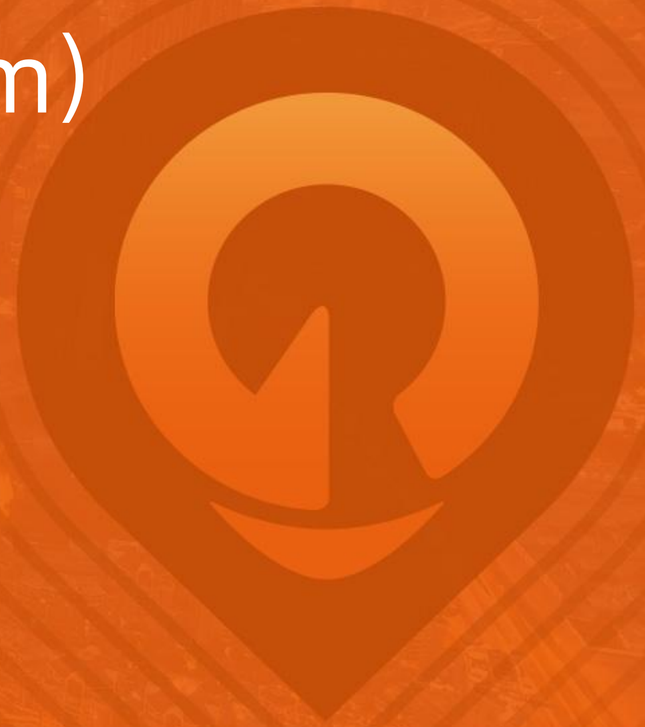
AI 算法进阶（Advanced AI System）



@Tyler

7. 连接主义：Transformer 与大模型

目录

万物皆可向量

极客时间训练营

- 我们如何让机器理解世界?
- **文本：**
  「阳光」 → [0.12, 0.85, –0.23, …]
- **图像：**
  猫的照片 → [0.91, –0.47, 0.33, …]
- **用户行为：**
  你在社交网络上的行为 → [0.27, 0.68, 0.14, …]

- **核心目标：** 将世界上所有事物，无论是具体的还是抽象的，都转化成计算机可以理解和计算的数学对象——**向量**。
- 这个过程，我们称之为 **表示学习 (Representation Learning)**。

自监督学习（Self-Supervised Learning）

极客时间训练营

- **现实困境：**
- **海量原始数据**（文本、图片、语音）触手可及
  精确标签数据极其稀缺 & 昂贵

- 举例：
- 给10万张猫狗图片手动标注 → 成本巨大
- 给每句评论标记情绪 → 主观、耗时

- **传统监督学习的限制：**
- 依赖标签
- 训练数据扩展难
- 通用性不足

自监督学习（Self-Supervised Learning）

极客时间训练营

- **自监督学习（Self-supervised Learning）**
- 一种让模型“自学成才”**的方法**
  它不依赖人类标签，而是从数据本身构造学习目标
  —— 例如预测下一个词、还原被遮挡的图像块……

- **它的核心目标：**
- **在大规模无标签数据上进行预训练（Pre-training）**
  学到语言、图像、行为等的**通用表示与能力**

- 自监督学习 = 用**海量无标签数据**，创造“自提问、自回答”的学习方式
  最终获得一个**强大、可迁移的通用模型**

关键钥匙：“借口任务”（Pretext Task）

极客时间训练营

- **定义：**
- 为了让模型学习，我们**人为设计一些“谜题”**，让模型在**尝试解谜的过程中，掌握数据的规律与结构。**
- 这些谜题就是所谓的：**“借口任务”**（Pretext Tasks）

- 本质：
- **不是为了任务本身的结果，而是借助它来引导模型学习有用的表示。**

静态词嵌入（Word Embedding）

- **代表模型：Word2Vec（2013）**
- 核心思想：**语义相近的词，其上下文也相似，** **示例：**
- “king” → [0.21, –0.51, 0.74, …]
- “queen” → [0.19, –0.47, 0.70, …]
- “apple” → [0.02, 0.93, –0.18, …]

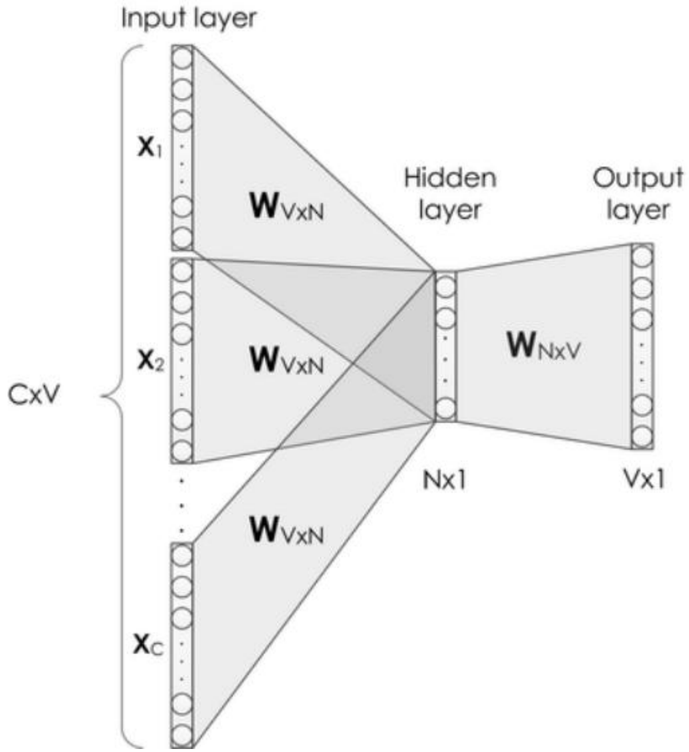
- **核心产出：**为每个词生成一个**固定的向量**，该向量**编码了该词的核心语义**
- 在向量空间中：
- 相似的词彼此接近（如“Paris” ~ “London”）
- 可以做类比运算：
- vector("king") – vector("man") + vector("woman") ≈ vector("queen")

静态词嵌入 (Word Embedding)

- **Word2Vec 的训练方式就是一种典型的“借口任务”：让模型猜词，从而学到词与上下文的语义关系。**

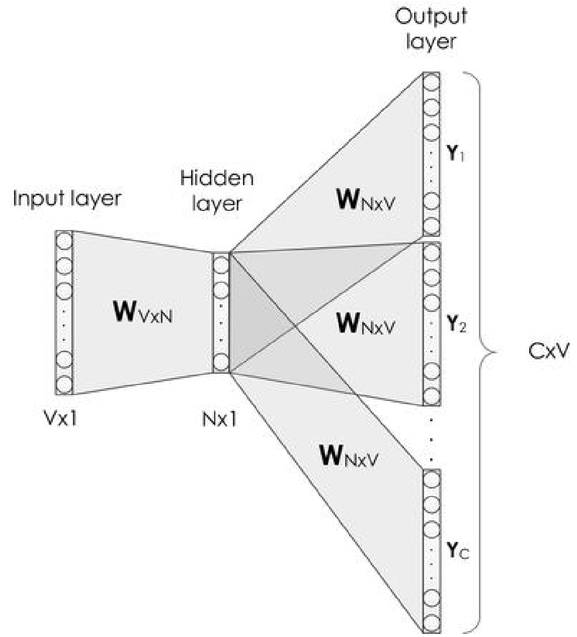
- 它主要有两种方式：

- **CBOW：根据上下文预测中心词**
- 模型看到左右的上下文词，任务是**猜出中间缺失的词**
- 例子：
- 输入：["今天", "天气", "____", "晴朗"]
  输出：预测中心词是 "非常"



