**Chapter 10. Transformers and Large Language Models**

引言：分布假说->Transformers->LLMs

在之前的章节中，提到了分布假说，分布假说认为，即使没有实际世界的接触，人们也能通过文本内容学习到词义，这种知识是基于单词及其共现词的复杂关联。

分布假说提供了一种理解和捕获词义的方法，而Transformers则是实现这种理解的强大工具。分布假说为理解词汇和构建语言模型提供理论基础，而Transformers提供了强大的技术手段。这种结合使得现代语言模型不仅能够捕捉词汇之间的直接关系，还能理解更长距离的依赖关系，这对理解复杂的文本结构非常关键。

Transformers模型，尤其是其核心组成部分——自注意力机制，使得模型能够处理和分析词汇在文本中的每一个出现的上下文环境。自注意力允许模型在处理每个词时，考虑到它在句子中与其他所有词的关系，从而有效地捕捉上下文信息。

Large Language Models（LLMs）如GPT和BERT，是在Transformers架构基础上发展起来的高级应用，它们扩展了Transformers的使用范围，并在多个方面进行了创新和优化。具体提升之处如下：

1. 规模的扩展：

LLMs特别强调模型规模的扩大，包括更多的参数、更大的训练数据集和更复杂的训练策略。例如，GPT-3拥有1750亿个参数，远超其前代模型。

通过增加模型的规模，LLMs能够捕捉更细致、更复杂的语言模式和依赖关系，这提高了模型在各种语言任务上的性能和通用性。

1. 预训练与微调：

LLMs采用预训练和微调的策略。在预训练阶段，模型在大规模的文本语料库上学习语言的广泛特征，然后在微调阶段，模型针对特定任务进行调整。

这种策略使得LLMs在没有显著改变架构的情况下，能够灵活适应从文本摘要到完全自动化的对话系统等多种不同的任务。

1. 自注意力机制的深化应用：

Transformers的核心是自注意力机制，而LLMs通过进一步优化和调整这一机制，提升了处理长距离依赖和复杂语境的能力。

高级的注意力策略，如多头注意力，使得模型能够从多个子空间同时学习信息，这增强了模型对语言多维度特性的理解。

1. 多样化的训练任务：

不同于初代Transformers模型通常专注于单一任务，LLMs通过在预训练阶段引入多种任务，如掩码语言建模（BERT）、下一句预测等，增强了模型的适应性和鲁棒性。

这种多任务学习策略使得LLMs能够更好地理解语言的广泛用法和复杂结构，提升在特定任务上的表现。

1. 生成能力的提升：

LLMs尤其擅长生成文本，这得益于其大规模的训练数据和复杂的模型架构。

在生成任务中，如文本生成、代码生成、图像描述等，LLMs表现出色，显示了其理解和创造性输出的能力。

**10.1 The Transformer: A Self-Attention Network**

这一部分对Transformers作了详细的介绍

1. Treansformer的基本架构

Treansformer接受一串单词作为输入，并输出下一个单词的预测以及表示每个输入单词上下文意义的上下文嵌入。

与第九章介绍的LSTM相比，Treansformer可以处理远距离信息，但不基于递归连接，因此在大规模实现时更高效。

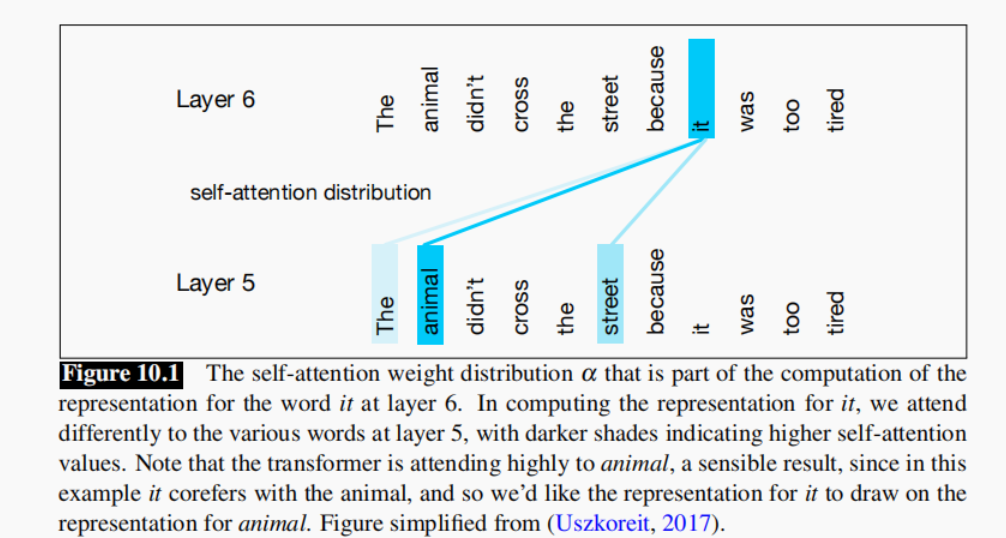
Treansformer由多个Treansformer块组成，每个块都是一个多层网络，将输入向量序列映射到同等长度的输出向量序列。

