**技术报告（偏向于实践）**

**如何使多模态大模型具备推理能力？**

**问题拆解：**

**1.多模态大模型的技术实现**

**1.1：Embedding（视频，音频等数据映射同一空间）**

**1.2：面对复杂高维的向量的解决办法（ANN: HNSW, PQ）**

**1.3：向量数据库**

**2.推理能力的技术实现**

**2.1：prefill decoding KV\_Cache的实现（编码器与解码器）**

**2.2：交叉注意力（让模态的Query关注另一个模态的Key-Value。）**

**2.3：transformer的三大关键机制**

**1.多模态大模型的技术实现**

**1.1：Embedding（视频，音频等数据映射同一空间）**

通用嵌入模型（如GPT-4 Embeddings）

技术框架

文本分支：直接使用LLM文本嵌入（如text-embedding-3-large）。

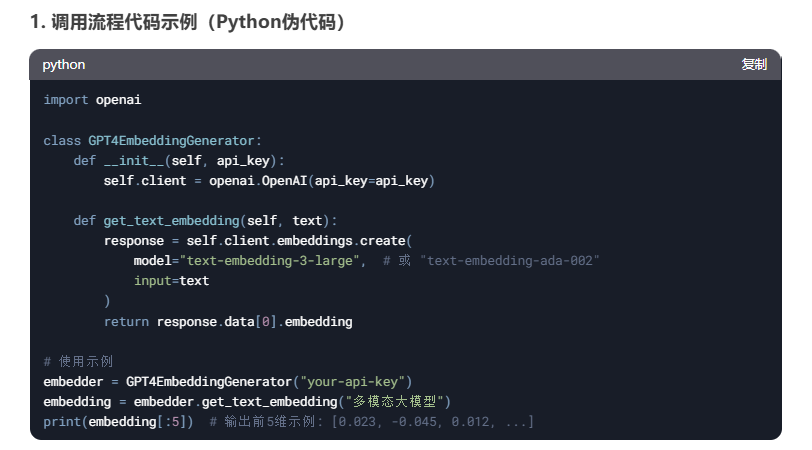
图像分支：外接视觉编码器（如CLIP-ViT）并微调投影层。

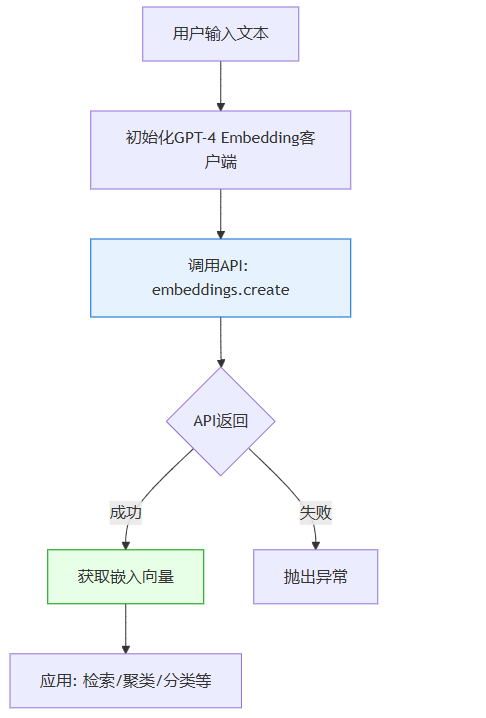
空间对齐：通过少量配对数据微调投影层，使图像向量与文本向量对齐。

关键点

代表实现：GPT-4V（扩展文本嵌入到多模态）。

灵活性：可兼容不同模态编码器的组合。





**1.2：面对复杂高维的向量的解决办法（HNSW,PQ）**

HNSW：上层用于模糊搜索，下层用于精确搜索

1. 核心思想

HNSW是一种用于\*\*高效近似最近邻搜索（ANN）\*\*的图结构算法，核心思想是：

分层导航：

构建多层图结构，上层是下层的稀疏“高速路”，下层是稠密“细节路”。

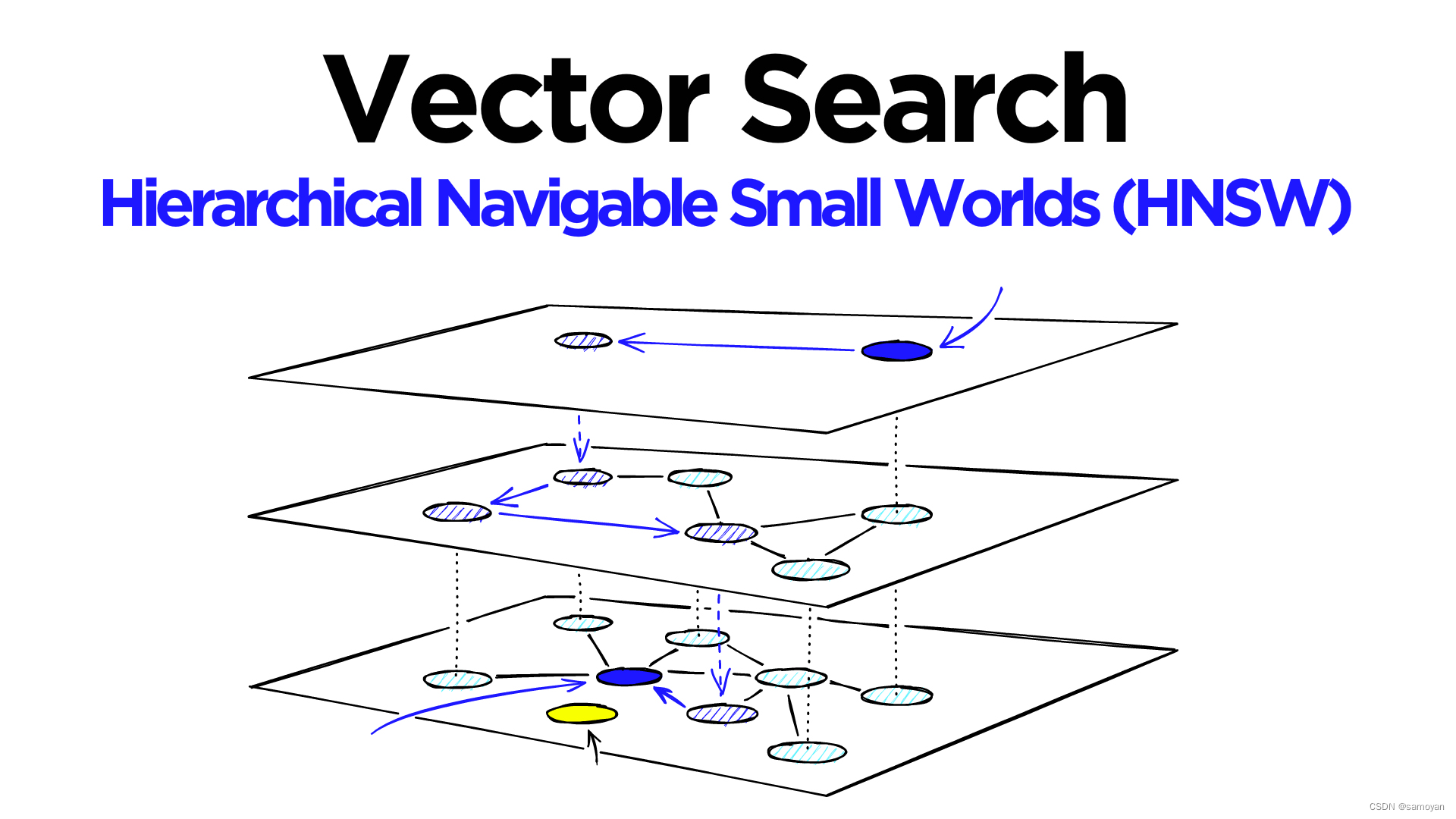
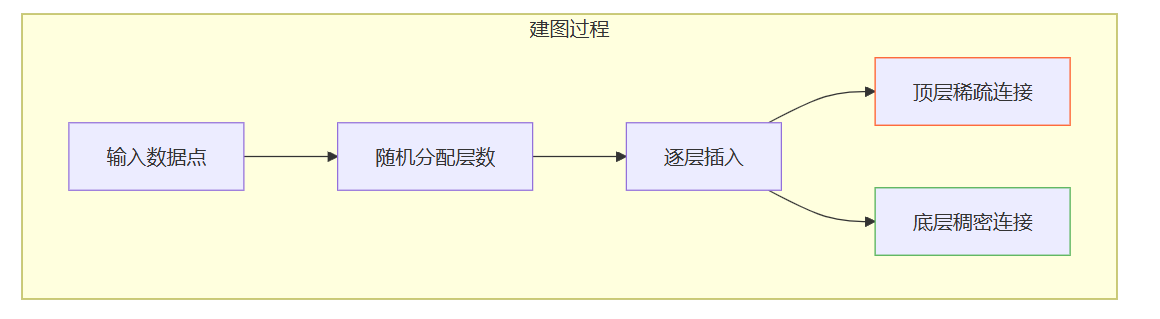
搜索时从顶层开始，逐步向下层细化，快速逼近目标。

