使用推测采样加速大语言模型解码

CharlieChen[1]、**Sebastian Borgeaud**[1]、**Geoffrey Irving**[1]、**Jean-Baptiste Lespiau**[1]、**Laurent Sifre**[1] **和 John Jumper**[1]所有作者均来自 DeepMind

arXiv:2302.01318v1 [cs.CL] 2 Feb 2023

我们提出了推测采样算法，通过使每次调用 Transformer 时能够生成多个词元，从而加速 Transformer解码。我们的算法基于这样一个观察：由速度更快但能力较弱的草稿模型生成的短延续序列，其并行评分的延迟与从更大的目标模型中采样单个词元的延迟相当。我们将这一点与一种新颖的改进拒绝采样方案相结合，该方案在硬件数值精度范围内保留了目标模型的分布。我们使用具有 70 个十亿参数的语言模型 Chinchilla 对推测采样进行了基准测试，在分布式设置下实现了 2–2.5× 的解码加速，同时不会损害采样质量，也无需修改模型本身。

引言

将 Transformer模型扩展至 5000 亿+ 个参数，使其在许多自然语言、计算机视觉和强化学习任务上取得了显著的性能提升（Arnab 等，2021年；Brown 等，2020年；Chowdhery 等，2022年；Dosovitskiy 等，2020年；Hoffmann 等，2022年；Rae 等，2021年）。然而，在这一规模下，Transformer解码仍然是一个成本高昂且效率低下的过程。

Transformer采样通常受内存带宽限制（Shazeer，2019年），因此，对于给定的一组硬件，Transformer模型生成单个词元所需的时间，与参数规模和Transformer内存规模的一阶近似值成正比。语言模型的规模也使得服务时必须采用模型并行，从而增加通信开销（Pope等人，2022年），并提高资源需求。由于每个新词元都依赖于此前生成的内容，因此采样出一个新序列需要多次调用此类Transformer。

我们提出一种算法，以加速延迟关键型应用中的Transformer采样，并将其称为推测采样（SpS）。其实现方式如下：

1. 生成长度为 K 的短草稿。这可以通过并行模型（Stern等人，2018年）实现，也可以通过调用更快的自回归模型 K 次来实现。我们将此模型称为**草稿模型**，并重点讨论它是自回归模型的情况。2. 使用我们希望从中采样的更大、更强大的模型为草稿评分。我们将此模型称为**目标模型**。3. 使用改进的拒绝采样方案，从左到右接受 K 个草稿令牌中的一部分，并在此过程中恢复目标模型的分布。

直观地说，通常会存在一些序列，其中下一个词元可能是“显而易见的”。因此，如果草稿模型和目标模型对给定词元或词元子序列的分布高度一致，这种方法就能在每次调用目标模型时生成多个词元。

我们证明，草稿令牌的预期接受率足以抵消

通讯作者：*ccharlie@deepmind.com*© 2023 DeepMind。保留所有权利

大型语言模型的草稿生成过程所带来的开销，从而形成一种有效且实用的方法，在无需修改目标模型或使样本分布产生偏差的情况下，降低采样延迟。根据评估领域的不同，SpS 可带来2–2.5× 倍加速比；使用 Chinchilla（Hoffmann 等人，2022年）进行采样时，尤为明显。值得注意的是，SpS 的平均每秒词元数通常超过由内存带宽所限定的自回归采样速度的理想化上限。

相关工作

大量研究致力于降低大型 Transformer 和其他自回归模型的采样延迟。

由于采样性能与模型在内存中的大小密切相关，量化至`int8` 甚至 `int4` （Dettmers 等，2022年；Yao 等，2022年）以及 Transformer 的知识蒸馏（Jiao 等，2020年；Sanh 等，2019年）是降低采样延迟且几乎不影响性能的有效技术。研究发现，模型大小对最终性能的影响低于预期（Hoffmann 等，2022年），这也推动了小型语言模型的普及。

