基础

大语言型号&文本生成

作者：Mohammadamin Barekta in、Anant Nawalgaria、Daniel J. Mankowitz、Majd Al Merey、Yaniv Leviathan、Massimo Mascaro、Matan Kalman、Elena Buchatskaya、Aliaksei Severyn 和 Antonio Gulli

Google

致谢

审稿人和贡献者

亚当·萨德沃夫斯基

吴永辉

安德鲁·戴

埃菲·科基奥波卢

查克·苏格内特

阿列克谢·弗拉先科

欧文·休伊曾加

策展人和编辑

安东尼奥·古利

阿南特·纳瓦尔加里亚

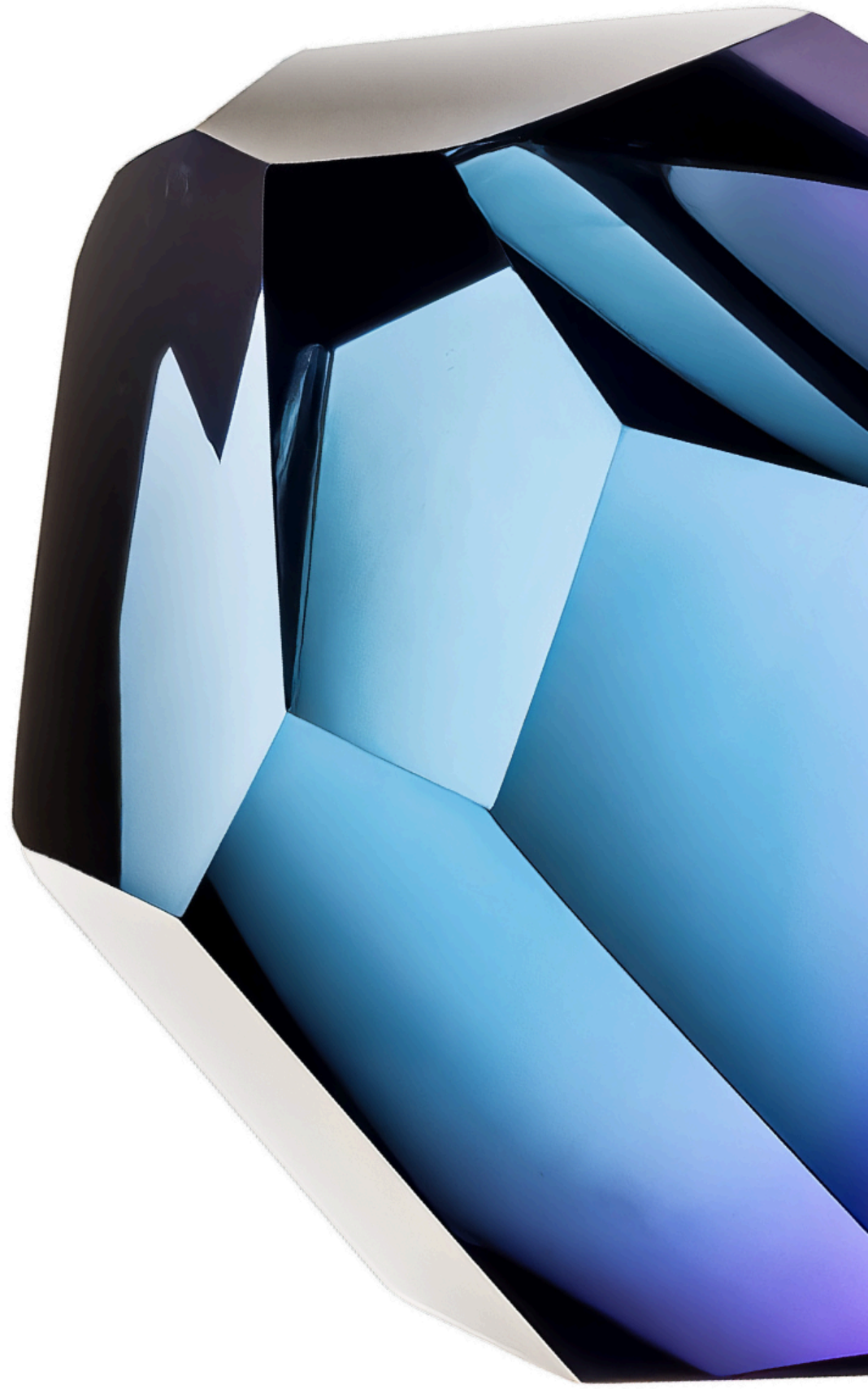
格蕾丝·莫里森

技术撰稿人

马克·艾弗森

设计师

迈克尔·兰宁



目录

我们相信，这种新技术有潜力随时为人们提供帮助、补充、赋权和激励几乎涵盖所有领域。

介绍

大型语言模型（ LLMs ）的出现代表了语言世界的巨大转变。人工智能。他们处理、生成和理解用户意图的能力是从根本上改变我们与信息和技术互动的方式。

LLM是一种先进的人工智能系统，专门从事处理、理解并生成类似人类的文本。这些系统通常实现为一个深度神经网络，并接受大量文本数据的训练。这使他们能够学习复杂的语言模式，使他们能够执行各种任务，比如机器翻译、创意文本生成、问答、文本摘要、以及更多推理和语言导向的任务。本白皮书深入探讨了构建大型语言的各种架构和方法的时间表发布时使用的模型和架构。它还讨论了精细-

为特定领域或任务定制LLM调整技术、方法
更有效的训练以及加速推理的方法。然后是这些
通过各种应用程序和代码示例。

为什么语言模型很重要

LLMs较之前的技术水平取得了令人印象深刻的性能提升
需要回答问题或复杂的各种不同且复杂的任务
推理，使许多新的应用变得可行。其中包括语言翻译、代码
生成和完成、文本生成、文本分类和问答，
仅举几例。尽管基础LLMs接受过大量各种任务的培训
的数据开箱即用并显示紧急行为（例如能够
执行他们没有直接训练过的任务）他们也可以适应解决
开箱即用的性能未达到流程所需水平的特定任务
称为微调。这需要的数据和计算资源比
从头开始培训LLM 。 LLMs可以进一步推动和引导以达到期望的目标
快速工程学科的行为：构成的艺术和科学
提示和LLM的参数以获得所需的响应。

最大的问题是：这些大型语言模型如何工作？下一节将探讨
LLMs的核心构建模块，重点关注变压器架构及其从
最初的“Attention is all you need”论文适用于最新型号，例如 Gemini、Google 的
最有能力的LLM 。我们还涵盖培训和微调技术，以及方法
提高响应生成速度。白皮书最后给出了一些示例
语言模型在实践中的使用方式。

大型语言模型

语言模型预测单词序列的概率。通常，当给出
文本的前缀，语言模型为后续单词分配概率。例如，
给定前缀“美国最著名的城市是……”，语言模型可能会预测高
单词“New York”和“Los Angeles”的概率以及单词的低概率
“笔记本电脑”或“苹果”。您可以通过存储 n 元语法表来创建基本语言模型，同时
现代语言模型通常基于神经模型，例如 Transformer。

在变压器发明之前，循环神经网络（RNNS）是流行的
序列建模方法。特别是“长短期记忆”（LSTM）和
“门控循环单元”（GRU）是常见的架构。这个领域包括语言
机器翻译、文本分类、文本摘要和问题等问题
回答等。 RNN 按顺序处理输入和输出序列。他们
根据先前的隐藏状态和当前的隐藏状态生成一系列隐藏状态
输入。 RNN 的顺序性质使其计算密集且难以并行化
在训练期间（尽管最近状态空间建模的工作正在尝试克服
这些挑战）。

另一方面，变压器是一种可以处理序列的神经网络
由于自我关注机制，并行的代币数量。这意味着变压器
可以更好地对长期上下文进行建模，并且比 RNN 更容易并行化。这使得
与 RNN 相比，它们的训练速度明显更快，并且更强大，可以处理长
长序列任务中的术语依赖性。然而，原来的 self-attention 的成本
转换器的上下文长度是二次的，这限制了上下文的大小，而
RNN 理论上具有无限的上下文长度。变形金刚已成为最
近年来序列建模和转导问题的流行方法。

在这里，我们讨论变压器模型的第一个版本，然后继续讨论更多
最近的先进模型和算法。

变压器

Transformer 架构是 Google 于 2017 年开发的，用于翻译模型。
它是一种序列到序列模型，能够从一个域转换序列
到另一个域中的序列。例如，将法语句子翻译成英语
句子。原始的 Transformer 架构由两部分组成：编码器和
解码器。编码器将输入文本（例如法语句子）转换为表示形式，
然后将其传递给解码器。解码器使用此表示来生成
自回归地输出文本（例如，英文翻译）。值得注意的是，输出的大小
变压器编码器的输入大小是线性的。图 1 显示了设计
原始变压器架构。

变压器由多层组成。神经网络中的层包含一组
对数据执行特定转换的参数。在图中你可以看到
一些层的示例，包括多头注意力、添加和规范、前馈、
Linear、Softmax等。层可以细分为输入层、隐藏层和输出层。
输入层（例如输入/输出嵌入）是原始数据进入的层
网络。输入嵌入用于表示模型的输入标记。输出
嵌入用于表示模型预测的输出标记。例如，在
机器翻译模型，输入嵌入将代表源中的单词
语言，而输出嵌入将代表目标语言中的单词。
输出层（例如 Softmax）是产生网络输出的最后一层。这
隐藏层（例如，多头注意力）位于输入层和输出层之间
魔法发生的地方！

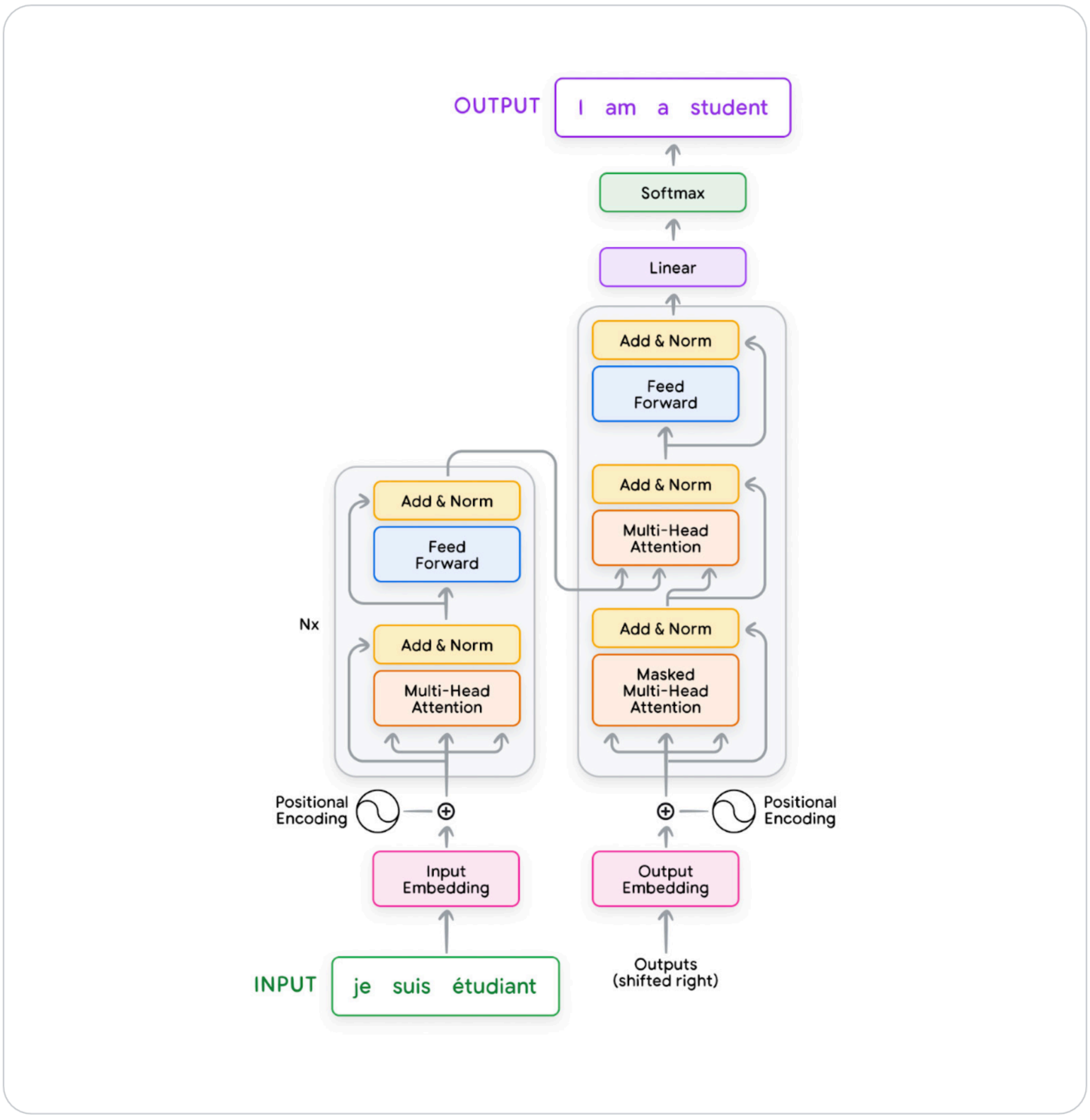

OUTPUT
I am a student
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
INPUT
je suis étudiant
Outputs (shifted right)

图1. 原始变压器(P.C:)

为了更好地理解变压器中的不同层，让我们使用法语到英语
以翻译任务为例。在这里，我们解释一下如何将法语句子输入到
变压器并输出相应的英文翻译。我们还将描述每一个
图 1 中变压器内部的组件。

输入准备和嵌入

为了为 Transformer 准备语言输入，我们将输入序列转换为标记并
然后进入输入嵌入。在高层次上，输入嵌入是一个高维向量
代表句子中每个标记的含义。然后将这个嵌入输入到
变压器进行处理。生成输入嵌入涉及以下步骤：

1. 标准化（可选）：通过删除多余的空白来标准化文本，
   口音等

2. 标记化：将句子分解为单词或子词并将它们映射为整数
   词汇表中的标记 ID。

3. Embedding：将每个token ID转换为其对应的高维向量，
   通常使用查找表。这些都是可以在训练过程中学到的。

4. 位置编码：添加有关序列中每个标记的位置的信息
   帮助转换器理解词序。

这些步骤有助于为变压器准备输入，以便它们能够更好地
理解课文的意思。

多头注意力

将输入标记转换为嵌入向量后，将这些嵌入输入
多头注意力模块（见图 1）。自注意力是一个重要的机制
变压器；它使他们能够专注于与以下内容相关的输入序列的特定部分
手头的任务并更有效地捕获序列内的远程依赖性
与传统的 RNN 相比。

理解自我注意力

考虑下面的句子：“老虎从树上跳下来喝水，因为它
渴了。”自注意力有助于确定不同单词之间的关系
句子中的短语。例如，在这句话中，“the Tiger”和“it”是同一个宾语，
所以我们期望这两个词有很强的联系。自注意力实现了这一点
通过以下步骤（图2）：

