PACER：采用自适应长度的推测解码分块预验证

Situo Zhang[1*]、**Yifan Zhang**[1*]、**Zichen Zhu**[1]、**Hankun Wang**[1]、**Da Ma**[1]、**Danyang Zhang**[1]、**Lu Chen**[1,2,3]、**Kai Yu**[1,2,3]
[1]X-LANCE Lab、计算机科学学院、人工智能教育部重点实验室、上海交通大学人工智能研究院、上海交通大学，中国上海[2]江苏省语言计算重点实验室，中国苏州[3]苏州实验室，中国苏州

arXiv:2602.01274v1 [cs.CL] 1 Feb 2026

摘要

推测解码（SD）是一种强大的技术，可以在不牺牲准确性的前提下加速大语言模型（LLM）的推理过程。通常，推测解码会使用一个小型草稿模型生成固定数量的草稿令牌，再由目标模型并行验证。然而，我们的实验发现，不同解码步骤的最优草稿长度存在显著差异。这种差异表明，使用固定草稿长度限制了进一步提升解码速度的潜力。为应对这一挑战，我们提出了 PACER，这是一种新方法，利用轻量级、可训练的预验证层动态控制草稿长度。该层会在草稿令牌发送给目标模型之前，以分块方式对其进行预验证；如果分块预验证失败，草稿模型便可停止生成词元。我们在多个推测解码模型对上实现了 PACER，并在各种基准测试中评估其性能。结果表明，与自回归解码相比，PACER 可实现最高 2.66× 的加速比，并且始终优于标准推测解码。此外，与 Ouroboros 集成后，PACER 可实现最高 3.09× 的加速比。

1 引言

近年来，大语言模型（LLM）彻底改变了人工智能领域 Achiam 等（2023年）；Guo 等（2025年）；Yang 等（2025年），在广泛的任务中展现出卓越能力，包括对话代理 Chiang 等（2024）、代码生成 Jimenez 等（2024）以及复杂推理 Luong 等（2025）。尽管当前 LLM 的性能令人印象深刻，但其自回归特性会按顺序生成词元，从而引入较高的推理延迟。这种延迟主要由内存带宽限制（内存瓶颈）而非计算限制（计算瓶颈）造成 Shazeer（2019年）；Cai 等（2024），导致 GPU并行性未得到充分利用。因此，LLM 仍难以部署于对时间要求严格的应用以及资源受限的边缘设备上 Stern 等（2018）；Ivanov 等（2021）。

人们提出了许多方法来应对这一挑战，其中推测解码（SD）作为一种加速大语言模型推理的有效技术脱颖而出Stern 等人 （2018年）；Leviathan 等人 （2023年）；Chen 等人 （2023年）。其核心思想是将解码过程分为两个阶段：草拟和验证。一个小型且高效的草稿模型会提前预测一系列 γ 词元，随后由较大的目标模型通过一次前向传播对其进行验证。这种方法在保持生成输出质量的同时，提高了推理速度。

*贡献相同。

大多数现有方法在整个解码过程中都使用固定数量的草稿令牌，记为 γ（也称为窗口大小），该窗口大小通常通过测试不同取值来手动确定，以在实验中获得最佳加速比。然而，我们观察到，不同解码步骤之间的接受长度存在显著差异，如图 2(a)所示。这种差异表明，使用固定窗口大小通常会导致次优的草稿生成行为。固定窗口大小的局限性有两点：（1）过于保守的窗口大小（较小的 γ）会导致不必要的目标模型验证开销（图1（b））；而（2）过大的窗口（较大的 γ）会在最终被拒绝的草稿上浪费计算量（图1（c））。我们的预备实验表明，与使用固定窗口大小相比，为每个解码步骤动态设置最优窗口大小，能够将整体吞吐量提高最多 1。4×，如图 2(b)所示。这些发现凸显了需要一种有效且高效的方法，在每个解码步骤动态确定最优草稿窗口大小。

已有多项研究尝试利用草稿模型的固有输出（如令牌概率或熵）来估计令牌接受率 JoaoGante (2023年）；Brown 等人(2024）；Wang 等人(2025年）；Agrawal 等人(2024）；Zhang 等人(2024b）。虽然这些指标在一定程度上反映了模型的语义不确定性，但研究表明，它们的准确性往往会随任务和数据分布的变化而波动，因此难以确保性能稳健且一致 Valentin 等人(2024）。此外，大语言模型容易出现校准失准，尤其表现为过度自信 Quevedo 等人(2024）；Valentin 等人(2024）。仅依赖这些指标会忽略生成上下文及其与目标模型的契合程度，从而导致接受率估计过于粗略且不可靠。另一方面，PEARL Liu 等人(2025年）提出了一种并行运行草稿模型和目标模型的框架。然而，在低资源场景下，这种方法难以高效调度 GPU 资源，尤其是在两个模型部署于同一设备时。它们会争用计算资源，最终拖慢执行速度。此外，PEARL 要求草稿模型每个步骤的总推理成本与目标模型处于同一数量级，这限制了其实际应用价值。

