln(p_1p_2p_3) = -\ln p_1 - \ln p_2 - \ln p_3.$$

所以一段文字的训练目标，可以写成各位置损失之和或平均。平均只是再除以参与计分的位置数量，方便比较尺度。

这就是把“给训练文本较高概率”变成一个方便求导的优化目标。对单个确定目标，这个负对数也就是常说的交叉熵损失。

8.4 但损失1.204到底能告诉模型什么

单独一个数不能指出要修改WQ的哪一项。它只告诉我们“这套参数当前产生的损失是多少”。

真正关键的是：损失来自一连串确定的计算。因此，我们可以追问：**如果某个参数略微改变，最终损失会往哪个方向变、变多少？**

这个变化率，就是导数或偏导数。下一章的反向传播，正是沿着已经算过的依赖关系，把这些变化率算出来。

第 9 章：模型怎样知道该改哪些数字

[返回目录](#)

9.1 先只考虑一个可改的数

假设模型只有一个参数 w ，损失是它的函数 $L(w)$ 。如果当前导数是正数，说明在这一点附近增加 w 会让损失上升，那么我们就向减小 w 的方向走一点。

反过来，导数为负，就向增大 w 的方向走一点。把两种情况合在一个式子里：

$$w_{\text{new}} = w_{\text{old}} - \eta \frac{dL}{dw}.$$

η 读作eta，是学习率，控制每次走多远。比如导数是 -0.7 ，学习率 0.1 ，则 w 增加 0.07 。

为什么不走得越远越好？因为导数描述的是当前位置附近的变化，走远后方向和曲率可能改变。这个公式给的是局部下降方法，不是一步找到最优参数的保证。

9.2 真正模型只不过有很多可改的数

WQ 不是一个数，而是一张表。我们对其中每一项分别问同样的问题：改变它一点， L 会怎样变？

这些偏导数排成同样大小的表，就叫 WQ 的梯度，记为 $\partial L / \partial W_Q$ 。这个记号不是两个矩阵相除。

然后按同样规则逐项更新：

$$W_Q^{\text{new}} = W_Q^{\text{old}} - \eta \frac{\partial L}{\partial W_Q}.$$

其他参数也是如此。实际大型训练通常使用更复杂的优化器调整步长，但仍以这些梯度为主要信息。本笔记手算使用普通梯度下降，便于看清发生了什么。

9.3 损失怎样追到很前面的 WQ

按顺序看这条路径：

WQ改变 → q改变 → 匹配分数改变 → 混合比例改变 → z改变 → 最终h改变 → 候选分数改变 → 正确目标概率改变 → L改变。

每一段都由明确函数连接。链式法则允许我们把各段变化率组合起来。

最简单的标量例子： $q=wx$ ， $z=2q$ ， $L=z^2$ 。把 x 视为已知，则：

$$\frac{dL}{dw} = \frac{dL}{dz} \frac{dz}{dq} \frac{dq}{dw} = (2z) \cdot 2 \cdot x.$$

这不是倒着猜 w 该是多少，而是在求“ w 通过这条计算路径影响 L 的变化率”。真正网络更大，有矩阵也有分支，原理仍然是多元链式法则。

反向传播负责算梯度，优化器负责用梯度改参数。算出梯度时，旧参数还没有被更新。

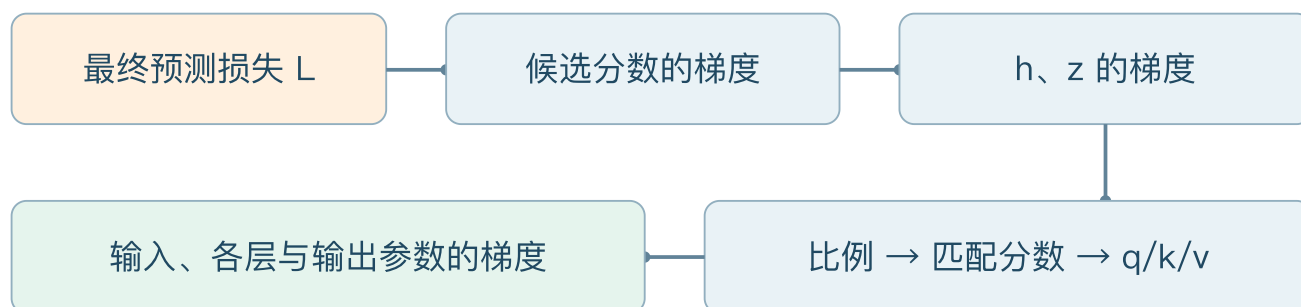


图4：沿计算依赖反向求导，再更新参数；不是把前向过程逆运算还原。

9.4 为什么从最后往前算

最终损失对输出分数的导数比较容易算。得到这一层的梯度后，就能继续求它对 h 的影响；有了 h 的梯度，再向前求 z 、比例、 $Q/K/V$ 。

中间结果可以被复用，不必为每个参数都重新跑一遍完整网络。如果同一变量通过两条路径影响 L ，就把两条路径的梯度加起来。残差连接正有这样一条额外路径。

所以“反向”说的是导数计算的顺序，前向计算本身并没有被逆运算还原。

9.5 输出层是不是要另外训练

通常的全参数训练中，不需要。输入嵌入、各层 $WQ/WK/WV$ 、FFN、归一化参数以及输出 U ，都属于同一个预测系统。

一次前向算出损失后，同一次反向传播可以求出它们各自的梯度，再由优化器一起更新。我们不必预先判断“这次主要是输入没表示好，还是输出没读好”。

“一起训练”不代表每个参数每一步都改变。有的行没有被用到，有的路径被屏蔽，有的梯度恰好为0。冻结部分参数的微调，也是另一种有意设置。

下一章不再只说“可以求导”，而是让你看到预测误差怎样真的传回 WQ ，再看到更新后概率怎样变化。

第 10 章：把一次预测和训练真的算完

[返回目录](#)