1. 创建查询、键和值：每个输入嵌入乘以三个学习的值
   权重矩阵（Wq、Wk、Wv）来生成查询（Q）、键（K）和值（V）向量。这些
   就像每个单词的专门表示。

   - 查询：查询向量帮助模型询问“序列中还有哪些单词”
     与我有关吗？”

   - 关键：关键向量就像一个标签，帮助模型识别单词的含义
     与序列中的其他单词相关。

   - 值：值向量保存实际的单词内容信息。

2. 计算分数：计算分数以确定每个单词应获得的分数
   “参加”其他词。这是通过取 1 的查询向量的点积来完成的
   word 与序列中所有单词的键向量。

3. 归一化：分数除以关键向量维度（dk）的平方根
   为了稳定性，然后通过一个softmax函数来获得注意力权重。这些
   权重表示每个单词与其他单词的关联程度。

4. 加权值：每个值向量乘以其对应的注意力权重。
   对结果进行总结，为每个单词生成上下文感知的表示。

图2. 多头注意力模块中计算self-attention的流程(P.C:)

实际上，这些计算是通过堆叠查询、键来同时执行的
将所有标记的值向量转化为 Q、K 和 V 矩阵，并将它们相乘为
如图3所示。

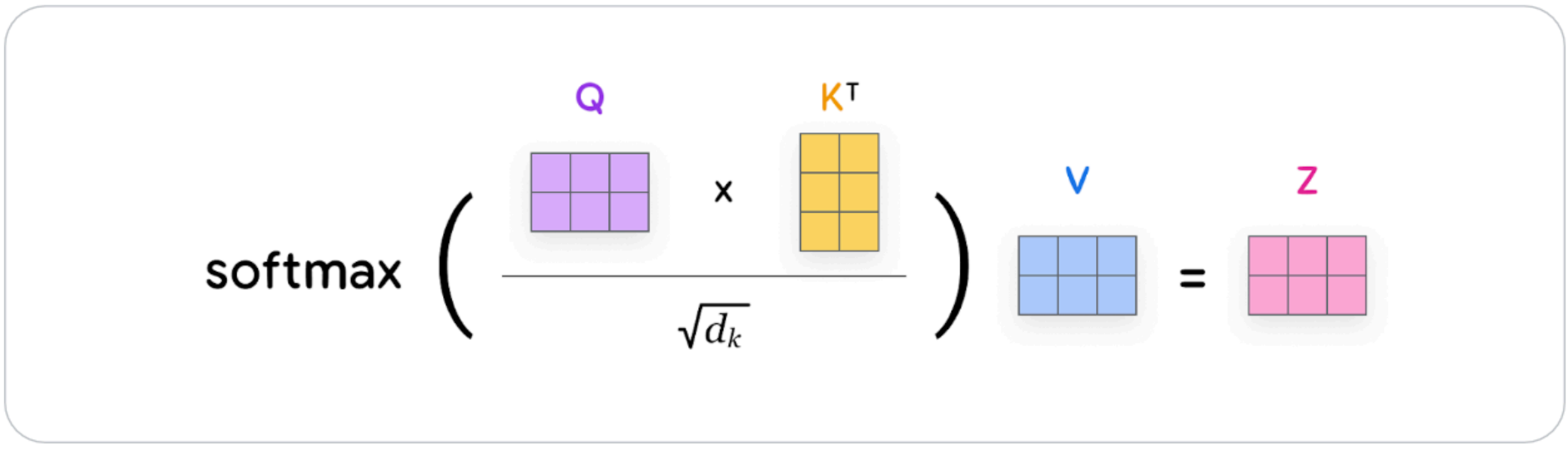


图 3. 注意力的基本操作，其中 Q=query，K=Keys 和 V=Value，Z=Attention，d_k = 查询和键的维度（P.C:）

多头注意力：多样性的力量

多头注意力采用多组 Q、K、V 权重矩阵。这些并行运行，
每个“头”都可能关注输入关系的不同方面。输出
来自每个头部的数据被连接并线性变换，使模型更加丰富
输入序列的表示。

多头注意力的使用提高了模型处理复杂语言的能力
模式和远程依赖性。这对于需要细致入微的任务至关重要
理解语言结构和内容，例如机器翻译、文本
总结、提问。该机制使变压器能够考虑
输入的多种解释和表示，这增强了其性能
这些任务。

层归一化和残差连接

Transformer 中的每一层均由多头注意力模块和前馈模块组成
层，采用层归一化和残差连接。这对应于添加
和图1中的Norm层，其中“Add”对应于残差连接，“Norm”对应于
对应于层归一化。层归一化计算均值和方差
激活的数量以标准化给定层中的激活。这通常是为了
减少协变量偏移并改善梯度流，以在期间产生更快的收敛
培训并提高整体表现。

残余连接将输入传播到一层或多层的输出。这有
使优化过程更容易学习的效果，也有助于处理消失问题
和爆炸梯度。

Add 和 Norm 层应用于多头注意力模块和 feed-
前向层将在下一节中描述。

前馈层

多头注意力模块和后续“Add and Norm”层的输出为
馈入每个变压器块的前馈层。该层应用位置方式
序列中每个位置独立地对数据进行转换，这使得
将额外的非线性和复杂性纳入模型的表示中。这

前馈层通常由两个具有非线性激活的线性变换组成
函数，例如 ReLU 或 GELU，介于两者之间。这种结构进一步增加了代表性
模型的力量。数据经过前馈层处理后，
另一个“添加和规范”步骤，有助于深度学习的稳定性和有效性
变压器模型。

编码器和解码器

原始的 Transformer 架构依赖于编码器和解码器的组合
模块。每个编码器和解码器由一系列层组成，每个层
包括关键组件：多头自注意力机制、位置式馈送
前向网络、归一化层和残差连接。

编码器的主要功能是将输入序列处理成连续的
保存每个标记的上下文信息的表示。输入的顺序是第一个
标准化、标记化并转换为嵌入。位置编码被添加到
这些嵌入保留序列顺序信息。通过自注意力机制，
序列中的每个令牌都可以动态地关注任何其他令牌，从而理解
序列中的上下文关系。编码器的输出是一系列
表示整个输入序列的嵌入向量 Z。

解码器的任务是根据提供的上下文生成输出序列
由编码器的输出Z。它以逐个令牌的方式运行，从start-开始
无序标记。解码器层采用两种类型的注意力机制：
自注意力和编码器-解码器交叉注意力。屏蔽自注意力确保
每个位置只能关注输出序列中较早的位置，从而保留
自回归性质。这对于防止解码器访问至关重要
输出序列中的未来标记。编码器-解码器交叉注意力机制
允许解码器利用上下文关注输入序列的相关部分
由编码器生成的嵌入。这个迭代过程一直持续到解码器
预测序列结束标记，从而完成输出序列生成。

最近的大多数 LLMs 采用了变压器架构的仅解码器变体。这
该方法放弃了传统的编码器-解码器分离，而是直接关注
从输入生成输出序列。输入序列经历类似的

在输入解码器之前进行嵌入和位置编码的过程。这
然后解码器使用屏蔽自注意力为每个后续的生成预测

基于先前生成的令牌的令牌。这种简化的方法简化了
用于特定任务的架构，其中编码和解码可以有效地合并。

培训变压器

在谈论机器学习模型时，重要的是要区分
训练和推理。训练通常是指修改模型的参数，
并涉及损失函数和反向传播。推理是仅使用模型时的情况
用于预测输出，而不更新模型权重。模型参数为
在推理过程中固定。到目前为止，我们了解了变压器如何在
推理。接下来，我们关注如何训练 Transformer 执行一项或多项给定任务。

数据准备

第一步是数据准备，其中涉及几个重要步骤。首先，清洁
通过应用过滤、重复数据删除和标准化等技术来处理数据。下一个
步骤是标记化，其中使用以下技术将数据集转换为标记：
字节对编码和一元标记化。标记化生成词汇表，
这是 LLM 使用的一组唯一标记。该词汇表用作模型的
用于处理和理解文本的“语言”。最后，数据通常分为
用于训练模型的训练数据集以及用于评估模型的测试数据集
性能模型。

训练和损失函数

典型的 Transformer 训练循环由几个部分组成：首先，批量输入序列是从训练数据集中采样的。对于每个输入序列，有一个相应的目标序列。在无监督预训练中，目标序列是从输入序列本身派生。然后将这批输入序列送入变压器。变压器生成预测的输出序列。区别使用损失函数（通常是交叉函数）来测量预测序列和目标序列之间的关系熵损失）。计算该损失的梯度，优化器使用它们来更新变压器的参数。重复这个过程直到变压器收敛到一定水平的性能或直到它已经接受了预先指定数量的令牌的训练。

有不同的方法来制定变压器的训练任务，具体取决于关于所使用的架构：

- 仅解码器模型通常在语言建模任务上进行预训练（例如，参见尾注）。解码器的目标序列只是输入的移位版本顺序。给定一个训练序列，例如“猫坐在垫子上”，各种输入/目标可以为模型生成对。例如，输入“the cat sat on”应该预测“the”，随后输入“the cat sat on the”应该预测目标序列“垫”。

- 仅编码器模型（如 BERT）通常通过破坏输入序列来进行预训练以某种方式让模型尝试重建它。一种这样的方法被掩盖了语言建模 (MLM)。在我们的示例中，输入序列可以是“[MASK] sat on the mat”，序列目标将是原始句子。

- 编码器-解码器模型（如原始变压器）是在序列到序列上进行训练的序列监督任务，例如翻译（输入序列“Le chat est assis sur le tapis”和目标“The cat sat on the mat”），问答（其中输入序列是一个问题，目标序列是相应的答案），并且

摘要（其中输入序列是一篇完整的文章，目标序列是它的相应的总结）。这些模型也可以通过无监督的方式进行训练将其他任务转换为序列到序列格式。例如，训练时在维基百科数据上，输入序列可能是文章的第一部分，目标序列包括本文的其余部分。

训练期间要考虑的另一个因素是“上下文长度”。这是指模型可以“记住”并用于预测下一个标记的先前标记的数量顺序。更长的上下文长度允许模型捕获更复杂的关系以及文本中的依赖关系，可能会带来更好的性能。然而，更长上下文还需要更多的计算资源和内存，这可能会减慢速度训练和推理。选择适当的上下文长度涉及平衡这些根据具体任务和可用资源进行权衡。

变压器的演变

接下来的部分概述了各种变压器架构。这些包括仅编码器、编码器-解码器以及仅解码器变压器。我们从 GPT-1 和 BERT，最后是 Google 最新的 LLMs 系列（称为 Gemini）。

GPT-1

GPT-1（生成式预训练变压器版本 1）是开发的仅解码器模型由 OpenAI 于 2018 年开发。它在 BooksCorpus 数据集（大约包含数十亿字）并且能够生成文本、翻译语言、编写不同类型的文本创意内容，并以翔实的方式回答问题。主要创新点在于 GPT-1 是：

- 结合 Transformer 和无监督预训练：无监督预训练
  是在大量未标记数据的语料库上训练语言模型的过程。然后，
  监督数据用于针对特定任务（例如翻译）微调模型
  或情感分类。在之前的工作中，大多数语言模型都是使用
  监督学习目标。这意味着该模型是在以下数据集上进行训练的
  标记数据，其中每个示例都有相应的标签。这个方法主要有两个
  限制。首先，它需要大量的标记数据，这可能是昂贵的并且
  收集起来很费时间。其次，该模型只能泛化到相似的任务
  到它接受培训的任务。半监督序列学习是最早的序列学习之一
  研究表明，无监督的预训练和随后的监督训练是有效的
  优于单独的监督训练。

  无监督预训练通过在大型数据集上训练模型来解决这些限制
  未标记数据的语料库。收集这些数据比标记数据更容易、更便宜
  数据。此外，该模型可以推广到与现有任务不同的任务。
  它受到了训练。 BooksCorpus 数据集是一个大型 (5GB) 未标记文本语料库，
  用于训练 GPT-1 语言模型。该数据集包含 7,000 多个未发布的
  书籍，为模型提供了大量的数据可供学习。此外，
  语料库包含很长的连续文本，这有助于模型学习长文本
  范围依赖性。总的来说，无监督预训练是一种强大的技术，可以
  用于训练比模型更准确和更通用的语言模型
  仅使用监督学习进行训练。

- 任务感知输入转换：有不同类型的任务，例如文本任务
  蕴含和问答需要特定的结构。例如，
  文本蕴含需要前提和假设；回答问题需要一个
  上下文文档；一个问题和可能的答案。 GPT-1的贡献之一
  正在将需要结构化输入的这些类型的任务转换为
  语言模型可以解析，而不需要特定于任务的架构
  预训练的架构。对于文本蕴涵，前提 p 和假设 h 是

在 - [p, $, h] 之间连接一个分隔符标记 ($)。对于回答问题，
上下文文档 c 与问题 q 以及可能的答案 a 和 a 连接起来
问题和答案之间的分隔符 - [c,q,$,a]。

GPT-1在多个基准测试中超越了之前的模型，取得了优异的成绩。尽管
GPT-1 是自然语言处理（NLP）领域的重大突破，它有一些
限制。例如，该模型很容易生成重复的文本，尤其是当
给出的提示超出了其训练数据的范围。它也未能推理出多个
对话轮流，无法跟踪文本中的长期依赖关系。此外，其
衔接性和流畅性仅限于较短的文本序列，而较长的段落会
缺乏凝聚力。尽管存在这些限制，GPT-1 仍然展示了无监督的力量
预训练，为基于此的更大、更强大的模型奠定了基础
变压器架构。

BERT

BERT 代表 Transformers 的双向编码器表示，
与传统的编码器-解码器变压器模型的区别在于，
仅编码器架构。 BERT 不是翻译或生成序列，而是专注于
通过对掩码语言模型目标进行训练来深入理解上下文。在
在这个设置中，句子中的随机单词被替换为 [MASK] 标记，并且 BERT 尝试
根据周围的上下文预测原始单词。另一个创新的方面
BERT 训练机制的核心是下一个句子的预测损失，它学习确定
给定的句子是否在逻辑上遵循前一个句子。通过针对这些目标的培训，
BERT 从单词的左侧和右侧捕获复杂的上下文依赖关系，并且
它可以辨别句子对之间的关系。这些能力使得 BERT
特别擅长需要自然语言理解的任务，例如问题
回答、情感分析和自然语言推理等。由于这是一个
仅编码器模型，BERT 无法生成文本。

GPT-2

GPT-2是GPT-1的后继者，由OpenAI于2019年发布。主要创新点
GPT-2是直接放大，其参数数量和大小都增加了十倍
其训练数据集：

- 数据：GPT-2 在一个名为 WebText 的大型（40GB）且多样化的数据集上进行训练，该数据集包含来自 Reddit 的 4500 万个网页，Karma 评级至少为 3。业力是 Reddit 上使用的评分指标，值为 3 意味着所有帖子都是合理的质量水平。