在采样过程中，每个注意力层都会维护一个键值缓存；随着批大小增加，该缓存可能成为内存带宽瓶颈。多查询注意力等方法（Shazeer，2019年）旨在通过缩小缓存来提升采样性能。然而，这些技术最有助于提升吞吐量（在批大小较大时），而非降低延迟；对于较大的模型尤其如此，因为大部分内存带宽预算都被参数占用。

结合上述技术，并对 TPU 进行多项底层优化，Pope 等人（2022年）大幅提升了 PaLM540B 的服务延迟表现和效率。

已有不少类似研究利用 Transformer 和序列模型并行运行的效率。这些研究包括块并行采样（Stern 等人，2018年）、激进解码（Ge 等人，2022年），以及一些在图像领域并行化自回归模型的研究（Song 等人，2021年；Wiggers 和 Hoogeboom，2020年）。这些方法尚未适用于典型的语言模型用例，因为它们要么仅适用于贪心采样，要么会使结果产生偏差，要么面向其他模态。此外，据我们所知，这些技术都尚未扩展到分布式设置，而这对于参数量达数百亿或数千亿的最高成本解码器而言是必要的。

巧合的是，本手稿中的工作与Leviathan 等人（2022年）关于推测解码的工作是同期独立开展的。我们更侧重于大型模型的分布式服务场景，并提出了一些渐进式优化；除此之外，核心思想相同。

自回归采样

虽然 Transformer 可以在 TPU 和图形处理器上高效并行训练，但样本通常是通过自回归方式生成的（参见 算法1）。对于大多数应用，自回归采样（ArS）受内存带宽限制严重，因此无法有效利用现代加速器硬件（Shazeer，2019年）。受内存带宽限制的模型调用每批次中的每个序列只能生成一个词元，因此，在任何采用这种方式的系统中，生成多个词元都会带来很高的延迟。

随着模型参数数量增加，这个问题会尤为突出。由于所有模型参数都必须经过至少一个加速器芯片，因此，模型大小除以所有芯片的总内存带宽，便构成了自回归采样速度上限的硬性约束。较大的模型还需要部署在多个加速器上运行，而设备间通信开销会进一步增加延迟。

算法1 自回归（ArS）与自回归模型

给定自回归目标模型 $q(.|.)$ 和初始提示序列 $x_1, \ldots, x_t$ 以及目标序列长度 T。初始化 $n \leftarrow t$。**当** $n < T$ **时**采样 $x_{n+1} \sim q(x|x_1, \ldots, x_n) n \leftarrow n + 1$**结束循环**

算法 2 自回归目标模型和草稿模型的推测采样（SpS）

给定前瞻 K 和最小目标序列长度 T。
给定自回归目标模型 $q(.|.)$，以及自回归草稿模型 $p(.|.)$，初始提示序列 $x_0, \ldots, x_t$.
初始化 $n \leftarrow t$。
当 $n < T$ **时执行**
  对于 $t = 1 : K$，**执行**
    自回归地采样草稿模型 $\tilde{x}_t \sim p(x|, x_1, \ldots, x_n, \tilde{x}_1, \ldots, \tilde{x}_{t-1})$
  结束循环
  并行计算来自草稿 $\tilde{x}_1, \ldots, \tilde{x}_K$ 的 $K + 1$ 组逻辑值:

$$q(x|, x_1, \ldots, x_n), \; q(x|, x_1, \ldots, x_n, \tilde{x}_1), \; \ldots, \; q(x|, x_1, \ldots, x_n, \tilde{x}_1, \ldots, \tilde{x}_K)$$

