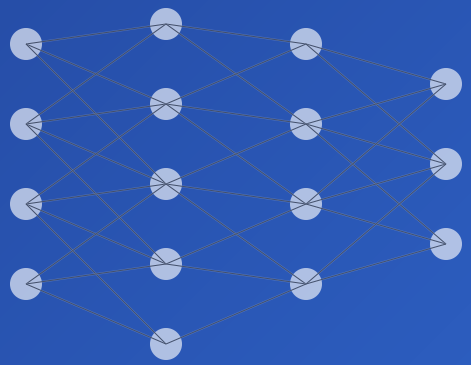
Transformer模型中的Embedding概念

深入理解NLP模型的基础表示方法

自然语言处理 | 深度学习 | 向量表示



什么是Embedding?

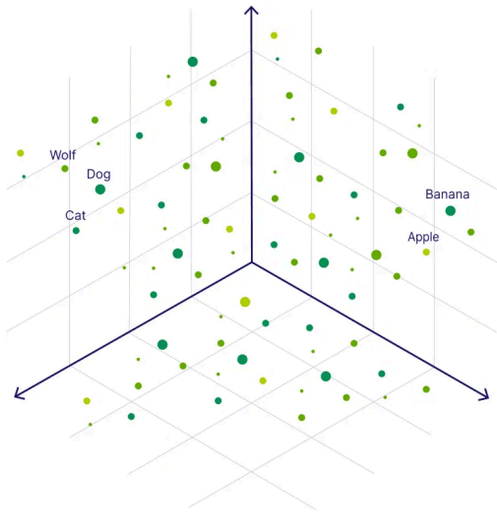
Embedding是将离散符号（如单词、字符或标记）映射到连续向量空间的过程。

在自然语言处理中，Embedding将文本转换为计算机可以理解和处理的数值表示。

通过Embedding，语义相似的词在向量空间中的距离更近，使模型能够捕捉词语之间的关系。

为什么需要Embedding?

- 计算机无法直接处理文本，需要数值表示
- 传统的独热编码（One-hot）无法表达语义关系
- 降低维度，解决稀疏性问题
- 捕捉词语之间的语义相似性和关系
- 为深度学习模型提供更有意义的输入表示