- 参数：GPT-2 有 15 亿个参数，数量级大了比以前的型号。更多参数可以提高模型的学习能力。这作者训练了四种语言模型，分别为 117M（与 GPT-1 相同）、345M、762M 和 1.5B (GPT-2)参数，发现参数最多的模型表现更好关于每一个后续任务。

这种扩展产生了一个能够生成更加连贯和真实的文本的模型
比GPT-1。它产生类似人类反应的能力使其成为各种有价值的工具
自然语言处理任务，例如内容创建和翻译。具体来说，
GPT-2 在捕获远程依赖性和
常识推理。虽然它在某些任务中表现良好，但并没有优于状态 -
最先进的阅读理解、总结和翻译。 GPT-2最重要的
成就在于它能够对各种任务进行零样本学习。零射击任务
迁移是模型在未经训练的情况下泛化到新任务的能力，这
要求模型根据给定的指令理解任务。例如，对于
英语到德语的翻译任务，模型可能会给出一个英语句子
通过单词“德语”和提示符（“:”）。然后模型将被期望理解
这是一个翻译任务并生成英语句子的德语翻译。
GPT-2 能够执行机器翻译、文本摘要等任务
没有任何明确监督的阅读理解。

研究发现，零样本任务的性能以对数线性方式提高
随着模型容量的增加。 GPT-2 表明，在更大的数据集上进行训练并具有
更多参数提高了模型理解任务并超越状态的能力
零样本设置中许多任务的艺术。

GPT-3/3.5/4

GPT-3，或生成预训练 Transformer 模型的第三次迭代，代表
与其前身 GPT-2 相比，主要在规模、功能和
灵活性。最显着的区别是 GPT-3 的庞大规模，拥有巨大的
1750 亿个参数，而 GPT-2 最大的模型有 15 亿个参数。
模型大小的增加使得 GPT-3 能够存储和调用更大量的数据
信息，理解细致入微的指令，并生成更加连贯和符合上下文的内容
较长段落的相关文本。

虽然 GPT-2 可以通过额外的训练数据对特定任务进行微调，但 GPT-3 可以
只需几个示例，有时甚至不需要任何示例，即可理解并执行任务
明确的示例——仅基于所提供的说明。这凸显了 GPT-3 的更多功能
动态理解和适应能力，减少对特定任务的精细化需求
调整在 GPT-2 中更为普遍。

最后，GPT-3的大模型规模和多样化的训练语料库带来了更好的结果
泛化到更广泛的任务。这意味着开箱即用，无需
通过任何进一步的训练，GPT-3 在各种 NLP 挑战中表现出更好的性能，从
与 GPT-2 相比，翻译为问答。还值得注意的是，此次发布
方法不同：虽然 OpenAI 最初由于担心滥用而推迟了 GPT-2，
他们选择将 GPT-3 作为商业 API 提供，反映了它的实用性和
组织对部署的不断变化的立场。

然后通过 InstructGPT 引入了指令调整，这是 GPT-3 的一个版本，非常好——
使用监督微调，对人类演示所需的数据集进行调整
模范行为。然后对该模型的输出进行排序，然后进一步细化-
使用人类反馈的强化学习进行调整。这导致了教学的改进
以下在模型中。 1.3B 参数 InstructGPT 模型具有更好的人类评估
比175B参数GPT-3模型。它还显示出在真实性和
减少毒性。

GPT-3.5 型号（包括 GPT-3.5 Turbo）比 GPT-3 有所改进，因为它能够
理解并生成代码。它针对对话进行了优化。并且它能够
接收最多 16,385 个令牌的上下文窗口，并可生成最多 4,096 个输出
代币。

GPT-4 将 GPT-3.5 扩展为能够处理图像和图像的大型多模态模型
文本输入并生成文本输出。具体来说，接受文本或图像作为输入
并输出文本。该模型具有更广泛的常识和高级推理
能力。它可以接收最多 128,000 个令牌的上下文窗口，并且具有最大
输出 4,096 个代币。 GPT-4 通过解决复杂任务展现出卓越的多功能性
跨越数学、编码、视觉、医学、法律和心理学等不同领域
无需专门说明。它的性能往往与人类相当甚至超过
功能并且显着优于 GPT-3.5 等早期模型。

这个MDA

谷歌的 LaMDA，代表“对话应用程序的语言模型”是另一个
对大规模语言模型领域的贡献，主要旨在参与
开放式对话。与运行环境更加受限的传统聊天机器人不同
和预定义域，LaMDA 旨在处理广泛的主题，提供

更自然、流畅的对话。 LaMDA 接受了以对话为中心的数据培训，
鼓励持续的对话流，而不仅仅是孤立的响应，确保用户可以
更广泛和探索性的对话。

虽然 GPT 模型，尤其是像 GPT-3 这样的后来的迭代，一直致力于解决
同时执行多项任务，从文本生成到代码编写，LaMDA 的主要任务
重点是保持和增强对话的深度和广度。 GPT模型
展现他们制作连贯的长篇内容和执行各种任务的能力
需要最少的提示，而 LaMDA 则强调对话的流程和进展，
努力模仿人类对话的不可预测性和丰富性。

地鼠

Gopheris 是一个基于仅解码器变压器的 2800 亿参数语言模型
架构，由 DeepMind 于 2021 年开发。它可以生成文本、翻译语言、
撰写不同类型的创意内容，并以翔实的方式回答您的问题。
与 GPT-3 类似，Gopher 专注于提高数据集质量和优化技术：

- 数据集：研究人员策划了一个名为 MassiveText 的高质量文本数据集，该数据集
  包含超过 10 TB 的数据和来自网页、书籍、新闻的 2.45B 文档
  文章和代码 (GitHub)。他们只训练了 300B 个 token，占数据集的 12%。
  重要的是，他们通过过滤（例如删除
  重复文本并删除重复的类似文档。这显着改善了
  模型在下游任务上的表现。

- 优化：研究人员使用 1,500 步的预热学习率，然后
  使用余弦时间表对其进行衰减。他们还有一个有趣的规则，即当他们
  增加了模型大小，降低了学习率并增加了数量
  每批中的令牌。此外，他们发现裁剪梯度最大为 1
  基于全局梯度范数有助于稳定训练。

Gopher 接受了各种任务的评估，包括数学、常识、逻辑
推理、常识、科学理解、道德和阅读理解。
Gopher 在 81% 的任务中表现优于之前最先进的模型。具体来说，
Gopher 在知识密集型任务上表现良好，但在推理繁重的任务上表现不佳
抽象代数等任务。

作者还研究了模型大小对不同类型的影响
任务。图 4 显示了该消融研究的结果。具体来说，作者发现
增加参数数量对逻辑推理和
阅读理解，但它并没有提高诸如以下任务的表现
一般知识，性能最终几乎趋于稳定。

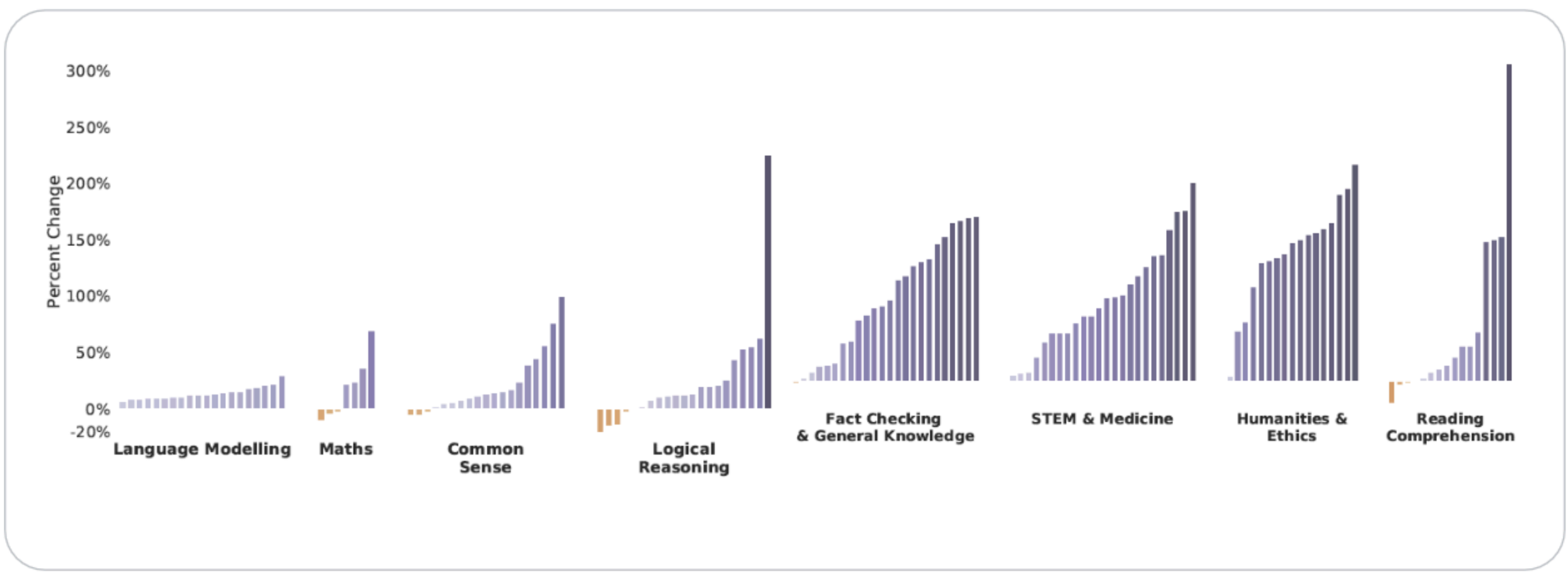


图 4. 模型大小对 Gopher 不同类型任务性能影响的消融研究

GLaM

GLaM（通才语言模型）是第一个稀疏激活的专家混合模型
语言模型。基于专家混合的模型计算效率更高
给定他们的参数数量。这是通过仅激活其参数的子集来实现的

（即专家）每个输入令牌。 GLaM 由 1.2 万亿个参数组成，但仅使用 ⅓
用于训练 GPT-3 的能量和一半用于推理的 FLOP，同时实现更好的效果
与 GPT-3 相比的整体性能。

龙猫

到 2022 年，LLMs 主要通过增加模型大小和使用以下数据集进行扩展：
按照当前标准，相对较小（最大模型最多 3000 亿个代币）。
这种方法是由 Kaplan 等人的研究提供的，该研究检查了性能如何
通过交叉熵损失来衡量的语言模型的性能随计算量的变化而变化
预算、模型大小和数据集大小。具体来说，考虑到计算量增加了 100 倍
资源 (C)，Kaplan 等人建议将模型大小缩放约 28.8 倍
(N∝C)，同时数据集大小仅增加 3.5 倍 (D∝C)。

Chinchilla 论文重新审视了计算最优缩放法则并使用了三种不同的
方法发现随着参数和数据的增加，接近相等的缩放是最佳的
计算。因此，计算量增加 100 倍应该转化为计算量增加 10 倍
数据大小和模型大小。

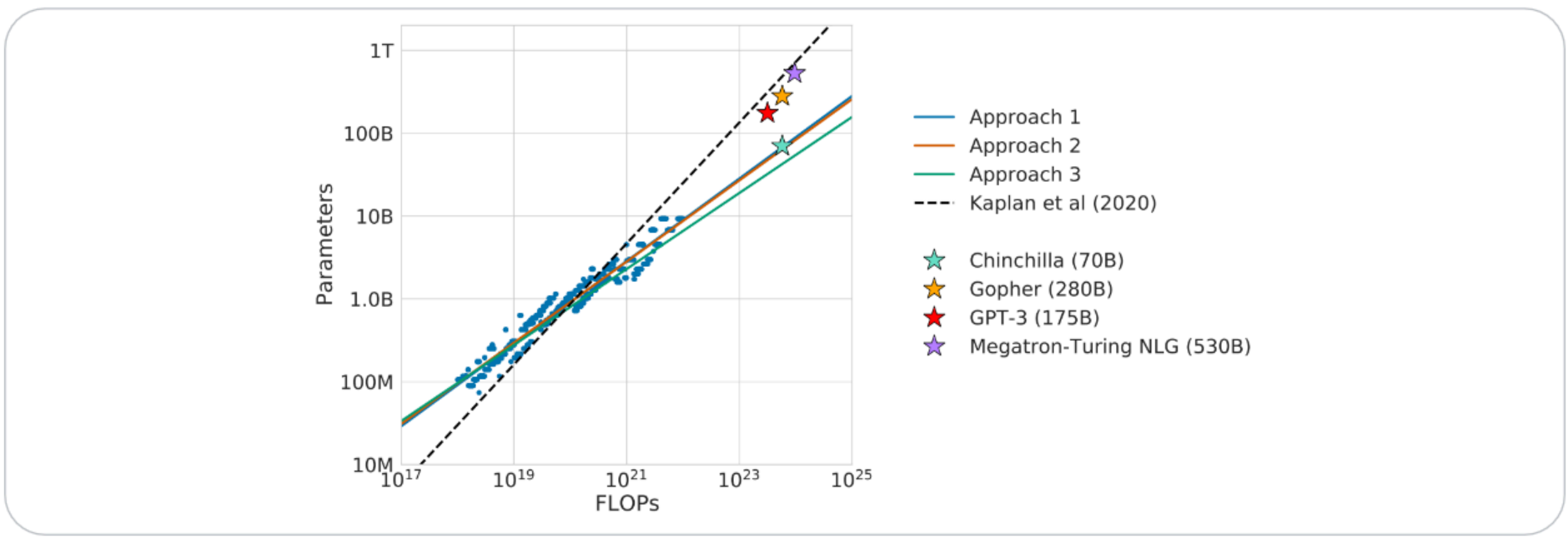


图 5. Chinchilla 论文中三种不同方法的叠加预测以及 Kaplan 等人的预测

为了验证更新后的缩放定律，DeepMind 训练了一个 70B 参数模型（称为 Chinchilla）使用与之前训练的 Gopher 模型相同的计算预算。Chinchilla 均匀且显着优于 Gopher (280B)、GPT-3 (175B) 和 Megatron-Turing NLG (530B) 执行大量下游评估任务。由于是比 Gopher 小 4 倍，Chinchilla 的内存占用和推理成本均为也更小。

龙猫的发现对未来LLMs的发展产生了重大影响。
重点转向寻找扩展数据集大小（同时保持质量）的方法
增加参数数量。推断这一趋势表明训练数据集大小
可能很快就会受到可用文本数据量的限制。这导致了新的研究
Muennighoff 等人探索数据约束条件下的缩放定律。

PaLM

Pathways语言模型（PaLM）是一个基于5400亿个参数的大型变换器
谷歌人工智能开发的语言模型。它接受了海量文本数据集的训练
代码并能够执行广泛的任务，包括常识推理，
算术推理、笑话解释、代码生成和翻译。