这些块由简单的线性层、前馈网络和自注意力层组合而成，自注意力层是Treansformer的关键创新，允许网络直接从任意大的上下文中提取和使用信息。

使用的自注意力机制

自注意力允许模型在计算时考虑到输入序列中任意远的上下文信息。

通过自注意力，每个单词的表示都是通过考虑其在前一层的表示和邻近单词的表示来计算的，目标是产生每个位置上每个单词的上下文化表示。



使用Treansformer进行语言建模

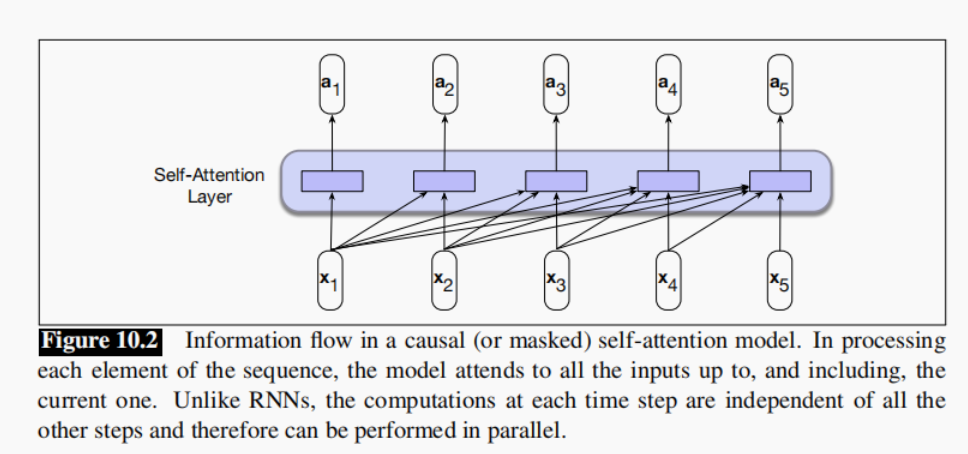
可以将Treansformer块与某些输入和输出机制结合使用，作为语言模型来预测上下文中先前单词的后续单词。

每层Treansformer都构建越来越丰富的输入单词或令牌的上下文化表示。这种表示旨在表达单词在特定上下文中的意义，与第六章中看到的静态向量相比，这些向量更具上下文化。

1. 因果或回溯型自注意力 (Causal or backward-looking self-attention

上下文定义: 自注意力中的上下文有两种使用方式。在因果或回溯型自注意力中，上下文指的是所有先前的词语。而在一般的双向自注意力中，上下文可以包括未来的词语。

焦点限定: 本章重点讨论因果、回溯型自注意力；将在第11章介绍双向自注意力。



信息流动: 图10.2展示了单个因果或回溯型自注意力层中的信息流动。与整个Transformer一样，自注意力层将输入序列 (x1,...,xn) 映射为相同长度的输出序列 (a1,...,an)。