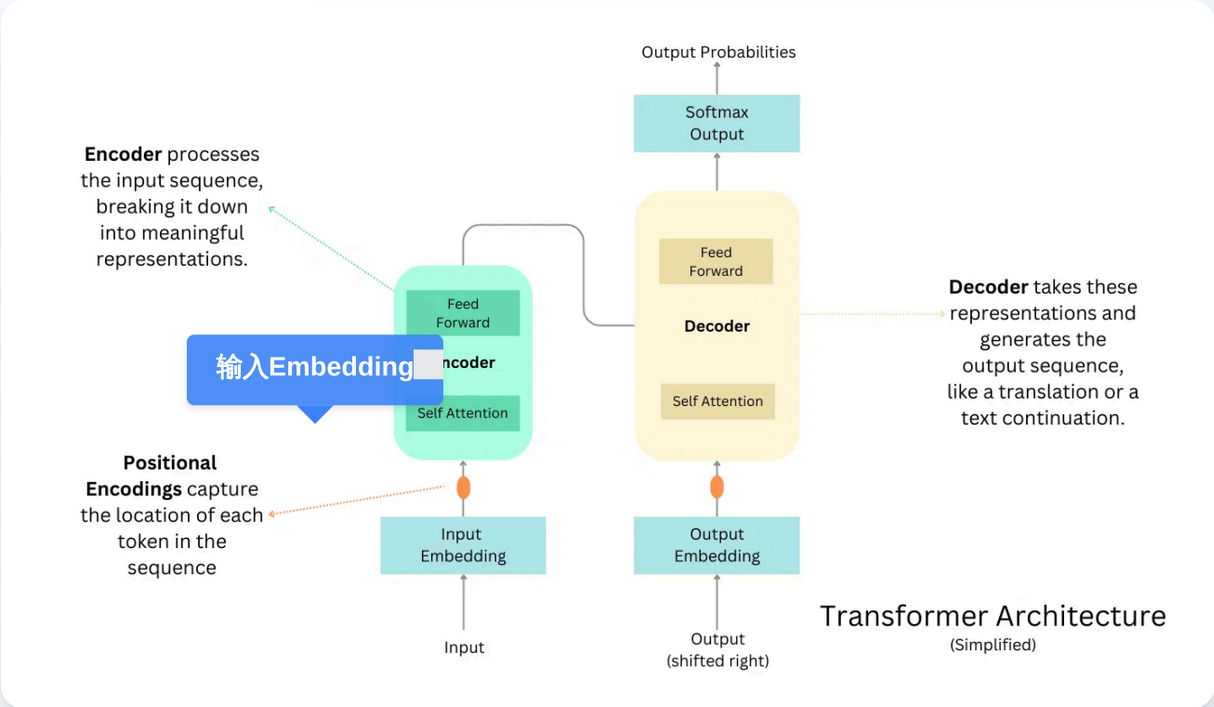


词向量空间示意图：语义相近的词在空间中距离更近

从离散符号到连续向量

"猫" → [0.2, 0.8, -0.1, ...]