在发布时，PaLM 还能够在
许多语言基准测试，例如 GLUE 和 SuperGLUE。

PaLM 的关键特性之一是其高效扩展的能力。这要归功于 Pathways系统，Google开发的用于分布式大语言的训练跨两个 TPU v4 Pod 的模型。

帕LM 2

PaLM 2 是 2023 年 5 月发布的 PaLM 的继任者。感谢许多人
架构和培训增强，PaLM 2 比 PaLM 更强大，
总参数更少。它擅长高级推理任务，包括代码生成、
数学、分类、问答和翻译。

PaLM 2 也被证明比 PaLM 更高效，并成为
Google 作为 Google Cloud Generative AI 的一部分发布的商业模型数量。

双子座

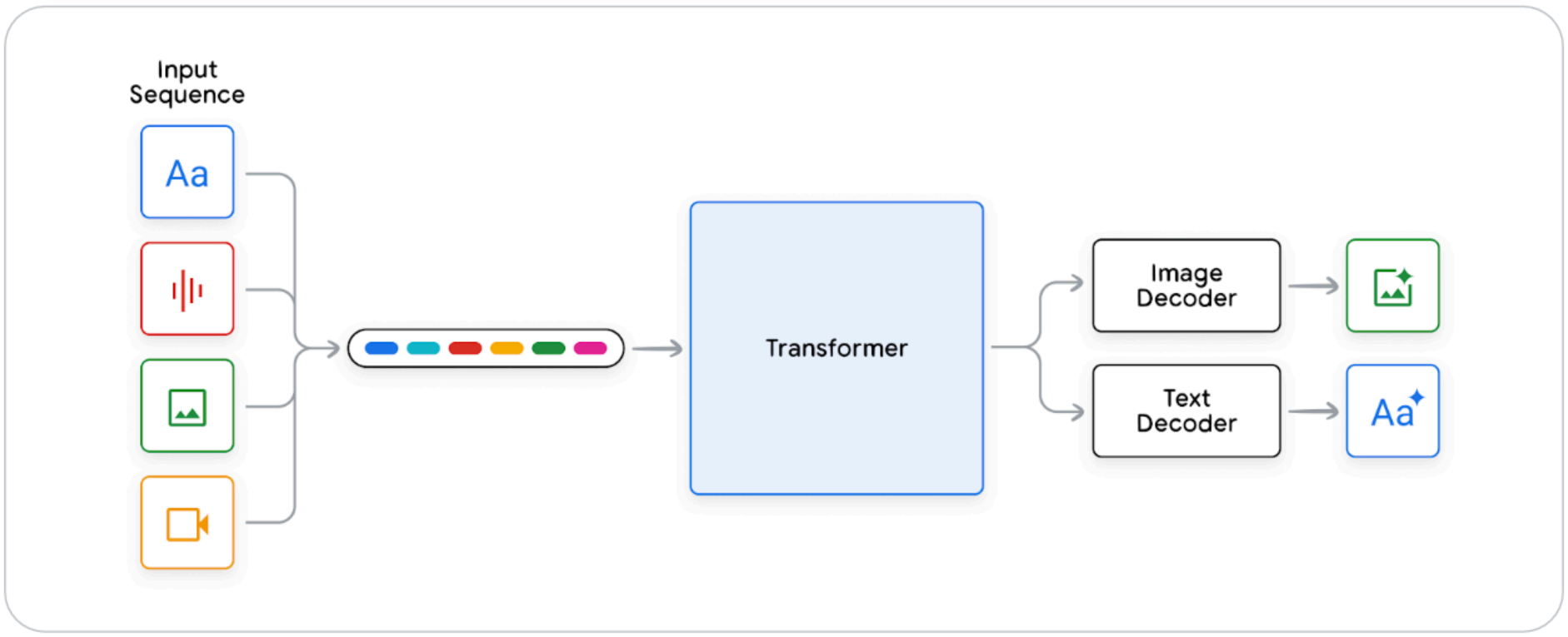


图 6.Gemini 可以接收多模式输入，包括文本、音频、图像和视频数据。这些都是
标记化并输入到其变压器模型中。转换器生成可以包含图像的输出
和文字

Gemini（图 6）是最先进的多模态语言模型家族，可以
将文本、图像、音频和视频的交错序列作为输入。它建立在
变压器解码器并具有规模和优化的架构改进
对 Google 张量处理单元 (TPU) 的推理。在当前的 1.5 版本中，这些模型
经过训练可支持不同大小的上下文，Gemini 1.5 Pro 中最多支持 2M 令牌
Vertex AI 上的版本并采用多查询注意力等机制来提高效率。
Gemini 模型还采用混合专家架构来优化效率和
模型的能力。多模态允许模型处理文本、图像和视频
投入方面，预计未来投入和产出方式将更加丰富。

Gemini 模型在 Google 的 TPUv5e 和 TPUv4 处理器上进行训练，具体取决于
尺寸和配置。预训练数据由网络文档、书籍、代码等组成
图像、音频和视频数据。

使用相同的方法训练更大的模型以计算最佳数量的令牌
Chinchilla 论文中的方法，而小模型则接受更多标记的训练
而不是计算最优值以提高给定推理预算的性能。

Gemini 系列型号针对不同尺寸进行了优化：Gemini Ultra、Gemini Pro、Gemini
纳米和闪存。 Gemini Ultra 用于高度复杂的任务并实现了最先进的
art 在 32 项基准任务中取得了 30 项成绩。 Gemini Pro 支持大规模部署
Gemini Nano 专为设备上应用而设计。 Gemini Nano 型号利用
蒸馏等先进技术可为小型设备提供最先进的性能
总结和阅读理解等任务的语言模型。作为双子座
模型本质上是多模态的，可以看出跨多种模态的训练确实
确实导致了一个能够在每个领域实现强大功能的模型。

2024年初，谷歌推出了Gemini家族的最新型号，
Gemini 1.5 Pro，一种计算效率高的多模式专家混合模型。这
该模型还将上下文窗口的大小显着增加到数百万个标记
并且能够对数百万个令牌进行回忆和推理，包括多个
长文档和数小时的视频和音频。 Gemini 1.5 Pro表现出色
跨不同领域的能力：

- 代码理解：它可以处理大量代码库并回答高度具体的问题
  代码相关问题。
- 语言学习：模型可以学习训练时从未观察到的新语言
  仅基于其输入中提供的参考材料
- 多模态推理：它理解图像和文本，使其能够定位著名场景
  摘自根据简单草图改编的小说《悲惨世界》。
- 视频理解：可以分析整部电影，回答细节问题
  以极高的准确性精确定位特定的时间戳。

Google 的 Gemini 1.5 Pro 模型擅长检索很长的信息
文件。在他们的研究中，它证明了对多达 530,000 个文档的 100% 召回率
代币，对 100 万个代币的召回率超过 99.7%。令人印象深刻的是，它保持了99.2%
在多达 1000 万个标记的文档中查找信息时的准确性。

此外，Gemini 1.5 Pro 在LLMs遵循复杂性方面实现了重大飞跃
指示。在406个多步骤提示的严格测试中，它的表现明显优于
之前的 Gemini 型号。该模型准确地遵循了近 90% 的指令，并且完全
完成了66%的复杂任务。

Gemini Flash 是 Gemini 型号家族的新成员，也是最快的 Gemini 型号
在 API 中提供服务。它针对大规模、高频率的任务进行了优化，
服务成本高效，并具有突破性的 100 万个代币长上下文窗口。
虽然是比 1.5 Pro 更轻的重量模型，但多模态推理能力很强
涵盖大量信息，并以其规模提供令人印象深刻的质量。

此外，最近先进的 Gemma 是一个轻量级、最先进的开放式产品系列
采用与创建 Gemini 模型相同的研究和技术构建的模型。
Gemma 的第一个模型拥有 256,000 个单词的大词汇量，并接受过训练
庞大的 6 万亿代币数据集。这使得它成为公开可用的有价值的补充
LLM集合。此外，2B 参数版本很有趣，因为它可以在
单个 GPU。

Gemma 2由Google AI开发，代表了人工智能领域的重大进步
打开大型语言模型。设计注重效率，270 亿个参数
该型号的性能可与 Llama 3 70Bon 标准等更大的型号相媲美
基准。这使得 Gemma 2 成为适用于各种 AI 的强大且易于使用的工具
开发商。它与基于云的解决方案的各种调整工具链兼容
到流行的社区工具，进一步增强了其多功能性。凭借其强劲的性能，
高效的建筑和可亲近的自然，Gemma 2 在推动创新方面发挥着至关重要的作用
并使人工智能能力民主化。

其他开放型号

开放LLMs的格局正在迅速发展，越来越多的模型
代码和预训练权重都是可公开访问的。下面我们重点介绍一些
已知的例子：

- LLaMA 2：LLaMA 2 由 Meta AI 发布，是一系列经过预训练和微调的 LLMs 参数范围从 7B 到 70B。它显示出显着的改进 前身 LLaMA 1，包括大 40% 的预训练数据集（2 万亿代币），加倍上下文长度（4096 个标记），以及使用分组查询注意力。这 微调版本 LLaMA 2-Chat 针对对话进行了优化并显示出竞争力 与相同规模的闭源模型相比的性能。

- LLaMA 3.2：LLaMA 3.2 由 Meta AI 发布，是其开放式 LLMs 的下一代。 Llama 3.2 包括多语言纯文本模型 (1B, 3B) 和视觉 LLMs (11B, 90B)，1B 和 3B 的量化版本尺寸平均缩小 56%，尺寸缩小 2-3 倍 加速，非常适合设备上和边缘部署。 LLaMA 3.2 利用分组查询 注意力和 128K 令牌词汇表可提高性能和效率。

- Mixtral：Mistral AI 开发的 Mixtral 8x7B 是稀疏专家混合 (SMoE) 模型。虽然其总参数数为 47B，但每个参数仅使用 13B 个活动参数。 推理过程中使用令牌，从而加快推理速度并提高吞吐量。这个型号 擅长数学、代码生成和多语言任务，通常表现出色 LLaMA 2 70B 在这些领域。 Mixtral 还支持 32k 令牌上下文长度，从而能够 它可以处理更长的序列。其指令调整版本 Mixtral 8x7B-Instruct，在人类评估基准上超越了多个闭源模型。

- Qwen 1.5：阿里巴巴的LLM系列有六种尺寸：0.5B、1.8B、4B、7B、14B和 72B。 Qwen 1.5 模型统一支持高达 32k token 的上下文长度并显示 在各种基准测试中表现强劲。值得注意的是，Qwen 1.5-72B 的表现优于 LLaMA2-70B 在所有评估基准上均表现出卓越的能力 语言理解、推理和数学。

- Yi：由01.AI创建，Yi模型系列包括预训练的6B和34B基础模型 基于 3.1 万亿个 token 的海量英文和中文数据集。它强调数据质量 经过严格的清洁和过滤过程。 34B型号实现性能

在许多基准上与 GPT-3.5 相当，并且可以有效地服务于消费者
具有 4 位量化的 GPU 级。 Yi 还提供了诸如 200k 上下文模型、

视觉语言模型（Yi-VL）和深度升级的 9B 模型。

- Grok-1：由 xAI 开发，Grok-1 是一个 314B 参数 Mixture-of-Experts 模型，
  给定代币上活跃权重的 25%。它是来自的原始基础模型检查点
  预训练阶段，并且没有针对对话等特定任务进行微调。 Grok-1 运行
  上下文长度为 8k 个标记。

LLMs 的创新步伐很快，并且没有放缓的迹象。那里
在学术和商业领域对该领域做出了许多贡献。
arxiv.org 上发表了超过 20,000 篇关于 LLMs 的论文，无法一一列举
为 LLMs 的发展做出贡献的模型和团队。然而，一个
感兴趣的开放模型的缩写列表可能包括 EleutherAI 的 GPT-NeoX 和 GPT-J，
斯坦福大学的 Alpaca、LMSYS 的 Vicuna、xAI 的 Grok、TII 的 Falcon、微软的 PHI、
NVLM 来自 Nvidia、DBRX 来自 Databricks、Qwen 来自阿里巴巴、Yi 来自 01.ai、Llama 来自
上面提到的元以及许多其他内容。一些著名的商业开发公司
基础 LLM 模型包括 Anthropic、Cohere、Character.ai、Reka、AI21、Perplexity、xAI
除了前面几节提到的 Google 和 OpenAI 之外，还有许多其他公司。这是
使用模型时确认许可证适合您的用例很重要，因为
许多型号都提供了非常具体的使用条款。

比较

在本节中，我们观察了基于 Transformer 的语言模型是如何演变的。他们
最初是经过数亿个参数训练的编码器-解码器架构
数以亿计的代币，并已发展成为大规模的仅解码器架构
具有数十亿个参数并在数万亿个代币上进行训练。表 1 显示了如何
本白皮书中讨论的所有模型的重要超参数都已经演变

随着时间的推移。数据和参数的缩放不仅提高了性能
LLMs 在下游任务上，但也导致了紧急行为和零或很少
对新任务进行概括。然而，即使是最好的LLMs仍然有很多
限制。例如，他们不擅长进行像人类一样的对话，他们的
数学技能是有限的，并且它们可能不符合人类道德（例如，它们可能是
偏见或产生毒性反应）。在下一节中，我们将了解很多此类问题
正在解决。

	模型						
	注意力 (2017)	GPT (2018)	GPT-2 (2019)	GPT-3 (2020)	这个MDA (2021)	地鼠 (2021)	龙猫 (2022)
优化器	ADAM	ADAM	ADAM	ADAM	ADAM	ADAM	ADAM-W
# 参数	213M	117M	1.5B	175B	137B	280B	70B
词汇量	~37K	~40K	~50K	~50K	~32K	~32K	~32K
嵌入方面	1024	768	1600	12288	8192	16384	8192
关键尺寸	64	64	64	128	128	128	128
# 个头 (H)	16	12	25	96	128	128	64
# 编码器层数	6	N/A	N/A	N/A	N/A	N/A	N/A
# 解码器层数	6	12	48	96	64	80	80
前馈方面	4 * 1024	4 * 768	4 * 1600	4 * 12288	8 * 8192	4 * 16384	4 * 8192
上下文令牌 Size	N/A	512	1024	2048	N/A	2048	2048
预训练代币	∼160M	∼1.25B	~10B	∼300B	∼168B	∼300B	∼1.4T

表 1. 基于 Transformer 的大型语言模型的重要超参数

A. 该数字是根据报告的数据集大小估算的。

微调大型语言模型

大型语言模型通常经历多个训练阶段。第一阶段，往往
称为预训练，是对 LLM 进行大规模训练的基础阶段，
多样化且未标记的文本数据集，其任务是在给定的情况下预测下一个标记
之前的上下文。此阶段的目标是利用大量、普遍分布的数据
并创建一个擅长从这个一般分布中采样的模型。语言之后
模型预训练，生成的 LLM 通常表现出合理的语言水平
跨各种不同任务的理解和语言生成技能
通常通过零样本或少样本提示（增强指令）进行测试
有一些例子/演示）。就时间而言，预训练是最昂贵的
（从几周到几个月取决于模型的大小）和所需的数量
计算资源（GPU/TPU 小时）。