处理原则: 在处理每个输入项时，模型能够访问到当前项及之前的所有输入，但无法访问当前项之后的信息。每个项的计算与其他项相互独立。

应用与优势: 保证了该方法可以用于创建语言模型并用于自回归生成，意味着可以轻松地并行化模型的前向推理和训练。

1. 自注意力的形式化描述

直观理解: 自注意力是一种通过整合前一层词语的信息来计算给定层次上一个词的表示的方法。

关键操作: 核心直观理解是，通过比较一个感兴趣的项与其他项集合来揭示它们在当前上下文中的相关性。这些比较的结果用于为当前输入序列计算输出序列。

自注意力计算的核心直观是比较感兴趣的项目与当前上下文中其他项目的相关性。例如，计算输出序列时，会利用这些比较结果。

1. 并行化自注意力

由于每个输出独立计算，整个自注意力过程可以并行化，利用矩阵乘法来有效地计算整个输入序列的自注意力

1. 掩蔽未来的信息

在语言建模中，为避免提前泄露信息（如预测下一个单词时已知该单词），会通过在矩阵中将未来单词的部分设置为零（或负无穷）来掩蔽这些信息。

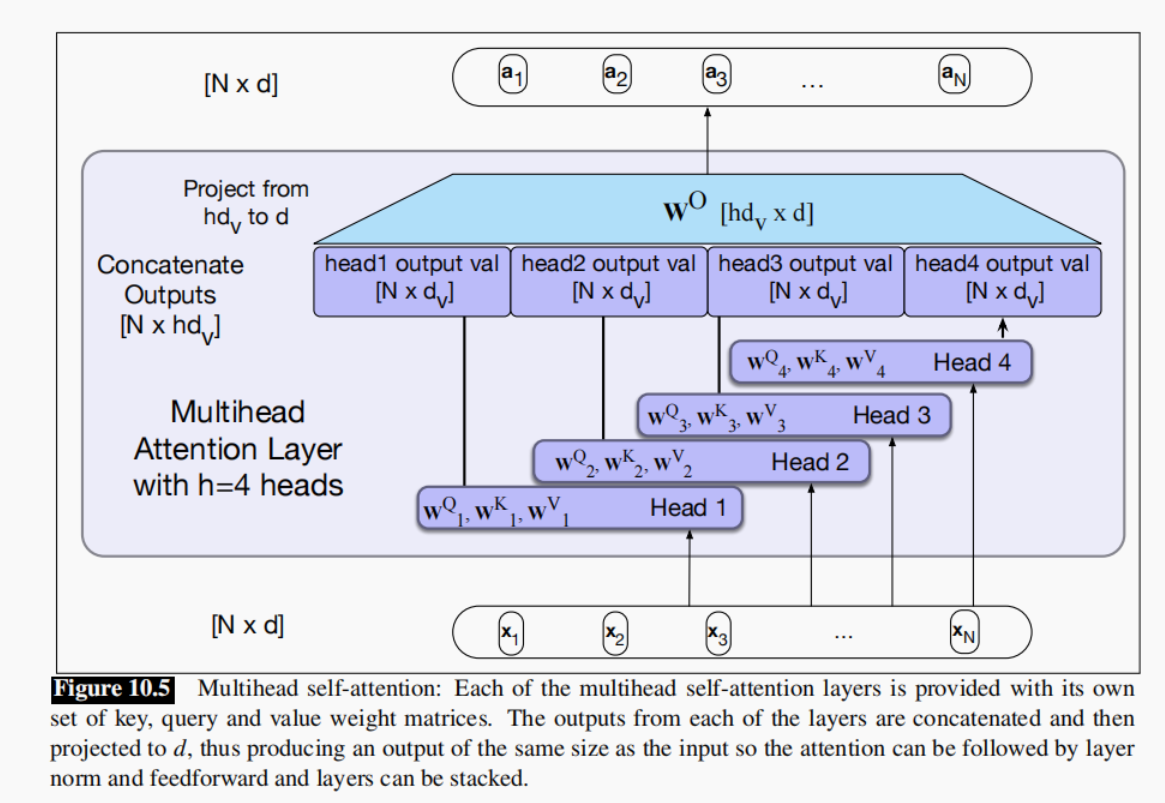
**10.2 Multihead Attention**

多样化的词语关系：在句子中，不同的词语之间可能同时存在多种关系，例如语法、语义和话语关系。单一自注意力模型难以捕捉所有这些并行的关系。

多头自注意力层：Transformer通过多头自注意力层来解决这一问题。这些层包含多个称为“头”的自注意力层，它们并行存在于模型的同一深度，每个头拥有独立的参数集，能够学习输入之间关系的不同方面。

实现方式：每个头i都有自己的键、查询和值矩阵（WiK, WiQ, 和WVi）。这些矩阵将输入投影为各自头的键、值和查询嵌入，其余自注意力计算保持不变。

多头输出与整合：每个头的输出维度为N×dv，多头层的输出为h个头输出的矩阵合并，形成维度为N×hdv的单一输出。然后通过另一个线性投影WO ∈ Rhdv×d，将其重新塑形为每个标记的原始输出维度。