为解决这一问题，我们提出了 PACER，这是一种解码框架，利用可训练的分块预验证模块，根据上下文信息在推测解码过程中动态调节草稿窗口大小。具体而言，PACER 会在目标模型正式验证之前，预先验证草稿令牌是否可能被接受。如图 1(a) 所示，草稿模型首先以大小为 b 的块生成令牌，随后引入预验证层，对每个包含 b 个令牌的块进行预验证。如果当前块中的令牌通过预验证，草稿模型将继续生成下一个包含 b 个令牌的块；否则将停止生成。该方法减少了对单个草稿令牌进行错误分类的情况，并摊销了预验证模块引入的额外推理开销。此外，PACER 为草稿令牌引入了位置编码，从而能够更准确地预测接受概率，并更好地优化动态窗口大小。

我们在多个推测解码模型对上实现了 PACER，包括 DeepSeek-Coder Guo 等人（2024）、Llama-2 Touvron 等人（2023）和 Qwen-2.5Yang 等人 （2024），并在多种文本生成基准上评估了其性能，例如代码生成Chen 等人 （2021）、数学推理 Cobbe 等人 （2021）和文本摘要 See 等人（2017年）。实验结果表明，与普通自回归解码相比，PACER 的加速比最高可达 2.66×，并且始终优于采用固定窗口大小的推测解码。此外，与 Ouroboros 集成后，PACER 相较于自回归解码的加速比最高可达 3.09×。总而言之，我们的主要贡献如下：

- 我们系统分析了自适应草稿长度所带来的性能提升，并给出了关键的实证洞见，直接为我们的动态草稿长度框架设计提供了依据。

- 我们提出了 PACER，这是一种高效且有效的框架，通过对草稿词元进行分块预验证并利用草稿位置信息，动态控制草稿窗口大小，使其达到最优。

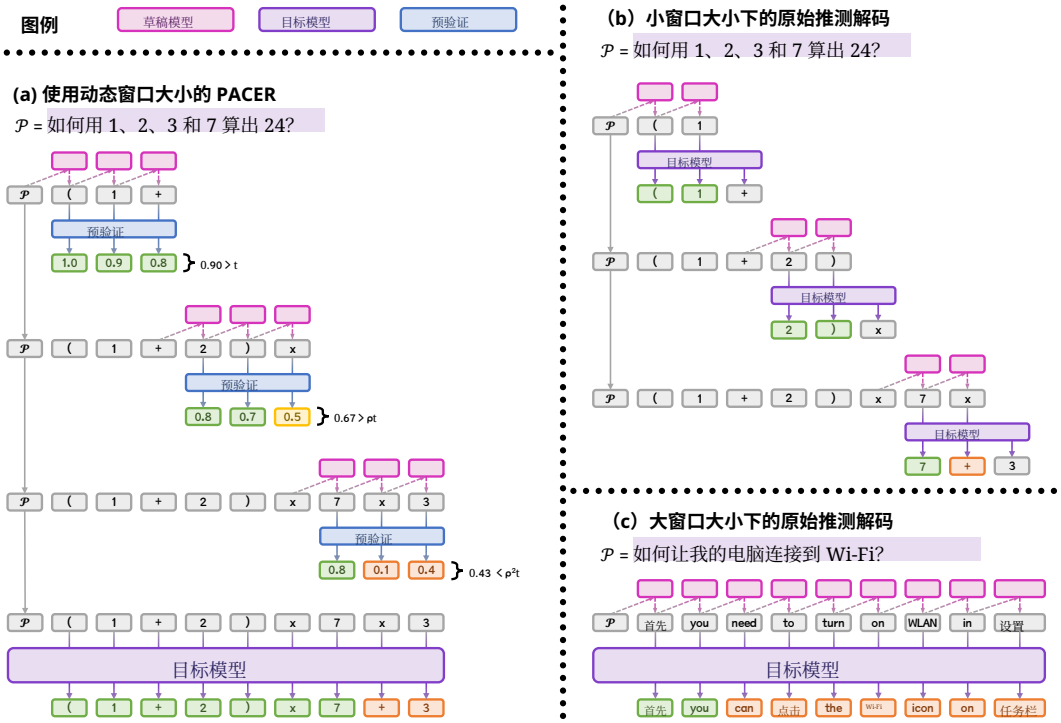


图 1：PACER 与采用固定小窗口大小和大窗口大小的原始推测解码（SD）流程对比。(a) PACER 按大小为 $b = 3$ 的块生成草稿，执行三轮预验证，总共生成 **9** 个草稿令牌，其中 **7** 个被接受，且只需进行 一**次** 目标模型前向传播。(b) 采用小窗口大小（$\gamma = 2$）的原始推测解码会导致草稿模型过早停止，从而产生**3 次代价高昂的**目标模型前向传播和 **6 次**草稿模型前向传播。(c) 采用大窗口大小 $\gamma = 9$ 的原始推测解码会生成 **9** 个草稿令牌，但只有 **2** 个被接受，导致草稿计算量浪费。