训练后，模型可以通过微调进一步专业化，通常称为
指令调整或简单的监督微调（SFT）。 SFT 涉及在 a 上训练 LLM
一组特定于任务的演示数据集，其性能也通过
一组特定于领域的任务。以下是一些可以采取的行为示例
使用微调改进：

- 指令调整/指令跟随：LLM 作为输入指令提供
  接下来可能包括总结一段文字、编写一段代码或编写
  具有某种风格的一首诗。
- 对话调整：这是指令调整的一种特殊情况，其中 LLM 被微调
  以问题和答复的形式处理对话数据。这通常被称为
  多轮对话。

- 安全调整：这对于减轻与偏见、歧视和歧视相关的风险至关重要。
  有毒的输出。它涉及多管齐下的方法，包括仔细的数据选择，
  人机交互验证，并纳入安全护栏。技术如
  具有人类反馈的强化学习 (RLHF) 使 LLM 能够优先考虑安全
  和道德回应。

与预训练相比，微调的成本要低得多，数据效率更高。
有许多技术可以进一步优化成本，稍后将对此进行讨论
本白皮书。

监督微调

如上一节所述，SFT 是提高 LLM 性能的过程
通过在特定领域的标记数据上进一步训练它来完成特定任务或一组任务。这
数据集通常比预训练数据集小得多，并且通常是人类的
精心策划且高品质。

在此设置中，每个数据点由输入（提示）和演示（目标
回复）。例如，问题（提示）和答案（目标响应）、翻译自
一种语言（提示）到另一种语言（目标响应），一份总结文件
（提示），以及相应的摘要（目标响应）。

值得注意的是，虽然微调可以用来提高性能
如上所述的特定任务，它也可以达到帮助LLM的目的
改进其行为，使其更安全、毒性更小、更具对话性、更善于跟随
指示。

从人类反馈中强化学习

通常，执行 SFT 后，会发生第二阶段的微调，称为
来自人类反馈的强化学习（RLHF）。这是一个非常强大的微调
使 LLM 能够更好地与人类偏好的响应保持一致的技术（即，使
它的反应更有帮助、更真实、更安全等）。

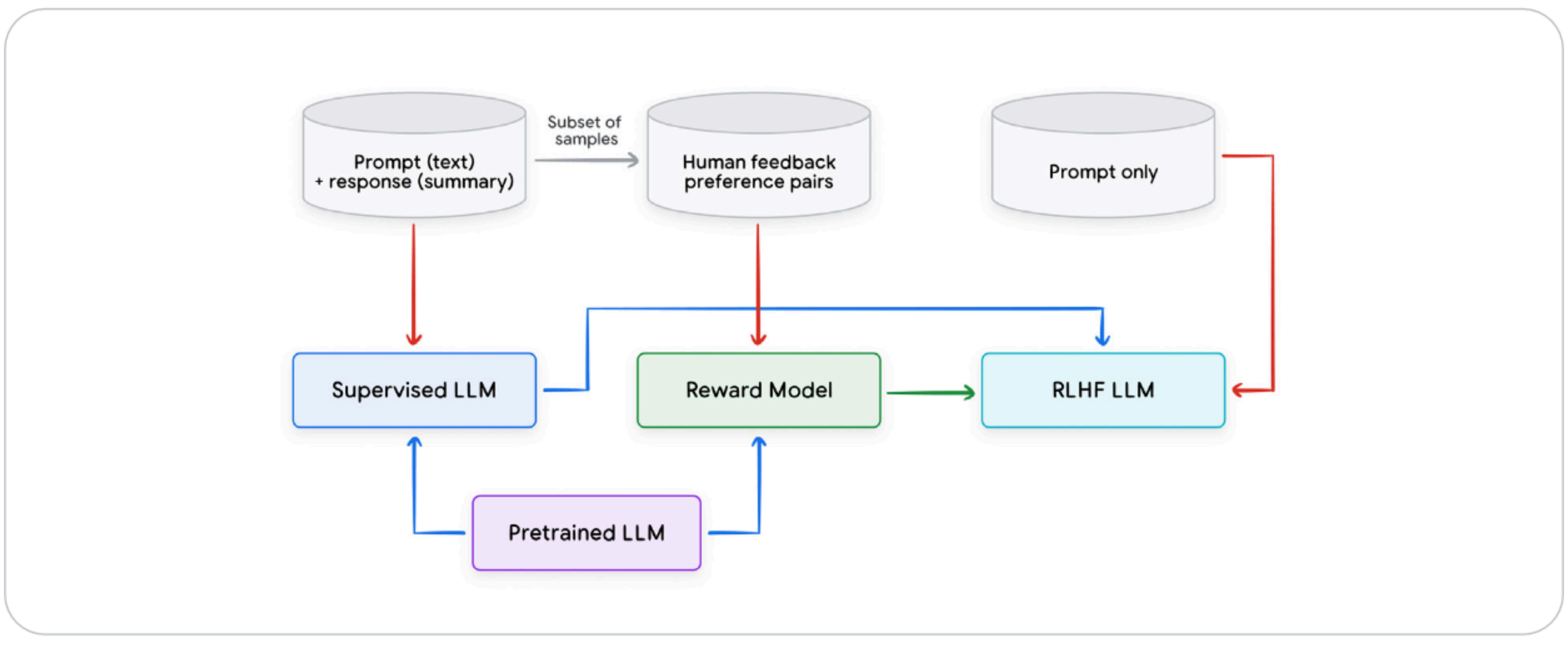


图 7. RLHF 过程示例

与 SFT 相反，SFT 中 LLM 仅暴露于正面示例（例如高质量的
演示数据），RLHF 还可以利用负输出，从而
当 LLM 生成表现出不良属性的响应时对其进行惩罚。惩罚
负输出使其不太可能产生无益或不安全的反应。

为了利用 RLHF，奖励模型 (RM) 通常需要使用类似的过程进行训练
如图 7 所示。RM 通常使用预训练的 Transformer 模型进行初始化，通常也使用
一种是SFT。然后它根据人类偏好数据进行调整，这些数据要么是单面的（带有一个
提示、响应和分数）或由一个提示和一对响应以及

偏好标签，指示首选两个响应中的哪一个。例如，给定
同一篇文章的两个摘要 A 和 B，人工评分者选择首选摘要
（依赖于详细指导）。我们将提供的偏好标签称为人类
反馈。偏好可以采用李克特量表的二元形式（例如“好”或“坏”），
当评估 2 名以上候选人时的排名顺序，或更详细的评估
总结质量。偏好信号还可以包含许多捕获的维度
定义高质量响应的各个方面，例如安全性、帮助性、公平性和
真实性。

图 7 显示了一个典型的 RLHF 管道，其中奖励模型在
偏好对。 RM 经过训练后，就会被强化学习使用
(RL)策略梯度算法，进一步微调先前指令调整的LLM
生成更符合人类偏好的响应。

为了更好地扩展 RLHF，人工智能反馈强化学习 (RLAIF) 利用人工智能反馈而不是人类反馈
反馈以生成偏好标签。也可以消除培训的需要
RLHF 通过利用直接偏好优化 (DPO) 等方法。两者 RLHF
RLAIF 可以在 Google Cloud 上使用。

参数高效微调

就计算时间和所需的加速器而言，SFT 和 RLHF 仍然非常昂贵，
特别是当对整个 LLMs 进行数十亿个参数的量级微调时。幸运的是，
有一些真正有用且有效的技术可以显着地进行微调
与预训练和全面微调相比，更便宜、更快。有这样一个家庭
方法是参数高效微调（PEFT）技术。

在高层次上，PEFT 方法附加了一组明显较小的权重（例如，在
数千个参数的顺序），用于“扰乱”预训练的 LLM 权重。
扰动具有微调LLM以执行新任务或一组任务的效果。
与传统方法相比，这具有训练明显更小的权重集的好处
对整个模型进行微调。

一些常见的 PEFT 技术包括适配器、低阶自适应和
软提示：

- 基于适配器的微调采用称为适配器的小模块来预调整
  训练有素的模型。仅训练适配器参数，从而显着减少
  参数优于传统的SFT。

- 低秩适应 (LoRA) 以不同的方式解决效率问题。它使用两个较小的矩阵
  近似原始权重矩阵更新，而不是微调整个LLM。
  该技术冻结原始权重并训练这些更新矩阵，显着
  以最小的额外推理延迟减少资源需求。此外，
  LoRA 改进了 QLoRA 等变体，它使用量化权重
  更高的效率。 LoRA 模块的一个很好的优点是它们可以即插即用，
  这意味着您可以训练专门从事一项任务的 LoRA 模块，并轻松地将其替换为
  另一个 LoRA 模块接受不同任务的训练。它还使传输变得更加容易
  模型，因为假设接收器具有原始矩阵，只需要更新矩阵
  待提供。

- 软提示是一种调节冻结大语言模型的技术
  可学习的向量而不是手工制作的文本提示。这些向量称为软向量
  提示，针对训练数据进行了优化，并且可以少至五个标记，从而使它们
  参数高效并支持混合任务推理。

对于大多数任务，完全微调仍然是性能最高的，其次是 LoRA 和 Soft
提示，但当涉及到成本时，顺序相反。这三种方法都比较
内存效率高于传统的微调并达到可比的性能。

Python

```
# 在开始之前运行此命令： # pip install --upgrade --user --quiet google-
cloud-aiplatform # 运行 pip install 后确保重新启动内核


从 vertexai.generative_models 导入 vertexai 从
vertexai.preview.tuning 导入 GenerativeModel 导入 sft


# TODO : 根据您的要求设置值 # 项目和存储常量 PROJECT_ID
= '' REGION = ''


vertexai.init(项目=PROJECT_ID, 位置=REGION)

# 定义训练和评估数据集。
TRAINING_DATASET = 'gs://cloud-samples-data/vertex-ai/model-evaluation/
peft_train_sample.jsonl' # 设置基础模型并指定调整模型的名称 BASE_MODEL = 'gemini-1.5-
pro-002' TUNED_MODEL_DISPLAY_NAME = '双子座微调-v1'


# 启动微调作业 sft_tuning_job =
sft.train( source_model=BASE_MODEL,
train_dataset=TRAINING_DATASET,

# # 可选:
tuned_model_display_name=TUNED_MODEL_DISPLAY_NAME, )


# 获取调优作业信息。
sft_tuning_job.to_dict()

# 调整后的模型端点名称tuned_model_endpoint_name =
sft_tuning_job.tuned_model_endpoint_name
# 使用调整后的模型tuned_genai_model =
GenerativeModel(tuned_model_endpoint_name)
print(tuned_genai_model.generate_content(contents='什么是LLM?'))
```

片段 1. Google 云上的 SFT 微调

使用大型语言模型

及时的工程和采样技术对性能有很大影响
LLMs。提示工程是设计和完善文本输入（提示）的过程
您将其输入 LLM 以实现所需的相关输出。抽样技术
确定选择输出标记的方式并影响正确性，
最终产出的创造力和多样性。接下来我们讨论提示的不同变体
工程和采样技术以及一些重要参数
会对 LLM 性能产生重大影响。

及时工程

LLMs非常强大，但他们仍然需要指导才能充分发挥潜力。迅速的
工程是指导 LLM 产生所需输出的关键组成部分。这可能
包括为模型奠定基础以产生事实反应或释放人们的创造力
讲故事或写歌的模型。即时工程的示例包括提供
对 LLM 进行清晰的说明，举例说明，使用关键字并格式化以强调
重要信息，提供额外的背景详细信息等。

您经常会在文章中听到“零样本”、“少样本”和“思维链”等术语。
即时工程的背景。我们对这些术语的定义如下：

- 少量提示：这是您向 LLM 提供任务描述的情况
  作为一些（例如三到五个）精心挑选的示例，这将有助于指导 LLM
  回复。例如，您可以为模型提供一些国家/地区的名称
  及其首都城市，然后要求它为不在其中的新国家生成首都
  的例子。

- 零次提示：这是当您直接向 LLM 提供提示时
  指示。您通常会为 LLM 提供任务描述，而 LLM 很大程度上依赖于其
  现有知识输出正确的响应。这不需要额外的数据或
  例子，因此被称为“零样本”，但可能不如少样本提示可靠。

- 思想链提示：该技术旨在提高复杂的性能
  推理任务。您不是简单地向 LLM 询问问题，而是提供提示
  演示如何使用逐步推理来解决类似问题。这
  LLM 然后针对新问题生成自己的思路链，将其分解为
  较小的步骤并解释其推理。最后，它根据自己的情况给出了答案
  推理过程。

快速工程是一个活跃的研究领域。

采样技术和参数

可以采用多种采样技术来确定模型如何选择
序列中的下一个标记。它们对于控制质量、创造力和
LLM 输出的多样性。以下是不同采样技术的细分
及其重要参数：

- 贪心搜索：每一步选择概率最高的标记。这是
  最简单的选择，但它可能会导致重复且可预测的输出。

- 随机采样：根据概率分布选择下一个token，其中
  每个标记都按照其预测概率按比例进行采样。这样可以产生更多
  令人惊讶且富有创意的文本，但也更有可能产生无意义的输出。

- 温度采样：通过温度参数调整概率分布。
  较高的温度促进多样性，较低的温度有利于高概率的代币。

- Top-K 采样：从前 K 个最可能的标记中随机采样。 K值
  控制随机性程度。
- Top-P 采样（核心采样）：来自令牌动态子集的样本，其
  累积概率总计为 P。这使得模型能够调整潜在的数量
  候选人取决于其信心，在不确定和不确定时倾向于更多多样性
  自信时专注于较小的一组高概率单词。
- Best-of-N 采样：生成 N 个单独的响应并选择被认为最好的一个
  根据预定的指标（例如，奖励模型或逻辑一致性检查）。
  这对于简短的片段或逻辑和推理的情况特别有用
  是关键。

通过将即时工程与采样技术相结合并正确校准
超参数，您可以极大地影响 LLM 的响应，使其更加相关，
富有创意，并且能够满足您的特定需求。

到目前为止，我们已经了解了各种类型的LLM架构，它们的底层技术，
以及用于训练、调整和调整这些模型以适应各种任务的方法。让我们
现在看一些关于如何加快 LLMs 中的解码过程的关键研究
显着地产生更快的响应。

加速推理

Kaplan 等人最初探索的 LLMs 的缩放定律仍在继续。
保持今天。语言模型的规模一直在不断增加，这已经
这些模型的质量和准确性大幅提高的直接贡献者
过去几年。由于参数数量的增加提高了 LLMs 的质量

还增加了运行它们所需的计算资源。多种方法
作为开发人员，已用于尝试提高 LLMs 不同任务的效率
被激励降低模型用户的成本和延迟。平衡费用
在时间、金钱、精力方面服务模型被称为成本性能权衡
并且经常需要针对特定用例进行调整。

LLMs 使用的两个主要资源是内存和计算。技术
提高推理效率或速度主要关注这些资源。这
内存和计算之间的连接速度也很关键，但通常是硬件
受到限制。随着 LLMs 的大小从数百万个参数增长到数十亿个参数，增长了 1000 倍。
附加参数会增加保存模型所需的内存大小和
产生模型结果所需的计算。