静态词嵌入 (Word Embedding)

- **Skip–gram：根据中心词预测上下文词**
- 例子：
- 输入：["非常"]
  输出：预测周围词是 "今天", "天气", "晴朗”



- CBOW 向内看，Skip–gram 向外看；

模型	输入	输出	图示方向
CBOW	上下文词	中心词	外 → 中
Skip-gram	中心词	上下文词	中 → 外

图表示学习 (Graph Representation Learning)

极客时间 训练营

- **文本中的表示学习，只处理一维的词序列。**
  但在现实世界中，我们面临的是更复杂的结构：
- 社交网络、人际关系图
  分子结构图、生物网络
  知识图谱、推荐系统

- **核心问题：**
- 如何将表示学习从线性序列**泛化到复杂的图结构**?

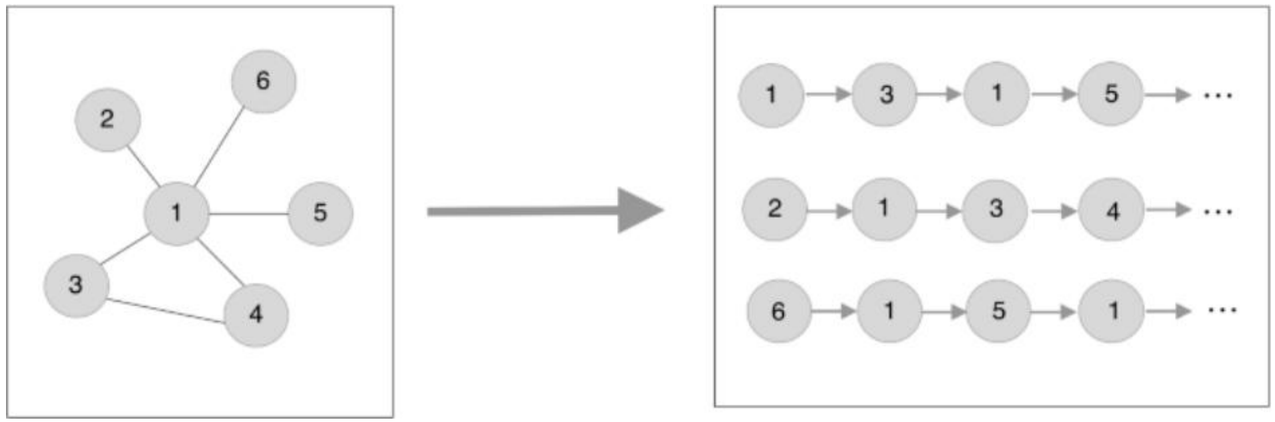
图表示学习 (Graph Representation Learning)

- **代表模型：DeepWalk（2014）**
- **1. 核心目标：为网络中的每个节点绘制“数字肖像”**
- **使命：** 为图中的每一个节点（如用户、商品）生成一个低维、稠密的向量（Embedding）。
- **本质：** 这个向量不是简单的ID，而是节点的“数学身份证”，它必须同时编码两大信息：
  - **自身属性：** 节点自身的特征。
  - **结构角色：** 节点在网络局部邻域中的位置和关系。

图表示学习 (Graph Representation Learning)

极客时间 训练营

- **核心思想：像理解句子一样，理解网络**
- **跨界联想：** DeepWalk的突破在于，它认为**节点在图中的游走序列**，本质上等同于**文本中的单词序列**。
- **关键类比：**
  - **随机游走路径** (Node A –> Node B –> Node C) **<==> 自然语言中的句子** (Word A Word B Word C)
- **方法论：** 既然节点序列 ≈ 句子，那么我们就可以借用成熟的NLP模型（Word2Vec）来学习节点的表示。



图表示学习 (Graph Representation Learning)

极客时间 训练营

- **第一步：数据生成 → 构造“伪句子” (Random Walk)**
- **起点：** 从图中每一个节点开始。
- **游走：** 在其邻居中随机选择下一步，持续进行，形成一条节点序列。
- **产出：** 大量由节点构成的“伪句子”语料库。
  - *示例 (电商网络):* [用户A] → [iPhone 15] → [Apple充电器] → [用户B] → [AirPods Pro]

图表示学习 (Graph Representation Learning)

极客时间 训练营

- **第二步：模型训练 → 学习上下文 (Skip-gram)**
  - **套用框架：** 将上述“伪句子”喂给Word2Vec中的Skip-gram模型。
  - **训练任务（Pretext Task）： 用中心节点预测其上下文节点**。例如，在序列中给定[iPhone 15]，模型需要学习去预测它旁边的[用户A]和[Apple充电器]。
  - **最终产出：** 模型训练得到的副产品——每个节点的向量表示（Embedding）。

- 通过在 Item-User 关系图上进行有偏随机游走，生成了大量的节点序列。这条序列就像一个句子，描述了节点之间的“上下文”关系。 这个场景也催生了后续的 Node2Vec 和 Item2Vec 等技术。

目录

因果语言建模（Causal Language Modeling, CLM）

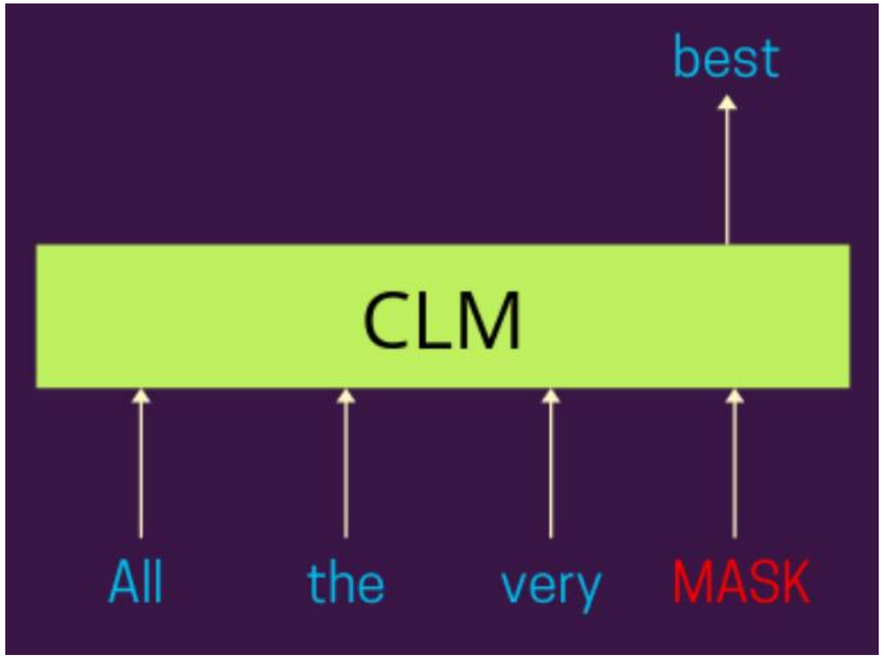
- **CLM（Causal Language Modeling）**
  是 GPT 预训练时使用的**核心借口任务**
- **任务描述：**
  给定前面的所有词，预测下一个最有可能出现的词
  （即学习条件概率：P(下一个词 | 前缀)）

