1. 也就是q和k计算相似度时需要考虑相对位置，即便是相同的两个向量，因为位置不同相似度也不同吗?
   如果是这样，那么是不是要求每一个，注意是每一个计算q和k的x一定是语义的向量，如果不是语义向量，q向k岂不是查了个寂寞？
2. 从最原始的是随机embedding开始，经过一些列层，这其中每一个，注意是每一个计算q和k的x，还是语义向量吗，他为什么不是中间过程一些无意义的向量，为什么?
3. 还是说x经过变换的每一步都是语义向量，只不过刚开始不准确，为什么? 还是说一开始的计算q和k也是有用的信息很少，但随着训练，越来越能表征语义

你这个问题非常关键，几乎触及 Transformer 内部 **向量语义演化的本质**。我来仔细梳理一遍，从头到尾，把你关心的"Q/K 的向量什么时候有语义、为什么有语义、训练如何演化"全讲明白。

---

1 Q/K 的语义不是一开始就有的

假设你模型刚初始化，embedding 和线性层都是随机的：

- **token embedding**：随机向量或者小的高斯初始化 → 基本没有语义
- **线性映射 Wq/Wk**：随机 → Q/K 也是随机向量

此时 **Q·K 计算出的注意力矩阵**也几乎是随机的，确实可以说"查了个寂寞"。

◇ **结论**：刚开始训练时，Q/K 的向量不具备语义，点积只是随机相似度。

---

2 随着训练，Q/K 向量逐渐有语义

Transformer 的训练是 **梯度从 loss 反向传回 Q/K 的生成网络**：

1. loss 告诉模型：某些 token 的输出要预测成某个正确值
2. 反向传播梯度经过 Q/K 的线性层和 embedding
3. 权重更新 → Q/K 的向量逐渐**编码出对任务有用的特征**

结果：

- 同义词 token 的 Q/K 越来越接近
- 不相关 token 的 Q/K 方向越来越远
- 注意力矩阵逐渐体现真实语义相关性

换句话说：

> Q/K 向量 **一开始无语义，随着训练慢慢成为"任务相关的语义向量"**。

---

3 为什么中间层的 Q/K 也有语义？

Transformer 有多层注意力，每一层都会产生自己的 Q/K/V：

1. 第一层：token embedding → Q/K → attention
    - Q/K 是最原始语义 → 低级特征（比如单词或 subword 的局部信息）
2. 第二层：上一层输出 → Q/K → attention
    - Q/K 结合了上下文 → 高级语义
3. 第 N 层：逐渐抽象 → 句子级、段落级语义

☑ **核心原因：**

- 每一层 Q/K 都是前一层输出经过线性变换得到
- 前一层输出已经融合了部分上下文信息（经过 attention 加权求和）
- 因此 Q/K **天然带有语义向量，只是语义级别逐层递进**

---

4 为什么每个位置的 Q 都要是语义向量？

因为 Q·K 计算的是**相似度/匹配程度**：

$$
\text{Attention}(i,j) \sim Q_i \cdot K_j
$$

- 如果 Q_i 或 K_j 根本不含有语义 → 点积毫无意义 → attention 权重随机
- 训练的目标就是让 Q/K 逐渐学到"谁应该关注谁" → 注意力矩阵才有用

> 换句话说，Transformer 的训练本质是 **学习 Q/K 的线性变换，让 dot-product 表征语义相关性。**

---

5 随着训练的演化过程

可以用时间线描述：

训练阶段	Q/K 特征	注意力矩阵
初始化	随机向量	随机、无规律
前几步	薄弱特征	注意力有些噪声，但能稍微区分某些 token
中期训练	局部语义	attention 开始聚焦在有用 token 上
后期训练	高级语义	attention 几乎捕捉到长距离依赖、语法/语义关系

---

6 总结一句话理解

1. **一开始** Q/K 是随机向量 → 没有语义
2. **训练中** Q/K 学习线性变换 → 越来越表征"token 在上下文中的语义"

3. **每一层的 Q/K** 都是有语义向量，只是语义抽象层级不同
4. **注意力点积**依赖 Q/K 的语义，否则注意力就像“盲点对盲点”

> 换句话说，Transformer 训练的核心，就是让 Q/K 成为**上下文敏感的语义向量**，这样 Q·K 才能“查到有用信息”。