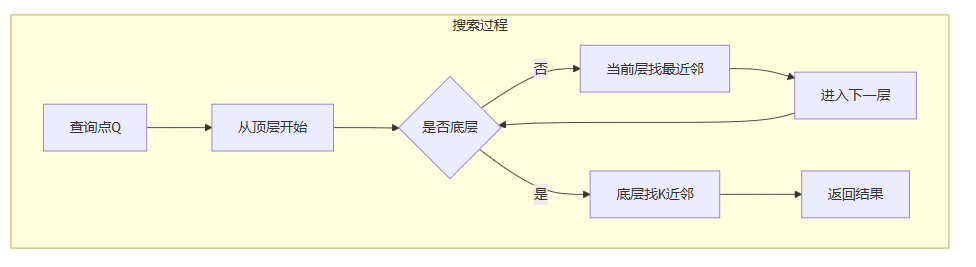
小世界特性：

每层图中每个节点与少量邻居连接，但任意两节点间只需少量跳跃即可到达（类似社交网络的“六度空间”）。

启发式搜索：

基于贪心策略，每次选择距离目标更近的邻居移动。



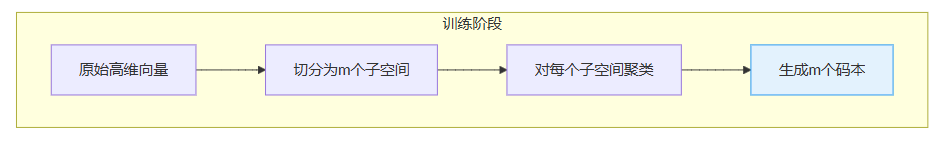
PQ：高维向量转化成低维向量

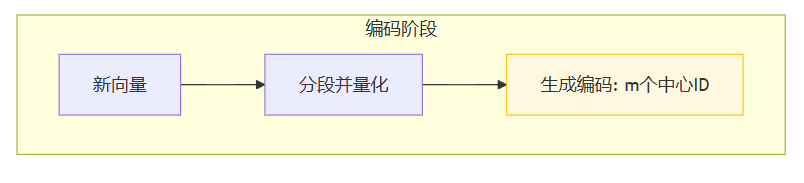
1. 核心思想

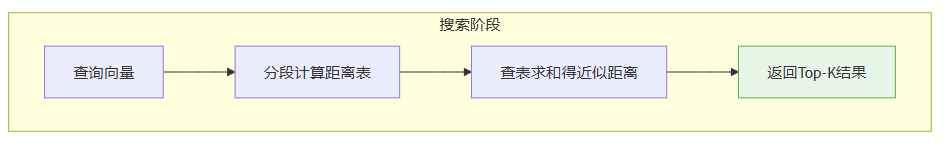
PQ算法通过分段量化和笛卡尔积组合，将高维向量压缩为紧凑的编码，从而高效实现近似最近邻搜索（ANN）。其核心分为两步：

