预训练语言模型：架构、范式与设计哲学

概要

- 范式一：自编码模型 (AE)
- 范式二：自回归模型 (AR)
- 范式三：编码器-解码器模型

根本问题：知识获取与迁移的鸿沟

- 浅层、上下文无关的表示 (Shallow, Context-Independent Representations)
  - Word2Vec, GloVe 等词嵌入技术，一个词对应一个固定向量
  - 无法解决一词多义问题 (例如 "bank" 在 "river bank" vs "investment bank" 中含义不同)
  - 模型从零开始学习语法和语义，效率低下
- 对任务特定标注数据的重度依赖 (Heavy Reliance on Task-Specific Labeled Data)
  - 每个下游任务 (情感分析、实体识别等) 都需要一个独立的、从头训练的模型 (如 LSTM, GRU)
  - 浪费了互联网上唾手可得的海量无标签文本中蕴含的通用语言知识。

- 能否构建一个通用的、深度的语言知识模型，通过在海量无监督数据上预训练，然后快速迁移到各种下游任务？
  - 类比： 寻求NLP领域的“ImageNet时刻”

预训练 (Pre-training) & 微调 (Fine-tuning) 范式

- 此两阶段的过程，根本性地改变了NLP任务的执行方式
- 预训练阶段 (Pre-training Phase)
  - 目标： 学习通用的语言表示
  - 数据： 海量无标签文本 (如维基百科、书籍、网页)
  - 方法： 设计一种自监督学习 (Self-supervised Learning) 任务，让模型从数据自身中学习
  - 产出： 一个包含了丰富语法、语义和世界知识的深度神经网络——预训练语言模型 (PLM)
- 微调阶段 (Fine-tuning Phase)
  - 目标： 适配特定的下游任务
  - 数据： 少量的任务相关标注数据 (如带情感标签的评论)
  - 方法： 在预训练模型的基础上，添加一个小的任务头 (Task-specific Head)，并用任务数据进行端到端的权重更新
  - 优势： 极大地提升了性能、数据效率和泛化能力

架构基石：Transformer 与自注意力机制

- Transformer架构的出现，是实现大规模预训练成为可能的关键。
- 核心创新：自注意力机制 (Self-Attention)
  - 完全替代了RNN的循环结构。
  - 为序列中的每个词元计算一个加权平均的上下文表示，权重由词元间的相关性动态决定。
  - 公式核心：Attention(Q, K, V) = softmax(QK^T / sqrt(d_k))V
- 带来的革命性优势：
  - 并行计算能力： 彻底摆脱序列依赖，所有词元的计算可以同时进行，极大提升了训练效率。
  - 长距离依赖捕捉： 任意两个位置间的路径长度为 O(1)，能高效捕捉远距离的语义关联。
  - 可扩展性 (Scalability): 使得训练拥有数百亿甚至数千亿参数的超深模型成为现实。
- 结论： Transformer为承载和学习海量文本中的复杂知识提供了完美的架构载体。

三大主流预训练范式

- 基于Transformer，根据预训练任务和架构选择的不同，演化出三大主流范式。它们的设计哲学决定了其能力边界。
- 自编码模型 (Auto-Encoding Models)
  - 哲学： 理解是核心 (To Understand)
  - 代表： BERT
- 自回归模型 (Auto-Regressive Models)
  - 哲学： 生成是核心 (To Generate)
  - 代表： GPT
- 编码器-解码器模型 (Encoder-Decoder Models)
  - 哲学： 转换是核心 (To Transform)
  - 代表： T5, BART

范式一：自编码模型 (AE)

语言的歧义性与上下文依赖

- 语言的根本困境在于其固有的歧义性（Ambiguity）
- 一个孤立的词语，甚至一个不完整的句子，其含义都是不确定的。词语的精确意义是在其所处的上下文（Context）中被动态“塑造”和“确定”的
  - 起决定性作用的上下文，同时包含了左侧上下文（前文）和右侧上下文（后文）
    - 左侧上下文提供背景和历史信息
    - 右侧上下文则提供限定、修正和澄清
  - 忽略任何一方，都会导致对语言的理解出现偏差甚至完全错误。这便是所有语言理解任务（Natural Language Understanding, NLU）面临的根本挑战

前BERT时代的困境：上下文表示的“方向性”难题

- 静态词向量 (Word2Vec, GloVe)
  - 上下文无关的。无论“bank”出现在“river bank”还是“investment bank”中，其向量都是固定的，这从根本上无法捕捉语义的丰富性
- 单向上下文模型 (如从左到右的LSTM, GPT-1)
  - 它们在预测或表示一个词时，只能利用其左侧（历史）信息。这对于生成任务是自然的，但对于理解任务则损失了一半的关键信息。一个词的精确含义往往由其左右两侧的语境共同决定
- 浅层双向模型 (ELMo, Bi-LSTM)
  - ELMo通过独立训练一个前向LSTM和一个后向LSTM，然后将它们的隐状态拼接起来，为每个词生成上下文相关的表示。但两个网络信息只在最顶层被简单地“粘合”在一起。这意味着，左侧上下文如何影响右侧上下文的表示（反之亦然）这一深层交互被忽略了

自编码模型 (AE): 双向的语境理解者

- 核心思想： 通过破坏输入文本，训练模型恢复原文，从而学习深度的上下文表示
- 类比
  - 完形填空专家: 它的任务是根据一个词左右两边的所有线索，推断出被遮盖的词
  - 语言考古学家: 修复残缺的文献，需要对语言结构有深刻的理解

- 模型设计目标： 生成对自然语言理解 (NLU) 任务最优的、深度融合双向上下文的特征表示

AE核心机制：掩码语言模型 (Masked Language Model)