  对 $t = 1 : K$ **执行**
    从均匀分布中采样 $r \sim U[0, 1]$。**如果** $r < \text{最小值}\left(1, \frac{q(x|x_1,\ldots,x_{n+t-1})(x|x_1,\ldots,x_{n+t-1})}{p(x|x_1,\ldots,x_{n+t-1})}\right)$，**则**
      设置 $x_{n+t} \leftarrow \tilde{x}_t$ 和 $n \leftarrow n + 1$。
    else
      采样 $x_{n+t} \sim (q(x|x_1, \ldots, x_{n+t-1}) - p(x|x_1, \ldots, x_{n+t-1}))_+$ 并退出循环。
    结束 if
  结束 for
  如果所有词元 $x_{n+1}, \ldots, x_{n+K}$ 都被接受，则采样额外词元 $x_{n+K+1} \sim q(x|, x_1, \ldots, x_n, x_{n+K})$ 并设置 $n \leftarrow n + 1$。
结束 while 循环

推测采样

条件评分

对于推测采样（参见 算法 2），我们首先注意到，并行计算 K 个词元的短续接的逻辑值，其延迟与采样单个词元的延迟非常相似

词元。我们重点关注采用 Megatron 风格分片的大型 Transformer（Shoeybi 等人，2019年）。对于这些模型，大部分采样时间可归因于三个组成部分：

1. **线性层：** 对于较小的批大小，每个线性层只处理少量嵌入。这会导致前馈层中的稠密矩阵乘法、查询、键和值的计算，以及最终的注意力投影都受内存限制。对于较小的 K，这些操作仍会受内存限制，因此耗时也大致相同。

2. **注意力机制：** 注意力机制同样受内存限制。在采样期间，我们会缓存序列中先前词元的所有键和值，以避免重复计算。这些键值缓存占用空间很大，并且消耗了注意力机制所需的大部分内存带宽。不过，由于键值缓存的大小不会随 K 增大而变化，这部分的耗时几乎没有变化。

3. **全归约：** 随着模型规模增大，其参数需要分布到多个加速器上 - 器上，从而产生通信开销。在 Megatron 中，这表现为每个前馈层和注意力层之后都会执行一次全归约。由于传输的只是少量词元的激活值，因此无论是采样还是评分（对于较小的 K），该操作通常都受延迟限制，而非受吞吐量限制。同样，这两种情况下所花费的时间也大致相同。

根据具体的 Transformer 实现，还可能存在其他开销。因此，位置编码的选择、解码方法（例如，核采样可能需要排序）、硬件限制等因素，仍可能导致评分与采样之间出现一些差异。不过，如果满足上述组件占主导地位的条件，那么对于较小的 K，评分不应明显更慢。

修正拒绝采样

我们需要一种方法，利用来自草稿模型的样本以及两个模型对这些词元给出的逻辑值，恢复目标模型的分布。

为此，我们针对草稿词元引入以下拒绝采样方案。给定一个词元序列 $x_1, \ldots\ldots, x_n$，以及从 $p(\text{。}|\text{。})$ 生成的 K 个草稿词元 $\tilde{x}_{n+1}, \ldots\ldots, \tilde{x}_{n+K}$，我们以如下概率接受 $\tilde{x}_{n+1}$：

$$\min\left(1, \frac{q(\tilde{x}_{n+1}|x_1, \ldots, x_n)}{p(\tilde{x}_{n+1}|x_1, \ldots, x_n)}\right)$$

其中，$q(\tilde{x}_{n+1}|x_1, \ldots\ldots, x_n)$ 和 $p(\tilde{x}_{n+1}|x_1, \ldots\ldots, x_n)$ 分别是目标模型和草稿模型在给定当前上下文时的概率。

如果该词元被接受，我们设置 $x_{n+1} \leftarrow \tilde{x}_{n+1}$，并对 $\tilde{x}_{n+2}$ 重复此过程，直到某个词元被拒绝或所有词元都被接受。