随着 LLMs 越来越多地应用于大规模和低延迟用例，发现
优化推理性能的方法已成为优先事项和积极的研究
具有重大进展的主题。我们将探索多种方法和一些
加速推理的权衡。

权衡

许多高产推理优化方法要求权衡一些
因素，这可以根据具体情况进行调整，以便采用量身定制的方法
不同的推理用例和要求。我们的一些优化方法
稍后将讨论这些权衡范围内的某个地方。

权衡一个因素与另一个因素（例如延迟与质量或成本）并不意味着
我们完全牺牲了这个因素，这只是意味着我们正在接受可能发生的事情
质量、延迟或成本的边际下降，以获得显着改善的好处
另一个因素。

质量与延迟/成本的权衡

通过接受可以显着提高推理速度和成本
模型准确性的下降可能微乎其微，可以忽略不计。一个这样的例子
正在使用较小的模型来执行任务。另一个例子是量化，我们
降低模型参数的精度，从而导致更快和更少的内存
密集的计算。

在进行这种权衡时，一个重要的区别是理论
质量损失的可能性与模型执行所需任务的实际能力之间的关系
任务。这是特定于用例的，探索它通常会带来显着的加速，而无需
以有意义或明显的方式牺牲质量。例如，如果我们想要的任务
执行的模型很简单，那么较小的模型或量化模型可能能够
做好这个任务。参数容量或精度的降低不会自动
意味着该模型在该特定任务上的能力较差。

延迟与成本的权衡

这种权衡的另一个名称是延迟与吞吐量的权衡。其中吞吐量指的是
系统有效处理多个请求的能力。相同条件下吞吐量更高

硬件意味着我们的LLM推理成本降低了，反之亦然。

就像传统的软件系统一样，通常有多种权衡机会
延迟与 LLM 推理成本的关系。这是自 LLM 推断以来的一个重要权衡
往往是整个堆栈中最慢且最昂贵的组件；平衡延迟
有意降低成本是确保我们根据产品或用途定制LLM性能的关键
一个例子是批量推理用例（例如离线标记）
其中成本可能是比任何特定请求的延迟更重要的因素。上
另一方面，LLM 聊天机器人产品将更加重视请求延迟。

现在我们已经介绍了优化时需要考虑的一些重要权衡
推理，让我们来看看一些最有效的推理加速技术。作为
在权衡部分讨论，一些优化技术可能会影响
模型的输出。因此，我们将这些方法分为两种类型：输出近似法
和输出保持。

输出近似方法

量化

LLMs 基本上由多个数值矩阵（也称为模型权重）组成。
在推理过程中，矩阵运算被应用于这些模型权重以产生
数字输出（也称为激活）。量化是减少数值的过程
权重和激活的存储、传输和操作的精度。这
权重和激活的默认表示通常是 32 位浮点数，其中
量化我们可以将精度降低到 8 位甚至 4 位整数。

量化具有多种性能优势，它减少了内存占用
该模型允许在相同的硬件上安装更大的模型，它还减少了
一个芯片内和跨芯片的权重和激活的通信开销
分布式推理设置 - 因此加速推理，因为通信是
延迟的主要贡献者。此外，降低权重/激活的精度
可以像某些加速器硬件一样在这些模型上实现更快的算术运算
（例如 TPU/GPU）本机支持更快的矩阵乘法运算，适用于某些较低的
精确表示。

量化对质量的影响可能非常轻微甚至不存在，具体取决于用途
案例和型号。此外，在量化可能引入质量回归的情况下，
与性能增益相比，回归可能很小，因此允许
有效的质量与延迟/成本权衡。例如，Benoit Jacob 等人报告了 2X
MobileNet SSD 上的 FaceDetection 任务的速度提升了 2%，但准确性下降了 2%。

量化可以用作仅推理操作，也可以合并
进入训练（称为量化感知训练 QAT）。 QAT一般是
被认为是一种更有弹性的方法，因为该模型能够恢复一些
训练期间与量化相关的质量损失。确保我们获得最佳的成本/质量
权衡，我们调整量化策略（例如，为权重选择不同的精度
与激活）以及我们将量化应用于张量的粒度（例如通道
或分组）。

蒸馏

使用较小的模型来执行任务是最有效的推理优化之一
然而，较小的模型可能会表现出质量的显着下降
与较大的同行相比。

蒸馏是一组旨在提高较小模型质量的训练技术
（学生）使用更大的模型（老师）。此方法可能有效，因为较大
即使模型都在相同的数据上进行训练，模型的性能也优于较小的模型，这主要是由于
参数能力和训练动态。随着训练的进行，性能差距仍在继续
数据集的增长如图 8 所示。

值得注意的是，即使训练数据量很少，大型模型也已经可以
表现出比相应训练的较小模型更好的性能，这一事实
引出了蒸馏的第一个变体，称为数据蒸馏或模型
压缩。我们使用一个大型模型，该模型是根据我们必须生成的数据进行训练的
更多的合成数据来训练较小的学生模型，数据量的增加将有所帮助
与仅对原始内容进行培训相比，使学生沿着质量线进一步前进
数据。需要仔细处理合成数据，因为它需要具有高质量和
否则可能会导致负面影响。

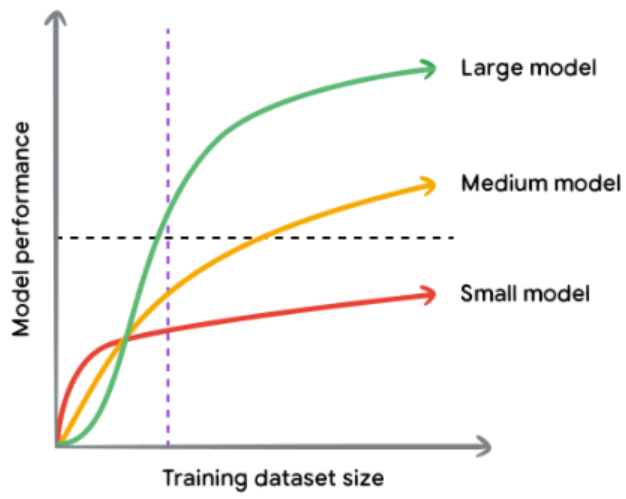


图 8. 不同规模模型的性能随训练变化的图示
数据集的大小

其他蒸馏技术试图使学生模型更接近教师
比合成数据生成更精细的级别。一项突出的技术是
知识蒸馏，在这种方法中，我们尝试对齐输出令牌分布
将学生模型与教师模型进行比较，这比样本效率更高
数据蒸馏。在政策蒸馏是另一种利用反馈的技术
强化学习中学生生成的每个序列的教师模型
设置。

输出保留方法

这些方法保证是质量中立的，它们不会导致模型发生变化
输出通常使他们在面对问题之前优化推理的第一步显而易见
近似方法的更细致的权衡

闪光注意

缩放点积注意力，这是该领域的主要注意力机制
变压器架构，是对输入长度的二次运算。优化自我
注意力计算可以带来显着的延迟和成本优势。

Tri Dao 等人提出的 Flash Attention 通过以下方式优化了注意力计算：
注意力算法 IO Aware，特别是尝试最小化我们移动的数据量
较慢的 HBM（高带宽内存）与较快的内存层 (SRAM/VMEM) 之间
TPU 和 GPU。计算attention时，操作顺序改变了，多个
层被融合，因此我们可以尽可能有效地利用更快的内存层。

Flash Attention是一种精确算法，它维护注意力的数值输出
由于减少了 IO 开销，Tri 可以产生显着的延迟优势
Dao 等人展示了注意力计算中 2-4 倍的延迟改进。

前缀缓存

LLM 推理中计算最密集且最慢的操作之一是
计算我们传递给输入的注意力键和值分数（又名 KV）
LLM，这个操作通常被称为预填充。预填充的最终输出就是所谓的
KV Cache，其中包括变压器每层的注意力键和值分数
对于整个输入。该缓存在产生输出的解码阶段至关重要
令牌，KV 缓存允许我们避免重新计算每个输入的注意力分数
自回归解码步骤。

Prefix Caching是指在后续的KV Cache之间对KV Cache本身进行缓存的过程
推理请求，以减少预填充操作的延迟和成本。道路
自注意力机制的工作使得重用 KV 缓存成为可能，因为令牌将
只注意序列中位于它们之前的标记。如果有新的输入
附加到模型之前见过的输入，那么我们就可以避免
重新计算旧输入的预填充。

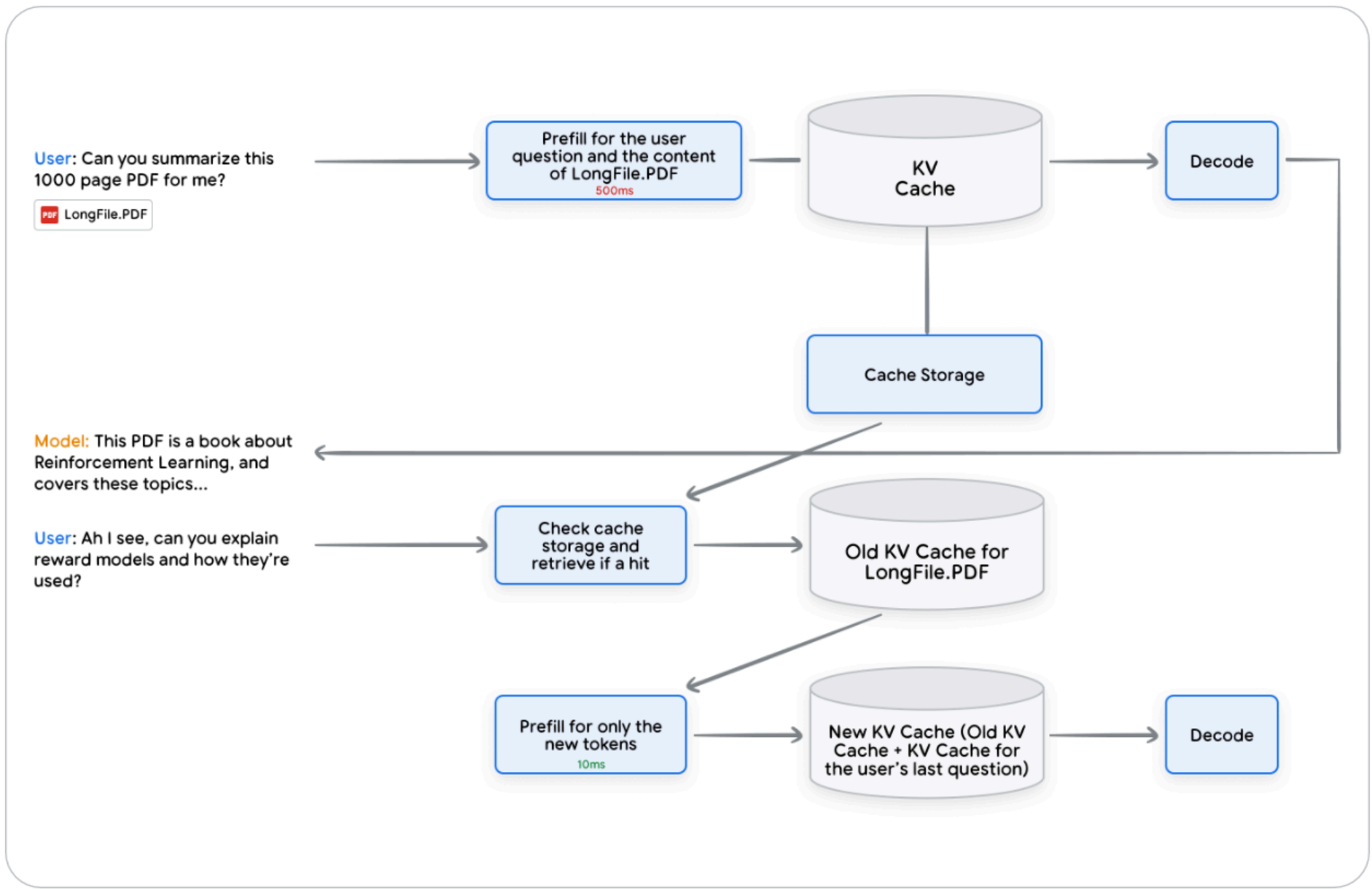


图 9. 聊天场景中前缀缓存的图示

图 9 说明了前缀缓存如何在文档上传的多轮场景中工作。
在第一个用户轮到时，预填充操作必须处理整个文档，以便采取
500ms，然后存储生成的 KV 缓存，以便在第二个用户回合时，我们可以检索
直接从存储中缓存并避免为长文档重新计算它，从而节省
大量的计算和延迟。

前缀缓存可以存储在内存中或磁盘上并按需获取。一
重要的考虑因素是确保输入结构/模式保留前缀 -
缓存友好，我们应该避免在后续请求中更改前缀，因为这会
使所有后续令牌的缓存无效例如，将新的时间戳放在
每个请求的一开始都会使缓存完全失效，因为每个后续请求
请求将有一个新的前缀。

许多 LLM 用例自然适合前缀缓存。例如，LLM 聊天机器人
用户将进行多轮对话，对话内容可能跨越 10 到 1000 个代币，
我们可以避免为会话的前面部分重新计算 KV 缓存。大的
文档/代码上传是另一个用例，其中用户上传的工件将保留
从一个请求到下一个请求都没有变化。改变的只是用户提出的问题
询问，因此缓存文档的 KV 缓存（特别是对于较大的工件）可能会导致
显着降低延迟并节省成本。

前缀缓存可作为 Google AI studio 上名为上下文缓存的服务使用
Google Cloud 上的 Vertex AI。

推测性解码

LLM 推理的第一阶段（称为预填充）由于矩阵较大而受到计算限制
对许多令牌的操作并行发生。第二阶段称为解码，是
通常，由于令牌一次自动回归解码一个，因此会受到内存限制。

天真地使用额外的并行计算能力来加速解码并不容易
考虑到需要等待当前令牌生成才能计算什么
下一个令牌应该是（根据自注意力机制），解码过程是
本质上是串行的。

推测性解码（Leviathan 等人）旨在通过查找来克服解码中的这一限制
一种利用空闲计算能力使每个解码步骤更快的方法。主要思想
就是使用一个更小的辅助模型（通常称为绘图员）来跑在前面
主模型并预测更多标记。 （例如前面有 4 个令牌）。这会很快发生
因为绘图仪比主模型更快、更小。然后我们使用主模型
针对 4 个步骤中的每一步并行验证起草者的假设（即第一个标记、
前两个令牌，前 3 个令牌，最后全部 4 个令牌），然后我们选择接受的
具有最大标记数的假设。例如：