10.1 先缩小教具，但不把数字与文字脱开

完整模型参数太多，不适合手算。我们做一个极小的教学模型，只留下“注意力 → 残差 → 输出 → 损失”这条主线。FFN、归一化和位置编码暂时不放进去；因此它不是完整GPT。

为了仍然对应那句填空，这一章专门使用一个只有三项的玩具词表：

简写	这一项代表的文本
A	患者服用华法林后
B	出现
C	黑便

把一整段前文当成一个Token，只是为了缩成两个输入位置。这是本章特设的玩具分词，和前文七块分词不同，也不是说合并以后与原模型计算等价。本章不处理句号或结束标记。

输入是A、B，即“患者服用华法林后出现”；目标是C，即“黑便”。注意力要让“出现”这个位置读取前文A，再由它的最终向量预测C。

10.2 先列出本次所有参数

给三项Token各指定一个二维嵌入。把这张表叫E：

$$E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}.$$

三行依次对应A、B、C。于是输入A、B查出 $(1, 0)$ 、 $(0, 1)$ ，按行排列：

$$X = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

指定匹配变换和内容变换：

$$W_Q = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad W_K = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad W_V = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

WQ把二维输入变成一维q，WK把二维输入变成一维k。WV在这个初值下不改变向量。q/k只有一个坐标，所以本例dk=1，缩放因子也是1。

再设注意力输出投影WO也为二维单位矩阵，并指定词表输出矩阵：

$$W_O = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}.$$

U三列依次给A、B、C打分。这里没有偏置，E与U也不共享。以上参数都是为了方便核验而指定的起点，不是已经学会医学知识的一套数。

10.3 逐行算 Q、K、V

对第一位置A， $\mathbf{x}_1 = (1, 0)$ ：

$$q_1 = 0, \quad k_1 = 1, \quad v_1 = (1, 0).$$

对第二位置B， $\mathbf{x}_2 = (0, 1)$ ：

$$q_2 = 1, \quad k_2 = 0, \quad v_2 = (0, 1).$$

为什么第二位置q等于1？就是 $(0, 1)$ 乘WQ那一列，得到 $0 \times 0 + 1 \times 1 = 1$ 。每一个数都来自普通乘加。

现在考察第二位置“出现”。它对前文A的分数是 $q_2 k_1 = 1$ ，对自己B的分数是 $q_2 k_2 = 0$ 。

所以它需要分配的是两个分数 $(1, 0)$ 。Softmax给出：

$$\left(\frac{e}{e+1}, \frac{1}{e+1} \right) \approx (0.731059, 0.268941).$$

为了后面算式短一点，把这两个数暂叫a和c。a不是Token A，c也不是Token C；它们只是两个比例：a约0.731059，c约0.268941，而且a+c=1。

10.4 取内容，再加回原表示

第二位置读到的结果是：

$$z_2 = av_1 + cv_2 = (a, c).$$

再经过WO投影。因为本例WO恰好是单位矩阵，投影不变。与第二位置原输入 $(0, 1)$ 相加：

$$h = x_2 + z_2 W_O = (a, 1 + c) \approx (0.731059, 1.268941).$$

这里h就是本章小模型的最终向量。请注意：第一个坐标约0.731059不是“黑便概率”，第二个坐标也不是另一个Token概率。h仍是给输出层读取的两个数。

第一位置只能读自己，权重恒为1。若同时把两行写出，注意力比例表为：

$$A_{\text{attn}} = \begin{bmatrix} 1 & 0 \\ a & c \end{bmatrix}.$$

加上下标attn是为了把这张表与词表中的Token A区分开。

10.5 终于得到三个候选的概率

把h乘U：

$$\ell = hU = (a, 1 + c, -a) \approx (0.731059, 1.268941, -0.731059).$$

第三项是C、“黑便”的分数。当前参数居然给它负分，但负分没有问题：logits只是相对分数，接下来才变概率。

对三项取指数再归一化，得到：

$$p \approx (0.339660, 0.581626, 0.078714).$$

即A约33.97%，B约58.16%，C“黑便”只有约7.87%。输入中没有C，也不妨碍输出层给它打分。

训练目标是C，所以损失为：

$$L = -\ln p_C \approx 2.541928.$$

这一次概率不是凭空指定的30%。它从输入、投影、比例、内容、残差与输出矩阵逐步算出来。

10.6 先看损失怎样要求输出分数改变

对“Softmax加负对数损失”，可以推导出一个非常实用的结果：

$$\frac{\partial L}{\partial \ell_j} = p_j - \mathbf{1}_{j=C}.$$

符号 $\mathbf{1}_{j=C}$ 的意思是：如果当前候选j正是目标C，取1，否则取0。推导在第13章，不需要凭记忆接受。

代入本例，三个分数的梯度约为：

$$(0.339660, 0.581626, -0.921286).$$

这说明：若能独立改这三个分数，降低A、B分数，提高C分数，会在当前位置附近降低损失。但它们并非独立参数，而是由h和U算出，所以还要继续求到h与U。

10.7 再看误差怎样传到“读多少”