向量分段量化：将高维向量切分为子空间，每个子空间独立聚类生成码本。

笛卡尔积组合：通过子空间码本的组合表示原始向量，大幅降低计算复杂度。







**1.3：向量数据库**

1. Milvus 核心功能

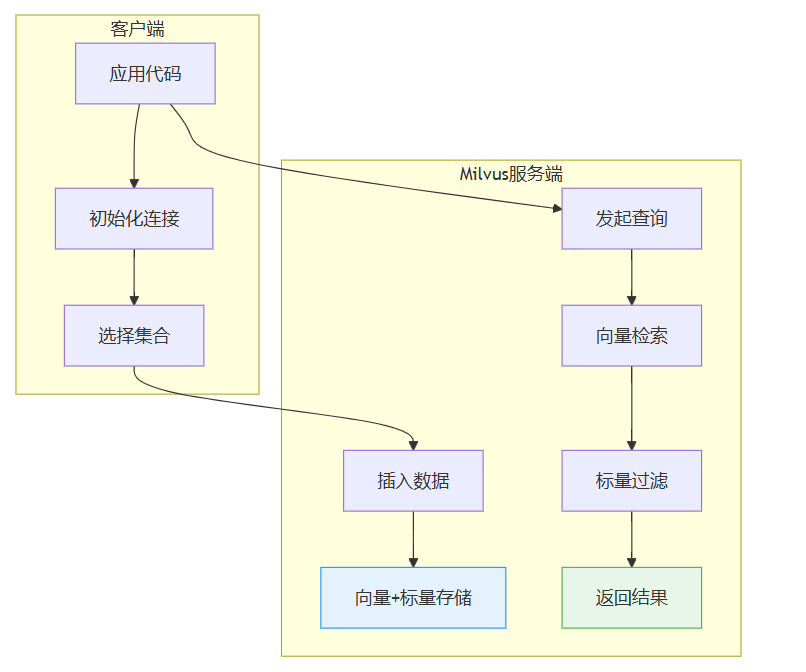
向量检索：支持ANN搜索（HNSW、IVF、PQ等算法）。

标量过滤：结合结构化数据（如ID、标签）进行混合查询。

分布式架构：支持水平扩展，处理亿级向量。





调用的类流程图如下：  
  
  
**2.推理能力的技术实现**

**2.1：prefill decoding KV\_Cache的实现（编码器与解码器）**

1. 编码器（Encoder）工作过程

输入句子："the cat sat on the"

步骤说明：

输入表示：

Token化：将句子拆分为Token序列：["the", "cat", "sat", "on", "the"]。

词嵌入：每个Token转换为向量（如"cat" → [0.2, -0.3, ..., 0.7]）。

位置编码：添加位置信息（如"cat"在位置2 → 向量 + [sin(2), cos(2), ...]）。

多层自注意力：

Self-Attention计算：

每个Token生成Query（Q）、Key（K）、Value（V）矩阵。

示例："sat"的Q会与所有Token的K计算权重，发现"cat"权重最高（主语-动作关系）。

输出更新：加权求和所有Token的V，生成上下文感知的向量。

更新后："sat"的向量包含"cat"和"the"的信息。

前馈网络：

对每个Token的向量做非线性变换，增强表达能力。

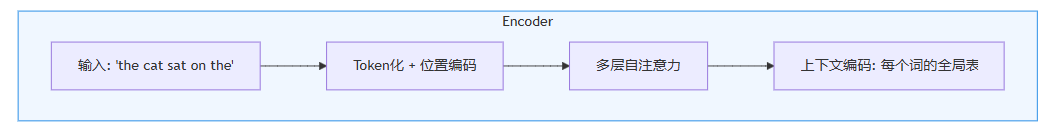
输出上下文编码：

每个Token的最终向量包含全局上下文信息。

示例输出：

"cat"的编码向量：[0.5, -0.1, ..., 0.3]（已知它是"sat"的主语）。

"the"的编码向量：[0.2, 0.4, ..., -0.2]（已知它修饰"cat"或"mat"）。



2. 解码器（Decoder）工作过程

目标：预测缺失词 "mat"