- 我们在多个文本生成基准测试上对 PACER 进行了广泛评估，结果表明，PACER 能够持续优于基线方法以及采用固定窗口大小的推测解码方法，展现出其有效性。
- 我们进一步表明，PACER 与其他旨在提升草稿生成质量的推测解码方法兼容，并且能够与这类技术无缝集成，从而进一步提升性能，验证了我们方法的普适性。

2 预备知识

2.1 符号说明

本文用M_T表示目标模型，用M_D表示草稿模型，并用M_B表示我们的分块预验证层。令$\mathcal{V}$表示大小为V的词汇表，令γ表示窗口大小，即M_D在由M_T验证之前生成的草稿令牌数。给定长度为n的输入前缀序列$\mathbf{x} = [x_1, x_2, \ldots, x_n]$，我们将使用模型$M$自回归生成$m$个词元定义如下：

$$(y_1, p_1), (y_2, p_2), \ldots, (y_m, p_m) = M^m(\mathbf{x}), \tag{1}$$

其中$y_1, \ldots, y_m \in \mathcal{V}$是生成的词元，而$p_1, \ldots, p_m \in \mathbb{R}^V$是它们对应的解码分布。每个草稿token y_i都根据其分布$y_i \sim p_i$进行采样，其中$i = 1, \ldots, m$。

对于模型 M 的并行前向传播，我们有：

$$p_1, \ldots, p_{m+1} = M(y_1, \ldots, y_m; \mathbf{x}), \tag{2}$$

其中 p_{m+1} 表示y_m之后下一个词元的分布。

2.2 推测解码

推测解码包含两个阶段：草拟和验证。给定输入前缀$\mathbf{x}$，草稿模型首先自回归地生成 γ 个草稿词元：

$$(y_1, q_1), \ldots, (y_\gamma, q_\gamma) = M_D^\gamma(\mathbf{x}), \tag{3}$$

其中 $q_1, ...q_\gamma$ 表示候选分布。随后，目标模型通过对拼接后的前缀$\mathbf{x}$执行一次并行前向传播，验证草稿$y_1, ...y_\gamma$：

$$p_1, \ldots, p_{\gamma+1} = M_T(y_1, \ldots, y_\gamma; \mathbf{x}). \tag{4}$$

每个草稿 token y_i 的接受率计算如下：

$$\alpha_i = \begin{cases} 1, & \text{if } p_i[y_i] \geq q_i[y_i], \\ \dfrac{p_i[y_i]}{q_i[y_i]}, & \text{otherwise.} \end{cases} \tag{5}$$

如果词元 y_i 被拒绝，之后的所有词元 $y_{i+1}, \ldots, y_\gamma$ 都会被丢弃，并从归一化分布 $\text{norm}(\max(0, p_i - q_i))$中重新采样一个新词元。如果所有草稿都被接受，推测解码会从分布 $p_{\gamma+1}$中额外采样一个词元。这个过程使推测解码能够在一个并行步骤中验证多个词元，大幅减少自回归解码所需的顺序步骤。重要的是，最终输出分布仍与目标大语言模型的普通自回归解码分布一致 Leviathan 等人 (2023年)。

2.3 接受长度

接受长度 L_A 表示单个推测解码步骤中接受的草稿令牌数量，该步骤包括草拟和验证阶段。其上限为草稿窗口大小 γ，因此 $0 \leq L_A \leq \gamma$。L_A 的值由草稿模型和目标模型共同决定，反映了草稿模型对目标模型下一个词元分布的拟合程度。

较大的固定窗口大小通常会带来更长的接受长度，但也会增加可能被拒绝的草稿令牌数量，从而浪费草稿模型的计算量。相反，较小的固定窗口大小虽能减少草稿浪费，却会限制可达到的接受长度，导致目标模型前向传播更加频繁且代价更高。这些权衡凸显了采用动态窗口大小的重要性，即在解码过程中根据观察到的接受情况进行调整。