示意图与应用：图10.5展示了具有四个自注意力头的多头自注意力的实现。在Transformer中，通常使用多头层替代单一自注意力层。

**10.3 Transformer Blocks**

构成：Transformer块的核心是自注意力计算，除此之外还包括前馈层、残差连接和归一化层（通常称为“层归一化”）。

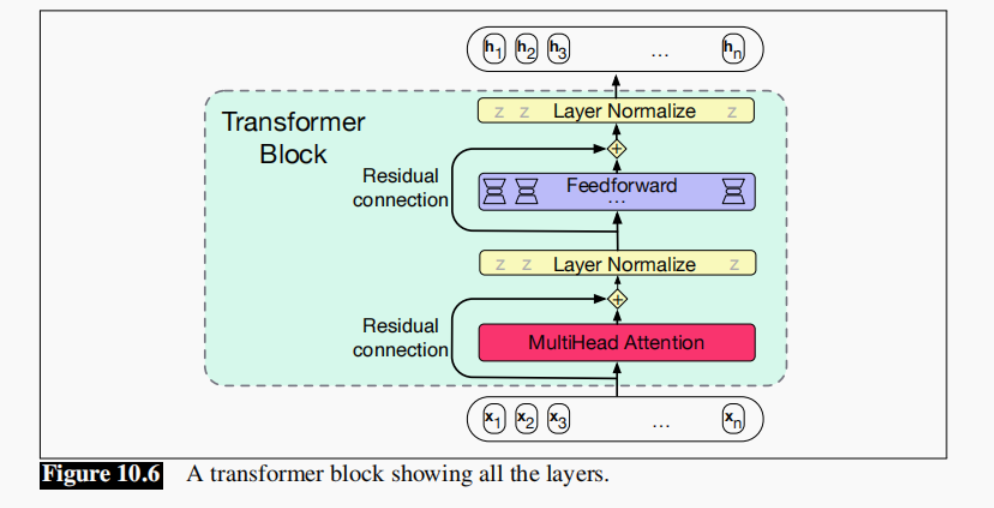
前馈层：前馈层包含N个逐位置网络，每个位置各有一个。这是一个全连接的两层网络，其权重在每个位置相同，但每层的参数不同。与注意力不同，前馈网络在每个位置是独立计算的，因此可以并行处理。

残差连接：残差连接是从低层直接传递到高层的连接，不经过中间层。这种设计有助于学习，使高层能直接访问来自低层的信息。

层归一化：归一化处理通过层归一化完成，它是用于深度神经网络中改善训练性能的归一化形式之一，可以保持隐藏层值在有利于基于梯度的训练的范围内。层归一化基于统计学中的标准分，对单个向量进行应用。

整合函数：Transformer块的功能可以表达为两个主要的连续操作——自注意力和前馈网络的输出，这两个输出都经过残差连接和层归一化的处理。

输入输出匹配：Transformer块的输入和输出维度匹配，这样可以堆叠多个块。每个输入标记xi具有维度d，因此输入X和输出H的形状均为[N×d]。



**10.4 The Residual Stream view of the Transformer Block**

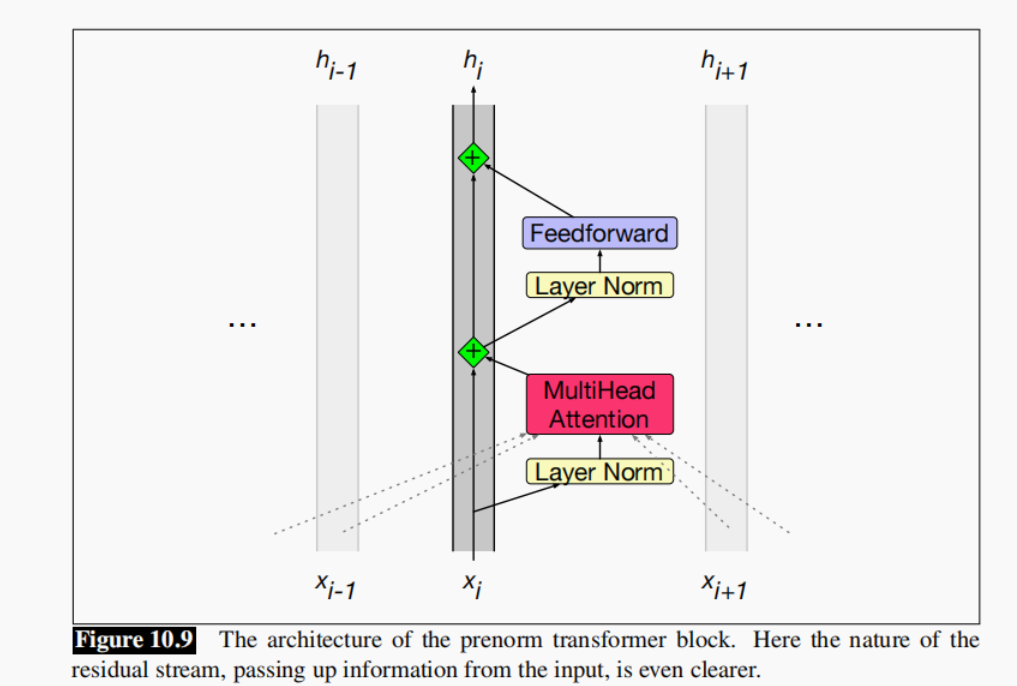
输入和输出形式：在之前的讨论中，Transformer块处理的是形状为[N×d]的整个N标记输入X，并产生同样形状的输出。尽管这种打包方式在计算上高效，但要理解Transformer的工作原理，考虑单个标记向量xi在每个Transformer块中的处理可能更为清晰。