Embedding在Transformer中的位置



在Transformer架构中，**Embedding层**是模型的第一个组件，负责将输入的离散标记转换为连续的向量表示。

Embedding在Transformer中的关键作用

- 将输入标记（如单词）转换为固定维度的向量
- 捕捉词语之间的语义关系
- 为自注意力机制提供初始输入
- 与位置编码结合，提供序列信息

Embedding层的输出直接传递给Transformer的编码器或解码器堆栈，经过多头自注意力机制和前馈神经网络处理。

Embedding与其他组件的关系

Embedding	→ 提供初始向量表示
自注意力机制	→ 处理标记间的关系
前馈神经网络	→ 进一步转换特征

Transformer中的三种Embedding

Transformer模型中通常使用三种不同类型的Embedding，它们共同构成了模型的输入表示。

Token Embedding

将词汇表中的每个标记（单词或子词）映射为固定维度的向量，捕捉语义信息。

Position Embedding

编码序列中标记的位置信息，使模型能够理解词序，可以是固定的或可学习的。

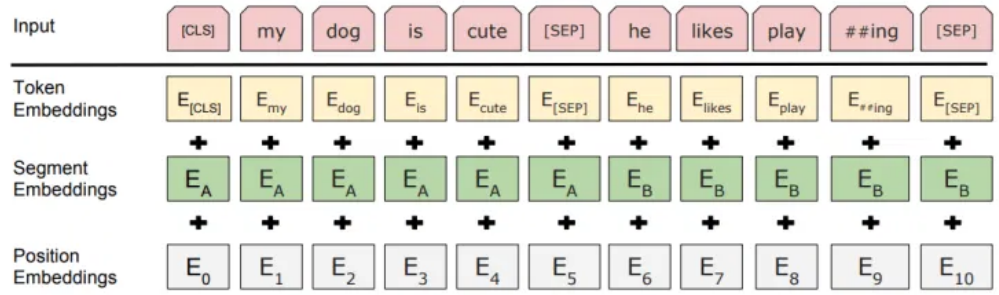
Segment Embedding

主要用于BERT等模型，区分不同句子或段落，支持多句子输入任务（如问答、文本蕴含）。

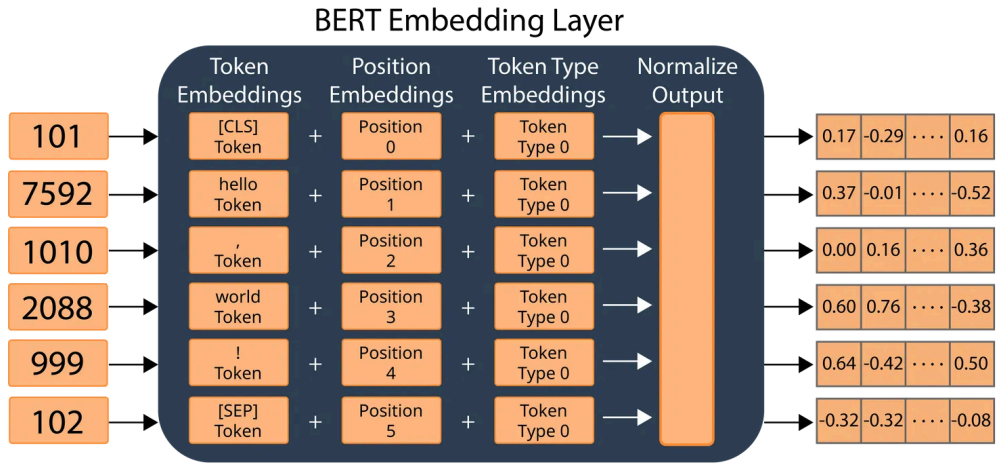
Embedding的组合

最终的输入表示是这三种Embedding的元素级相加，形成丰富的向量表示：

```
Input = Token + Position + Segment
```



三种Embedding的组合示意图



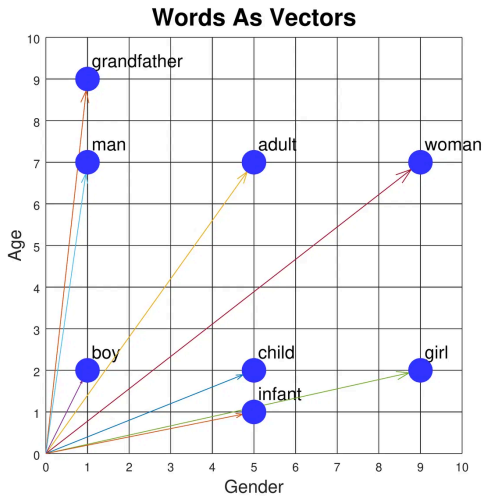
BERT模型中的Embedding层结构

Token Embedding详解

Token Embedding是Transformer模型中最基础的嵌入类型，负责将离散的词汇标记转换为连续的向量表示。

Token Embedding的关键特性

- **词汇表大小**：通常包含数万个标记（10k-50k）
- **向量维度**：常见维度为256、512或768
- **可学习参数**：在训练过程中不断优化
- **语义编码**：相似词汇在向量空间中距离更近



词向量空间中的语义关系示例

Token Embedding本质上是一个查找表（Look-up Table），将词汇表中的每个标记映射到一个唯一的向量。

```python
# Token Embedding的简化实现
class TokenEmbedding(nn.Module):
    def __init__(self, vocab_size, d_model):
        super().__init__()
        # 创建嵌入矩阵
        self.embedding = nn.Embedding(
            vocab_size,  # 词汇表大小
            d_model      # 嵌入维度
        )
        self.d_model = d_model

    def forward(self, x):
        # 查找并返回对应的嵌入向量
        return self.embedding(x) * math.sqrt(self.d_
```

Token Embedding的训练过程

1. **随机初始化**
   每个标记的向量最初是随机生成的
2. **反向传播**
   通过模型训练过程中的梯度更新调整向量
3. **语义捕捉**
   随着训练进行，向量逐渐捕捉词语的语义关系
4. **语义空间形成**
   最终形成一个语义丰富的向量空间

Position Embedding详解

在Transformer模型中，**Position Embedding**解决了序列顺序信息的编码问题，使模型能够理解词语在句子中的位置。

为什么需要位置编码？

与RNN或LSTM不同，Transformer的自注意力机制是**并行**处理所有输入标记的，这导致模型无法自然地感知序列顺序。

没有位置信息，模型将无法区分以下句子的不同：

"猫追狗" vs "狗追猫"