- 代表模型: BERT (Bidirectional Encoder Representations from Transformers)
- 为了实现深度双向的目标，BERT不能使用传统的“预测下一个词”的语言模型任务，因为这会不可避免地引入单向性。如果模型能看到要预测的词本身，任务就变得毫无意义。BERT的团队为此设计了两个巧妙的自监督任务
- 子主题解析：掩码语言模型 (Masked Language Model, MLM)
- 子主题解析：下一句预测 (Next Sentence Prediction, NSP)

- 架构： 仅编码器 (Encoder-Only)

子主题解析：掩码语言模型

- 输入： 随机将输入句子中约15%的词元（Token）替换为一个特殊的 [MASK] 标记
  - The quick [MASK] fox jumps over the lazy dog.
- 掩码策略 (80-10-10规则): 对于这15%被选中的词元：
  - 80% 的概率，用一个特殊的 [MASK] 标记替换它。 (例: the quick [MASK] fox)
  - 10% 的概率，用一个随机的其他词元替换它。 (例: the quick apple fox)
  - 10% 的概率，保持原样不变。 (例: the quick brown fox)
- 目标： 训练模型，使其仅利用 [MASK] 位置的最终隐藏层状态 h_[MASK]，来预测原始词元 ("brown")。
- 在预测 [MASK] 时，模型可以无限制地访问其左侧和右侧的所有上下文，因此是双向 (Bidirectional) 的。

子主题解析：下一句预测(NSP)

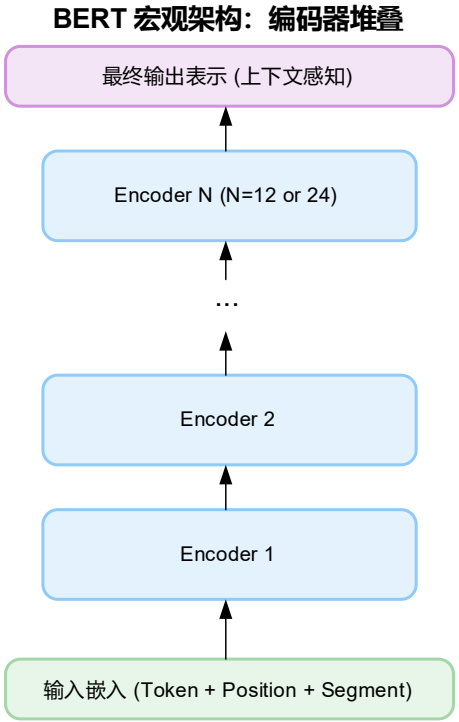
- 工作流程:
  - 构造样本: 训练时，输入由两个句子对 (A, B) 构成。
  - 50% 的概率，B是A在原始语料中的真实下一句（标签：IsNext）。
  - 50% 的概率，B是语料库中随机抽取的一个句子（标签：NotNext）。
  - 预测目标: 模型需要预测B是否是A的真实下一句。这个预测任务由输入序列开头的特殊标记 [CLS] 的最终输出来完成。
- 后续演进与批判: 尽管初衷很好，但后续的研究（如RoBERTa, ALBERT）发现NSP任务的设置过于简单，模型可能更多地是学习到句子间的主题相关性而非真正的逻辑关系，甚至可能对性能产生负面影响。因此，许多后来的自编码模型都放弃或改进了NSP任务。

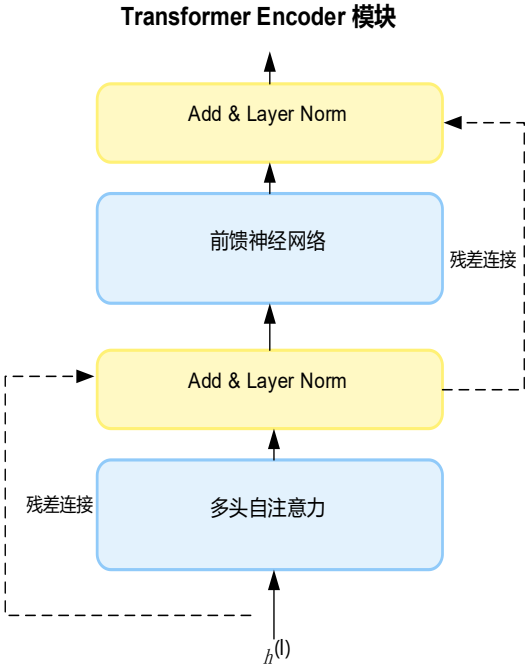
BERT 架构与掩码语言模型 (MLM)

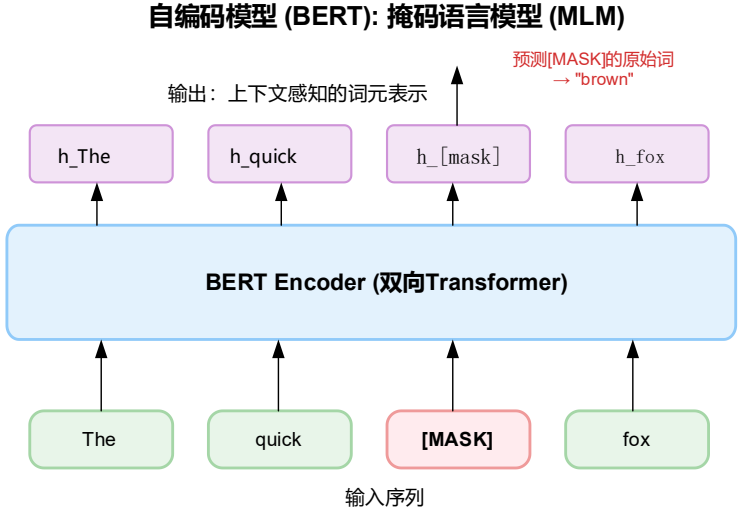
- BERT只使用Transformer的编码器
  - 一个典型的BERT-Base模型包含12层编码器，BERT-Large则有24层。
- 每个输入词元的表示由三部分相加而成
  - 词元嵌入 (Token Embeddings): 词元本身在词汇表中的向量
  - 片段嵌入 (Segment Embeddings): 用于区分句子对中的句子A和句子B
  - 位置嵌入 (Position Embeddings): 用于向模型提供词元在序列中的位置信息，因为Transformer本身不具备序列顺序的概念
- 最终输入形式为：[CLS] Sentence A [SEP] Sentence B [SEP]