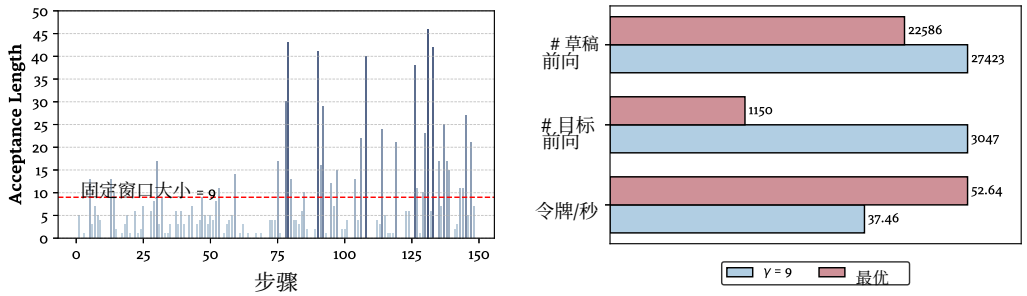
3 方法

本节首先介绍解码步骤之间接受长度的观察结果，并展示采用最优草稿长度所实现的加速比。随后，我们介绍 PACER，这是一种通过分块方式预验证草稿令牌来动态控制草稿窗口大小的解码框架。最后，我们描述 PACER 的训练过程。

3.1 观察结果

3.1.1 不同步骤之间的接受长度差异显著

我们使用 DeepSeek-Coder 模型进行初步实验Guo 等人(2024)，在 Hu- manEval 上使用 1.3B 模型作为草稿模型，33B 模型作为目标模型Chen 等人(2021)。为获得最大接受长度$L^\star_A$，我们将草稿



(a) (b)

图2: （a）各解码步骤中的最大接受长度。最优固定窗口大小（$\gamma = 9$）以红色水平线标示。接受长度的显著变化凸显了采用固定草稿窗口大小的低效。（b）比较采用最优固定窗口（$\gamma = 9$）和最优动态窗口大小（$\gamma^\star$）的推测解码。使用动态草稿长度可大幅减少草稿模型和目标模型的前向传播次数，从而提高解码速度（令牌/秒）。

在解码过程中将窗口大小设置得足够大。不同解码步骤的接受长度绘制于图2(a)。

我们的分析表明，不同解码步骤之间的接受长度存在显著差异，说明固定窗口大小并非最优。这会造成两种主要低效：

1. 当 $\gamma > L^\star_A$ 时，尤其是当 $L^\star_A \approx 0$ 时，计算资源会浪费在生成最终被拒绝的草稿词元上。2. 当 $\gamma < L^\star_A$ 时（图 2(a) 中接受长度较长的柱条），草稿模型会过早终止，错失生成更多被接受词元的机会。这会导致目标模型进行不必要的前向传播，而这类计算开销很大。

因此，动态调整窗口大小对于优化推测解码的效率至关重要。

3.1.2 最优草稿长度提升推测解码效率 *ncy*

我们将每个解码步骤的最优草稿长度定义为 $\gamma^\star$ 最大接受长度$L^\star_A$。如图2(b)所示，与窗口大小固定为 9 且性能最佳的推测解码方法相比，使用 $\gamma^\star$ 可将草稿前向传播次数减少 4,837 次，并将目标前向传播次数从 3,047 次大幅降至 1,150 次，从而使总体加速比达到 1。4×。这一改进源于减少了草稿模型和目标模型中冗余前向传播所带来的计算开销。因此，准确估计 $\gamma^\star$ 可以显著提升推测解码的效率。

3.1.3 接受率随草稿位置增加而下降

我们分析了接受率如何随草稿 token 的位置变化，并在图6中将这一趋势可视化。结果显示，随着草稿位置增加，接受率急剧下降，这表明草稿序列中较晚出现的词元被目标模型接受的可能性显著降低。这说明，在预测接受概率时，可以纳入位置信息。

3.2 PACER

基于第3.1节中的观察，我们注意到，接受长度强烈依赖于生成上下文。因此，我们提出 PACER，这是一种可训练模块，旨在

利用草稿上下文，以分块方式近似预验证草稿 token。PACER 的工作流程如图 1(a)所示。为兼顾预测准确性与计算开销，PACER 在草稿模型之上采用单层 Transformer 来捕获上下文，并以分块方式进行预验证预测。

我们的关键洞见是，准确预测草稿接受率需要考虑全部先前的草稿上下文。然而，我们观察到预验证层的前向传播延迟不可忽略。为了平衡准确性与计算效率，我们以大小为b的块进行预验证，从而有效摊销这一开销。具体而言，给定前缀$\mathbf{x}$，草稿模型M_D 首先生成b 个草稿令牌：

$$(y_1, h_1), \ldots, (y_b, h_b) = M_D^b(\mathbf{x}), \tag{6}$$

其中$h_1, \ldots, h_b$ 表示这b 个草稿的隐藏状态。随后，预验证层M_B将这些隐藏状态与位置嵌入一并处理：

$$\hat{\alpha}_1, \ldots, \hat{\alpha}_b = M_B\left([h_1 + e_1], \ldots, [h_b + e_b]; \mathbf{h_x}\right), \tag{7}$$

