GPT 嵌入层

GPT的嵌入层是模型的第一个组件，负责将输入的文本 token 转换为高维向量表示。
嵌入层将离散的 token ID 映射到连续的向量空间中。每个 token （通常是子词或字符）都对应一个固定维度的实数向量，这个向量包含该 token 的语义信息。

主要组成

Token 嵌入（Token Embedding）

- 将词汇表中的每个 token 映射到固定维度的向量(例如 GPT2 small 是 768 维)
- 这些向量在训练过程中学习得到，捕获 token 的语义和语法特征
- 向量维度通常为 512,768,1024等，通常是 64 的倍数。大模型通常为 2048,4096,8192等

位置嵌入 (Positional Embedding) 由于 Transformer 架构本身不包含位置信息，GPT 使用位置嵌入来编码 token 在序列中的位置：

- 绝对位置嵌入：为每个位置学习一个固定的向量
- 相对位置嵌入：在一些变体中使用，更好的处理长序列

嵌入矩阵

token 嵌入矩阵的形状是 [词表尺寸，嵌入向量维度]
位置嵌入矩阵的形状是 [最大序列长度，嵌入向量维度]

比如，GPT2 small 模型中，两个矩阵形状分别是：

- token 嵌入矩阵：[50257,768]
- 位置嵌入矩阵: [1024,768]，其中 **1024** 是上下文最大序列长度

前向传播

前向传播包含 token 嵌入和位置嵌入，然后将两个嵌入之后的向量相加，作为模型的下一层输入。（下一层通常是一个 Dropout 层）

准备工作

在嵌入之前，需要将输入的训练文本或者推理文本使用 tokenizer 转换为 token ID 序列
在训练时，可能包含多个批次，每个批次有多个 token 序列。在推理时，通常只有一个批次，包含一个 token 序列。

token 嵌入过程

假设输入 token 是一个二维张量，形状为 [B,T]，B 表示批次，T 表示 token序列。

示例文本: "What is the capital city of France?"
tokenizer 之后得到如下形如 [1,8] 的二维张量：

$$\begin{bmatrix}2061 & 318 & 262 & 3139 & 1748 & 286 & 4881 & 30\end{bmatrix}$$

将以上 [1,8] 的二维张量输入到嵌入层之后，以第二维的各个元素作为索引，在 token 嵌入矩阵中查找对应位置的嵌入向量，最终输出的张量形状为 $[B, T, C]$，其中 C 就是嵌入矩阵的特征向量长度，对于 GPT2 small来说，C 的长度为 768.

位置嵌入

对于输入的 token 张量:

$$\begin{bmatrix}2061 & 318 & 262 & 3139 & 1748 & 286 & 4881 & 30\end{bmatrix}$$

token 序列长度为 8，则会生成一个形如 [B,T] 的二维张量:

$$\begin{bmatrix}0 & 1 & 2 & 3 & 4 & 5 & 6 & 7\end{bmatrix}$$

将以上的二维位置张量输入到位置嵌入层，以第二维的各个元素作为索引，在位置嵌入矩阵中查找对应位置的嵌入向量，最终输出的张量形状为 $[B, T, C]$，其中 C 就是嵌入矩阵的特征向量长度，对于 GPT2 small来说，C 的长度为 768.

token 嵌入张量 + 位置嵌入张量

上述两个张量形状完全一致，直接将对应位置元素相加，最终生成一个 $[B, T, C]$ 的张量作为下一层（通常是 Dropout 层）的输入。

到此就完成了嵌入过程。

反向传播过程

在训练初始阶段，token 嵌入矩阵和位置嵌入矩阵的参数采用正态分布随机初始化（均值=0，标准差=0.02），使之可以在训练过程中进行学习。

计算过程

当梯度反向通过 Dropout 层之后，即可作为 token 嵌入层和位置嵌入层的反向传播输入梯度。

在嵌入层的反向传播过程中，输入的梯度形状是 $[B, T, C]$. 我们只需要依据输入张量的索引，将相应的梯度累加到嵌入矩阵指定位置即可。

假如输入梯度如下（形状为[1,8,768]）：

$$\begin{bmatrix}\begin{bmatrix}0.1 & 0.2 & 0.3 & 0.4 & \dots & 0.123\\ 0.1 & 0.2 & 0.3 & 0.4 & \dots & 0.123\\ 0.1 & 0.2 & 0.3 & 0.4 & \dots & 0.123\\ 0.1 & 0.2 & 0.3 & 0.4 & \dots & 0.123\\ 0.1 & 0.2 & 0.3 & 0.4 & \dots & 0.123\\ 0.1 & 0.2 & 0.3 & 0.4 & \dots & 0.123\\ 0.1 & 0.2 & 0.3 & 0.4 & \dots & 0.123\\ 0.1 & 0.2 & 0.3 & 0.4 & \dots & 0.123\end{bmatrix}\end{bmatrix}$$

在训练过程中，嵌入层缓存了输入张量，比如：

$$\begin{bmatrix}2061 & 318 & 262 & 3139 & 1748 & 286 & 4881 & 30\end{bmatrix}$$

比如对于 梯度张量的第一个批次的第一行向量，我们根据前向传播的输入张量，可以知道这个行向量对应于嵌入矩阵索引为 [2061] 的所在行.则，我们将这个行向量各个元素累加到梯度矩阵索引为 [2061] 所在行向量上。因为一个序列中相同的 token ID 可能出现多次，所以使用是累加，而不是直接赋值。

位置嵌入也是一样的过程,并且，token 嵌入和位置嵌入均使用 Dropout 反向传播的输出梯度作为输入梯度。

补充

在 GPT2 模型中，token 嵌入中的嵌入矩阵与模型最后一个线性层共享嵌入矩阵，所以，token 嵌入矩阵在一次反向传播中会被累加两次梯度。当然输入的梯度应该是不同的。