U第一行是 $(1, 0, -1)$ 。h第一个坐标增加一点，会让A分数同量增加、C分数同量减少。因此L对h第一坐标的导数是：

$$0.339660 - (-0.921286) = 1.260945.$$

U第二行是 $(0, 1, 0)$ ，所以h第二坐标只直接影响B分数，对应导数是0.581626。

于是h的梯度约为 **(1.260945, 0.581626)**。由于本例的残差直连和WO初值很简单，这份梯度也沿z2向前传。

而z2是 $av_1 + cv_2$ 。若先把a、c当成两个独立变量，增加a就沿 **(1, 0)** 方向改变z2，对损失的影响率是1.260945；增加c则沿 **(0, 1)** 方向改变，影响率是0.581626。

所以损失对两项混合比例的梯度，是这两个数。到这里，我们已经从“C预测得太低”，追到了“怎样的内容混合更能降低损失”。

10.8 比例互相牵制，不能各自随便改

a与c经过Softmax产生，必须相加为1。如果提高第一个匹配分数，a会增加，c会相应减少。因此要把这两种影响同时算进去。

对分数 (s_1, s_2) ，两项Softmax中有：

$$\frac{\partial a}{\partial s_1} = ac, \quad \frac{\partial c}{\partial s_1} = -ac.$$

把两条影响加起来：

$$\frac{\partial L}{\partial s_1} = ac(1.260945 - 0.581626) \approx 0.133562.$$

同理，第二个分数的梯度约为-0.133562。这里终于出现了“匹配分数该往哪个方向变”的数值。

注意，它不是简单地说“让看起来有关的前文权重越来越高”。这个方向取决于当前V内容、输出U和整个损失，不是人为指定的语义关系。

10.9 现在才到 WQ

本例q2来自第二行输入乘WQ，而两个分数是：

$$s_1 = q_2 k_1 = q_2, \quad s_2 = q_2 k_2 = 0.$$

求 q_2 的局部影响时，把当前 k 视为给定，得到：

$$\frac{\partial L}{\partial q_2} = 0.133562 \times 1 + (-0.133562) \times 0 = 0.133562.$$

再看WQ第二个元素，记为 w 。因为输入第二行是 $(0, 1)$ ，有

$q_2 = 0 \times W_{Q,1} + 1 \times w = w$ ，故损失对 w 的导数也是0.133562。

取学习率0.1，它从1更新为：

$$w_{\text{new}} = 1 - 0.1 \times 0.133562 \approx 0.986644.$$

至此，“预测错了怎样影响WQ中的一个数”已经走通：输出误差 $\rightarrow h \rightarrow$ 内容比例 $\rightarrow$ 匹配分数 $\rightarrow q \rightarrow$ WQ。

10.10 不是只改这一个数，而是共同更新

同样的链式法则也能算出WK、WV、WO、U和输入E的梯度。第13章列出矩阵规则，配套程序保存全部24个参数元素的结果。

例如，U第二行、C列最初为0。它的梯度为：

$$\frac{\partial L}{\partial U_{2,C}} = h_2(p_C - 1) \approx 1.268941 \times (-0.921286) \approx -1.169057.$$

注意这里 h_2 是本章二维 h 的第二个坐标，不是“第2位置的向量”。因为本节已经固定最后位置，只对向量内部坐标编号。写明这一点，是为了不把位置下标与坐标下标混用。

学习率0.1时，这个输出参数从0增加到约0.116906。输入表示的数也会更新，Q/K/V的各变换同样参与。

所有梯度先基于同一套旧参数算出，再一起执行更新。不能先改U，再拿新U去拼接旧前向的梯度，那是在混用不同计算点。

10.11 全部更新后，再跑一次看看

使用学习率0.1联合更新所有可训练矩阵，再从查表开始重新计算：

观察的量	更新前	更新后
“出现”读取前文A的比例	0.731059	0.713748
输出“黑便”的概率	0.078714	0.171807
正确目标损失	2.541928	1.761382

目标概率从约7.87%提高到17.18%，损失降低。但“黑便”还不是概率最大的候选，一次训练不必立刻答对。

读取前文A的比例反而下降了。说明训练可以通过同时改变输入表示、传递内容和输出打分来进步，不是“把每条看起来相关的注意力都加大”。

这个小模型共有24个参数元素。解析梯度与中心差分逐项核对，最大绝对误差小于 3×10^{-11} 。这支持本例计算与程序实现一致，不等于证明真实模型学到了医学知识。

想核查数值，可以打开[全部参数和梯度](#)或[运行程序](#)。正文显示值经四舍五入，程序始终使用完整精度。

第 11 章：训练完以后，它怎样一句句输出

[返回目录](#)