此处，$\hat{\alpha}_1$，…，$\hat{\alpha}_b$ 表示草稿 $y_1, \ldots, y_b$的估计接受率，而 $e_1, \ldots, e_b$表示对应于草稿位置 1 到 b 的位置嵌入。此外，h_x 表示当前草拟与验证步骤之前已接受词元的隐藏状态。我们计算该块内估计接受率的均值 mean($\hat{\alpha}_1, \ldots, \hat{\alpha}_b$)，若该均值低于预定义阈值 t，则停止草拟，并由目标模型 M_T触发验证。否则，M_D 继续生成另一个包含 b 个词元的块。在后续预验证轮次中，草稿位置变为 b + 1，…，2b，依此类推。对于经过 k 轮预验证的草稿，该解码步骤的最终窗口大小为$\gamma = k \cdot b$。同时，阈值 t 每轮按增长因子 $\rho > 1$提高，从而使后续步骤中更容易停止草稿生成。详细算法见第 A 节。

3.3 训练

为确保训练与推理之间的一致性，我们首先使用目标模型 M_T根据训练数据集中的指令生成响应。随后，我们应用推理过程中使用的相同推测解码过程，构建带标签的接受数据。具体而言，草稿模型 M_D 从给定的前缀序列 $\mathbf{x}$ 开始，生成一系列草稿 token $y_1, \ldots, y_\gamma$，并使用较大的窗口大小（例如 $\gamma = 50$）。我们将这些草稿与目标模型生成的词元进行比较：如果草稿 token y_i 与对应的目标词元不同，则将所有后续词元 $y_i, \ldots, y_\gamma$ 标记为 0（拒绝），而将词元 $y_1, \ldots, y_{i-1}$ 标记为 1（接受）。此过程生成的带标签训练数据与推理时的验证过程精确对齐。我们使用标准交叉熵损失训练预验证层 M_B。为提高训练效率，我们将多个解码步骤中的草稿 token 打包到一个序列中，并精心设计注意力掩码。训练详情见第 D节。

4 实验

4.1 实验设置

评估数据集。 我们在多种具有代表性的文本生成基准测试 上评估 PACER，以展示其有效性。具体而言，对于代码生成任务，我们采用两个广泛使用的基准测试：HumanEval（Chen 等人，2021）和 MBPP（Austin 等人，2021）。对于文档摘要，我们使用成熟的 CNN/Daily Mail（CNN/DM）数据集（See 等人，2017年）。此外，对于算术推理，我们使用 GSM8K 数据集（Cobbe 等人，2021）。有关评估数据集的更多详情，请参见第 C.1节。

训练数据集。 我们使用 CodeAlpaca 数据集（Chaudhary，2023年），该数据集包含约 20,000 个指令-响应对，涵盖广泛的编程场景和编码任务。我们使用其中的指令生成训练数据。

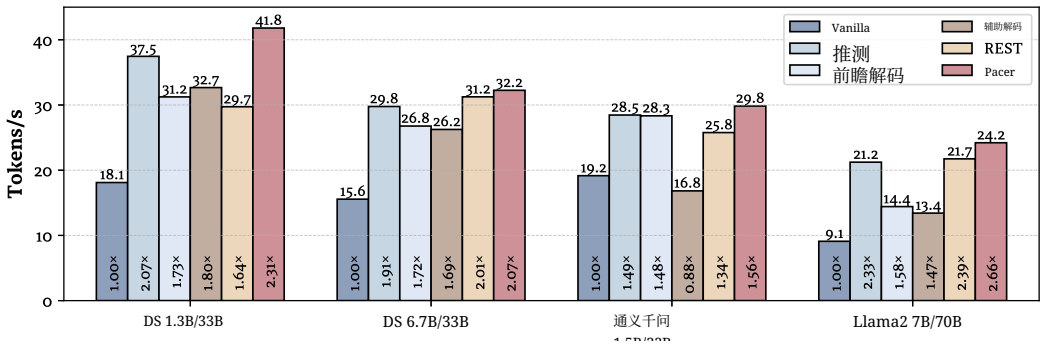


图 3：HumanEval数据集上不同方法的解码速度（令牌/秒）和加速比对比。PACER 在不同模型上始终优于所有基线方法。

模型。 我们采用当前研究中广泛使用的多个最先进大语言模型，包括 DeepSeek-Coder 系列（Guo 等人，2024）、Llama2 系列（Touvron 等人，2023年）以及 Qwen2.5 系列（Yang 等人，2024）。具体而言，我们采用 DeepSeek-Coder 系列的 **1.3B/33B**和**6.7B/33B**模型、Llama2-chat 系列的 **7B/70B** 模型，以及 Qwen2.5 系列的**1.5B/32B** 模型，将参数较少的模型用作草稿模型，并将其较大的对应模型用作目标模型。这些模型涵盖了广泛的模型大小，能够代表大语言模型研究中的典型规模。