BERT的网络架构

- 完全由Transformer的编码器（Encoder）模块堆叠而成

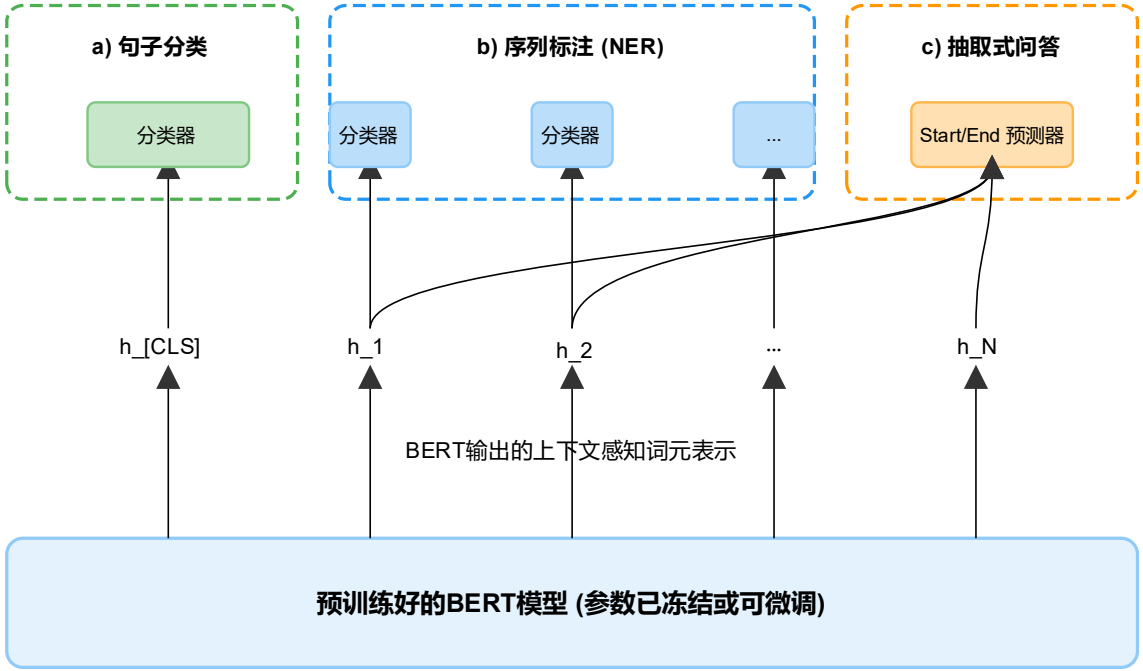






下游任务微调 (Fine-tuning for Downstream Tasks)

- 核心思想: 在预训练好的BERT模型之上，添加一个或多个简单的、通常是随机初始化的网络层（称为“任务头”，Task-specific Head），然后在特定任务的有标签数据上对整个模型（或仅仅是任务头）进行端到端的训练



AE: 优势与局限

- 优势
  - 强大的NLU能力： 由于其双向性，它在需要深度语境理解的任务上表现卓越
  - 应用： 情感分类、命名实体识别 (NER)、句子关系判断 (NLI)、问答 (QA)
- 局限
  - 不擅长生成：预训练任务决定了它不是一个自然的文本生成器。其生成能力有限且通常不连贯，结果往往是无序和重复的
  - 预训练-微调差异: [MASK] 标记在预训练时存在，但在微调和推理时不存在，造成了训练和推理阶段的数据分布不一致
- BERT是深度双向表示的开创者，它站在ELMo（浅层双向）和GPT-1（单向）的肩膀上，通过MLM任务和Transformer编码器，将预训练模型的性能推向了新的高度，真正开启了NLP的“预训练-微调”范式时代。它更像一个强大的特征提取器，为下游任务提供高质量的语义表示

范式二：自回归模型 (AR)

如何对语言进行有效的概率建模与生成？

- 在Transformer时代之前，语言生成主要面临两大技术瓶颈
  - 有限的上下文依赖。以n-gram模型为代表的传统统计方法，其核心缺陷在于马尔可夫假设。无法捕捉决定语义的关键长距离依赖
  - 序列计算的效率与梯度问题。循环神经网络及其变体在实践中受到两大制约
    - 其固有的序列计算模式使其难以利用现代硬件进行大规模并行训练，严重限制了模型的扩展能力
    - 长序列下普遍存在的梯度消失/爆炸问题，使得模型难以学习到真正跨越长距离的依赖关系。
- 因此，自回归预训练模型（以GPT为代表）的核心目标
  - 构建一个可扩展的、能够捕捉长距离依赖的通用语言生成器
  - 通过“预测下一个词”海量无监督文本上进行训练，从而将语言的语法结构、语义关系、乃至蕴含的世界知识，内化到一个统一的深度神经网络中
  - 其终极愿景是，任何语言任务都可以被重新定义为一个生成任务，通过提供不同的“提示”（Prompt），模型就能生成相应的“答案”