11.1 回到原来的七块教学分词

上一章缩小了词表，现在回到：

患者 | 服用 | 华法林 | 后 | 出现 | 黑便 | 。

生成时，用户只提供前五块。模型计算这些位置的表示，最后用 h_5 给整个词表打分、得到概率。

为什么用最后一行？因为每一行都在预测它后面的一块。前几行对应的后续已经是输入的一部分，最后一行才对应现在还没有提供的下一块。

这不是说 h_5 无损储存了整句话，只是说它可以依赖允许前缀，并被训练来提供下一Token预测所需的特征。

11.2 有了概率，也还没真正选词

Softmax只输出分布。生成还要执行选择规则。

最直接的是取概率最大的Token，叫贪心解码。另一种是按概率随机抽样，概率较小的候选也有机会被选中。

温度、top-k、top-p等规则会改变分布的尖锐程度或候选范围。它们是在决定“这一刻怎样选”，不是通过反向传播重新训练参数。

而且，选择当前最可能的一项，不保证整段续写的联合概率最大：这一步选择会改变后面的上下文，后续概率也跟着变。

11.3 选出一个，就把它接回去

假设选中“黑便”。新前缀变成：

患者 | 服用 | 华法林 | 后 | 出现 | 黑便

现在使用第6位置的表示预测下一块。若选中句号，就得到完整句子。它还可以继续预测句号之后的内容，直到选中结束Token、达到长度限制或触发其他停止规则。

整个过程中，输入序列变长，中间向量改变，缓存也可能增长，但模型参数通常不变。

若原本输入就已经是“患者服用华法林后出现黑便。”，模型不会把已提供的“黑便”当成下一步待填内容；它从这个完整前缀的末尾继续生成。

11.4 每多一个字，都得从头重算整句吗

从概念上可以这样做，但通常太浪费。因果约束告诉我们：加上新Token，并不会改变旧位置原本能看到的前文。

在参数、位置规则与计算设置不变的条件下，旧位置已经算好的各层K/V可以保存下来。新位置算自己的Q/K/V，再去读这些历史K/V。这就是KV Cache。

注意是**各层的K/V**，不只是最后一个句子向量。新位置仍能读取分布在历史位置的信息；并没有把所有历史永久压成一个不再变化的h。

缓存减少重复计算，但要占用额外内存，也不消除新位置读取历史的开销。若改参数、重排位置或改变窗口规则，能否复用需要另行检查。

11.5 训练能并行，为什么生成常常逐个

训练文本已经给出所有真实前文，各位置可以在同层一起算，掩码防止泄漏。生成时，下一步的前文包含这一步实际选出的Token；在选出来之前，后面的输入还没确定。

所以标准自回归生成要一步步推进。这里说的是基础解码过程；工程上还有其他加速方法，不改变我们此处理解依赖关系的主线。

在相同参数、相同前缀、相同掩码和位置条件下，关闭dropout等随机差异，训练与生成在对应位置的预测分布应相同。区别是之后用来评分更新，还是用来选择追加。

第 12 章：这些计算为什么值得这样组织

[返回目录](#)

12.1 “不就是加权平均吗”，这个疑问很合理

若权重永远固定成 $(0.2, 0.5, 0.3)$ ，无论什么输入都这么混合，确实只是固定规则。

Attention的区别是：**比例由当前输入通过可训练的匹配规则算出**。前文不同，Q/K不同，分数与比例可以不同。即使参数固定，读取方式仍可随输入改变。

如果把这种依赖写出来，就是：

$$z_i = \sum_j a_{ij}(X) v_j.$$

括号X是在提醒你，a不是预先写死的常数。读什么、读多少，是这次输入参与决定的。

12.2 多层是在对已经加工过的信息继续工作

第一层的输入比较接近初始Token表示。第二层输入已经受到第一层上下文混合的影响，因此它匹配和传递的内容，不再只是原始词向量。

再配合FFN的非线性处理、多头与残差，模型能构造复杂得多的映射。但不能因此规定“第一层识别药物、第二层识别出血、第三层做推理”。真实模型未必按人类讲解顺序分工。

这说明架构提供了怎样的处理空间，不证明随机设置这些矩阵就能理解语言。

12.3 学习让这些表示逐渐有用

如果某种前文读取方式有利于预测，损失经过反向传播，就可能推动相关参数向降低训练目标的方向变化。反复在大量数据上训练，各部分逐渐形成配合：输入表示适合后续加工，隐藏表示适合输出层读取。

所以不需要每个坐标都能被人类命名，后续网络仍可能利用它们完成任务。我们在第10章看到的，是这种机制的极小、可核验例子。

从这里到真实模型的语言能力，还涉及数据、模型规模、优化是否有效和泛化表现等问题。“能表达复杂函数”与“实际学会可靠推理”，是不同结论，不能用一句“堆叠足够多层”把中间都省掉。

12.4 预测文本，不等于保证事实

损失会奖励模型给训练文本中的后续较高概率，不自动判断文本真伪。因此模型可能学到有用知识，也可能学到错误、偏见、表面共现或训练样本记忆。

同样，一条注意力边权重很大，只表示这个头对相应Value用了较大系数。最终答案还受内容方向、投影、其他头、残差和后续模块影响，不能直接把它当成答案的因果解释。

12.5 Transformer、GPT、聊天系统还要分清

Transformer有不同的信息流组织方式。原始编码器—解码器架构常用于翻译：编码器处理源句，解码器一边读取源句表示，一边逐个生成目标句。解码器读取编码器结果的部分叫交叉注意力：Q来自正在生成的一侧，K/V来自已经编码的一侧。