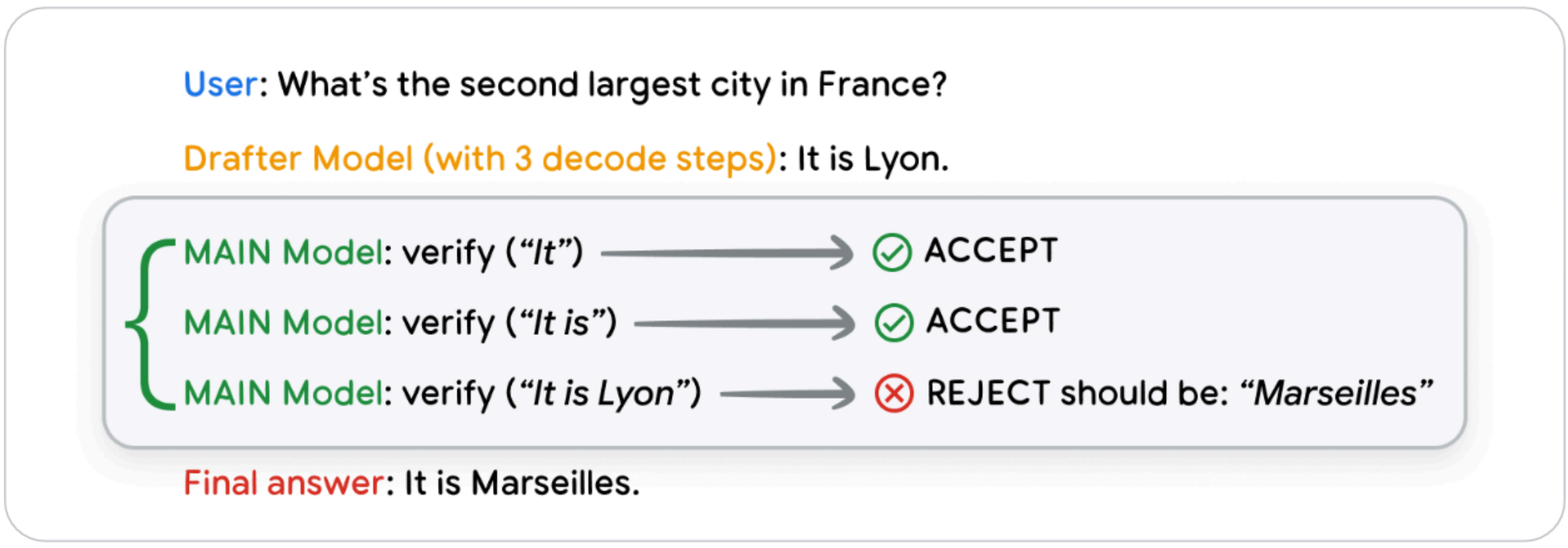


图 10. 对 3 个令牌进行推测性解码的图示

请注意，3 个主要模型步骤是并行运行的。因为我们不受计算限制
解码时，我们可以利用空闲容量来获得更好的解码延迟。在示例中
上面，假设单个主模型步骤需要 10ms，而绘图员需要 1ms。没有
推测性解码，我们需要 3 * 10ms = 30ms 来产生响应，推测性

解码时，由于并行化，关键路径上只有一个主要模型步骤，因此我们
需要 3 * 1ms + 10ms = 13ms。显着改善延迟。这个技术完全是
质量中立，主模型将拒绝任何它自己无法预测的令牌
首先，所以推测解码所做的唯一事情就是提前运行并呈现
主模型可以并行接受或拒绝的假设。

推测解码有效工作的一个重要条件是起草者模型
与主模型具有良好的一致性，否则我们将无法接受任何
的代币。因此，投资绘图员模型的训练质量是值得的
更好的延迟。

现在我们已经看到了一些使 LLM 更快地生成响应的方法，让我们
查看一些如何将这些模型应用于各种任务的示例以获得一个想法
如何使用它们。

批处理和并行化

到目前为止，我们讨论的大多数优化技术都是特定于机器学习的
特别是 Transformer 架构。然而，就像任何软件系统一样，
是通过组合 1) 批处理来提高吞吐量和延迟的机会
计算密集型操作较少（即我们可以在同一硬件上运行多个请求
同时更好地利用空闲计算）和 2）并行化更多计算
计算的密集部分（即我们可以划分计算并将其分成
更多的硬件实例可以获得更多的计算能力，从而获得更好的延迟

LLMs 中的批处理在解码端最有用 - 正如我们在推测中所解释的
解码部分，解码不受计算限制，因此有机会
批量处理更多请求。我们需要小心，以这样的方式进行批处理计算：

能够利用加速器（例如 TPU）上可以实现的备用容量
和 GPU）。我们还需要确保我们保持在内存限制内，因为解码是一个
内存密集型操作，批处理更多请求会给空闲带来更大压力
内存可用。批处理已成为大多数高通量处理的重要组成部分
LLM 推理设置。

鉴于各种机会，并行化也是一种广泛使用的技术
用于跨更多硬件实例水平扩展的变压器。有多个
跨模型输入的并行技术（序列并行）、模型层
（管道并行性）和单层内（张量并行性）。最重要的之一
并行性的考虑因素是之间的通信和同步的成本
我们分配给其他机器的不同分片。沟通很重要
开销，如果我们不这样做，就会削弱增加更多计算能力的好处
小心使用哪种并行化策略。另一方面，选择正确的
平衡额外计算需求和通信成本的策略可以产生
显着的延迟优势。

现在我们已经看到了一些使 LLM 更快地生成响应的方法，让我们
查看一些如何将这些模型应用于各种任务的示例以获得一个想法
如何使用它们。

应用领域

大型语言模型正在彻底改变我们与信息交互和处理信息的方式。
凭借前所未有的理解背景和生成内容的能力，他们
改变文本、代码、图像、音频和视频领域的众多应用程序。
这里我们收集了一些应用领域的例子，但读者应该记住
这不是一个全面的清单，并且许多新想法正在不断涌现

关于如何最好地利用这些新工具的功能。欲了解更多信息
根据以下内容优化构建和部署功能应用程序
用例请参考后续白皮书。

使用以下任一方法为您的用例生成基于文本的响应也非常简单
Google Cloud Vertex AI SDK 或以开发人员为中心的 AI 工作室。片段 3 显示
这些 SDK 中的代码示例，用于使用 Gemini 生成对文本提示的响应
模型。请注意，Gemini 的多模式方面已在各自的章节中进行了介绍。
专用白皮书。

Python

```
# 在开始之前运行此命令：  # pip install --upgrade --user --quiet google-
cloud-aiplatform # 运行 pip install 后确保重新启动内核


从 vertexai.language_models 导入 vertexai 从 vertexai.preview.generative_models 导入
TextGenerationModel 导入 GenerationConfig,GenerativeModel


# 根据您的要求设置值 PROJECT_ID = '' # 设置为您的project_id
vertexai.init(project=PROJECT_ID, location='us-central1')


PROMPT= '什么是 LLM? ' # 在此设置提示 model =
GenerativeModel('gemini-1.5-pro-002')


# 调用 Gemini API response =
model.generate_content( PROMPT)


打印（响应.文本）


# google AI Studio SDK import
google.generativeai as genai import os


# 使用您的 API 密钥更新
genai.configure(api_key=os.environ[“GOOGLE_API_KEY”])


# 选择模型 model = genai.GenerativeModel('gemini-
pro')


response = model.generate_content('什么是 LLM?') # 在此处设置提示


打印（响应.文本）
```

片段 3. 使用 Vertex AI 和 Google AI studio SDK 实现单峰文本基因

代码和数学

生成模型可以理解并生成代码和算法以增强性能
通过在许多应用领域为开发人员提供帮助。一些流行的用例
代码包括：

- 代码生成：LLMs可以用自然语言提示生成代码
  特定的编程语言来执行某些操作。输出可以用作
  草稿。
- 代码完成：LLMS可以在用户键入时主动建议有用的代码。这
  可以节省开发人员的时间并提高代码质量。
- 代码重构和调试：LLMs可以通过重构帮助减少技术债务
  调试代码以提高质量、效率和正确性。
- 代码翻译：LLMs 可以显着帮助开发人员节省时间和精力
  将代码从一种编程语言转换为另一种编程语言。例如，LLM 可能
  将 Python 代码转换为 Java。
- 测试用例生成：可以提示LLMs为提供的生成单元测试
  代码库可以节省大量时间并减少错误。
- 代码文档和理解：LLMs可以以对话方式使用
  进行自然语言聊天以帮助您理解代码库。他们还可以
  生成适当的评论、了解版权状态并创建发行说明。

最近，在竞争领域取得了许多令人兴奋的进展
编码和数学。 AlphaCode 2，将 Gemini 的推理能力与
搜索和使用工具来解决竞争性编码问题。它接收作为输入
要解决的问题的描述，并输出解决该问题的代码解决方案。它

现在在流行的 Codeforces 竞争中跻身前 15% 的竞争程序员之列
编码平台。 FunSearch 使用基于配对的进化过程
经过预先训练的 LLM 和系统评估器。它解决了上限设置问题，开放
数学问题，还发现了更有效的装箱算法
用于许多应用，例如提高数据中心的效率。另一个最近的
称为 AlphaGeometry 的方法解决了寻找复杂几何证明的问题
定理。它包括由神经语言模型和
符号推演引擎。 AlphaGeometry 成功解决了 30 场奥林匹克竞赛中的 25 场
几何问题，人类金牌得主的平均得分为 25.9。

机器翻译

LLMs 能够生成流畅、高质量且上下文准确的翻译。
这是可能的，因为 LLM 对语言细微差别、习语和
语境。以下是一些可能的现实用例：

- 即时通讯应用：在通讯平台中，LLMs可以提供即时通讯
  感觉自然的翻译。与以前可能翻译单词的算法不同
  换句话说，LLMs 理解俚语、口语和地区差异，增强
  跨语言交流。

- 电子商务：在速卖通等全球平台上，产品描述会自动生成
  翻译的。 LLMs 帮助确保产品中的文化差异和惯用表达
  细节得到适当翻译，从而减少误解。

- 旅行应用程序：在谷歌翻译等应用程序中，旅行者可以获得实时口语翻译。
  使用LLMs，翻译后的对话更加流畅，可以用外语进行交互
  国家更省力。

文本摘要

文本摘要是本白皮书中提到的许多LLMs的核心功能。
有许多自然的潜在用例，其中包括：

- 新闻聚合器：LLMs 可以精心制作摘要，不仅捕获主要内容
  事件的同时也体现了文章的情绪和基调，为读者提供了更多的信息
  整体理解。
- 研究数据库：LLMs可以帮助研究人员生成概括的摘要
  科学论文的核心发现和影响。
- 聊天管理：在 Google Chat 等平台中，基于 LLM 的系统可以生成
  捕捉紧迫性和语气的线索摘要，帮助用户确定优先级
  他们的反应。

问答

老一代的 QA 系统通常通过关键字匹配来工作，经常丢失
用户查询的上下文深度。 LLMs 然而，请深入了解上下文。他们可以
推断用户意图，遍历大量信息库，并提供符合上下文的答案
丰富而精确。可以使用此功能的一些示例包括：

- 虚拟助理：LLMs可以提供天气预报的详细解释
  考虑用户的位置、一年中的时间和最近的天气趋势。
- 客户支持：在业务平台中，基于 LLM 的机器人可以提供以下答案：
  考虑到用户的购买历史、过去的查询和潜在的问题，
  个性化的帮助。

- 学术平台：在像 Wolfram Alpha 这样的学术平台上，LLMs 可以满足
  通过了解学术问题的深度和背景，提供用户查询
  适合从高中生到研究生研究员的每个人的答案。

生成答案的质量以及相应的引文和来源
可以通过使用高级搜索系统（例如基于
检索增强生成（RAG）架构）以相关内容扩展提示
信息，以及生成响应后的事后接地。清除
说明、应该和不应该用来回答问题的内容的角色，以及
先进的提示工程方法（例如思想链和搜索/RAG
架构），结合较低的温度值等也可以
有很大帮助。

聊天机器人

早期的聊天机器人遵循脚本化的路径，导致“机械”对话。 LLMs
通过提供动态的、类人的交互来改变这个空间。他们可以分析
情感、背景，甚至幽默，让数字对话感觉更加真实。一些
可以使用此功能的示例包括：

- 客户服务：Zara 等零售平台上的聊天机器人不仅可以回答产品-
  相关查询，还根据当前趋势提供时尚建议。
- 娱乐：在媒体LLM驱动的聊天机器人可以动态地与用户互动，
  对流中的实时事件做出反应，并通过上下文理解来调节聊天。

内容生成

文本生成并不新鲜，但LLMs带来的是前所未有的能力
生成与上下文相关且细节丰富的类人文本。早期型号
在较长的段落中常常会失去上下文或连贯性。 LLMs，以其巨大的
知识和细致入微的理解，可以制作跨越各种风格、语气和内容的文本
复杂性，将事实与创造力（取决于上下文）有效地结合起来
机器生成的内容和人类编写的内容之间的差距。以下是一些
现实世界的例子：

- 内容创建：平台可以利用LLMs来帮助营销人员开发广告。
  LLMs 可以生成有创意的、有针对性的和
  特定于受众的消息。

- 剧本写作：LLMs可能有助于制作电影或电视剧本
  显示。作家可以输入主题或情节点，模型可以建议对话或
  场景描述，增强创作过程。

文本生成是一项广泛的任务，涵盖各种用例，包括
那些生成输出的正确性或多或少比其创造力更重要的情况/
语言的多样性。采样方法和温度等参数应符合
进行相应调整。有关更多信息，请参阅提示工程和架构
LLM 应用白皮书。

自然语言推理

自然语言推理（NLI）是确定给定文本是否
假设可以从文本前提进行逻辑推断。

传统模型难以处理微妙的关系或需要更深入的关系
对上下文的理解。 LLMs，凭借对语义和上下文的复杂掌握，表现出色
在此类任务中，使准确性水平接近人类表现。以下是
一些现实世界的例子：

- 情绪分析：企业可以利用 LLMs 来推断客户情绪
  产品评论。他们不仅可以提取基本的正面或负面标签，
  微妙的情绪，如“满意”、“失望”或“兴高采烈”。
- 法律文件审查：律师事务所可以使用LLMs来推断影响
  和合同中的意图，确保不存在矛盾或潜在的冲突
  有问题的条款。
- 医疗诊断：通过分析患者的描述和病史，LLMs 可以提供帮助
  医生推断潜在的诊断或健康风险，确保早期干预。

关于特定领域LLMs、提示工程和LLM架构的白皮书
应用程序可以进一步深入了解这些用例。

文本分类

文本分类涉及将文本分类到预定义的组中。虽然传统
算法很有效，但它们经常遇到模糊或重叠的类别。
LLMs，鉴于他们对上下文的深刻理解，可以以更高的精度对文本进行分类，甚至
当面对细微的差别时。这方面的一些例子包括：

- 垃圾邮件检测：电子邮件服务可以利用 LLMs 将电子邮件分类为垃圾邮件或
  合法的。这些模型不仅仅是基于关键字的检测，还可以理解上下文
  和意图，可能减少误报。