自回归模型 (AR): 单向的序列生成者

- 核心思想： 严格遵循从左到右的顺序，根据已生成上文，预测下一个最可能的词元
- 直觉类比
  - 故事续写者 (Story Writer): 给定一个开头，它能逐词逐句地创作出连贯的后续内容
  - 语言预言家 (Language Oracle): 永远在预测“接下来会发生什么”
- 设计目标： 建模文本的联合概率分布$P(x_1, \ldots, x_n)$，使其成为一个强大的自然语言生成 (NLG) 引擎

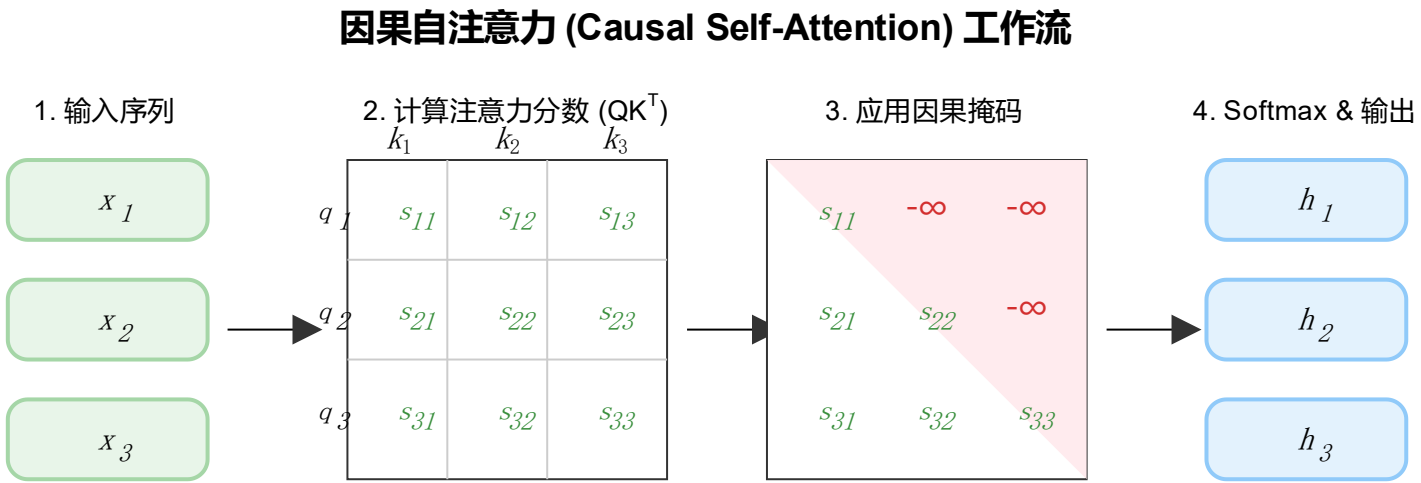
AR核心机制：因果语言模型 (Causal Language Model)

- 输入： The quick brown fox
  - 目标： 在每个时间步 t，模型根据前面的序列$x_1, \ldots, x_{t-1}$ 来预测 x_t
  - 信息流： 通过在自注意力层使用因果掩码 (Causal Mask)，模型在预测位置 t 时，只能访问 t 及其之前的位置。信息流是单向 (Unidirectional) 的
- 目标： 对于由T个词元组成的序列 $X = (x_1, x_2, \ldots, x_T)$,其联合概率
  - $P(X) = P(x_1) \cdot P(x_2|x_1) \cdot P(x_3|x_1, x_2) \cdots P(x_T|x_1, \ldots, x_{T-1}) = \prod_{t=1}^{T} P(x_t|x_{<t})$
  - 联合概率问题，转化成了一系列“预测下一个词”的条件概率问题
  - 自回归模型的全部任务，就是学习一个强大的函数，来精确地建模这个条件概率 $P(x_t|x_{<t})$

架构实现：Transformer解码器与因果自注意力

- 自回归模型（如GPT）仅采用Transformer架构的解码器部分，其核心机制是因果自注意力（Causal Self-Attention），即掩码自注意力（Masked Self-Attention）
- 实现“不能偷看未来”的关键
  - 标准自注意力：在标准自注意力中，一个词元（Query）会与序列中所有其他词元（Keys）计算注意力分数，从而聚合整个序列的信息
  - 引入因果掩码。在因果自注意力中，计算注意力分数矩阵 $\text{Scores} = \text{QK}^T$ 之后，会应用一个上三角掩码矩阵。该矩阵将所有 j > i 的位置（即未来位置）的分数设置为一个极大的负数
  - Softmax效应。 当对这个被掩码的分数矩阵应用Softmax函数时，那些被设置为 -∞ 的位置的注意力权重会变为0
- 确保计算第 i 个位置的输出表示时，模型只能关注到从第 1 到第 i 个位置的信息，而完全无法获取任何未来（i+1 及之后）的信息，从而严格保证了自回归的单向信息流