评估方法。我们将本方法与若干成熟的基线方法进行比较，包括普通自回归解码、推测解码（Leviathan 等人，2023年；Chen 等人，2023年）、前瞻解码（Fu 等人，2024）、基于启发式的动态窗口方法（辅助生成）（Joao Gante，2023年），以及基于检索的解码（REST）（He 等人，2024）。为确保在我们的硬件环境下进行公平比较，我们使用各基线方法的官方实现和默认参数复现了每种方法。在所有实验中，包括基线方法和我们提出的方法，我们均将批量大小设为 1，这也是推测解码框架中的标准设置。我们使用以下指标评估模型性能：**解码速度（令牌/秒）**，**加速比**，以及**平均接受长度**（τ）。详细的评估设置见第C.1节。

4.2 主要结果

我们在此前所述的基准测试上进行了实验。如图3所示，在 HumanEval 数据集上，PACER 在所有评测的模型系列中始终优于普通自回归解码、推测解码、前瞻解码、辅助生成和 REST。具体而言，在使用 DeepSeek-Coder 1.3B/33B 的实验中，PACER 达到了 41.8 令牌/秒的解码速度，相比普通自回归解码实现了 2.31× 的加速比，并超过了原始推测解码达到的 2.07× 加速比。对于 Llama-2 7B/70B，加速比从 2.33× 提升至 2.66×。表1进一步展示了多个数据集上各种方法的解码速度及相应加速比。PACER 在所有模型上始终实现了最高的解码速度和加速比，优于所有测试方法。值得注意的是，在 GSM8K 数据集上，使用 DeepSeek-Coder 1.3B/33B 时，加速比从 1.92× 提升至 2.17×；使用 Llama-2 7B/70B 时，则从 1.94× 提升至 2.08×。这些结果表明，利用分块预验证层动态控制草稿长度，能够有效发挥草稿模型的能力，从而在多种任务和模型配置下实现显著的性能提升。

4.3 与其他动态草稿长度方法的比较

我们将 PACER 与若干采用动态草稿长度的先进方法进行基准测试对比，包括 AdaEDL（Agrawal 等人，2024），其生成草稿的依据是

算法	Deepseek 1.3B/33B		Deepseek 6.7B/33B		Llama-2 7B/70B	
	令牌/秒	加速比	令牌/秒	加速比	令牌/秒	加速比
			MBPP			
Vanilla	17.53	1.00	15.86	1.00	9.28	1.00
推测	30.72	1.75	26.62	1.68	18.65	2.01
前瞻解码	29.55	1.69	23.67	1.49	13.30	1.43
辅助解码	26.60	1.52	21.07	1.33	16.30	1.76
REST	29.41	1.69	25.64	1.60	15.38	1.66
PACER	**32.93**	**1.88**	**28.57**	**1.80**	**19.90**	**2.14**
			CNN/DM			
Vanilla	17.11	1.00	15.37	1.00	8.46	1.00
推测	23.55	1.38	20.55	1.34	14.40	1.70
前瞻解码	18.21	1.08	17.64	1.15	8.56	1.01
辅助解码	22.27	1.30	18.52	1.20	11.36	1.34
REST	18.50	1.06	16.70	1.09	9.62	1.14
PACER	**23.77**	**1.39**	**21.51**	**1.40**	**14.50**	**1.71**
			GSM8K			
Vanilla	18.25	1.00	16.51	1.00	9.19	1.00
推测	35.08	1.92	29.36	1.78	17.81	1.94
前瞻解码	33.82	1.85	29.09	1.76	12.07	1.31
辅助解码	34.86	1.91	28.80	1.74	14.02	1.53
REST	23.25	1.27	22.22	1.34	12.66	1.38
PACER	**39.69**	**2.17**	**31.07**	**1.88**	**19.07**	**2.08**

Table 1：不同方法与大语言模型家族的解码速度（令牌/秒）和加速比 s
on MBPP、CNN/DM 和 GSM8K 基准测试。

方法	MT-Bench	翻译	摘要	QA	数学推理	RAG	平均值
Vanilla	9.04	9.31	8.63	9.32	9.29	8.32	8.99
辅助解码	17.77	17.95	16.42	17.10	20.46	15.60	17.58
AdaEDL	17.55	17.85	16.10	17.34	20.21	15.69	17.47
SpecDec++	18.18	17.62	**16.57**	17.41	20.81	**15.80**	17.79
PACER	**18.53**	**18.06**	16.19	**17.42**	**21.45**	15.48	**17.95**

表 2：Llama-2 7B/70B 在 SpecBench 上的解码速度（令牌/秒）。

草稿词元分布的熵，以及 SpecDec++(Huang 等人，2024），它通过额外的预测头控制草稿长度。我们在 HumanEval 基准测试上进行了有针对性的比较。如表 3 所示，PACER 的表现优于所有其他动态方法，实现了最高的解码速度（24.20 令牌/秒）和显著更长的平均接受长度（7.46）。这些结果凸显了我们的预验证层的有效性。