- **输入：**
  “今天天气真不错，我们一起去 ___”
- **模型输出预测：**
- “公园”
- “散步”
- “喝咖啡”

因果语言建模 (Causal Language Modeling, CLM)

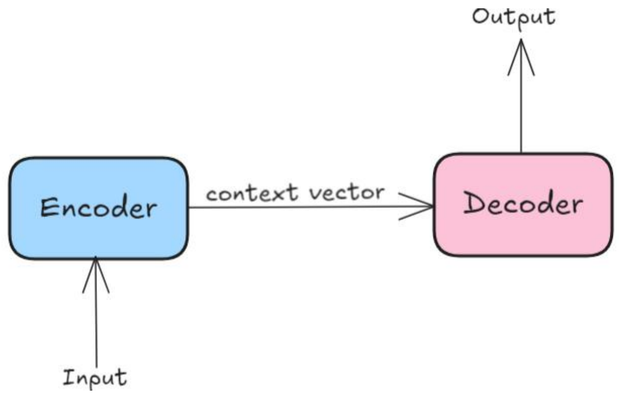
极客时间 训练营

- 目标是：预测最合理的下一个词
- 这个过程没有标签，模型**用海量文本自己制造训练数据**



因果语言建模（Causal Language Modeling, CLM）

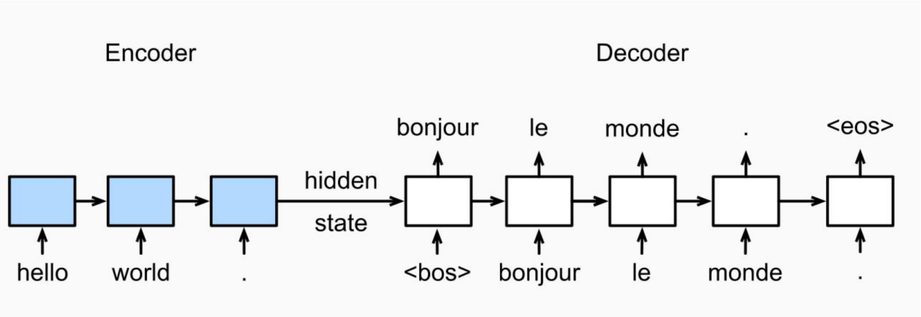
- **为了准确预测下一个词，模型必须学会：**
  - **语法规律**（词法、句法结构）
  - **上下文理解**（主语–谓语一致、代词指代）
  - **事实知识**（“巴黎”是法国首都）
  - **因果逻辑**（“因为…所以…”）
  - **社会常识与语用推理**（“我们一起去…”大概率不是“吵架”）
- **模型甚至隐式构建了一个“记忆宫殿”来支持预测**



- 更显式的记忆建模机制：
  Encoder–Decoder（Seq2Seq）架构
  用于显式建构输入语境与目标输出之间的对齐映射

Encoder–Decoder 架构：显式建模上下文与目标

- **基本结构：**
- ➢ **Encoder：**
- 将**输入序列**编码成一个连续的向量表示（通常为上下文表示）
- 输入：如一句话“我今天很开心”
- 输出：一个压缩后的上下文向量或一组表示（隐藏状态序列）



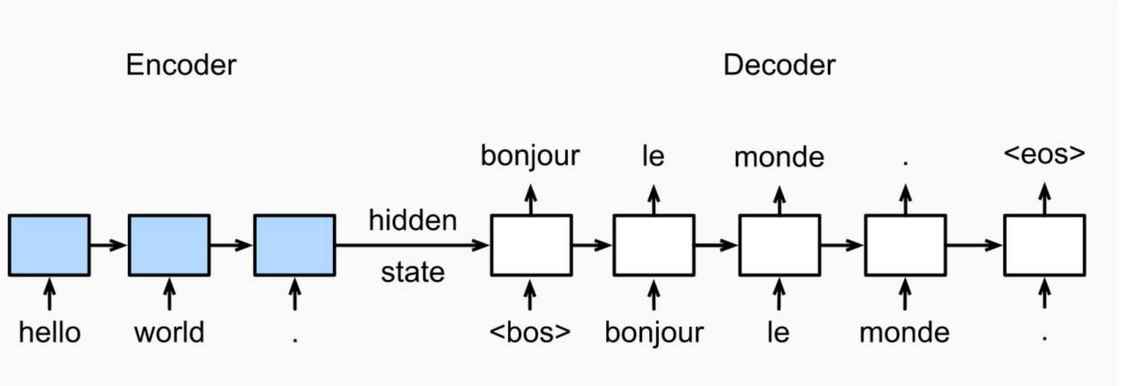
- ➢ **Decoder：**
- 根据**编码结果生成目标结果**
- 输入：编码器输出
- 输出：任务的输出（例如，输入的语种 / 输入的翻译 / 输入的情感 ...）

因果语言建模 (Causal Language Modeling, CLM)

极客时间 训练营

- 看似“简单”的任务，却要求模型全面理解语言背后的深层结构
- 结果：GPT 模型拥有了强大的生成、理解、推理、对话能力

- **上下文建模能力的限制：**
- **隐藏状态（Context Vector）容量有限**
  → 在长文本中容易“遗忘”前文信息
- Decoder 在处理长输入时
  → 记忆模糊、上下文稀释