因果自注意力机制图解

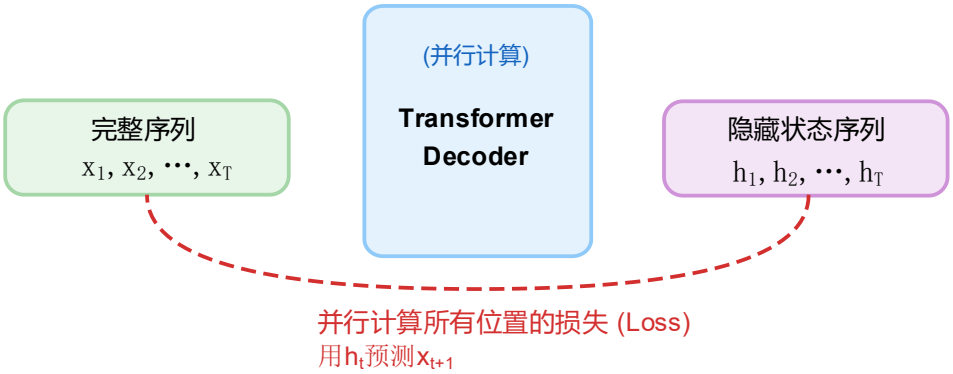


$h_1 = f(x_1)$ (只依赖于 x_1)
$h_2 = f(x_1, x_2)$ (依赖于 x_1 和 x_2)
$h_3 = f(x_1, x_2, x_3)$ (依赖于 x_1, x_2, 和 x_3)
在预测第 i 个词的输出 h_i 时，模型无法获取任何未来词元 (j > i) 的信息

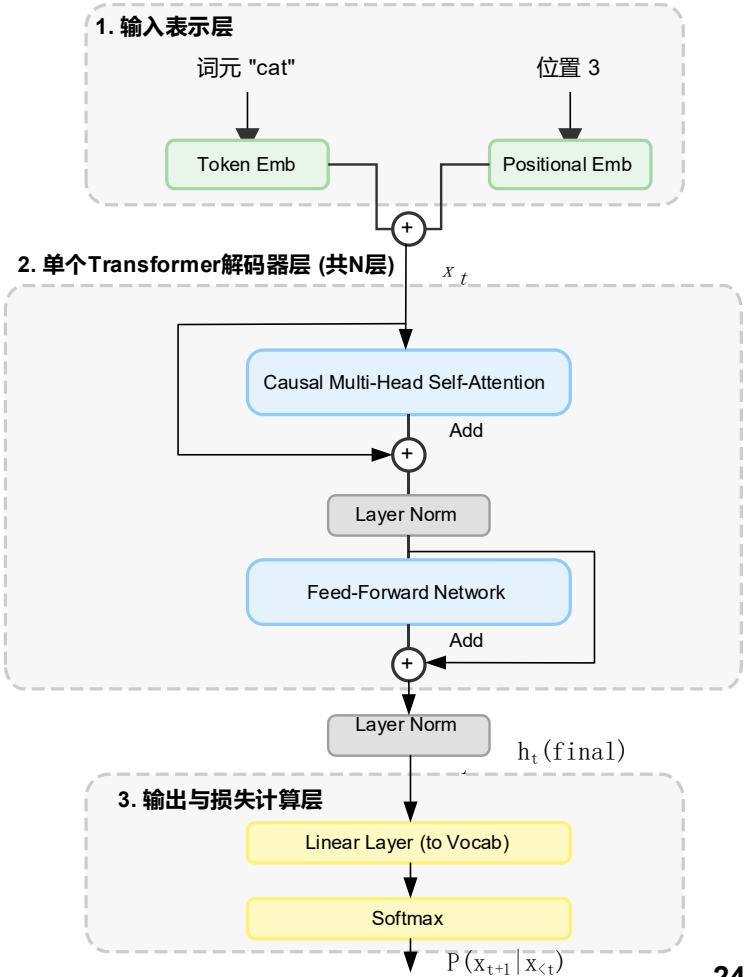
预训练 (Pre-training) 网络

- 以并行模式高效处理整个序列以计算损失

预训练计算流程 (并行模式)



预训练架构细节展开



输入表示层 (Input Representation)

- 分词器 (Tokenizer)
  - 作用: 将原始文本字符串（如 "thinking machines"）切分为一系列的词元（Tokens）
  - 机制: 现代大模型几乎都采用子词（Subword）分词算法，如BPE (Byte-Pair Encoding)
- 嵌入层 (Embedding Layer): 将离散的词元ID映射为连续的、稠密的向量
  - 词元嵌入 (Token Embedding)
  - 位置嵌入 (Positional Embedding)
- 最终输入向量 x = Token Embedding + Positional Embedding

核心计算引擎

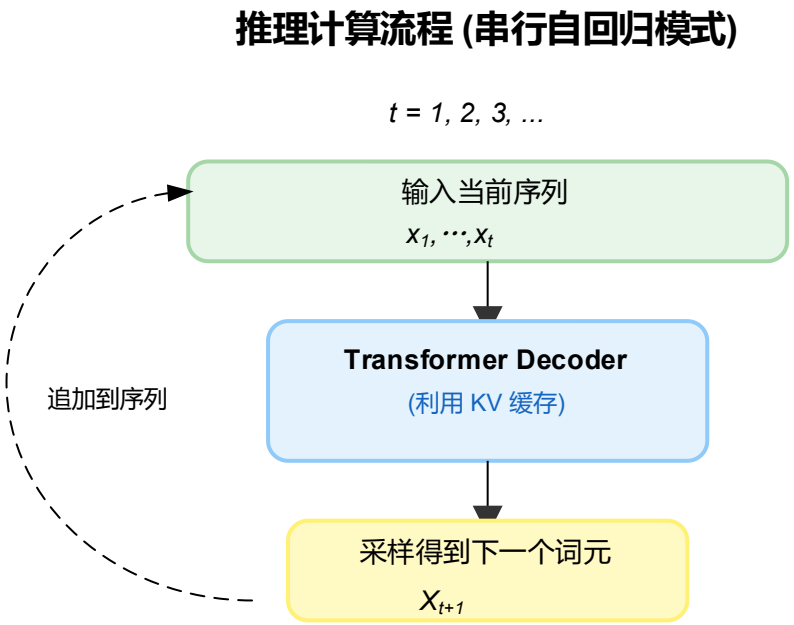
- 由 N 个完全相同的解码器层 (Decoder Layer) 堆叠而成。数据每经过一层，其表示就会被进一步提炼，变得更加抽象和上下文感知
- 子层1：因果多头自注意力 (Causal Multi-Head Self-Attention)
  - 模型理解上下文的核心。它让每个词元能够“回顾”并加权聚合其前面所有词元的信息
- 子层2：逐位置前馈网络 (Position-wise Feed-Forward Network, FFN)
  - FFN对每个位置的向量进行一次独立的、非线性的深度加工
- 连接件：残差连接 (Add) & 层归一化 (Norm) 每个子层（注意力层和FFN层）都被包裹在一个 Add & Norm 结构中
  - 残差连接: 将子层的输入 x 直接加到子层的输出 SubLayer(x) 上，即 x + SubLayer(x)
  - 层归一化: 对每个样本、每个层的输出向量进行归一化，使其均值为0，方差为1