残差流的定义和视觉化：每个标记xi通过Transformer块的处理可以视为一个d维表示的流，称为残差流。这个流的输入是一个标记的嵌入，这个嵌入通过残差连接传递，并加上前馈和注意力层的输出，然后通过层归一化处理，生成此块的输出hi，作为下一个层块的输入。图10.7展示了这一过程。

多头注意力的特殊作用：在所有组件中，只有多头注意力组件会从其他标记（其他残差流）读取信息。这意味着，注意力的输出虽然添加到了该标记的嵌入流中，但它实际上是从相邻标记的残差流中“移动”信息到当前流中。

数学表示和处理：对于单个Transformer块，残差流中标记的处理可以通过一系列的数学操作来描述，包括多头注意力、加法操作、层归一化和前馈网络。每一步都对应一个操作，以确保每个组件的输入和输出都保持在适当的维度。

预归一化与后归一化架构：Transformer还有一种常用的变体架构，称为预归一化架构，其中层归一化发生在注意力层和前馈层之前，而不是之后。这种架构有助于在许多情况下提高性能。预归一化变体中，信息的流向和处理在视觉上更为清晰（如图10.9所示），最后的输出也通过一个额外的层归一化进行处理。



**10.5 The input: embeddings for token and position**

输入X的来源：Transformer的输入X来源于序列中N个标记的嵌入，每个词在上下文中都有一个嵌入表示。

标记嵌入和位置嵌入：Transformer通过计算两种嵌入来处理输入——标记嵌入和位置嵌入。标记嵌入为每个输入标记提供初始表示，而位置嵌入指定每个位置的独特表示。

嵌入的获和使用：给定一个标记序列，首先将每个标记转换为词汇表索引，然后使用这些索引从嵌入矩阵E中选择相应的行。此外，还可以通过一种称为绝对位置的方法来随机初始化位置嵌入，这些嵌入在训练过程中学习并调整。

组合嵌入：为了表示每个标记在序列中的位置，将标记嵌入和位置嵌入相加，产生一个新的嵌入，这个新嵌入作为进一步处理的输入。

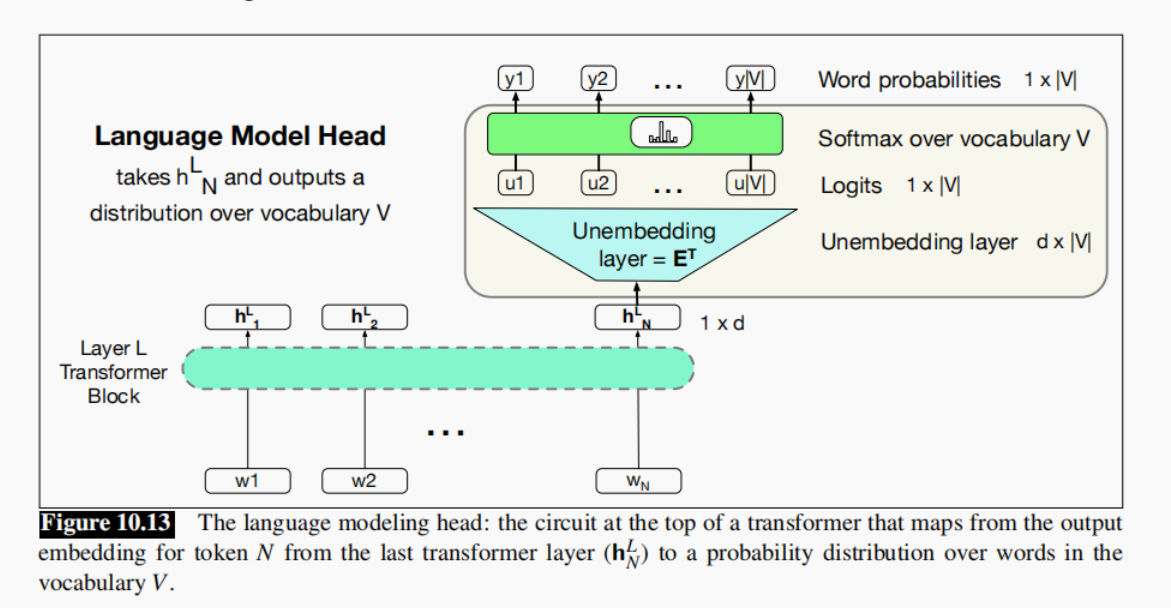
位置嵌入的替代方法：绝对位置嵌入的一个潜在问题是对于输入中的初位位置有大量的训练示例，而在外部长度限制的位置较少，这可能导致测试时泛化不良。作为替代，可以选择一种静态函数（如正弦和余弦函数的组合），这种方法在原始的Transformer工作中使用，以捕捉位置之间的内在关系。