我们主线讲的是Decoder-only的因果自注意力语言模型。还有BERT一类双向编码器，让输入位置使用前后文，训练任务与逐个续写不同。它们不是同一掩码设置，也不能用同一个训练例子混着解释。

聊天时，系统与用户消息、历史回答、角色标记会被组织成模型上下文，再从助手回答开始的位置续写。让模型学会按这种格式回答，还需要相应的训练数据与训练过程。

因此，“下一个Token预测”解释了基础计算任务，但没有单独解释所有对话行为。参数训练、在当前上下文里利用信息、应用把信息存下来以后再提供，也分别是不同机制。

第 13 章：想追到底时，再看这些数学推导

[返回目录](#)

这一章供回查，不要求先读完才理解主线。每节都对应前文一个具体问题，而不是另外搭一套符号体系。

13.1 一次矩阵乘法为什么等于逐行投影

X有n行，每行d个坐标；WQ有d行dk列。定义 $Q = XW_Q$ ，则第i行第r个数是：

$$Q_{ir} = \sum_{u=1}^d X_{iu}(W_Q)_{ur}.$$

固定i，让r遍历所有输出坐标，得到的正是 $q_i = x_i W_Q$ 。同一张WQ作用于每一行，因而同层参数跨位置共享。

接着 QK^T 的第i行第j列为 $\sum_r Q_{ir}K_{jr} = q_i \cdot k_j$ 。所以“同时算所有位置间点积”是矩阵乘法定义直接给出的结果。

最后， $Z = AV$ 的第i行第u个坐标为 $\sum_j a_{ij}V_{ju}$ ，也正是逐个v加权组合后的第u坐标。

13.2 几种维度不要混成一件事

符号	指什么	举例
n	当前输入多少个位置	5个Token
d	每个位置在层间传递多少个数	每行512个数
dk	单个头里q/k各有多少坐标	每行64个数
dv	单个头里v有多少坐标	每行64个数
N	词表里有多少候选Token	50000项

X形状是nxd，Q/K是nxdk，V是nxdv。A是nxn，Z是nxdv。U是dxN。注意力矩阵的两条轴都是输入位置；输出概率的候选轴是词表。

dk、dv相等是常见配置，不是点积运算强制V也与Q同长。不同头输出拼接后，要通过WO投影回d维，才能接回同形状残差。

13.3 学习两个投影，究竟增加了什么自由

用行向量写：

$$q_i k_j^T = x_i W_Q W_K^T x_j^T.$$

中间矩阵 $B = W_Q W_K^T$ 可训练，秩至多 $\min(d_Q, d_K)$ ，通常不要求对称。若强制 $W_Q = W_K$ ，那么 B 变成 $W_Q W_Q^T$ ，额外受到对称半正定约束。

一个小例子： $x_a = (1, 0)$, $x_b = (0, 1)$ ，取 $W_Q = (1, 0)^T$, $W_K = (0, 1)^T$ ，则 a 读 b 时点积为1， b 读 a 时为0。原来两个输入向量的直接点积却是0。可见投影后的匹配不是原始向量相似度的简单重复。

13.4 为什么方差可以变成平方的期望

你原来追问的等式是 $\text{Var}(X + Y) = \mathbb{E}[(X + Y)^2]$ 。它不是无条件成立，前提是 $X+Y$ 的均值为0。

对任意具有有限二阶矩的随机变量 Z ，方差定义为：

$$\text{Var}(Z) = \mathbb{E}[(Z - \mathbb{E}Z)^2].$$

先展开平方，再用期望的线性性质：

$$\text{Var}(Z) = \mathbb{E}[Z^2] - 2\mathbb{E}Z \mathbb{E}Z + (\mathbb{E}Z)^2 = \mathbb{E}[Z^2] - (\mathbb{E}Z)^2.$$

所以均值为0时，第二项消失。若 $\mathbb{E}X = \mathbb{E}Y = 0$ ，由期望线性性知 $\mathbb{E}(X + Y) = 0$ ，就得到前述简化。这一步不要求 X/Y 独立。

继续展开平方，会出现交叉项 $2\mathbb{E}[XY]$ 。若 X/Y 独立且各自均值为0，它才等于0；更一般地，方差相加需要协方差为0。

$$\text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y) + 2 \text{Cov}(X, Y).$$

协方差描述两变量相对各自均值共同波动的程度。独立足以保证协方差为0，但反过来不一定成立。

13.5 用这个依据推到根号dk

假设q与k各坐标全部相互独立，均值0、方差1。这里是在建立一个理想化概率模型，用来分析尺度，不是断言训练好的向量真的满足这些条件。

把第r项乘积记成 $Y_r = q_r k_r$ 。独立性使：

$$\mathbb{E}Y_r = \mathbb{E}q_r \mathbb{E}k_r = 0, \quad \mathbb{E}[Y_r^2] = \mathbb{E}[q_r^2] \mathbb{E}[k_r^2] = 1.$$

所以每项方差是1；不同r的项也独立，dk项相加的方差为dk，标准差为根号dk。

$$\text{Var}\left(\frac{q \cdot k}{\sqrt{d_k}}\right) = \frac{1}{d_k} \text{Var}(q \cdot k) = 1.$$