如果 $\tilde{x}_{n+1}$ 被拒绝，我们从以下分布中重新采样 x_{n+1}：

$$x_{n+1} \sim (q(x|x_1, \ldots, x_n) - p(x|x_1, \ldots, x_n))_+$$

其中 $(.)_+$表示：

$$(f(x))_+ = \frac{\max(0, f(x))}{\sum_x \max(0, f(x))}$$

依次应用这一过程，我们便能在硬件数值精度范围内，还原目标模型对已接受词元的分布（证明见定理 1）。请注意：

- 草稿-接受循环始终至少会生成一个词元——如果第一个词元被拒绝，就会重新采样一个有效词元。
- 由于草稿模型的最后一个词元会给出下一个词元的逻辑值，因此，如果所有草稿词元都被接受，我们就可以照常从中采样。与朴素实现（每次循环只返回 K 个词元）相比，这样每次循环最多可返回 $K+1$ 个词元。

采用核采样、top-k采样和调整温度等标准采样方法时，我们可以在应用这一拒绝采样方案之前，相应地修改概率。我们观察到，无论使用何种具体参数，整体接受率都很稳定。

由于此方法不会与 Transformer 本身的主体交互，因此可以结合许多其他技术来加速采样或优化其内存使用，例如量化和多查询注意力。

草稿模型的选择

由于接受准则能够保证样本服从目标模型的分布，因此只要生成续写的方法能够提供逻辑值，并且接受率足够高和／或延迟足够低，能够达到收支平衡，我们就可以自由选择草稿生成方法。这里有几种可行方案：

- 将草稿生成整合到目标模型中，并从一开始就训练模型。这是Stern等人（2018年）采用的策略：在 Transformer 中加入多个注意力头，以生成多个词元。
- 使用序列级蒸馏（Kim和Rush，2016年）生成第二个模型，使其能够并行预测 K 词元。这一策略由Ge等人(2022年)采用。
- 将目标模型的一部分激活值作为草稿模型的输入，并使用这些输入训练草稿模型。

尽管这些方法可能会产生性能强大的草稿模型，但它们需要大量由目标模型生成的数据，或需要对目标模型进行改动。尤其是序列级蒸馏，需要大量计算预算。因此，这些方法不太适用于大规模应用。

尽管大型语言模型生成的样本质量更高，但直观来看，有些词元更容易预测，小型模型或许就足以胜任。因此，我们可以直接使用目标语言模型的较小版本作为草稿模型，从而获得较高的接受率。从工程和工作流程的角度来看，这也很方便，因为训练目标模型时，本来就应该已经具备适用于这类模型的稳健工具链。

结果

我们在 16 台 TPU v4 上训练了一个 40 亿参数的草稿模型，并针对采样延迟进行了优化——这与通常用于研究目的运行 Chinchilla 的硬件相同。该模型使用与 Chinchilla 相同的分词器和数据集进行训练，宽度略小，且只有 8 层。相对较少的层数使其能够达到 `1.8ms/token` 的采样速度，而 `14.1ms/token` 是 Chinchilla 的采样速度。详情请参阅表 2中的超参数。

对于分布式设置，不能简单地选择一个小模型作为草稿模型，因为不同模型的最佳推理配置各不相同。例如，通常会以服务方式部署 Chinchilla70B

表格 1 | **Chinchilla 在 XSum 和 HumanEval 上采用朴素采样和推测采样时的性能与速度，批大小为 1 和** $K = 4$。XSum 使用核采样参数 $p = 0.8$，HumanEval 使用 $p = 0.95$，温度为 0.8。

采样方法	基准测试	结果	平均令牌时间	加速比
ArS（核采样）	XSum（ROUGE-2）	0.112	14.1毫秒/令牌	1×
SpS（核采样）		0.114	7.52毫秒/令牌	1。92×
ArS（贪心）	XSum（ROUGE-2）	0.157	14.1毫秒/令牌	1×
SpS（贪心）		0.156	7.00毫秒/令牌	2。01×
ArS（核采样）	HumanEval（100次提示）	45.1%	14.1毫秒/令牌	1×
SpS（核采样）		47.0%	5.73毫秒/词元	2。46×