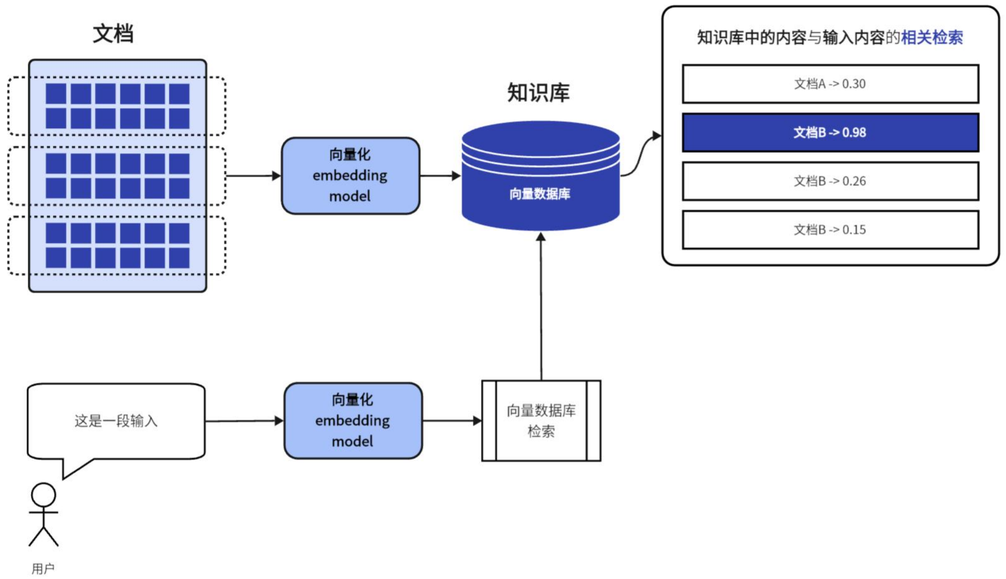


- **一个比喻**
  - 想象一下，您需要将一整本书的内容翻译给另一个人，但您只能给他一份**摘要**。无论摘要写得多好，翻译的质量都将受限于摘要所包含的信息量，大量细节会丢失。

革命性解决方案：注意力机制登场

- 在注意力机制中，编码器（Encoder）的角色发生了根本性转变。它不再是仅仅生成一个单一的、概括性的上下文向量，而是将整个输入序列转化为一个结构化的、可供随时查询的**“键值对(Key-Value)数据库”**。

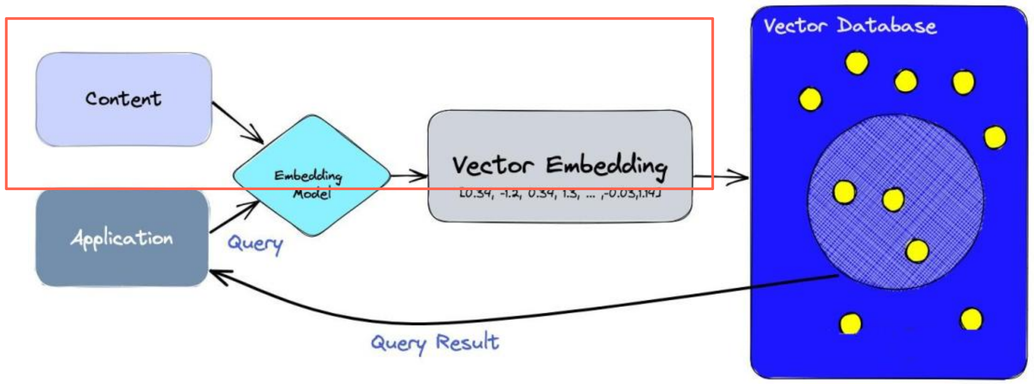
- 注意力查询的过程，与现代 **向量数据库（Vector Database）** 的 **模糊/相似度检索** 在概念上高度一致。它不是基于关键词的精确匹配，而是一次基于语义相似度的智能检索。



革命性解决方案：注意力机制登场

- **数据库构建流程：**
- 对于输入序列中的 **每一个位置**（例如，每一个单词），编码器（如RNN/LSTM）读取该位置的词元，并生成其对应的隐藏状态 h_t。
- **生成记录**: h_t 会通过两个独立的线性变换（投影层），生成两条并行的信息记录：
- **键 (Key) k_t：**随机初始化一个 embedding，作为该位置信息的 **“地址”**。
- **值 (Value) v_t：**随机初始化一个 embedding，作为该位置信息的**“内容”** 。

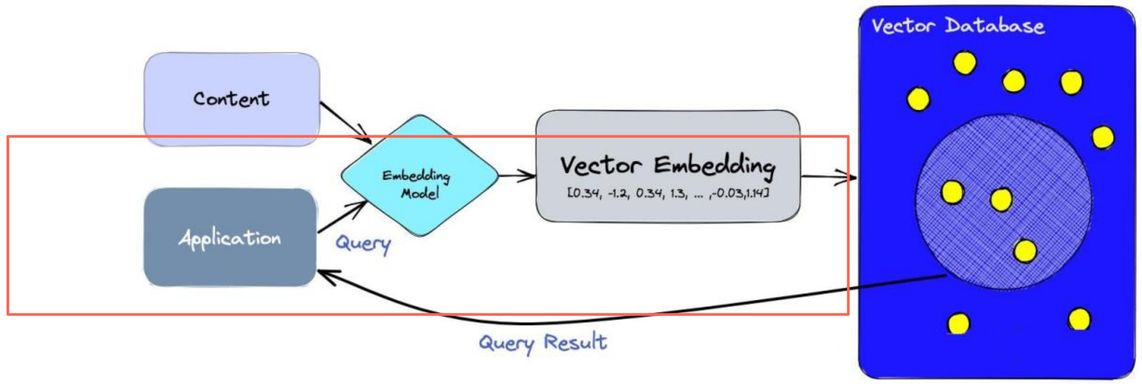
原始词元	键向量 (Key Embedding) k_t (用于匹配的 “索引” 向量)	值向量 (Value Embedding) v_t (待提取的 “内容” 向量)
"I"	[0.12, -0.45, 0.67, ..., 0.88]	[0.91, 0.23, -0.05, ..., -0.11]
"love"	[-0.24, 0.81, 0.11, ..., 0.43]	[0.65, 0.88, 0.12, ..., 0.99]
"learning"	[0.55, 0.09, -0.72, ..., -0.21]	[0.21, -0.15, 0.95, ..., 0.34]