输出与损失计算层

- 语言模型头 (Language Model Head):
  - 将最终的隐藏状态向量 h_t（维度为 d_model）转换为对下一个词元的预测。
- 损失函数
  - 交叉熵损失

推理/生成 (Inference/Generation) 网络架构

- 以串行自回归模式，通过循环迭代逐词生成新文本



生成过程

- 起始: 从一个初始的提示（Prompt）序列开始
- 迭代
  - 将当前序列输入模型，获得最后一个位置的输出表示
  - 将该表示通过一个线性层和Softmax函数，得到整个词汇表上关于下一个词元的概率分布
  - 从该分布中采样一个词元。采样策略至关重要，常见的有
    - Greedy Search: 总是选择概率最高的词。易导致重复、乏味
    - Top-k Sampling: 在概率最高的k个词中按其概率进行采样。增加多样性
    - Nucleus (Top-p) Sampling: 在累积概率超过阈值p的最小词元集中进行采样。动态调整候选集大小，是目前最主流的策略
  - 将新采样的词元追加到序列末尾
  - 重复此过程，直到生成结束标记或达到最大长度

AR: 优势与局限

- 优势：
  - 强大的NLG能力： 在开放式文本生成、对话、写作等任务上表现出色。
  - 上下文学习 (In-context Learning): 大规模AR模型展现出惊人的能力，无需微调即可通过示例 (Prompt) 完成任务。
- 局限：
  - 非最优的NLU表示： 单向信息流限制了其对整个句子上下文的深度理解，在NLU任务上天然弱于AE模型。
  - 暴露偏差 (Exposure Bias): 训练时接触真实数据，推理时依赖自身生成，可能导致错误累积。

范式三：编码器-解码器模型

现有范式的局限

- BERT (Encoder-only)
  - 卓越的“理解者”，它能为输入序列产出高质量的表示，但它本身不具备生成新序列的机制。它能判断一个句子好不好，但写不出来
- GPT (Decoder-only)
  - 强大的“生成者”，它可以写出连贯的文本。但在处理Seq2Seq任务时，源序列只能作为其生成开头的“提示(Prompt)”。由于其单向注意力机制，它在生成后续词元时，无法回头重新、完整地审视整个源序列，这限制了其对源序列信息的保真度和利用率，容易出现事实偏离（在摘要中）或意义漂移（在翻译中）

