注意力机制学习过程

**PyTorch实现多头注意力机制与KV Cache理解**

1.题前准备

**了解注意力机制**

1、【查询】:输入查询(query,Q),比如“笔记本”

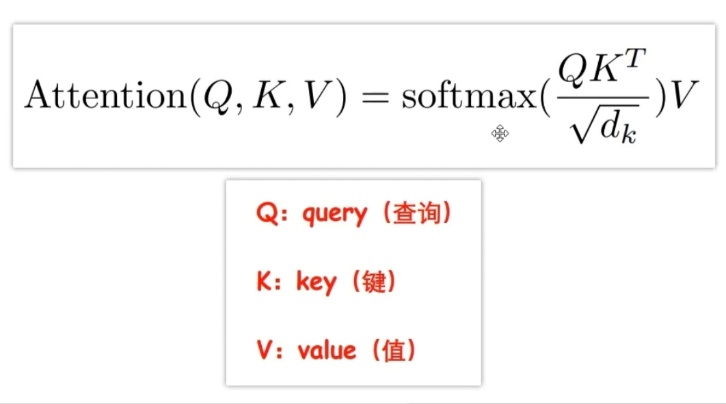
2、【计算相似性】:后台拿到这个查询，并用这个查询Q去和后台的所有的商品的关键字(title)(key,K)一的来对比，找到物品和我们查询的相似性(或者说物品对应的相似性的权重)，相似性越高，越可能推送给我们

3、【得到价值】:并且这个时候还要考虑物品的价值(value,V),这个V不是指物品值几块钱，而是这个物品在算法中的价值。如果商品物美价廉，评论好，点赞高，购买多，等等，那么算法就越有可能把物品排在前面推送给我们。

4、【计算带权重的价值】:我们拿刚刚的相似性，乘上物品在算法中的价值V，计算结果就是每件物品的最后的带相似性权重的价值，最后的算法就是返回这个带权重的价值，也就是把排好序的这些商品推送给我们。

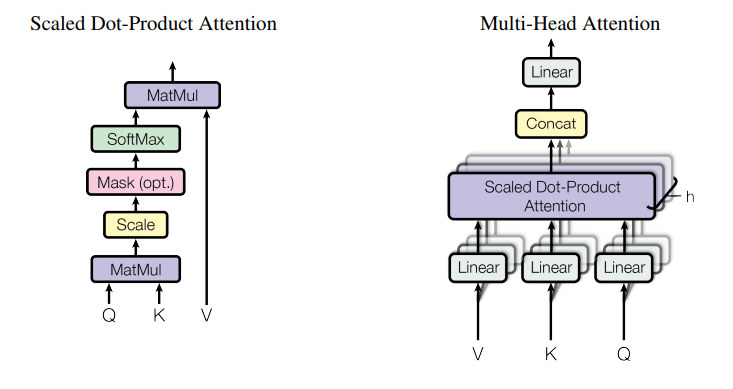
这就是一个最典型的注意力的过程。即推送在最前面给我们的商品，肯定就是它最希望获得我们注意力的商品。

那么注意力计算公式即为



多头注意力即将输入维度分割为多头运用矩阵乘法并行执行

即以下过程



**了解KV Cache**

KV Cache 是解码器在生成式任务中的一种优化技术，其核心思想是：

1.避免重复计算：在自回归生成过程中，每次生成新token时，历史token的K和V矩阵保持不变。

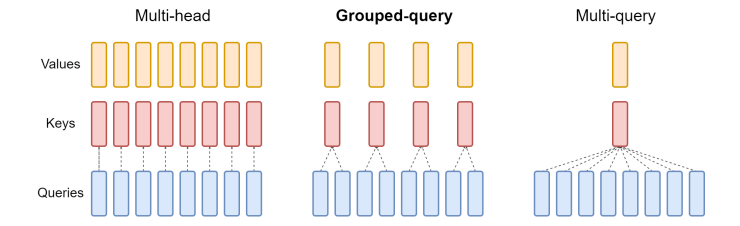
2.缓存机制：将已计算的K和V存储在缓存中，后续步骤只需计算当前token的Q与缓存的K、V交互。

3.节省计算量：将复杂度从O(n²)降低到O(n)，显著提升长序列生成效率。

**了解MHA、GQA、MQA对比**

|  |  |  |  |
| --- | --- | --- | --- |
| 方法 | 核心思想 | 优势 | 适用场景 |
| MQA | 所有查询头共享同一组键值对 | KV Cache减少H倍 | 极度关注推理效率的场景 |
| GQA | 分组共享键值对（如每组4个头共享1组KV） | 平衡性能与显存 | 需要权衡效果与资源的场景 |
| MHA | 每个头独立计算键值对 | 模型容量最大 | 训练阶段或计算资源充足的场景 |

即以下过程



学习资料

https://www.bilibili.com/video/BV1nL4y1j7hA/?share\_source=copy\_web&vd\_source=bf1feac98319a1bc5037b3043ddff954

Bangumi评论分数预测器的训练学习过程

**1. 数据预处理流程**

* **文本清洗**：
  + 保留中文字符：[^\u4e00-\u9fff] 正则表达式过滤非中文内容
  + 统一转换为小写：消除大小写差异
* **长度过滤**：
  + 统计文本中有效中文字数
  + 移除长度 <4 或 >128 的文本，保证输入有效性
* **评分标准化**：
  + 对评分进行 Z-score 标准化：(原始值 - 均值)/标准差
  + 目的：加速模型收敛，提升数值稳定性

**2. 模型架构设计**

* **基于 BERT 的回归模型**：
  + 使用 BertForSequenceClassification 架构
  + 修改 num\_labels=1 适应回归任务
  + 输出层直接预测标准化后的评分值
* **损失函数**：
  + 均方误差损失 (MSE)：torch.nn.MSELoss()
  + 适合连续值预测任务
* **早停机制：**
  + 监控验证集损失，连续 5 次不改善则终止训练
  + 保存最佳模型参数 (best\_model.pth)