**10.6 The Language Modeling Head**

语言模型头部的定义：在Transformer模型中，语言模型头部是指加在基本Transformer架构顶部的附加神经电路，用于执行特定的任务，如语言建模。语言模型头部的主要功能是进行词预测，即根据给定的词语上下文，为可能的下一个词分配概率。

词预测功能：语言模型通过给定的词语上下文计算下一个词的概率分布。例如，如果上文是“Thanks for all the”，模型将计算“fish”作为下一个词的概率P(fish|Thanks for all the)。这使我们能够为整个词汇表中的每个可能的下一个词分配条件概率。

上下文窗口大小：在Transformer语言模型中，上下文的大小是Transformer的上下文窗口的大小，这可以非常大，例如2048甚至4096个标记，以适应大型模型。

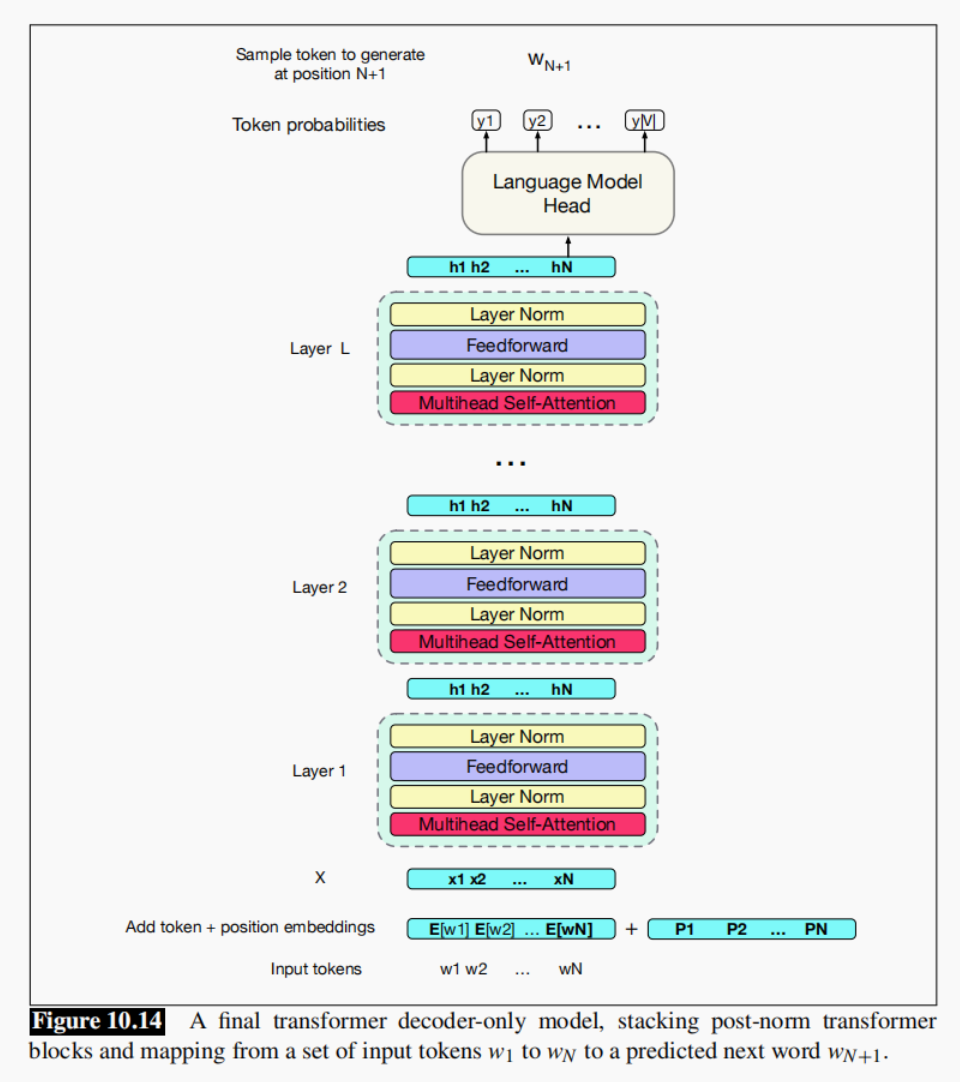


实现语言模型头部：语言模型头部的任务是利用最后一个Transformer层的输出，来预测位置N+1处的即将出现的词。如图10.13所示，这包括取最后一个标记在最后一层的d维输出嵌入，并生成词汇表上的概率分布。

线性层和逆嵌入层：语言模型头部的第一个模块是一个线性层，其任务是将最后一个标记的输出嵌入从维度[1×d]投影到一个对数向量（或得分向量），这个向量为词汇表V中的每个词提供一个得分。这个线性层通常与嵌入矩阵E的转置绑定（权重绑定），使得在输入阶段和输出阶段使用相同的权重。