注意力机制的本质：查询–键–值（QKV）模型

极客时间 训练营

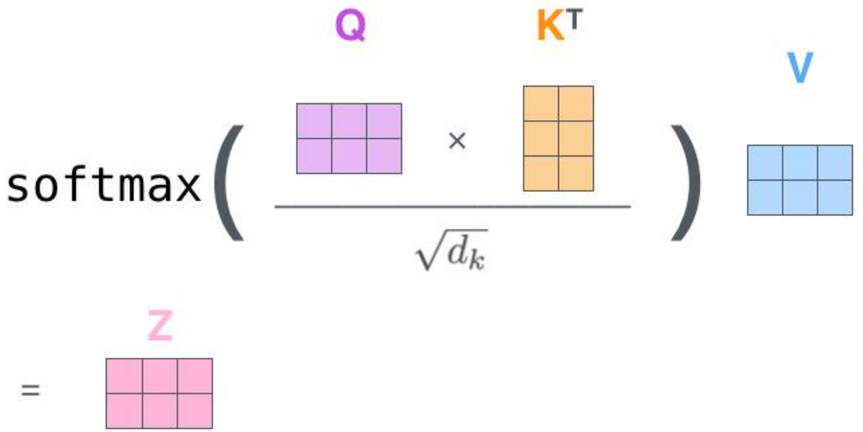
- 这个过程可以分解为四个连贯的步骤，如同一次高效的数据库查询：
- **1. 生成“搜索词”（ formulating the Query）**
- 解码器根据当前的需求（隐藏状态 h_z），生成一个**查询向量 q_z，目的是将 h_z 投影到 q_z 才能和 k_t 在同一个语义空间，让他们的空间距离可比，类似于我们用 Embedding 模型投影到语义空间**。
- **2. 并行相似度匹配（Similarity Matching）**
- 将这个“搜索词”向量 q_z，通过一次大规模的**矩阵运算**，与“向量数据库”中**所有的键向量 (Key) k_t** 进行并行匹配。
- **3. 按相关性“软排序”（"Soft" Sorting by Relevance）**
- 上一步得到的原始分数会通过一个 **Softmax** 函数，被转换成一个概率分布（即注意力权重）。



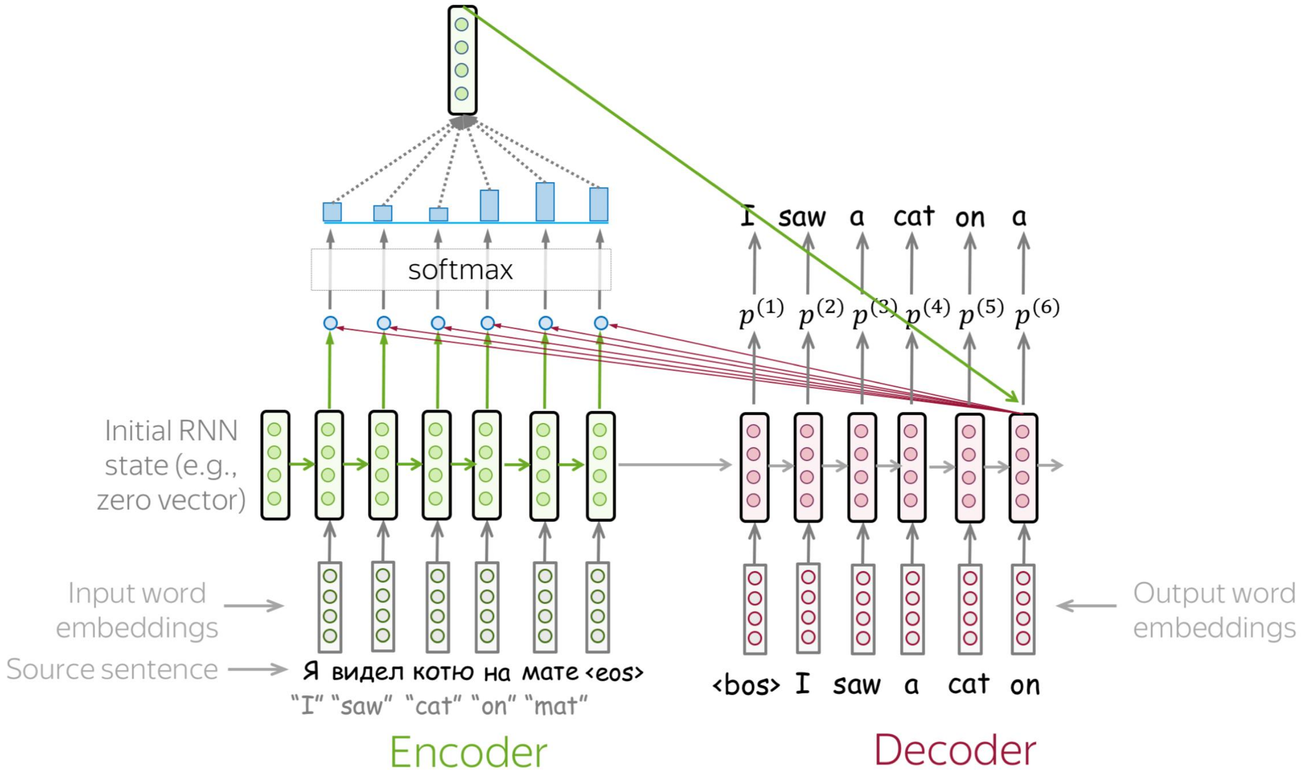
注意力机制的本质：查询–键–值（QKV）模型

极客时间训练营

- **4. 按权重提取内容（Weighted Retrieval）**
- 最后，根据上一步“排序”得到的权重，对数据库中每个“键”所对应的“值”向量 (Value) v_t 进行加权求和。



基于注意力机制的 RNN



Transformer 与并行化革命

- **RNN + 注意力机制的遗留问题**
  - **顺序计算依赖**：编码器必须按顺序处理输入，无法并行，效率低下。
  - **跨组件状态依赖**：解码器依赖编码器或自身的上一步状态，形成串行链路。
- **Transformer 的核心变革**
  - **完全摒弃循环结构 (RNN/LSTM)**。
  - **编码器**：使用 **自注意力机制 (Self-Attention)** 实现完全并行计算。
  - **解码器**：采用由查询驱动的生成模式，摆脱对单一隐藏状态的依赖。

目录

注意力机制

极客时间训练营

- **RAG：宏观的、外部的检索增强**
- 是一种 **外部知识接入机制**，适配开放领域问答等任务。
- 每个文档或段落 **d** 被编码为：
  - **Key**：通过 **向量模型** 得到的 Embedding
  - **Value**：提取文档 **原始内容** 作为值
- 所有文档的 k, v 组成一个 **知识库（retrieval corpus）**

注意力机制

- **注意力机制：微观的、内部检索增强**
- 是一种**模型内部的记忆访问机制**，无需依赖外部知识。
- 每个输入位置的隐藏状态 **h_t** 被映射为：
  - **Key**： 通过 **向量变换 K**(h_t) 得到的语义嵌入 **k_t**
  - **Value**： 通过 **向量变换 V**(h_t) 得到的内容表示 **v_t**
- 所有的 k_t, v_t 对构成一个**临时知识库**（即局部上下文）。

注意力机制

- **为何不像向量数据库那样，直接将 Value 设为隐藏状态 h_t 本身，还进行一次向量转换 V?**
- 如果 h_t 本身包含最关键信息，
  投影变换 V 可以学习成**恒等映射**，保持信息完整，无负面影响。
- 如果 h_t 中含有干扰或冗余信息，
  V 可以充当**提取器**，从中抽取出对生成最有用的语义内容。

注意力机制

- **换个角度思考：Value 是否总该是“原文”?**
- 在向量数据库中，检索到的向量**通常被映射回原始文档内容**。
- 但问题是：**原文直接用于生成任务，是否总是最优?**