步骤说明：

初始输入：编码器的输出 + 起始符 <bos>。

自回归生成（逐Token预测）：

第1步：输入<bos> → 预测"the"（概率最高）。

第2步：输入<bos> + "the" → 预测"cat"（结合编码器中"the cat"的上下文）。

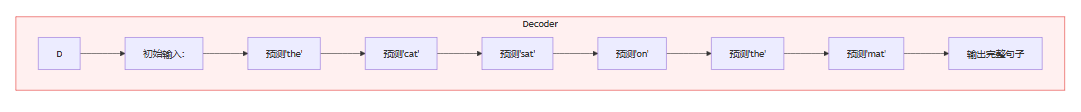
第3步：输入<bos> + "the" + "cat" → 预测"sat"（编码器提示"cat sat"是合法组合）。

第4步：输入<bos> + "the" + "cat" + "sat" → 预测"on"。

第5步：输入<bos> + "the" + "cat" + "sat" + "on" → 预测"the"。

第6步：输入<bos> + "the" + "cat" + "sat" + "on" + "the" → 预测"mat"（编码器提示"on the mat"常见）。

终止条件：生成<eos>或达到最大长度。



**2.2：交叉注意力（让模态的Query关注另一个模态的Key-Value。）**

1. 交叉注意力工作流程（以Decoder预测缺失词为例）

场景设定：

编码器输出：已编码的上下文向量（如 "the"、"cat"、"sat" 的向量表示）。

解码器输入：已生成的部分序列 <bos> + "the" + "cat" + "sat" + "on" + "the"，准备预测 "mat"。

步骤说明：

生成Decoder的Query（Q）：

当前解码器Token（如 "the" 的最后一个位置）生成 Q\_decoder。

获取Encoder的Key-Value（K-V）：

编码器输出的所有Token向量作为 K\_encoder 和 V\_encoder。

计算注意力权重：

Q\_decoder 与 K\_encoder 计算点积，得到权重：

"the"（解码器）可能关注编码器的 "on" 和 "the"（介词短语部分）。

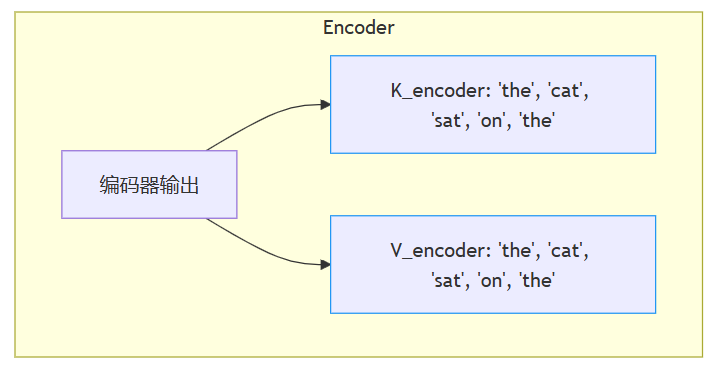
加权求和Encoder的Value：

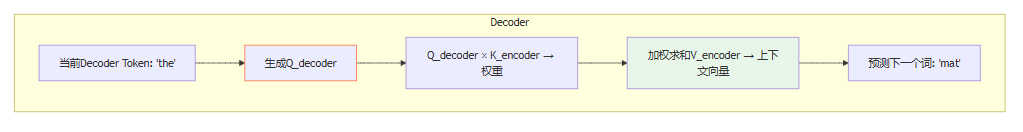
用权重对 V\_encoder 加权，生成上下文向量：

例如：0.6\*V\_on + 0.4\*V\_the。

输出融合结果：

上下文向量与解码器自身表示合并，用于预测下一个词（如 "mat"）。





**2.3：transformer的三大关键机制**

1. 自注意力（Self-Attention）处理过程

输入：Token序列 ["the", "cat", "sat", "on", "the"]