Softmax层和文本生成：Softmax层将对数向量转换为词汇表上的概率分布。我们可以使用这些概率来帮助分配给定文本的概率，或者通过从这些概率中采样一个词来生成文本。生成文本时可以采用“贪婪”解码（选择最高概率的词），或者使用更复杂的采样方法来增加生成文本的多样性。

概率分布和文本生成：模型通过计算每个可能的下一个词的概率来生成文本。选择概率最高的词进行输出通常被称为贪婪解码，但在实践中，为了避免文本过于通用和重复，通常不使用贪婪解码。相反，会使用更复杂的采样方法来引入更多的多样性。



总体架构：如图10.14所示，完整的Transformer解码器模型通过堆叠后规范化Transformer块，从一组输入标记w1到wN映射到预测的下一个词wN+1。这种架构使得模型能够在每一步都融入之前的上下文和自己生成的输出，从而在生成过程中保持对原始文章的直接访问。

**10.7 Large Language Models with Transformers**

条件生成的任务：Transformer基于的大型语言模型主要用于条件生成任务，即基于输入文本（提示）生成文本。Transformer的长上下文能力（最多可达4096个标记）使其在条件生成中特别有力，因为它可以追溯到很远的提示文本。

文本完成的例子：例如，语言模型可能会被给定一个文本前缀，并要求生成可能的补全。这一过程中，模型可以直接访问初始的上下文以及其自身随后生成的所有输出。这种能力使得基于Transformer的大型语言模型非常强大。

语言建模的实际应用：大型语言建模的洞见在于，许多实际的自然语言处理（NLP）任务可以被视为词预测任务，一个足够强大的语言模型可以高精度地解决这些任务。例如，情感分析可以通过给语言模型一个上下文并比较“positive”和“negative”词的条件概率来实现。

复杂任务的处理：更复杂的任务，如问答，也可以通过语言建模来解决。通过给定一个问题和一个提示（如"A:"表示答案应该接在后面），语言模型可以预测可能的答案。这种方式可以扩展到需要生成较长回应的任务，如文本摘要。

文本摘要：文本摘要任务要求从一篇长文章生成一个简短有效的摘要。通过给语言模型提供一篇文章并跟随一个如"tl;dr"的标记，模型可以生成一个摘要。这显示了Transformer在处理长文本和生成相关回应方面的能力。

解码策略：在文本生成时，常用的解码策略包括贪婪解码，即总是选择给定上下文中最可能的词。然而，由于贪婪解码往往生成预测性很强、重复性较高的文本，实际应用中更倾向于使用引入更多多样性的采样方法。

贪婪解码的局限性：贪婪解码因其可预测性而可能导致生成的文本通用且重复。这种方法在一些非常受限的任务中（如机器翻译）可能表现良好，但在大多数其他任务中，更复杂的采样方法更受欢迎。

**10.8 Large Language Models: Generation by Sampling**

生成过程的核心：大型语言模型的生成核心在于根据上下文和模型分配给可能词汇的概率选择下一个要生成的单个词。这一任务称为解码（decoding），具体来说，是在左到右的方式（或对于如阿拉伯语这样从右到左阅读的语言，则为从右到左）连续选择下一个词，这称为自回归生成或因果语言模型生成。

采样方法：在大型语言模型中，最常见的解码方法是采样。采样意味着根据模型为词汇分配的概率随机选择词汇。这样，我们更有可能生成模型认为在上下文中有高概率的词汇，而不太可能生成模型认为有低概率的词汇。

随机采样的问题：虽然随机采样主要会生成有意义且概率高的词汇，但分布尾部的许多低概率词汇加起来可能构成足够大的比例，使得它们被足够频繁地选择，从而生成奇怪的句子。因此，我们通常使用避免生成极不可能词汇的采样方法。

Top-k采样：

Top-k采样是贪婪解码的简单泛化。

在生成单个最可能的词汇之外，首先将分布截断为最可能的k个词汇，重新标准化以产生合法的概率分布，然后根据这些重新标准化的概率随机采样。

当k=1时，Top-k采样等同于贪婪解码。将k设置为大于1的数字，有助于选择不一定是最可能的，但仍然足够可能的词汇，从而生成更多样化但仍然足够高质量的文本。

核心采样或Top-p采样：

Top-p采样不保留顶部k个词汇，而是保留顶部p百分比的概率质量。

这种方法的目标相同：截断分布以移除非常不可能的词汇。

通过测量概率而不是词汇数量，希望这种度量在非常不同的上下文中更为稳健，动态增减候选词汇池。

温度采样：

温度采样不截断分布，而是重塑它。这种方法的直觉来自热力学，其中高温度的系统非常灵活，可以探索许多可能的状态，而低温度的系统可能只探索一部分较低能量（更好）的状态。