注意力机制

- **任务驱动的 Value 设计**
- 其实可以根据任务目标或文档类型，**提前对原文进行加工**，构造更适合生成或推理的 Value 表示。

- **举个具体例子**
- 假设你的知识库文档都是关于“模型发布时间”的描述：
- 原文：
- “2023年，OpenAI 发布了 GPT-4 模型，具备多模态能力，并显著提升了推理能力。”
- **普通做法（直接返回原文）：**
- 向量数据库检索后，直接将这段文本作为上下文供生成模型使用。
- 模型必须**再次解析自然语言**以提取关键信息，存在信息损耗或生成冗余。

注意力机制

- **改进做法（提前针对任务导向加工 Value）：**
- { "model": "GPT-4", "release_year": 2023, "features": ["multimodal", "reasoning"] }
- 这种结构化或摘要式 Value 表示，更直接、更高效地支持下列任务：
- “GPT-4 是哪一年发布的？”
  “有哪些关键能力？”
  “哪些模型支持多模态？”

- 因此，无论在注意力机制中，还是在向量数据库场景中，
- “Value 的优化不应只是还原原文，而应服务于生成目标。”
- ——任务导向地设计 Value，可以显著提升生成效果和推理效率。

注意力机制

极客时间训练营

- **从 RAG 检索类比注意力的解码器查询机制**
- **1 查询表示**
- 在 RAG 中，我们首先将用户的问题编码成一个查询向量（Query Embedding），用于搜索知识库。
- 在注意力机制中，同样会从当前解码器状态 h_z 中生成一个查询向量 q_z，通过一个线性变换 Q(h_z) 得出。

注意力机制

极客时间训练营

- **2 相似度计算**
- RAG 使用向量检索方法（如内积或余弦相似度）计算查询向量与所有文档向量的相似度。
- 注意力机制中，q_z 与所有 Encoder 侧的 Key 向量 k_t 计算点积相似度，衡量当前生成位置对每个输入位置的关注程度。

注意力机制

极客时间训练营

- **3 排序与选择**
- 在 RAG 中，系统会对所有文档按照相似度排序，优先使用排序靠前的结果。
- 而在注意力机制中，相似度排序更高 Key 被赋予更高权重。

注意力机制

极客时间训练营

- **4 内容获取与聚合**
- 在 RAG 中，系统将相似度排名靠前的文档内容，优先拼接在 Prompt 的前部，使其更容易被生成模型关注并利用。
- 而在注意力机制中，模型根据 Query 与 Key 的相似度，高相似度的 v_t 会获得更高的注意力权重，被着重提取并用于解码，而低相似度的部分则被自动弱化。

- 统一视角下的总结：
- 两者都遵循“根据相似度优先使用相关内容”的原则，
- 只是 RAG 是显式位置优先，注意力是隐式权重加权。

到自注意力机制的进化

极客时间训练营

- **问题：RNN + Attention 的局限**
- **1. 顺序计算限制**
- RNN 必须按时间顺序逐步生成隐藏状态 h_t，每一步都依赖前一步结果。
- 无法并行，导致**长文本处理效率低**，计算时间随序列长度**线性增长**。

- **2. 编码–解码分离**
- Encoder 负责构建“知识库”（Key + Value），Decoder 负责查询（通过 Query）。
- 但 Encoder 与 Decoder 内部仍沿时间维度单向流动信息，
- 限制了跨时间步的灵活信息交互与并行计算能力。

从注意力机制到自注意力机制的进化

极客时间 训练营

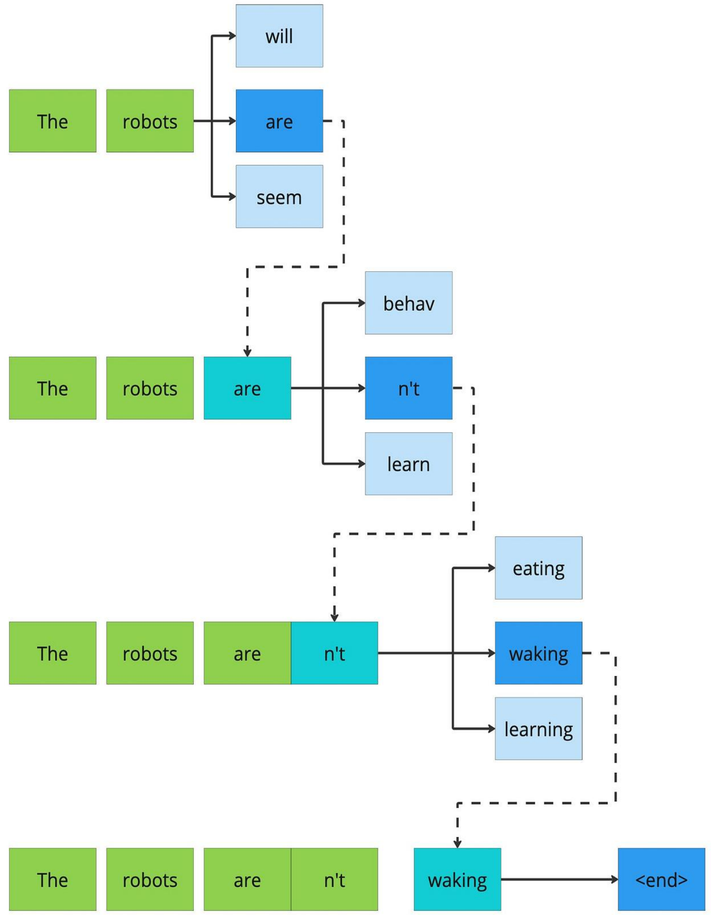
- **Transformer：打破顺序依赖，实现高效并行**
- Transformer 摒弃了 RNN 的顺序递归结构，
- 利用自注意力机制（Self-Attention）一次性计算序列中所有位置的表示，
- 无需等待前一步计算，极大提升了训练与推理效率。

- **1 Encoder 自注意力（Self-Attention）**
- Encoder 侧采用自注意力机制，
- **每个位置的表示向量同时作为 Query、Key 和 Value**，
- 通过上下文中其他位置的内容信息，构建我在全局中的一个知识条目，包括：
  - 其他输入中，谁的内容与我最相关?
  - 我在整个输入内容全局中的角色定位是什么?