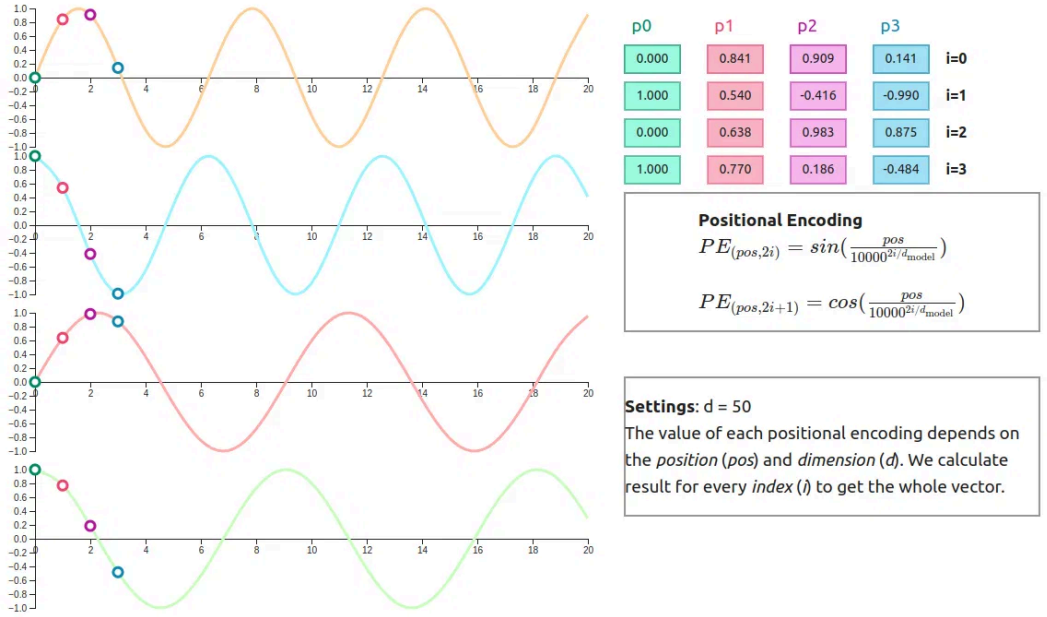
位置编码的两种主要方法

1. 固定式位置编码（原始Transformer）

使用正弦和余弦函数生成，不需要学习参数

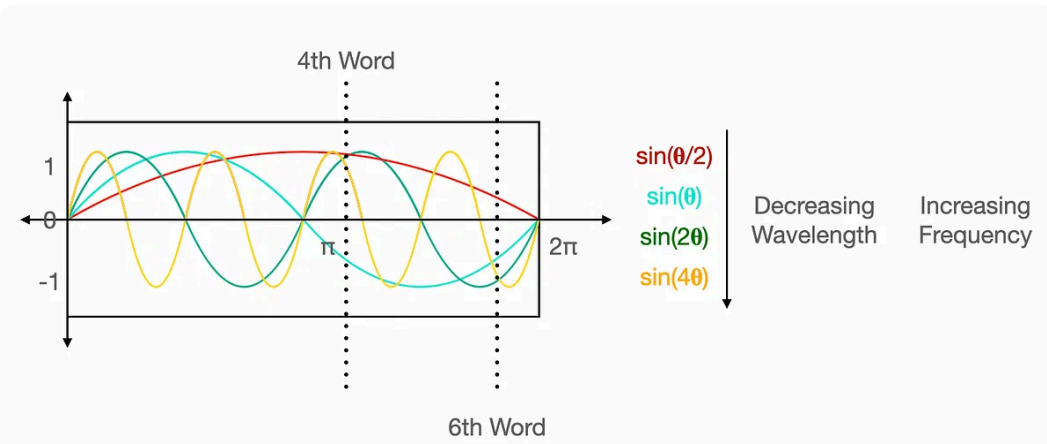
2. 可学习的位置编码（BERT等）

通过训练学习最优的位置表示



位置编码的正弦/余弦模式可视化

固定式位置编码公式



特点：

- 使用不同频率的正弦和余弦函数
- 每个位置有唯一的编码模式
- 相对位置关系可以通过线性变换学习
- 可以扩展到训练中未见过的序列长度

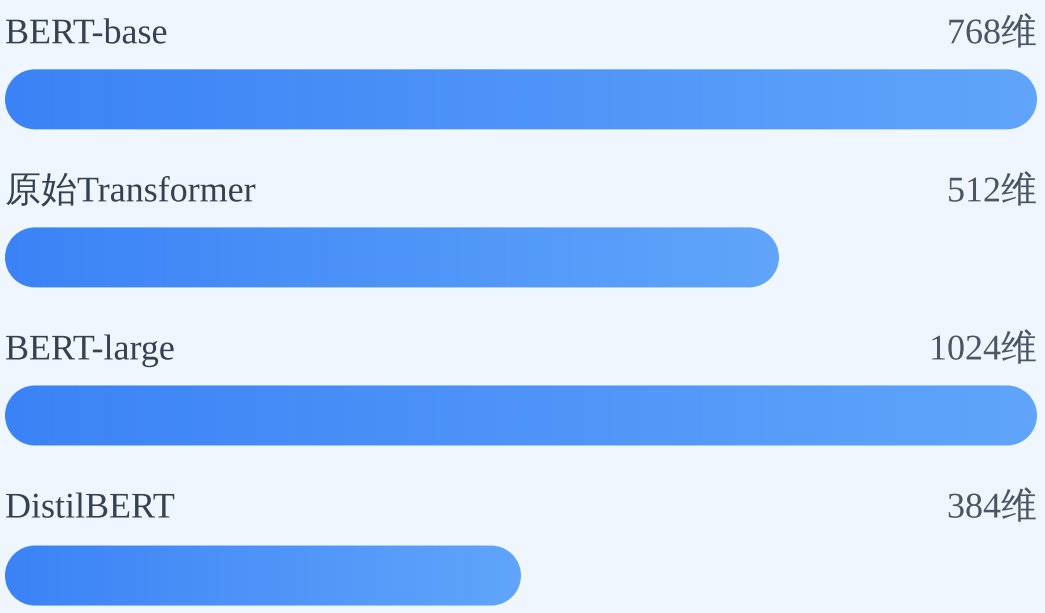
位置编码的特性

唯一性　可扩展性　相对关系　平滑过渡

Embedding的维度与计算