此外，我们在 SpecBench 基准测试（Xia 等人，2024a）上评估了这些方法。如表 2 所示，PACER 始终实现最高的平均解码速度，证实了我们的预验证机制具有良好的泛化能力。

4.4 与 Ouroboros 集成

为展示 PACER 的灵活性与正交性，我们将其与 Ouroboros（Zhao 等人，2024）相结合。Ouroboros 是一种近期提出的方法，通过从 n-gram 池中生成短语级草稿来提升草拟效率。Ouroboros 虽然提升了草稿质量，但采用固定的草稿步。相比之下，PACER 通过预验证动态调整草稿长度。我们推测，结合这两种互补策略可以进一步提升性能。我们使用 DeepSeek-Coder 1.3B/33B，在 HumanEval、MBPP 和 GSM8K 基准测试上开展了实验，并采用 Ouroboros 论文中的最优配置。

方法	令牌/秒	τ
推测	21.20	5.39
SpecDec++*	21.17	5.13
AdaEDL	22.53	4.57
PACER	**24.20**	**7.46**

表 3：使用 Llama-2 7B/70B 在 HumanEval 上进行动态草稿长度方法的解码速度（令牌/秒）和平均接受长度（τ）。由于与原论文相比聊天模板存在差异，*SpecDec++ 在此处的表现不如标准推测解码。我们在附录中给出了使用原始 SpecDec++ 模板获得的结果，其中 PACER 仍然更优。

数据集	方法	令牌/秒	加速比	τ
HumanEval	Vanilla	18.10	1.00	—
	Ouroboros	49.05	2.71	8.36
	PACER +Ouroboros	**51.07**	**2.82**	**10.90**
MBPP	Vanilla	17.53	1.00	—
	Ouroboros	47.99	2.74	3.81
	PACER +Ouroboros	**50.08**	**2.86**	**5.10**
GSM8K	Vanilla	18.25	1.00	—
	Ouroboros	52.66	2.89	4.43
	PACER +Ouroboros	**56.31**	**3.09**	**5.99**

表4：使用 Deepseek-Coder 1.3B/33B 将 PACER 与 Ouroboros 结合时，在基准测试 HumanEval、MBPP 和 GSM8K 上对解码速度（令牌/秒）和平均接受长度（τ）的比较。

如表4所示，集成方法始终能实现更高的解码速度和更长的平均接受长度。例如，在 GSM8K 上，加速比从单独使用 Ouroboros 时的 2.89× 提升至组合方法的 3.09× 。这些结果凸显了 PACER 的适应性及其在增强其他优化草稿过程的推测解码方法方面的有效性。

4.5 消融研究

方法	HumanEval			GSM8K		
	令牌/秒	加速比	τ	令牌/秒	加速比	τ
Vanilla	18.11	1.00	–	18.25	1.00	–
推测	37.46	2.07	7.38	35.08	1.94	5.87
PACER	**41.80**	**2.31**	9.67	**39.69**	**2.17**	7.23
不含位置嵌入	39.99	2.21	8.83	37.48	2.05	7.19
不含增长因子	40.36	2.23	**9.71**	37.84	2.07	**7.46**

表5：使用 DeepSeek-Coder 1.3B/33B 对草稿位置嵌入和阈值增长因子进行的消融实验。

我们进行了一系列消融研究，以考察我们方法的关键组成部分。具体而言，我们分析了草稿位置嵌入（**位置嵌入**）和阈值增长因子（**增长因子**）对整体性能的影响。我们还进行了额外的消融实验，以分析超参数设置和模型设计的关键方面。更多消融实验见附录。

位置嵌入的影响。 如表5所示，移除草稿位置嵌入会将解码速度降至39.99令牌/秒，表明位置信息对PACER的整体性能有显著贡献。这与以下观察一致：

草稿位置是接受率的强有力指标，将其纳入能够提高预测准确率。

增长因子的作用。如表5所示，移除阈值增长因子会增加平均接受长度 τ，但会降低解码速度。这表明，随着草稿序列变长，增长因子通过逐步提高提前停止的可能性，发挥着至关重要的作用。没有增长因子时，PACER 往往会生成过长的草稿，造成计算量浪费并降低效率。

5 讨论

本节将探究 PACER 加速推测解码的深层因素，并讨论关键的架构设计选择。我们将分析 PACER 对性能的贡献，考察草稿生成的替代停止准则，并评估预验证层中不同注意力范围的影响。