从注意力机制到自注意力机制的进化

极客时间训练营

- **2 Decoder 自注意力**
- Decoder 也使用自注意力，目的是看一下前面已经生成了什么。
- 在 Decoder 的每一层，自注意力之后，增加一个跨注意力层，
- **Decoder 的 Query 向量**用于查询 Encoder 输出的 Key–Value 对，
- 这是一个外部检索过程，Decoder 通过跨模块查询获取编码后的上下文信息，支持生成更准确的输出。



总结 Transformer

- **过去 (RNN/LSTM): 一次性的“状态交接”**
  - Encoder 将整个输入压缩成一个**固定大小的上下文向量**（Context Vector）。
  - 这就像一个“摘要笔记”，包含了输入的所有信息，一次性交给 Decoder。
- **现在 (Transformer): 持续性的“信息查询”**
  - **1. Encoder 成为“知识库构建者”**
    - 它完整阅读输入，为每个词都生成专属的“**键(Key)**”和“**值(Value)**”向量。
    - 这个包含全局信息的 **K–V 集合**，就是 Decoder 专属的、开放式的“**知识库**”。
  - **2. Decoder 成为“聪明的查询者”**
    - 在生成**每一个新词**时，它都会通过**交叉注意力 (Cross–Attention)** 机制...
    - ...主动向 Encoder 构建的“知识库”发出查询，动态地获取此刻最需要的信息。

总结 Transformer

极客时间 训练营

- **Transformer 的解码过程：**
- 在推理生成的过程中，Decoder 并不是简单地“看一眼”Encoder 就开始埋头写作，而是像一个正在进行“开卷考试”的学生，在写下每一个字时，都会同时做两件重要的事情：
- 回顾自己刚写下的句子，确保语法通顺、逻辑连贯。
- 翻阅桌上的参考资料（原文），确保自己写的内容忠实于原文。
- 这两个动作，在 Transformer 的 Decoder 模块中，由两个不同的“注意力”机制完美地分工协作完成。

总结 Transformer

- 让我们以将 "I am a student" 翻译成 "我是一个学生" 为例，一步步拆解 Decoder 在生成第三个词“一个”时的内心活动：
- **前提：Encoder 的准备工作**
- 源语言信息已输入： Encoder 模块已经完整地“听完”了整个句子 "I am a student"。
- 核心资料已备好： Encoder 将其对这句话的完整理解，形成了一份内容固定的“核心资料”（即一系列 Key-Value 向量），这份资料在整个翻译过程中随时可供查阅。

Transformer 的工作过程

- 现在，这位传译员（Decoder）开始工作，假设他已经说出了“我 是”，正准备说出下一个词。
- **生成下一个词的“两步思考法”**
- **第一步：回顾自己刚说的话（历史自注意力）**
- 这是 Decoder 与“过去的我”的交互，目的是确保语言的流畅和连贯。
- **当前进度：** Decoder 的脑中记着它已经说出的部分：<s> 我 是。
- 基于当前的结尾“是”，Decoder 会整合 <s> 我 是 的语境，形成一个关于“接下来该说什么”的初步念头。这个念头是通过一个名为自注意力的机制实现的。它会分析“是”这个词与前面“我”等词的关系，判断出当前的语法结构和语义流向。
- **生成初步意图：** 这个过程的产出是一个包含了历史上下文信息的“意图向量”，可以理解为传译员在衔接下一句话之前的“内心酝酿”。

Transformer 的工作过程

- **第二步：对照核心资料核实源头（交叉注意力）**
- 这是 Decoder 与“源语言信息”的交互，目的是确保翻译的准确和忠实。
- **带着“意图”去查询：** 在第一步中形成的那个“意图向量”，现在变成了一个**查询（Query）**。传译员带着“我刚说了‘我 是’，接下来该怎么说呢？”这个念头，去查阅核心资料。
- **在资料中定位关键信息：** 这个查询会被拿去和 Encoder 准备好的、关于 "I am a student" 的那份“核心资料”进行比对。模型会发现，当前的“意图”与原文中的 "a" 这个词关联性最强。
- **提取并融合关键信息：** 模型随即重点“借用”了 "a" 所代表的语义信息，并将其与自己的“初步意图”融合，形成一个更完整、更准确的最终意图。

如果只保留 Decoder 会怎么样?

极客时间训练营

- **核心转变：** 系统不再受知识库 (Encoder) 的直接约束。
- **行为：**内容完全依赖于“前缀（Prompt）”，并以此为起点，在自身内部知识的引导下，不断地、连贯地生发与成长。
- **支柱一：庞大的“内部知识”约束**
  - **来源：** 海量数据的无监督预训练。
  - **体现：** 模型参数内化了关于世界事实、语言规律、逻辑常识的“知识库”。
  - **效果：** 它“知道”基本事实（如“巴黎是法国的首都”），保证了内容的**合理性**和**准确性**。
- **支柱二：动态的“上下文”约束**
  - **来源：** Prompt + 已生成内容的实时序列。
  - **体现：** 自注意力机制时刻回顾历史，确保下一步的生成在语法和逻辑上是连贯的。
  - **效果：** 它“记得”自己刚说过的话，保证了内容的**流畅性**和**一致性**。

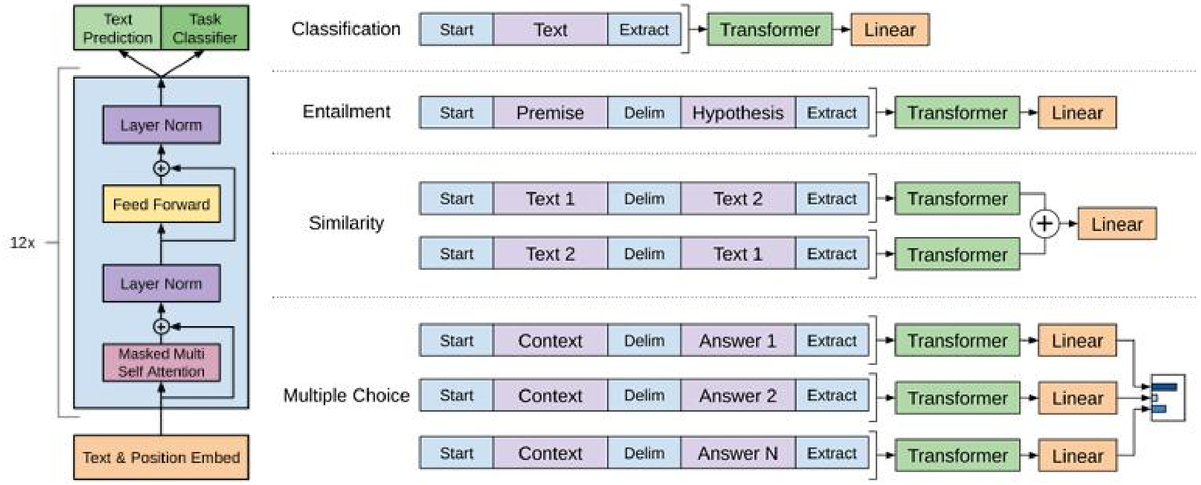
GPT–1 的诞生

- **这正是 GPT–1 的基本逻辑。**
- 通过**极大规模的参数**和**海量的预训练**，让模型仅依靠这两大“内部约束”，就足以生成满足前缀要求的高质量内容。这一思想直接开启了现代大语言模型的时代。