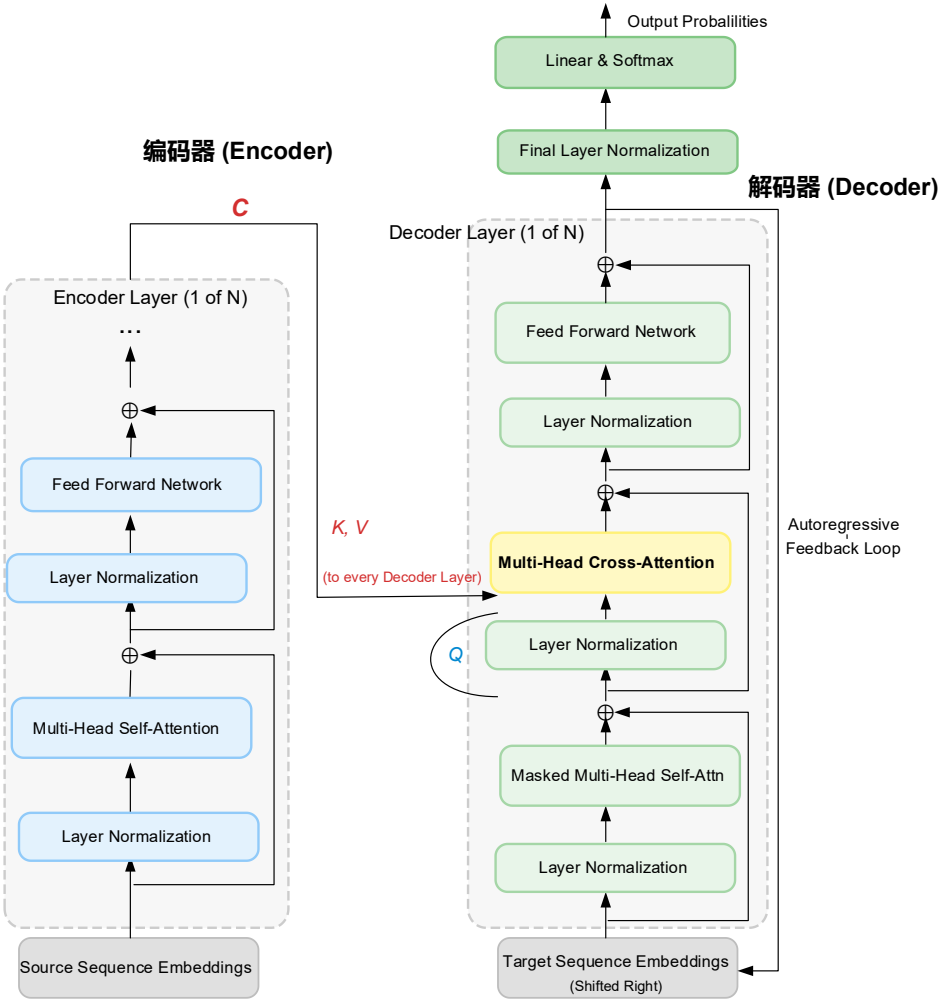
编码器-解码器 (Seq2Seq): 序列转换

- 核心思想： 结合AE和AR的优点，先用一个双向编码器理解整个输入序列，再用一个自回归解码器生成目标序列
- 直觉类比
  - 专业翻译家 (Professional Translator): 先完整阅读并理解源语言句子（编码），然后在脑中形成抽象语义，最后用目标语言流畅地表达出来（解码）
- 成为解决通用序列到序列 (Sequence-to-Sequence) 任务的强大框架

Seq2Seq核心机制：编码器-解码器与交叉注意力

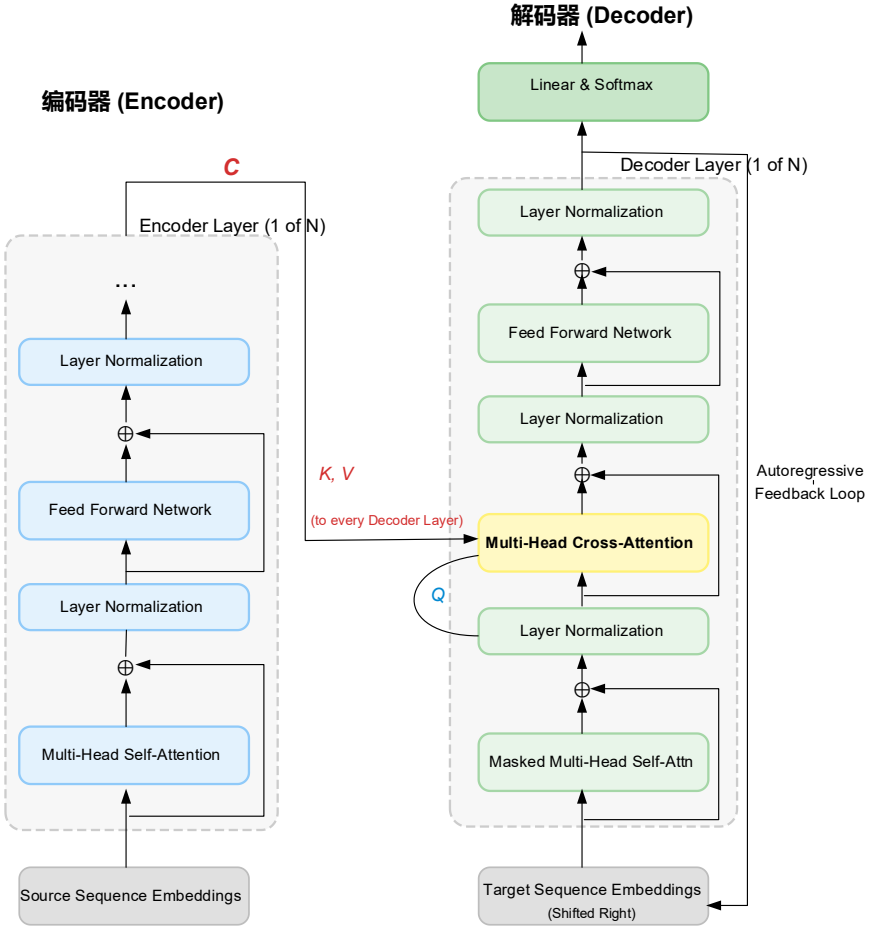
- 架构： 完整的Transformer，包含编码器 (Encoder) 和解码器 (Decoder)。
- 信息流：
  - 编码器: 对源序列进行双向编码，生成一套富含上下文的表示 memory。
  - 解码器: 在生成目标序列的每个词元时，会执行两种注意力：
    - 自注意力 (Self-Attention): 关注已生成的目标序列部分（单向）。
    - 交叉注意力 (Cross-Attention): 关注编码器输出的全部 memory。这是关键，它让解码器在每一步都知道完整的源序列信息。

T5架构，Pre-LN


Output Probalilities
Linear & Softmax
Final Layer Normalization
解码器 (Decoder)
编码器 (Encoder)
C
Decoder Layer (1 of N)
Encoder Layer (1 of N)
...
Feed Forward Network
Layer Normalization
Multi-Head Self-Attention
Layer Normalization
Source Sequence Embeddings
K, V
(to every Decoder Layer)
Q
Feed Forward Network
Layer Normalization
Multi-Head Cross-Attention
Layer Normalization
Masked Multi-Head Self-Attn
Layer Normalization
Target Sequence Embeddings
(Shifted Right)
Autoregressive
Feedback Loop

对比：Transformer架构

- Post-LN



Seq2Seq: 预训练、优势与局限

- 预训练任务 (多样化):
  - T5 (Text-to-Text Transfer Transformer): 将所有NLP任务统一为“文本到文本”的填空形式。
  - BART: 采用去噪自编码器 (Denoising Autoencoder) 思想。对原文进行各种破坏，然后让模型完整地恢复原文。
- 优势：
  - 强大的条件生成能力： 在给定输入条件下生成目标输出的任务上是SOTA。
  - 应用： 机器翻译、文本摘要、对话生成。
- 局限：
  - 模型复杂度和参数量大： 同时包含编码器和解码器，计算成本更高。