Embedding的维度选择对模型性能和计算效率有重要影响，需要在表达能力和计算成本之间取得平衡。

常见的Embedding维度



维度选择的考量因素

表达能力

更高维度可以捕捉更复杂的语义关系

计算成本

维度增加导致计算量和内存需求增加

数据量

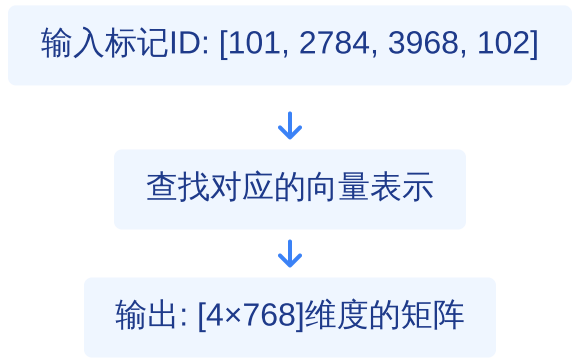
更高维度需要更多训练数据

任务复杂度

复杂任务可能需要更高维度

Embedding矩阵的形状与计算

Embedding查找过程



计算复杂度考虑

- 参数量 = 词汇表大小 × 嵌入维度
- BERT-base: 30,000 × 768 ≈ 2300万参数
- 查找操作计算复杂度为O(1)
- Embedding层通常占模型总参数量的20-30%