为什么不是标准差直接相加？四个独立的等概率±1相加，可能总和为-4、-2、0、2、4，概率分别是 (1, 4, 6, 4, 1)/16。均值0，平方的期望为4，故标准差2，不是4。随机正负相互抵消，使典型尺度与最大可能值不同。

这里使用了期望线性性、独立变量乘积期望可分解等概率基础定理；不要求正态分布。实际Q/K若有关联、均值或方差不同，方差结论便不再精确照搬。

13.6 Softmax的导数为什么是那样

设 $D = \sum_r e^{s_r}$, $a_j = e^{s_j}/D$ 。若对自己的分数sj求导，商法则给出：

$$\frac{\partial a_j}{\partial s_j} = \frac{e^{s_j} D - e^{2s_j}}{D^2} = a_j(1 - a_j).$$

若对别的分数sk求导，只有分母受影响：

$$\frac{\partial a_j}{\partial s_k} = -\frac{e^{s_j} e^{s_k}}{D^2} = -a_j a_k, \quad k \neq j.$$

这解释了“提高一项，会挤掉其他项的比例”。若一项已接近1、其他项接近0，上述导数都可能很小，所以过尖的注意力分配可能不易调整。缩放分数是帮助控制这种情况的方法，不是所有训练问题的保证解法。

为了数值稳定，所有分数可先减去最大值。Softmax不变，因为分子分母中的公共指数因子会约去。这能避免直接对很大的分数取指数。

13.7 掩码的严格定义与凸组合的范围

对位置 i ，允许集合是 $J_i = \{1, \dots, i\}$ 。分数仍是 $s_{ij} = q_i \cdot k_j / \sqrt{d_k}$ ，定义：

$$a_{ij} = \begin{cases} \frac{e^{s_{ij}}}{\sum_{r \in J_i} e^{s_{ir}}}, & j \in J_i, \\ 0, & j \notin J_i. \end{cases}$$

这样不用真的把负无穷当普通数求导。允许集合内是正数、总和为1，集合外为0。每行至少要有有一个允许位置；本文允许读取自身，不会出现全屏蔽的空集合。

无任何掩码且分数有限时，各位置数学上的权重严格正，但可以极小。机器计算还可能因浮点下溢变成0，不能把机器结果与严格实数性质混为一谈。

单头加权结果在当前 V 的凸包内，是由凸包定义得到的。多头投影、残差和FFN不必受同一个凸包约束，候选 V 本身也随输入与参数改变。

13.8 损失对输出分数的导数

对目标编号 y ，输出分数是 $\ell_1, \dots, \ell_N$ ，Softmax概率为 p 。将目标概率代入负对数：

$$L = -\ln \frac{e^{\ell_y}}{\sum_r e^{\ell_r}} = -\ell_y + \ln \sum_r e^{\ell_r}.$$

对第 j 个分数求导：第一项在 $j=y$ 时给-1，否则给0；第二项给 $e^{\ell_j} / \sum_r e^{\ell_r} = p_j$ 。所以：

$$\frac{\partial L}{\partial \ell_j} = p_j - \mathbf{1}_{j=y}.$$

因此即使输出对错误候选很自信，目标项梯度仍可能接近-1。不能仅根据Softmax本身的导数很小，就推断“输出错误且自信时一定无法学习”。合并损失后要看完整导数。

13.9 把第10章的反向计算写成矩阵

用上横线表示“损失对这个量的梯度”，比如 $\bar{Z} = \partial L / \partial Z$ 。梯度与原量同形状。对矩阵 X ，它的含义由微分关系规定：

$$dL = \sum_{ir} \bar{X}_{ir} dX_{ir}.$$

对输出 $\ell = hU + b$ ：

$$\bar{U} = h^\top \bar{\ell}, \quad \bar{h} = \bar{\ell} U^\top, \quad \bar{b} = \bar{\ell}.$$

第一式为什么是外积？因为第 j 分数是 $\ell_j = \sum_r h_r U_{rj} + b_j$ ，故 $\partial L / \partial U_{rj} = h_r \bar{\ell}_j$ 。多位置时将各贡献累加；若损失取平均，梯度也带同样的平均系数。

对于第10章定义的简化模块 $R = X + ZW_O, Z = AV$ ，给定 $\bar{R}$ ：

$$\bar{W}_O = Z^\top \bar{R}, \quad \bar{Z} = \bar{R} W_O^\top, \quad \bar{A} = \bar{Z} V^\top, \quad \bar{V} = A^\top \bar{Z}.$$

对允许的注意力分数，用Softmax各导数汇总：

$$\bar{S}_{ij} = a_{ij} \left(\bar{A}_{ij} - \sum_{r \in J_i} a_{ir} \bar{A}_{ir} \right).$$

被禁止位置梯度为0。然后：

$$\bar{Q} = \bar{S} K / \sqrt{d_k}, \quad \bar{K} = \bar{S}^\top Q / \sqrt{d_k},$$

$$\bar{W}_Q = X^\top \bar{Q}, \quad \bar{W}_K = X^\top \bar{K}, \quad \bar{W}_V = X^\top \bar{V}.$$