在16个TPUv4上（其 `14.1ms/token`达到了前述水平），而Chinchilla最优的7B模型则在4个TPUv4上实现最低采样延迟（其达到了 `5ms/token`）。对于较小的模型，额外的内存带宽和浮点运算能力不足以抵消更多设备之间额外的通信开销——在16个TPU上提供7B模型实际上会增加延迟。这意味着，如果采用其最优拓扑结构，7B模型作为草稿模型时只能带来有限的加速比，并且在草稿生成期间无法充分利用硬件。

我们可以通过训练一个层数相对较少的更宽模型来规避这一问题，从而尽量降低通信开销。已有研究观察到，语言模型的性能对于模型纵横比的变化相对稳健（Levine 等，2020年），因此，我们可以提供一个功能强大的草稿模型，并在与目标模型相同的硬件上快速完成采样。

在 XSum 和 HumanEval 上的评估

我们评估 SpS 用 Chinchilla 在两项任务上进行投机采样，并在 表 1 中总结结果：

- XSum（Narayan 等人，2018年）基准测试。这是一项自然语言摘要任务，使用 1 次提示，共采样 11,305 个序列，最大序列长度为 128。
- 100-shot HumanEval 任务（Chen 等人，2021年）。这是一项代码生成任务，需要生成 16,400 个样本，最大序列长度为 512。

即使采用贪心采样，一个词元因数值计算而出现偏差，也可能导致两个序列发生巨大分歧。由于 ArS 和 SpS 处理伪随机种子的方式不同，而且不同的计算图会导致数值结果不同，因此我们不能期待输出完全相同。不过，我们预期这些样本在数值计算精度范围内来自同一分布，并通过评估这些基准测试对这一点进行了实证验证。

我们使用 SpS 和 ArS，以批大小 1 运行这些任务。每次 SpS/ArS 循环所需的时间方差较小，可以直接从 TPU 配置文件中测量。为了获得平均加速比、标准差和其他指标，我们记录每次推测循环生成的词元数量。在表 1 中，我们展示了 Chinchilla 在 XSum 和 HumanEval 基准测试上进行朴素采样和推测采样的性能。

这两项任务都获得了显著加速，HumanEval 的加速比接近 2。5×。然而，我们的基准指标与之持平——底层样本分布经证明在数值精度范围内完全相同，这也从经验上验证了草稿模型不会使结果产生偏差。对于 HumanEval 和贪心 XSum，这种加速超过了硬件对自回归采样的理论内存带宽上限（模型大小除以总内存带宽）。

各领域的接受率变化

显然，接受率取决于应用场景和解码方法。HumanEval 获得的加速比显著更高——我们推测，这是多种因素共同作用的结果：代码包含许多常见子序列（例如，`for i in range(len(arr)):` 对草稿模型来说相对容易猜中）；代码通常会被分解为数量较少、长度较短的词元；此外，温度值会使草稿模型和目标模型的逻辑值分布都更加尖锐。