- **GPT–1 的发布（2018年），其意义远不止一个新模型，它为整个自然语言处理领域带来了三大革命性的贡献，完美诠释了“内功”与“招式”的哲学。**
- **贡献一：确立“生成式预训练”的核心地位**
- **革命性的“内功”修炼法：**
  - **GPT–1 的证明：** 在海量无标签文本上，仅通过“**预测下一个词**”这个极其简单的目标，就能迫使模型学到深刻的语法结构、语义关系乃至世界知识。
  - **结论：** 这为“内功”的修炼指明了一条可无限扩展的道路——只要有足够多的文本，模型的内在能力就可以持续增强。

GPT-1 的诞生

- **贡献二：NLP 大规模“预训练+微调”的统一范式**
- **化繁为简的“简洁招式”：**
  - **在此之前：** 不同的NLP任务（如文本分类、问答、句子关系判断）通常需要设计不同的、复杂的专属模型架构。
  - **GPT-1 的证明：** **同一个预训练好的模型**，几乎不做任何结构改动，只需在不同下游任务的少量标注数据上进行微调（Fine-tuning），就能取得当时顶级（State-of-the-art）的效果。



GPT-1 的诞生

- **贡献三：验证“Decoder-only”架构的巨大潜力**
- **对“架构之争”的有力回应：**
  - **GPT-1 的证明：** 大胆地证明了，一个纯粹的、单向的 **Transformer Decoder 结构**，只要其“内功”（预训练规模）足够深厚，就完全有能力成为一个强大的通用语言模型基础。
  - **结论：** 这为后续所有 GPT 系列模型的成功，以及 Decoder-only 架构的流行，奠定了坚实的基础。

目录

CLIP：用自然语言开启视觉理解新纪元

极客时间 训练营

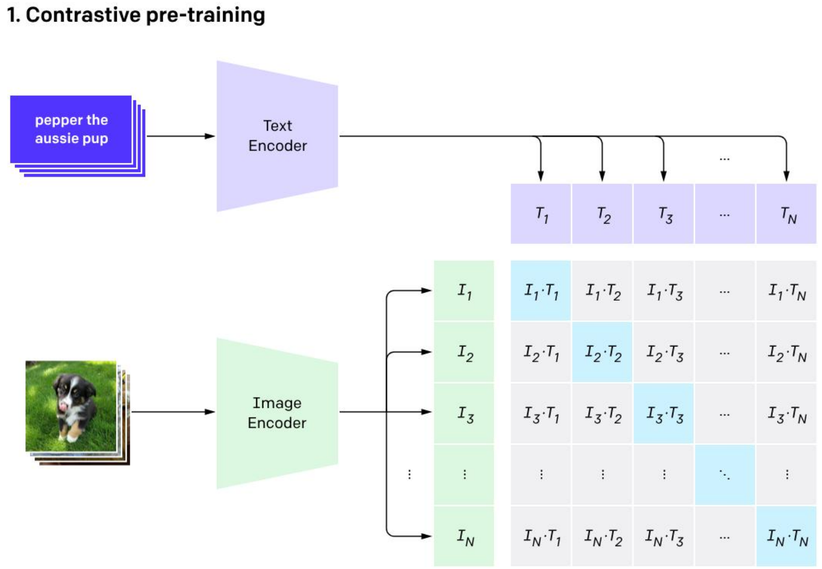
- **视觉领域的挑战**
- 传统有标签数据（如 ImageNet）规模有限，难以覆盖海量多样的视觉场景。
- **机遇**
- 互联网中存在海量“图片 + 描述文本”自然配对数据，为视觉理解提供了新的突破口。

- **核心创新：**
- 不再限制模型识别固定类别，
  而是让模型学会衡量**任意文本描述**与**任意图像**的匹配度。
- 通过对比学习（Contrastive Learning），CLIP建立了跨模态的共享向量空间，实现视觉和语言的深度融合。

连接文本与视觉

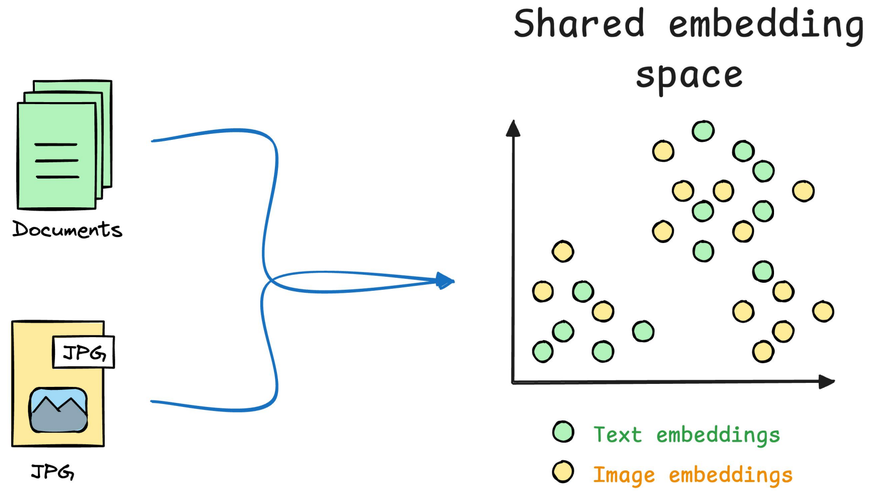
- **借口任务：图文对比学习（Contrastive Learning）**
- **从互联网海量获取**
- 成千上万的（图片，文本描述）配对数据

- **优化目标**
- **拉近匹配的图文向量距离**
- **推远不匹配的图文向量距离**
- 通过这种方式，模型学会了跨模态语义对齐



通过提示语实现“万物识别”

- 训练完成后，CLIP 解锁了前所未有的零样本图像分类能力。
- **如何识别一张图片?**
- 输入：一张狗的图片
- 构造多条文本提示：
  - “一张猫的图片”
  - “一张狗的图片”
  - “一张飞机的图片”
- 计算图片向量与各文本向量的相似度
- 选择相似度最高的文本作为分类结果
- **结果：**图片与“**一张狗的图片**”的相似度最高，成功完成分类



通过提示语实现“万物识别”

- **优势：**
- 无需针对新类别重新训练模型
- 只需修改或新增文本提示，即可识别任意新类别
- 灵活、高效、开放式视觉理解能力

练习

极客时间 训练营

1. **词表示：**训练一个 Word2Vec 词嵌入模型，理解词向量的生成与语义捕捉。
2. **句子表示：**使用 BERT 预训练模型，提取句子级别的向量表示，感受上下文语义的编码能力。
3. **生成模型：**利用 OPT 模型进行文本生成实验，掌握 Next Token Prediction 原理。
4. **多模态训练：**尝试基于文本–图像对数据，微调增强 CLIP 模型的跨模态表示学习能力。

推荐阅读

- Pattern Recognition and Machine Learning （PRML） 作者：Christopher Bishop
- 《深度学习：基础与概念》 作者：Christopher Bishop

THANKS