输入X通过残差与三条投影路径同时影响损失，故收到的梯度要相加：

$$\bar{X} = \bar{R} + \bar{Q}W_Q^T + \bar{K}W_K^T + \bar{V}W_V^T.$$

这些式子针对本节明确的小模块，不冒充整个完整GPT的反向公式。完整网络还要沿FFN、归一化和更多层继续用链式法则。

如果X来自嵌入查表，某个词表行收到的梯度，就是本次所有使用该行的位置梯度之和。没有用到的行不从输入路径收到梯度；重复Token的贡献相加。

13.10 第10章的其余数值，供核查

同一次前向得到的梯度约为：

$$\bar{W}_Q = \begin{bmatrix} 0 \\ 0.133562 \end{bmatrix}, \quad \bar{W}_K = \begin{bmatrix} 0.133562 \\ -0.133562 \end{bmatrix},$$

$$\bar{W}_V = \bar{W}_O = \begin{bmatrix} 0.921825 & 0.425203 \\ 0.339120 & 0.156423 \end{bmatrix},$$

$$\bar{U} = \begin{bmatrix} 0.248311 & 0.425203 & -0.673514 \\ 0.431008 & 0.738049 & -1.169057 \end{bmatrix},$$

$$\bar{E} = \begin{bmatrix} 1.055387 & 0.425203 \\ 1.466503 & 0.871611 \\ 0 & 0 \end{bmatrix}.$$

C“黑便”的嵌入行没作为输入出现，所以这里E第三行梯度为0；但U的C输出列有非零梯度。输入角色和输出角色是不同路径。WV与WO梯度相等，则是本例单位矩阵初值造成的特殊结果。

配套程序用 $[L(\theta + \epsilon e_r) - L(\theta - \epsilon e_r)]/(2\epsilon)$ 对24个参数逐项做中心差分， $\epsilon = 10^{-5}$ ， e_r 表示只在第r个参数位置改动。差分有数值误差，是对解析推导的核验，不代替推导本身。

13.11 更完整的Block与LayerNorm

主线只解释各模块用途，这里明确一种可计算的排列。令 X 是本层输入，LN1/LN2为两次归一化，MHA是多头注意力：

$$T = X + \text{MHA}(\text{LN}_1(X)),$$

$$Y = T + \text{FFN}(\text{LN}_2(T)).$$

这叫Pre-LN，因为归一化放在各子层前面。 Y 进入下一层。最后还可以做最终归一化，得到用于输出的 H 。原始2017年Transformer使用子层与残差相加之后再归一化的Post-LN排列，不要混成同一精确配置。[R1]

对一行 x ，LayerNorm先定义均值 μ 和平方偏差平均 v ：

$$\mu = \frac{1}{d} \sum_r x_r, \quad v = \frac{1}{d} \sum_r (x_r - \mu)^2.$$

再逐坐标计算：

$$\text{LN}(x)_r = \gamma_r \frac{x_r - \mu}{\sqrt{v + \varepsilon}} + \beta_r.$$

ε 为小正数，避免分母为0； γ/β 是可训练的逐坐标缩放和偏移。这里 v 是归一化定义的平方偏差平均，不是除以 $d-1$ 的统计样本方差。因为有 ε 及 γ/β ，不能声称最后输出严格均值0、方差1。[R3]

FFN可设 $B = XW_1 + b_1, C = \sigma(B), Y = CW_2 + b_2$ 。给定 $\bar{Y}$ ，按乘积与链式法则：

$$\bar{W}_2 = C^\top \bar{Y}, \quad \bar{C} = \bar{Y} W_2^\top, \quad \bar{B} = \bar{C} \odot \sigma'(B),$$

$$\bar{W}_1 = X^\top \bar{B}, \quad \bar{X} = \bar{B} W_1^\top.$$

符号 $\odot$ 表示逐坐标相乘；偏置梯度对位置相加。ReLU在0不可微，自动微分实现会使用约定值；平滑激活按自己的导数计算。

LayerNorm也可逐步求导。令 $s = \sqrt{v + \varepsilon}$ 、 $\hat{x} = (x - \mu)/s$ ，已知输出梯度 g ，令 $u = g \odot \gamma$ ，可整理为：

$$\bar{x} = \frac{1}{s} [u - \text{mean}(u)\mathbf{1} - \hat{x} \text{mean}(u \odot \hat{x})].$$

mean沿这一个位置的坐标取平均。用 $d\mu = \text{mean}(dx)$ 、 $dv = 2 \text{mean}((x - \mu) \odot dx)$ 与 $ds = dv/(2s)$ 代入 $d\hat{x}$ ，即可推出。 γ/β 梯度分别为 $g \odot \hat{x}$ 与 g ，多位置时相加。完整网络就是把这些局部规则按计算图连接起来。

13.12 位置信息与输入输出共享

没有位置编码、也没有位置相关掩码的全注意力，对输入位置的重排会相应重排输出，这叫置换等变。因为Q/K/V都随行重排，分数表随行列重排，每行Softmax与内容组合也随之重排。

因果掩码本身已经引入方向约束，不能把这个“没有掩码”的论证原封不动套给因果网络。显式位置信息还可提供位置或距离结构。

RoPE的一种理解是：先给不同位置的Q/K施加不同角度的二维旋转，再点积。本段专门采用列向量，写成 $\tilde{q}_i = R_i q_i, \tilde{k}_j = R_j k_j$ 。那么：

$$\tilde{q}_i^T \tilde{k}_j = q_i^T R_i^T R_j k_j.$$

若一个二维块的旋转角是位置乘固定频率 ω ，则 $R_i^T R_j = R((j - i)\omega)$ 。相对位置差 $j-i$ 便进入点积分数。换回主文行向量约定时，相应转置即可。[R4]