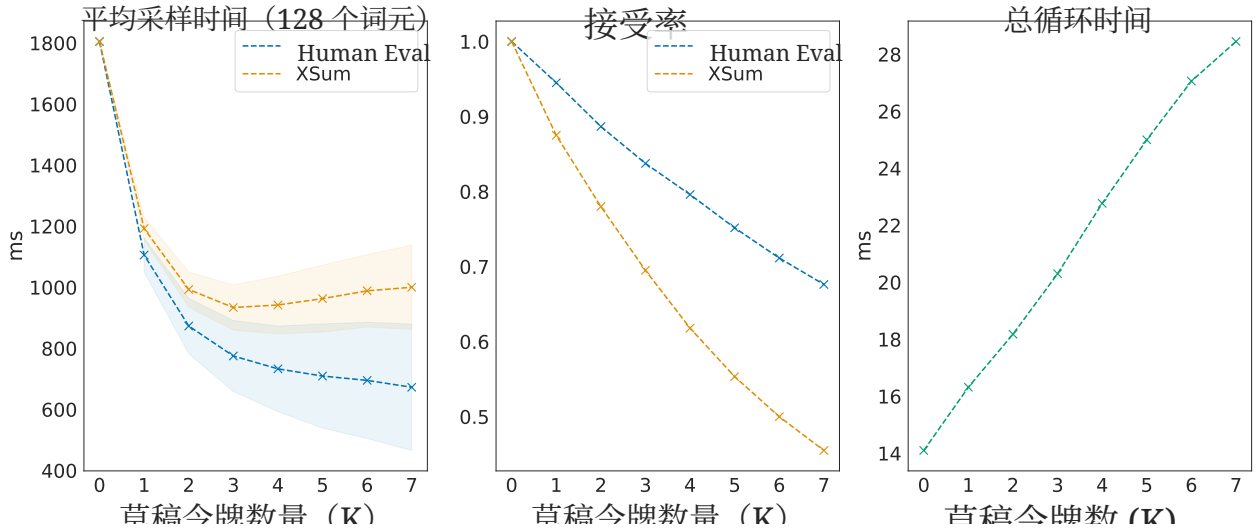

图 1 | **左图：**生成 128 个词元所需的平均时间及标准差。请注意，随着 K增加，整体加速比会趋于平稳，甚至有所下降；XSum 在 $K = 3$时表现最佳。方差始终随着 K增加。**中图：**接受的平均词元数除以$K + 1$ ——这可用于衡量改进的拒绝方案的整体效率，该效率会随着前瞻增加而降低。**右图：**每次循环的平均时间随 K近似线性增加，这是因为模型调用次数增加。请注意，由于核采样解码带来了额外开销，其梯度略高于草稿模型的采样速度。

更长的草稿与更频繁的评分之间的权衡

我们在 K（草稿模型采样的词元数量）增加时，对这种权衡进行了可视化，见图 1。随着 K 增加，为生成相同长度的序列，我们需要更少的大型模型评分调用，因此有望获得更大的加速比。不过，随着草稿模型调用次数增加，以及评分时间小幅增长，总循环时间也近似线性增加。随着 K 增加，已接受词元的比例所体现的整体效率会下降，因为后续词元取决于前序词元是否被接受。因此，平均加速比会趋于平稳，甚至随着 K 增大而下降（例如，使用核采样时，XSum 的延迟在 $K = 3$处达到最低），具体取决于领域。

此外，尽管在某些情况下，较大的 K 值可能会带来略高的平均加速比，但也会增加生成完整序列所需时间的方差。这对于关注 P90、P99 延迟的场景可能会造成问题。

结论

在这项工作中，我们展示了一种加速语言模型解码的新算法和工作流程。推测采样无需修改目标语言模型的参数或架构，在数值精度范围内可证明不会造成损失；配合适当的草稿模型时，它能够良好扩展，并且可以与许多现有技术互补，在小批大小场景下降低延迟。

我们优化了这项技术，并使用一个易于借助现有基础设施训练的草稿模型将其扩展至 Chinchilla 70B，展示了该方法在此过程中能够在各类基准测试任务和常见解码方法上实现显著加速。我们通过下游任务的实证验证，确认该方法确实不会造成损失。