Embedding的可视化

Embedding向量通常具有高维度（如768维），无法直接可视化。通过降维技术，我们可以将高维向量投影到2D或3D空间进行可视化，从而观察词语之间的语义关系。

常用的降维可视化技术

t-SNE

t-分布随机邻域嵌入，保留局部结构，适合可视化聚类关系，但计算成本较高。

PCA

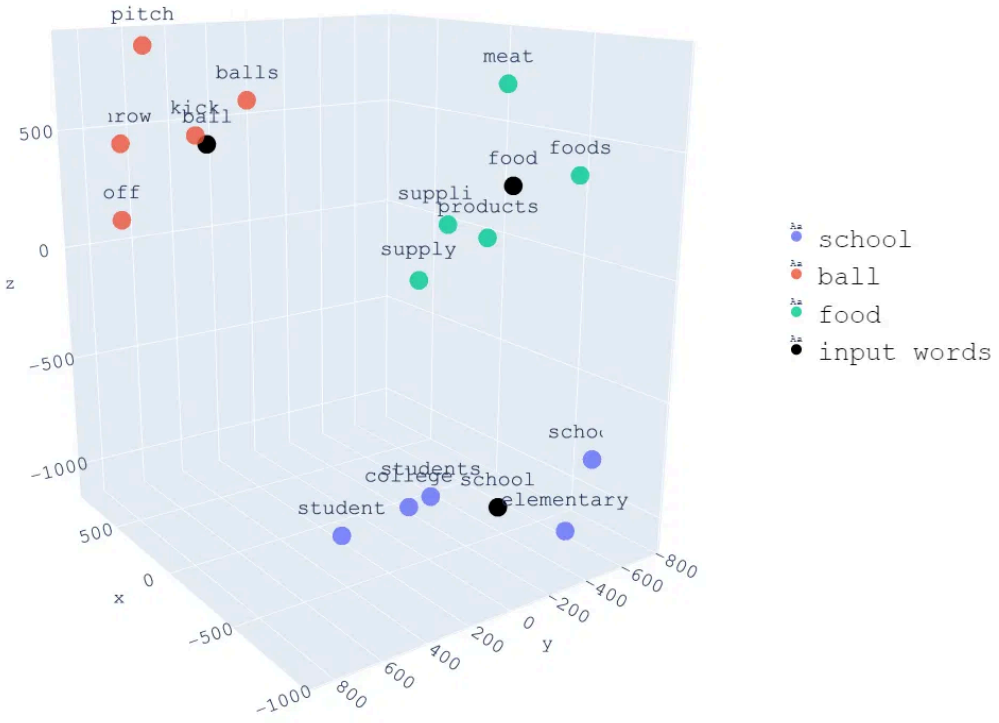
主成分分析，保留全局方差，计算效率高，但可能无法捕捉复杂的非线性关系。

UMAP

统一流形近似和投影，兼顾局部和全局结构，速度快于t-SNE，是近年来流行的可视化方法。

可视化的价值

- 直观理解词语之间的语义关系
- 发现潜在的语义聚类
- 评估Embedding质量
- 识别异常或错误的Embedding
- 辅助模型调试和优化



使用t-SNE降维的词向量可视化示例

可视化中的语义关系示例

同义词聚类

快乐　高兴　愉悦　开心

语义对比

男人　女人　国王　王后

语义渐变

冷　凉　温　热

语法关系

走　走了　走过　走着

Embedding的实际应用

Embedding技术在现代NLP系统中有着广泛的应用，是各种语言模型的基础组件。

预训练与微调

Embedding层在预训练阶段学习通用语义表示，微调阶段可以针对特定任务进一步优化。

迁移学习

预训练的Embedding可以迁移到下游任务，减少数据需求，加速模型收敛。

多语言模型

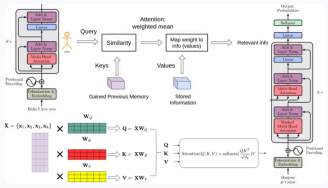
共享的Embedding空间可以捕捉不同语言之间的语义对应关系，支持跨语言任务。

语义搜索

利用Embedding的语义表示能力，实现基于语义而非关键词的搜索系统。

实际应用案例

BERT预训练任务



通过掩码语言模型(MLM)和下一句预测(NSP)任务，BERT学习丰富的Embedding表示。

特征提取与表示学习

使用预训练模型的Embedding层作为特征提取器，为下游任务提供高质量的输入表示。

特征提取

参数共享与知识迁移

参数共享

在多任务学习中，共享Embedding层可以促进任务间的知识迁移，提高整体性能。

Embedding应用的技术挑战

- 处理稀有词和未登录词(OOV)
- 适应领域特定的语义
- 平衡模型大小与性能
- 处理多语言和跨语言场景
- 解决偏见和公平性问题

总结与展望