通过在通过softmax标准化之前将logit除以温度参数τ来实现这一直觉。低温度采样中，τ ∈ (0,1]。当τ接近1时，分布变化不大；但τ越低，传递给softmax的分数越大，从而产生的分布将更倾向于高概率词汇，使分布更加贪婪。

**10.9 Large Language Models: Training Transformers**

自监督训练算法：

为了将Transformer训练为语言模型，我们采用自监督训练方法。这意味着我们不需要对数据添加任何特殊的标签；自然词序列本身就是监督信号。

训练过程中，模型每一时间步t都被要求预测下一个词。我们使用交叉熵损失函数来最小化预测真实下一个词的错误。

交叉熵损失：

交叉熵损失测量预测的概率分布与正确分布之间的差异。在语言模型中，正确的分布由下一个词的一热向量表示，模型为正确的下一个词分配的概率决定了语言建模的交叉熵损失。

模型的任务是在每个词位置t输入正确的词序列w1:t，并利用这些来计算可能的下一个词的概率分布，从而计算下一个词wt+1的模型损失。

教师强制：

在训练中，始终提供模型正确的历史序列来预测下一个词的做法称为教师强制。这意味着不管模型在上一步预测了什么，都使用正确的词序列来估计下一个词的概率。

并行处理能力：

与基于RNN的模型相比，Transformer允许在训练中并行处理每个序列元素的输出，因为每个元素的输出都是单独计算的。

训练数据：

大型语言模型主要训练于从网上抓取的文本数据，这些数据经过精心策划的增补。训练语料库很大，含有许多自然示例，如FAQ列表中的问答对、各种语言间的句子翻译等。

训练语料来源：

网络文本通常来自自动抓取的网页语料库，如Common Crawl。例如，C4语料库就是一个经过清理的1560亿令牌的英语语料库，去除了重复内容、非自然语言内容等。

模型规模的扩展定律：

大型语言模型的性能主要由三个因素决定：模型大小（参数数量）、数据集大小和用于训练的计算资源。

性能与这些因素的关系被称为扩展定律，表明大型语言模型的性能（损失）与这些训练属性呈幂律关系。

例如，通过增加参数（增加更多层或更宽的上下文等）、在更多数据上训练或进行更多迭代训练，都可以改善模型性能。

**10.10 Potential Harms from Language Models**

错误信息生成（幻觉现象）：

语言模型可能产生不真实的信息，这一问题被称为“幻觉”。尽管模型被训练以生成可预测和连贯的文本，但现有的训练算法无法保证生成的内容的准确性或真实性。这在需要准确信息的应用中会引起重大问题。

生成有害语言：

研究显示，即使是完全无害的提示也可能引导大型语言模型输出仇恨言论和辱骂用户的语言。此外，言模型还可能生成关于许多人口群体的刻板印象和负面态度。

偏见来源：

偏见的一个来源是训练数据。大型语言模型的训练数据集中可能包含从被禁网站抓取的有害文本。训练数据的生成还可能过度集中在来自美国和发达国家的作者上，这可能使生成的内容偏向这些群体的观点或话题。此外，语言模型可能放大训练数据中的人口统计学和其他偏见。

被恶意使用的风险：

语言模型可能被恶意行为者用于生成误导信息、网络钓鱼或其他社会有害活动的文本。研究显示大型语言模型能够生成模仿在线极端分子的文本，存在放大极端运动及其激进化和招募风险。

隐私问题：

语言模型可能泄露有关其训练数据的信息，例如个人姓名、电话号码和地址。如果大型语言模型是在私有数据集（如电子健康记录）上训练的，这将是一个问题。

版权问题：

大型语言模型通常在受版权保护的文本上进行训练。在某些国家，如美国，公平使用原则允许使用受版权保护的内容来构建语言模型，但如果使用它们生成的文本与它们训练所用的文本的市场竞争，则可能不被允许。

减轻这些危害的研究：

寻找减轻这些危害的方法是自然语言处理（NLP）中一个重要的当前研究领域。至少，仔细分析用于预训练大型语言模型的数据非常重要，这有助于理解有关毒性、偏见、隐私和公平使用的问题。确保语言模型包括数据表或模型卡，提供用于训练它们的语料库的完整可复制信息，这一点极为重要。开源模型可以明确其确切的训练数据。要求模型在这些方面的透明度也在被各国政府的法规逐步纳入。

提出问题：

大型语言模型在实际应用中如何处理输入文本的版权问题？有哪些常见的法律挑战和解决策略？

大语言模型现在有完善的水印技术，来判断文本是否由大预言模型生成吗？

标准的Transformer模型由于其全自注意力机制的特性，面对长文本时会遇到显著的计算和存储挑战，有什么方法来解决这个问题？并且如何来评估处理长文本的性能。