参考文献

A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lucic, and C. Schmid. Vivit：一种视频视觉变换器。载于 *2021*年 *IEEE/CVF*国际计算机视觉大会（*ICCV*），第 6816–6826 页。IEEE计算机学会，2021年。T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell 等。语言模型是少样本学习者。载于 神经信息处理系统进展，33：1877–1901，2020年。M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba。评估基于代码训练的大语言模型。载于 *CoRR*，abs/2107.03374，2021年。网址 https://arxiv.org/abs/2107.03374。A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann 等。PaLM：借助路径扩展语言建模。载于 *arXiv*预印本 *arXiv:2204.02311*，2022年。T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer。LLM.int8（）：面向大规模Transformer的8位矩阵乘法。载于 *arXiv*预印本 *arXiv:2208.07339*，2022年。A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Min- derer, G. Heigold, S. Gelly 等。一张图像胜过 16×16 个词：用于大规模图像识别的Transformer。载于 *arXiv*预印本 *arXiv:2010.11929*，2020年。T. Ge, H. Xia, X. Sun, S. Chen, and F. Wei。通过激进解码无损加速序列到序列生成。载于 *ArXiv*，abs/2205.10350，2022年。

J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark 等。训练计算最优的大语言模型。arXiv预印本 arXiv:2203.15556，2022年。X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu。TinyBERT：通过知识蒸馏BERT以实现自然语言理解。载于计算语言学协会论文集：EMNLP 2020 Findings，第 4163–4174 页，线上，2020年11月。计算语言学协会。doi：10.18653/v1/2020.findings-emnlp.372。网址 https://aclanthology.org/2020.findings-emnlp.372。Y. Kim and A. M. Rush。序列级知识蒸馏。CoRR，abs/1606.07947，2016年。网址http://arxiv.org/abs/1606.07947。Y. Leviathan, M. Kalman, and Y. Matias。通过推测解码加速Transformer推理。ArXiv，abs/2211.17192，2022年。Y. Levine, N. Wies, O. Sharir, H. Bata, and A. Shashua。自注意力中的深度与宽度关系。arXiv预印本 arXiv:2006.12467，2020年。S. Narayan, S. B. Cohen, and M. Lapata。别告诉我细节，只给我摘要！用于极限摘要的主题感知卷积神经网络。载于《2018年自然语言处理经验方法会议论文集》，第 1797–1807 页，比利时布鲁塞尔，2018年10月至11月。计算语言学协会。doi：10.18653/v1/D18-1206。网址 https://aclanthology.org/D18-1206。R. Pope, S. Douglas, A. Chowdhery, J. Devlin, J. Bradbury, A. Levskaya, J. Heek, K. Xiao, S. Agrawal, and J. Dean。高效扩展Transformer推理。arXiv预印本 arXiv:2211.05102，2022年。J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young 等。扩展语言模型：Gopher训练方法、分析与洞见。arXiv预印本 arXiv:2112.11446，2021年。V. Sanh, L. Debut, J. Chaumond, and T. Wolf。DistilBERT：BERT的精简版本，更小、更快、更便宜、更轻量。arXiv预印本 arXiv:1910.01108，2019年。N. Shazeer。快速Transformer解码：只需一个写入头。CoRR，abs/1911.02150，2019年。网址 http://arxiv.org/abs/1911.02150。M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro。Megatron-LM：使用模型并行训练数十亿参数语言模型。arXiv预印本 arXiv:1909.08053，2019年。Y. Song, C. Meng, R. Liao, and S. Ermon。通过并行非线性方程求解加速前馈计算。载于 M. Meila 和 T. Zhang 编，《第38届国际机器学习会议论文集》，第 139 卷《机器学习研究论文集》，第 9791–9800 页。PMLR，2021年7月18–24日。网址 https://proceedings.mlr.press/v139/song21a.html。M. Stern, N. Shazeer, and J. Uszkoreit。深度自回归模型的分块并行解码。CoRR，abs/1811.03115，2018年。网址 http://arxiv.org/abs/1811.03115。A. Wiggers and E. Hoogeboom。使用预测型自回归模型进行预测采样。载于 H. D. III 和 A. Singh 编，《第37届国际机器学习会议论文集》，第 119 卷《机器学习研究论文集》，第 10260–10269 页。PMLR，2020年7月13–18日。网址 https://proceedings.mlr.press/v119/wiggers20a.html。