- 新闻分类：新闻平台可以使用LLMs将文章分类为
  诸如“技术”、“政治”或“体育”等主题，即使文章模糊了界限
  类别之间。

- 客户反馈排序：企业可以通过以下方式分析客户反馈
  LLMs 将它们分类为“产品设计”、“客户服务”或“定价”等领域，
  确保有针对性的反应。

- 将 LLMs 作为自动评分器进行评估：LLMs 可用于对
  还生成其他 LLMs 的输出。

文本分析

LLMs 擅长深度文本分析 - 提取模式、理解主题和收集信息
来自大量文本数据集的见解。传统工具可能会触及表面，LLMs
深入研究，提供丰富且可行的见解。一些潜在的现实例子是：

- 市场研究：公司可以利用LLMs来分析消费者的对话
  社交媒体，提取趋势、偏好和新兴需求。

- 文学分析：学者可以使用 LLMs 来理解主题、主题和
  文学作品中的人物发展，为经典和经典提供新的视角
  当代文学。

多式联运应用

多模态LLMs，能够处理和生成文本、图像、音频和视频，具有
开辟了人工智能的新领域，提供了一系列令人兴奋和创新的应用程序
各个部门。以下是一些示例：

创意内容生成：

- 讲故事：人工智能系统可以观看图像或视频并讲述引人入胜的故事，
  将视觉细节与其知识库相结合。
- 广告和营销：制作有针对性且引起情感共鸣的广告
  基于产品照片或视频。

教育和无障碍环境：

- 个性化学习：根据个人学习风格定制教育材料
  将文本与交互式视觉和音频元素相结合。
- 辅助技术：多模式LLMs可以为描述图像、视频、
  以及为视觉或听觉障碍人士提供的音频。

商业和工业：

- 文档理解与总结：自动提取关键信息
  从复杂的文档中，将文本和视觉效果（如发票和合同）结合起来。
- 客户服务：多模式聊天机器人可以理解并响应客户的查询
  结合文字和图像，提供更丰富、更个性化的体验。科学
  和研究：

- 医学诊断：一起分析医学扫描和报告，识别潜在的可能性
  问题并为医生提供见解。
- 生物信息学和药物发现：整合来自不同数据源的知识，例如
  医学图像、蛋白质结构和研究论文，以加速研究。

这些例子只是冰山一角。随着研究的进展，应用
多式联运 LLMs 的数量预计只会增长，从而以多样化和多样化的方式改变我们的日常生活
深刻的方式。多式联运LLMs也从现有的方法中受益匪浅
单峰 LLMs（即基于文本的 LLMs）。

LLMs，由于他们理解和处理语言的能力，正在重塑我们的方式
与不同部门的文本进行交互、生成和分析。随着它们的不断发展，
它们的应用只会增长，从而提高机器和人类致富的能力
自然语言交互。

概括

在本白皮书中，我们讨论了变压器的基础知识，所有现代技术都以此为基础
LLMs 为基础。我们详细介绍了各种LLM模型架构的演变及其
成分。我们还看到了可用于训练和微调的各种方法
高效且有效地建立模型。我们简要讨论了即时工程和采样
极大地影响 LLM 输出的技术，并且还涉及可能的
该技术的应用。有一些关键要点需要牢记：

- Transformer 架构是所有现代 LLMs 的基础。跨越各种
  在本白皮书中提到的架构中，我们看到重要的是不仅要添加更多
  模型的参数，但数据集的组成同样重要。

- 用于微调的顺序和策略很重要，可能包括多个步骤
  比如指令调优、安全调优等。监督微调（SFT）很重要
  捕捉任务的本质。 RLHF 和潜在的 RLAIF 可用于改变
  通过以下的力量从预训练分布到更理想的分布
  奖励函数，可以奖励理想的行为并惩罚不良的行为。

- 使神经模型的推理变得高效是一个重要的问题，也是一个积极的问题
  研究领域。存在许多方法可以以最小的方式降低服务成本和延迟
  对模型性能的影响，一些精确的加速方法保证相同
  模型输出。

- 大型语言模型可用于各种任务，包括摘要、
  翻译、问答、聊天、代码生成等等。你可以
  使用 Vertex 和 Makersuite 文本生成服务创建您自己的任务
  利用 Google 最新的语言模型。模型经过训练和调整后，
  尝试工程提示很重要。你应该使用该技术
  最适合手头的任务，因为 LLMs 对提示 k 很敏感。
  此外，还可以增强特定任务的表现或创造力，
  通过调整与采样技术相对应的参数来实现多样性，例如
  Top-K、Top-P 和 Max 解码步骤可找到正确性、多样性、
  以及手头任务所需的创造力。

尾注

1. Vaswani, A.、Shazeer, N.、Parmar, N.、Uszkoreit, J.、Jones, L.、Gomez, A. N., ... & Polosukhin, I.，2017 年，注意是所有你需要的。神经信息处理系统的进展，30。

2. 维基百科，2024，Word n-gram 语言模型。可以在：https://en.wikipedia.org/wiki/Word_n-gram_language_model.

3. Sutskever, I.、Vinyals, O. 和 Le, Q. V.，2014 年，使用神经网络进行序列到序列学习。进展神经信息处理系统，27。

4. Gu, A.、Goel, K. 和 Ré, C.，2021，利用结构化状态空间对长序列进行高效建模。arXiv 预印本 arXiv：2111.00396。

5. 贾拉马尔，J.（日期不详）。图示变压器。可以在：https://jalammar.github.io/illustrated-transformer/.

6. Ba, J. L.、Kiros, J. R. 和 Hinton, G. E.，2016 年，层标准化。arXiv 预印本 arXiv：1607.06450。

7. 何凯、张旭、任胜、孙静，2016，图像识别中的深度残差学习。诉讼程序 IEEE 计算机视觉和模式识别会议。

8. HuggingFace.，2024，字节对编码。可以在：https://huggingface.co/learn/nlp-course/chapter6/5?fw=pt.

9. Kudo, T. 和 Richardson, J.，2018，Sentencepiece：一种简单且独立于语言的子词分词器和用于神经文本处理的去标记器。 arXiv 预印本 arXiv：1808.06226。

10. HuggingFace，2024，Unigram 标记化。可以在：https://huggingface.co/learn/nlp-course/chapter6/7?fw=pt.

11. 古德费洛等。等，2016，深度学习。麻省理工学院出版社。网址：http://www.deeplearningbook.org。

12. Radford, Alec 等人，2019，语言模型是无监督的多任务学习者。

13. Brown, Tom, et al., 2020，语言模型是小样本学习者。神经信息的进展处理系统，33，1877-1901。

14. Devlin, Jacob 等人，2018，BERT：用于语言理解的深度双向转换器的预训练。arXiv 预印本 arXiv：1810.04805。

15. Radford, A. 和 Narasimhan, K.，2018，通过生成预训练提高语言理解。

16. Dai, A., & Le, Q.，2015，半监督序列学习。神经信息的进展
处理系统。

17. Ouyang, Long, et al., 2022，训练语言模型以遵循人类反馈的指令。神经信息处理系统的进展，35, 27730-27744.-27744。

18. OpenAI.，2023，GPT-3.5。网址：https://platform.openai.com/docs/models/gpt-3-5。

19. OpenAI.，2023，GPT-4 技术报告。网址：https://arxiv.org/abs/2303.08774。

20. Thoppilan, Romal 等，2022，Lamda：对话应用程序的语言模型。
arXiv 预印本 arXiv：2201.08239。

21. Llama 3.2：通过开放、可定制的模型彻底改变边缘人工智能和视觉。可用的
网址：https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/。

22. Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., ... & Irving, G., 2021，缩放语言
模型：来自训练 Gopher 的方法、分析和见解。网址：https://arxiv.org/pdf/2112.11446.pdf。

23. Du, N., He, H., Dai, Z., McCarthy, J., Patwary, M. A., & Zhou, L., 2022, GLAM：语言的有效扩展
具有混合专家的模型。国际机器学习会议（第 2790-2800 页）。 PMLR。

24. Kaplan, J.、McCandlish, S.、Henighan, T.、Brown, T. B.、Chess, B.、Child, R., ... & Amodei, D.，2020，神经语言模型的缩放
定律。 arXiv 预印本 arXiv：2001.08361。

25. Hoffmann, Jordan 等，2022，训练计算最优大型语言模型。 arXiv
预印本 arXiv：2203.15556。

26. Shoeybi、Mohammad 等，2019，Megatron-LM：使用训练数十亿参数的语言模型
模型并行性。 arXiv 预印本 arXiv：1909.08053。

27. Muennighoff, N. 等人，2023，扩展数据约束的语言模型。 arXiv 预印本 arXiv：2305.16264。

28. Chowdhery、Aakanksha 等人，2023 年，Palm：使用路径扩展语言建模。机械学报
学习研究，24(240)，1-113。

29. Wang, Alex 等人，2019，SuperGLUE：通用语言理解的更具粘性的基准
系统。神经信息处理系统的进展，32。

30. Anil、Rohan 等人，2023 年，Palm 2 技术报告。 arXiv 预印本 arXiv：2305.10403。

31. DeepMind，2023，Gemini：一系列高性能的多模式模型。可以在：
https://storage.googleapis.com/deepmind-media/gemini/gemini_1_report.pdf.

32. DeepMind，2024，Gemini 1.5：解锁跨数百万个上下文标记的多模式理解。
网址为：https://storage.googleapis.com/deepmind-media/gemini/gemini_v1_5_report.pdf。

33. Google 开发者，2024 年，推出 PaLi-Gemma、Gemma 2 和升级的负责任的 AI 工具包。
网址：https://developers.googleblog.com/en/gemma-family-and-toolkit-expansion-io-2024/。

34. Touvron, H.、Lavril, T.、Izacard, G.、Martinet, X.、Lachaux, M.、Lacroix, T., ... & Jegou, H.，2023 年，Llama 2：公开基础和微调的聊天模型。 arXiv 预印本 arXiv：2307.09288。

35. 江阿Q.，2024，专家混合。 arXiv 预印本 arXiv：2401.04088。

36. Qwen，2024 年，推出 Qwen1.5。网址：https://qwenlm.github.io/blog/qwen1.5/。

37. Young, A.，2024，Yi：01.AI 开放基础模型。 arXiv 预印本 arXiv：2403.04652。

38. Grok-1，2024 年，网址：https://github.com/xai-org/grok-1。

39. 段浩东等，2023，BotChat：评估LLMs的多轮对话能力。
arXiv 预印本 arXiv：2310.13650。

40. Google Cloud，2024，通过人类反馈的强化学习来调整文本模型。可以在：
https://cloud.google.com/vertex-ai/generative-ai/docs/models/tune-text-models-rlhf.

41. 白云涛等人，2022，宪法人工智能：人工智能反馈的无害性。 arXiv 预印本 arXiv：2212.08073。

42. 维基百科，2024 年，李克特量表。网址：https://en.wikipedia.org/wiki/Likert_scale。

43. Sutton, R. S. 和 Barto, A. G.，2018，强化学习：简介。麻省理工学院出版社。

44. Bai, Yuntao, et al, 2022，宪法人工智能：人工智能反馈的无害性。 arXiv 预印本 arXiv：2212.08073。

45. Rafailov, Rafael, et al., 2023，直接偏好优化：你的语言模型是秘密的奖励
模型。 arXiv 预印本 arXiv：2305.18290。

46. Houlsby、Neil 等，2019，NLP 的参数高效迁移学习。在国际会议上
机器学习（第 2790-2799 页）。 PMLR。

47. Hu, Edward J., et al., 2021，LoRA：大型语言模型的低阶适应。
arXiv 预印本 arXiv：2106.09685。

48. Dettmers, Tim 等人，2023，QLoRA：量化 LLMs 的高效微调。 arXiv 预印本 arXiv：2305.14314。

49. Lester, B.、Al-Rfou, R. 和 Constant, N.，2021，参数高效提示调整的规模力量。 arXiv
预印本 arXiv：2104.08691。

50. HuggingFace.，2020，如何生成文本？网址：https://huggingface.co/blog/how-to-generate。

51. Google AI Studio 上下文缓存。可用的
位于：https://ai.google.dev/gemini-api/docs/caching?lang=python。

52. Vertex AI 上下文缓存概述。可用的
位于：https://cloud.google.com/vertex-ai/generative-ai/docs/context-cache/context-cache-overview。

53. Gu, A.、Goel, K. 和 Ré, C.，2021，利用结构化状态空间对长序列进行高效建模。
网址：https://arxiv.org/abs/2111.00396。

54. Hubara 等人，2016，量化神经网络：使用低精度权重和
激活。网址：https://arxiv.org/abs/1609.07061。

55. Benoit Jacob 等人，2017，仅进行高效整数算术的神经网络的量化和训练
推理。网址：https://arxiv.org/abs/1712.05877。

56. Bucila, C.、Caruana, R. 和 Niculescu-Mizil, A.，2006 年，模型压缩。知识发现和数据
矿业。网址：https://www.cs.cornell.edu/~caruana/compression.kdd06.pdf。

57. Hinton, G.、Vinyals, O. 和 Dean, J.，2015，在神经网络中提炼知识。
网址：https://arxiv.org/abs/1503.02531。

58. 张丽，费文，吴伟，何勇，楼正，周红，2023，双粒度量化：高效细粒度
LLM 的量化。网址：https://arxiv.org/abs/2310.04836。

59. Agarwal, R.、Vieillard, N.、Zhou, Y.、Stanczyk, P.、Ramos, S.、Geist, M.、Bachem, O.，2024，语言模型的政策蒸馏：从自我生成的错误中学习。可用的
网址：https://arxiv.org/abs/2306.13649。

60. Shazeer, N.、Mirhoseini, A.、Maziarz, K.、Davis, A.、Le, Q.、Hinton, G. 和 Dean, J.，2017 年，惊人的大型神经网络
网络：稀疏门控专家混合层。网址：https://arxiv.org/abs/1701.06538。

61. Schuster, T.、Fried, D. 和 Jurafsky, D.，2022，自信的自适应语言建模。可以在：
https://arxiv.org/abs/2207.07061.

62. Tri Dao 等人。 “闪光注意。可以在：
https://arxiv.org/abs/2205.14135.

63. Leviathan, Y.、Ram, O.、Desbordes, T. 和 Haussmann, E.，2022，通过 Transformer 进行快速推理推测性解码。网址：https://arxiv.org/abs/2211.17192。

64. Li, Y., Humphreys, P., Sun, T., Carr, A., Cass, S., Hawkins, P., ... & Bortolussi, L., 2022，竞赛级代码使用 AlphaCode 生成。科学，378（1092-1097）。 DOI：10.1126/science.abq1158。

65. Romera-Paredes, B.、Barekatain, M.、Novikov, A.、Novikov, A.、Rashed, S. 和 Yang, J., 2023，数学使用大型语言模型进行程序搜索的发现。自然。 DOI：10.1038/s41586-023-06924-6。

66. 维基百科，2024 年，帽子套装。网址：https://en.wikipedia.org/wiki/Cap_set。

67. Trinh, T. H.、Wu, Y. 和 Le, Q. V. 等人，2024 年，无需人类演示即可解决奥林匹克几何问题。自然，625, 476–482。 DOI：10.1038/s41586-023-06747-5。