最后，输入嵌入 E 为 $N \times d$ ，输出 U 为 $d \times N$ ，可以有意设置 $U = E^T$ ，这叫权重共享。此时输出是拿 h 与各Token嵌入点积，而不是对 E 求逆。共享参数收到输入、输出两条路径的梯度之和；不共享时分别训练。[R1、R6]

13.13 梯度下降的依据与边界

把所有参数排成向量 θ ，令 $g = \nabla L(\theta)$ 。可微时的一阶展开给出：

$$L(\theta - \eta g) = L(\theta) - \eta \|g\|^2 + o(\eta).$$

末项 $o(\eta)$ 表示比 η 更高阶的小量。当 g 不为0、 η 充分小时，负的一阶项占主导，当前损失会下降。它没有保证任意学习率都有效，没有保证找到全局最优，也没有保证训练批次改善就代表其他文本改善。

理解“训练”时，应同时保留机制与边界：梯度给出当前目标的局部方向，参数更新是有限一步，真实效果还需重新计算与评估。

第 14 章：按你的问题回查，以及阅读来源

[返回目录](#)

14.1 原来16次提问，现在分别在哪里

你的原始问题	本版解释位置
1 用我会的知识解释Transformer架构	任务从填空开始、模块怎样连接
2 点积转权重还是太快，不好理解	三个具体点积、Softmax两步
3 为什么64项的标准差是8	缩放的直觉、方差完整推导
4 为什么方差等于平方的期望	均值为0的条件与证明
5 等等dk是什么	在点积里数坐标、维度对照
6 这一步是在算WQ还是q	固定矩阵作用于不同输入
7 为什么这里又乘V	比例还不是内容

你的原始问题	本版解释位置
8 这东西有什么高级的	输入改变读取方式
9 x输出z有什么意义	表示的用途、输出层怎样读取
10 输入的是一句话，到底训练还是输出	整句矩阵、共同前向、训练对齐
11 继续：为什么最后位置能预测	每个位置预测谁、最后一行与生成
12 没有M，每个词都能联系其他词吧	可见范围与防泄漏
13 用医学句子展示训练和输出	训练的真实前文、缩小后算完、逐个生成
14 解释损失、梯度、更新	为什么用负对数、链式法则、数值走通
15 还得有个预测函数吧	从h到候选分数与概率
16 输出层单独训练，还是一起	同一损失联合训练、共享权重的补充

14.2 六个不用背公式的自检

问：WQ没变，为什么注意力比例会变？ 因为输入X变了，算出的Q/K也会变，匹配分数和比例随之改变。参数固定不等于中间结果固定。

问：有了注意力比例，为什么还不够？ 因为还不知道按比例混合的V内容是什么。只有比例不能确定z。

问：z或h的某一坐标是某个词的概率吗？ 一般不是。还要输出矩阵打分，再经Softmax，才得到词表概率。

问：训练输入里已有“黑便”，不就是答案吗？ 是用于评分的答案，但预测它的位置每层都不能读取它。数据已知不等于该前向路径可见。

问：推理时把刚生成的词接回输入，是更新参数吗？ 不是。这改变的是上下文与中间计算，参数通常不变。

问：正确目标概率提高了，就一定已经答对了吗？不一定。第10章“黑便”从7.87%变成17.18%，损失下降，却仍未成为最高概率项。

14.3 本版怎样处理严谨性

计算规则与导数在明确设定下推导；类比只解释角色，不当成证明。完整架构与手算小模型分开，小模型没有的FFN、归一化等不假装已验证。真实模型的语义分工与能力，不能从本例数字直接推出。

数学依赖包括有限维矩阵运算、指数与对数求导、多元链式法则、可微函数的一阶展开、概率链式法则、有限二阶矩下的期望与方差性质。与学习主线直接相关的步骤已展开，不再从测度论或实数构造重建这些基础。

14.4 对话与参考资料

原始来源：[解释Transformer架构的分享问答](#)。本地保留完整可读对话与提取核对。共有16次用户提问，会话链连续；另有一条源页面隐去的插件输出，不能声称恢复其内容。

第二版按你的反馈重新讲解，不只是给第一版术语加注释。第一版仍保留在旧版本目录，供比较；最新入口见[阅读索引](#)。

- **R1** Vaswani等，[Attention Is All You Need](#)：原始Transformer、缩放点积、多头、FFN与输入输出结构。
- **R2** Bengio等，[A Neural Probabilistic Language Model](#)：神经概率语言模型与分布式表示的背景。
- **R3** Ba等，[Layer Normalization](#)：LayerNorm的定义和方法。
- **R4** Su等，[RoFormer: Enhanced Transformer with Rotary Position Embedding](#)：旋转位置编码。
- **R5** Devlin等，[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#)：双向编码器与不同训练目标的对照。

- **R6** Press与Wolf, [Using the Output Embedding to Improve Language Models](#): 输入输出权重共享的讨论。

本文不需要联网才能阅读正文和公式。以上论文与本地跨文件资料属于补充链接；单文件分享HTML时，它们需要另行获取，但不影响正文理解。