总结与思考

总结与思考

- 表格

特性	自编码（AE）	自回归（AR）	编码器—解码器（Seq2Seq）
核心架构	Encoder－Only	Decoder－Only	Encoder－Decoder
信息流	双向	单向（因果）	编码器双向，解码器单向
预训练任务	掩码语言模型（MLM）	因果语言模型（CLM）	去噪 / 文本到文本
设计哲学	理解（Understand）	生成（Generate）	转换（Transform）
核心优势	NLU任务（分类，NER）	NLG任务（写作，对话）	Seq2Seq任务（翻译，摘要）
代表模型	BERT，RoBERTa	GPT系列，LLaMA	T5，BART，mBART

批判性思考 (1): 范式界限的模糊化

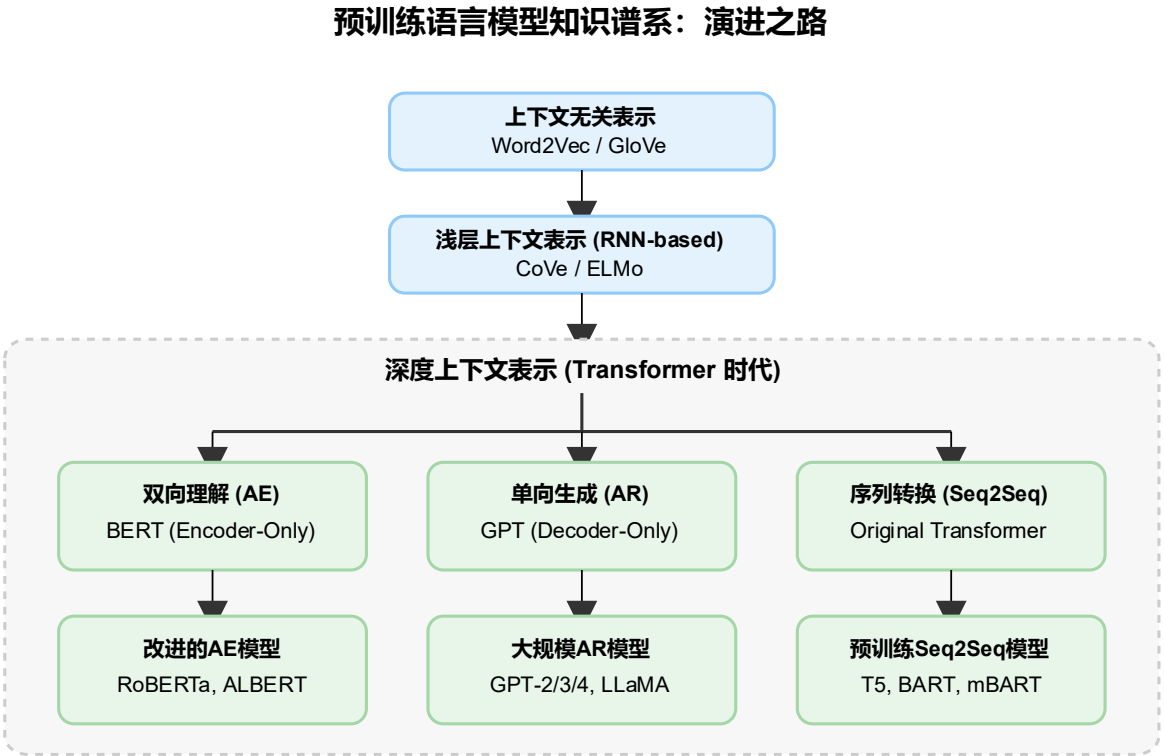
- 纯粹的架构分类正在变得不那么绝对
- Decoder-Only模型的“涌现”NLU能力
  - 通过指令微调 (Instruction Tuning) 和思维链 (Chain-of-Thought) Prompting，大规模的AR模型 (如GPT-3.5/4) 在复杂的NLU和推理任务上也能表现出色
  - 它们不是通过专门的“表示”来解决问题，而是学会了在上下文中“生成”一个推理过程来得到答案
  - 例子
    - BERT (AE): 直接在[CLS]向量上接分类头，判断情感
    - GPT (AR): 通过Prompt Review: "This movie was great!" Sentiment: Positive. Review: "I hated this film." Sentiment:，让模型续写出 Negative
- 结论： 规模和训练方法可以在一定程度上弥补架构上的天然倾向

批判性思考 (2): 统一的趋势与效率的挑战

- 统一的趋势 (The Unification Trend):
  - T5的“文本到文本”框架极具前瞻性，它预示了将所有任务统一到一个通用接口下的可能性。
  - 现代LLM（如GPT-4, Gemini）通过统一的聊天接口处理各种任务，本质上也是这种思想的延伸。
- 效率的挑战 (The Efficiency Challenge):
  - Transformer的自注意力机制具有 O(n^2) 的计算和内存复杂度，成为处理长文本的瓶颈。
  - 研究热点： 各种近似注意力机制 (Efficient Transformers)，旨在将复杂度降低到 O(n)。

知识谱系：演进之路

- 预训练语言模型知识谱系：演进之路



总结与展望

- 没有银弹： AE, AR, 和 Seq2Seq 是针对不同设计目标的权衡，不存在绝对的优劣，只有是否适合特定场景。
- 规模是关键： 模型参数、数据量和计算量的巨大提升，是解锁“涌现能力”和推动范式融合的核心驱动力。
- 未来方向：
  - 多模态： 将文本、图像、声音等信息融合到统一的表示空间。
  - 效率： 探索超越标准Transformer的更高效架构。
  - 可控性与对齐： 如何让模型生成的内容更可靠、可控，并与人类价值观对齐 (Alignment)。