步骤：

·生成Q/K/V：

每个Token通过线性变换生成三组向量：

"cat" → Q\_cat, K\_cat, V\_cat

"sat" → Q\_sat, K\_sat, V\_sat

·计算注意力权重：

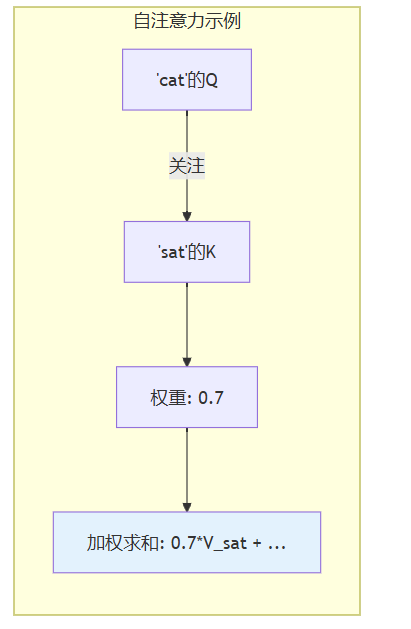
Q\_cat 与所有 K（包括自身的 K\_cat）计算点积，得到权重：

"cat" 与 "sat" 的权重较高（主语-动作关系）。

·加权输出：

对 V 加权求和，生成新表示：

"sat" 的新向量 = 0.6×V\_cat + 0.3×V\_the + ...



2. 多头注意力（Multi-Head Attention）处理过程

输入：同一组Token的嵌入向量

步骤：

并行多头计算：

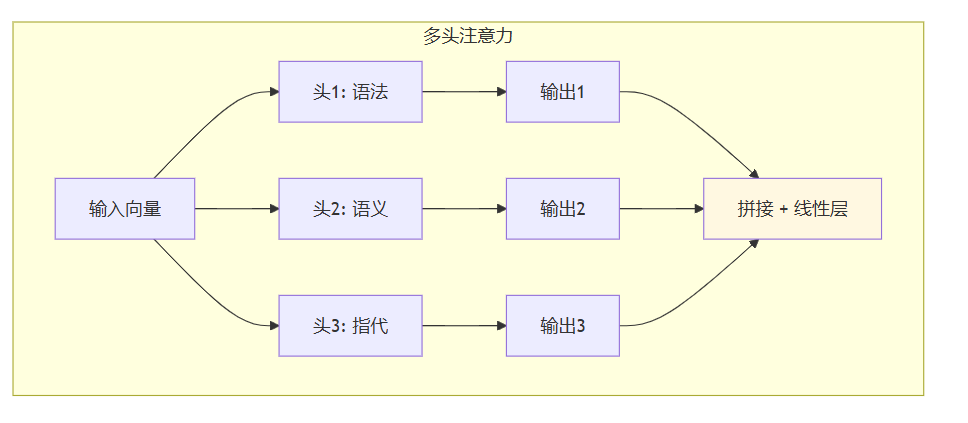
头1（语法）："cat" 关注 "sat"（主谓关系）。

头2（语义）："sat" 关注 "on"（动作-位置关系）。

头3（指代）："the" 关注 "cat" 或 "mat"（冠词修饰）。

拼接多头输出：

合并所有头的输出，通过线性层调整维度。



3. 位置编码（Positional Encoding）处理过程

输入：Token嵌入向量（无位置信息）

步骤：

生成位置编码：

对位置 pos=2（"cat"）计算正弦/余弦函数值：

PE(pos=2) = [sin(2/10000^(0/512)), cos(2/10000^(0/512)), ...]

添加到词向量：

"cat" 的最终输入 = 词嵌入 + PE(2)

