嵌入层(Embedding Layer)

概念与定义

在自然语言处理(NPL)任务中，第一步是将文本进行分词(tokenization)，为每个词元分配一个唯一的整数ID，即TokenID。但一维的ToeknID无法提供和承载丰富、复杂的语义信息，需要将其转换为更高维的向量。这就是嵌入层(Embedding Layer)的作用。简言之，嵌入层就是一个简单的查找表，用于将这些整数ID转换为固定大小的稠密向量表示。

实现与使用

nn.Embedding 是 PyTorch 中的一个模块，用于实现嵌入层。

我们也可以通过代码简单实现一个嵌入层(对原理不感兴趣的，直接跳过这里，看“嵌入层使用”即可)。

嵌入层实现

嵌入层本质是一个查找表，输入的是索引(token_ids)，返回的是权重矩阵中对应索引的行。一个简单的嵌入层实现如下：

```
import torch            # PyTorch 的核心库，提供了大量的张量操作
import torch.nn as nn   # PyTorch 中用于构建神经网络的库，包含了各种层(layers)和模块(modules)

class MyEmbedding(nn.Module):  # MyEmbedding 类继承自 nn.Module，这是 PyTorch 中所有神经网络
    def __init__(self, num_embeddings, embedding_dim): # 初始化方法接受两个参数: num_embedd
        super(MyEmbedding, self).__init__() # 调用父类 nn.Module 的初始化方法
        self.weight = nn.Parameter(torch.rand(num_embeddings, embedding_dim)) # 创建了一

    def forward(self, token_ids): # forward 定义嵌入层的前向传播方法，即数据通过嵌入层的计算过程
        return self.weight[token_ids] # 根据输入的 token_ids，返回权重矩阵中对应索引的行. 这里
```

由上面的实现代码可以看出，在嵌入层中，嵌入矩阵被视为一个可学习的参数，在训练过程中会被优化器更新。在训练过程中，模型会根据损失函数调整嵌入矩阵，以使其更好地表示词元的语义特征。

嵌入层使用

正如前面所说的，nn.Embedding 是 PyTorch 中的一个模块，可以直接使用：

```python
import torch
import torch.nn as nn

# 创建一个嵌入层，输入维度为10，输出维度为5
embedding = nn.Embedding(num_embeddings=10, embedding_dim=5)
# 输入一个整数ID
input_ids = torch.tensor([1, 2, 3, 4])
# 输出嵌入向量
embedded_vectors = embedding(input_ids)
print(embedded_vectors)
# 输出示例(一个形状为(4, 5)的张量，代表了4个词元的嵌入向量):
# tensor([[ 0.1237,  0.2345, -0.3456,  0.4567, -0.5678],
#         [ 0.5678, -0.4567,  0.3456, -0.2345,  0.1237],
#         [-0.1237,  0.2345, -0.3456,  0.4567, -0.5678],
#         [ 0.5678, -0.4567,  0.3456, -0.2345,  0.1237]], grad_fn=<EmbeddingBackward>)
```

除了上面示例代码中的 num_embeddings 和 embedding_dim 参数，nn.Embedding 还提供了其他一些有用的参数，例如 padding_idx（用于填充的索引）、max_norm（嵌入向量的最大范数）等。

- num_embeddings (int): 嵌入字典的大小，即离散符号的数量。例如，如果有 10000 个单词，则 num_embeddings=10000。
- embedding_dim (int): 每个嵌入向量的维度。例如，如果希望每个单词用 128 维的向量表示，则 embedding_dim=128。
- padding_idx (int, 可选): 如果指定，则该索引对应的嵌入向量始终为零向量。常用于填充符号（如 <PAD>）。
- max_norm (float, 可选): 如果指定了值，则会限制嵌入向量的范数不超过该值。
- norm_type (float, 默认 2.0): 计算范数时使用的类型，默认为 L2 范数。
- scale_grad_by_freq (bool, 默认 False): 是否根据词频缩放梯度。如果设置为 True，则出现频率较低的词对梯度的影响更大。
- sparse (bool, 默认 False): 是否使用稀疏梯度更新。如果为 True，可以节省内存，但仅支持某些优化器（如 SGD）。