Z. Yao、R. Y. Aminabadi、M. Zhang、X. Wu、C. Li 和 Y. He。Zeroquant：面向大规模 Transformer 的高效且经济的训练后量化。*arXiv* 预印本 *arXiv:2206.01861*，2022年。

补充材料

作者贡献

- **初步提案：** Charlie Chen、John Jumper 和 Geoffrey Irving
- **初始实现、优化与扩展：**CharlieChen
- **改进的拒绝采样方案：**JohnJumper
- **工程改进：**Jean-Baptiste Lespiau 和 Charlie Chen
- **实验：**Charlie Chen、Sebastian Borgeaud 和 Laurent Sifre
- **手稿草稿：**Charlie Chen 和 Sebastian Borgeaud
- **手稿反馈：**Laurent Sifre、Geoffrey Irving 和 John Jumper

致谢

我们谨向 Oriol Vinyals 和 Koray Kavukcuoglu 致以谢意，感谢你们给予的宝贵建议与领导。我们也要感谢 Evan Senter 对手稿提出的额外反馈，以及 Amelia Glaese 在协助我们完成出版流程方面给予的支持。最后，我们要感谢 BlakeHechtman 和 Berkin Ilbeyi 在 XLA 方面提供的宝贵建议，并感谢 Nikolai Grigoriev 与我们讨论了可应用于 Transformer架构的各种技巧。

超参数

表 2 | **超参数**（草稿模型）

模型	$d_{模型}$	注意力头	层	参数
目标模型 (Chinchilla)	8192	64	80	70B
草稿模型	6144	48	8	4B

证明

定理 1 (修正拒绝采样可恢复目标分布)。 给定离散分布q、 p和一个草稿样本 $\tilde{x} \sim p$，令 X 为最终得到的样本。要使 $X = x$ 成立，我们必须先采样得到 $\tilde{x} = x$ 并接受它，或者在拒绝 $\tilde{x}$ （任意值）后重新采样。因此：

$$\mathbb{P}(X = x)$$

$$= \mathbb{P}(\tilde{x} = x)\mathbb{P}(\tilde{x}\ accepted|\tilde{x} = x) + \mathbb{P}(\tilde{x}\ rejected)\mathbb{P}(X = x|\tilde{x}\ rejected)$$

对于第一项，我们应用接受规则：

$$\mathbb{P}(\tilde{x} = x)\mathbb{P}(\tilde{x}\ accepted|\tilde{x} = x)$$

$$= p(x)\min\left(1, \frac{q(x)}{p(x)}\right)$$

$$= \min\left(p(x), q(x)\right)$$

对于第二个条件项，我们应用重采样规则：

$$\mathbb{P}(X = x|\tilde{x}\ rejected) = (q(x) - p(x))_+$$

其中 $(.)_+$表示：

$$(f(x))_+ = \frac{\max(0, f(x))}{\sum_x \max(0, f(x))}$$

最后，我们计算拒绝概率：

$$\begin{aligned}
&\mathbb{P}(\tilde{x}\ rejected) = 1 - \mathbb{P}(\tilde{x}\ accepted) \\
&= 1 - \sum_{x'} \mathbb{P}(X = x', \tilde{x}\ accepted) \\
&= 1 - \sum_{x'} \min(p(x'), q(x')) \\
&= \sum_{x'} \max(0, q(x') - p(x')) \\
&= \sum_{x'} q(x') - \min(p(x'), q(x')) \\
&= \sum_{x'} \max(0, q(x') - p(x'))
\end{aligned}$$

这等于 $(q(x) - p(x))_+$ 的分母，因此：

$$\mathbb{P}(\tilde{x}\ rejected)\mathbb{P}(X = x|\tilde{x}\ rejected) = \max(0, q(x) - p(x))$$

因此：

$$\begin{aligned}
&\mathbb{P}(X = x) \\
&= \min(p(x), q(x)) + \max(0, q(x) - p(x)) \\
&= q(x)
\end{aligned}$$

由此，我们得到了所需的目标。