5.1 PACER 的加速比从何而来?

为了解 PACER 的性能优势，我们记录了草稿模型和目标模型的前向传播次数，以及平均接受长度。如表 6所示，在 HumanEval、CNN/DM 和 GSM8K 上，与采用最优固定窗口大小的推测解码相比，使用 DeepSeek-Coder 1.3B/33B 的 PACER 实现了显著提升。PACER 减少了草稿模型和目标模型的前向传播次数，同时提高了平均接受长度。

这些改进源于 PACER 能够自适应地调整草稿窗口大小。在预测不确定的区域，它会提前停止，以避免浪费草稿计算量；而在接受可能性较高的片段中，它会生成更长的草稿。这种自适应行为提升了数据移动效率，缓解了自回归解码固有的内存瓶颈，并在不同任务和模型规模下带来稳定的加速效果。

方法	HumanEval			CNN/DM			GSM8K		
	#草稿前向	#目标前向	τ	#草稿前向	#目标前向	τ	#草稿前向	#目标前向	τ
推测	27423	3047	7.38	23904	7968	1.78	59456	7432	5.87
PACER	**26376**	**2368**	**9.67**	**22252**	**7884**	**1.89**	**54024**	**6816**	**7.23**

表6：在不同基准测试中，比较采用最优固定窗口大小的原始推测解码（推测）与 PACER 在草稿模型和目标模型上的前向传播次数，以及平均接受长度（τ）。

5.2 草稿生成的最佳停止准则是什么?

为确定预验证阶段的有效停止准则，我们将所提出的平均令牌概率准则与两种更严格的替代方案进行比较：（1）**任意词元低于**t：当**任**一词元在该块中的预测接受率低于阈值t时即停止，以及（2）**末词元低于**t：仅根据**最后**一个词元的预测接受率停止。结果如表7所示。

表7：在 HumanEval 上使用 DeepSeek-Coder 1.3B/33B 时，采用不同停止准则的解码速度和平均接受长度（τ）对比。

停止准则	令牌/秒	τ
任意词元低于t	38.96	8.70
最后一个词元低于t	40.11	9.34
平均令牌概率t （**我们的方法**）	**41.80**	**9.67**

如表7所示，我们的平均令牌标准取得了最高的解码速度和最长的接受序列长度。这是因为对词元取平均可减轻偶发低置信度预测的影响。相比之下，更严格的准则往往会过早停止草稿，从而缩短接受长度并降低整体吞吐量。

5.3 注意力上下文如何影响预验证?

为验证预验证层中全上下文注意力的重要性，我们将其与两种受限变体进行比较：（1）**局部块**，其中注意力仅限于当前块；（2）**局部草稿**，其中注意力仅限于当前草稿步骤。结果汇总于表8。

表8：注意力上下文消融实验

上下文范围	令牌/秒	τ
局部块	22.29	5.32
局部草稿	22.58	5.45
完整上下文（**本文**）	**24.20**	**7.46**

扩大注意力范围能够持续提升解码速度和接受长度 τ。完整上下文注意力尤为有效，因为它能够捕捉草稿与前缀中的长程依赖，从而实现更可靠的接受预测。

6 相关工作

推测解码 推测解码（推测解码）采用草稿-验证范式实现无损加速。现有的推测解码框架通常可根据所使用的草稿模型类型进行分类（Xia 等，2024b）。（1）独立草稿模型：SpecDec（Xia 等，2023）提出了一种非自回归（non-AR）Transformer，可同时生成多个词元。其他若干工作（Leviathan 等，2023；Chen 等，2023；Spector 与 Re，2023；Sun 等，2023）提出利用同一大语言模型家族中较小的预训练模型（例如 Llama2-7B 与 Llama2-70B（Touvron 等，2023））来加速推理，从而避免额外训练，并凭借共享的分词器和训练语料库保持预测行为的一致性。SpecInfer（Miao 等，2024）和 DistillSpec（Zhou 等，2024）采用蒸馏技术来获得更小的草稿模型。（2）自草稿：该方法通过使用目标大语言模型自身进行草拟，从而无需外部草稿模型，进而确保高度一致。Stern 等人（2018）；Cai 等人（2024）；Hwang 等人（2024）；Gloeckle 等人（2024）训练多个预测头，以同时预测多个未来的草稿令牌。Eagle（Li 等，2024b）引入单层 Transformer，在特征层面进行自回归预测。此外，Santilli 等人（2023）；Yi 等人（2024）；Xiao 等人（2024）探索了用于生成草稿的非自回归解码方法。前瞻解码（Fu 等，2024）利用雅可比迭代构建 n 元语法池作为草稿。与此同时，Yang 等人（2023）；Zhang 等人（2024a）；Elhoushi 等人（2024）的工作研究了在目标模型中采用提前退出和跳层，以提升草拟效率。尽管已有这些进展，大多数方法仍采用固定窗口大小，自适应草稿生成这一挑战在很大程度上仍未得到解决。

自适应草稿长度 近期有多项研究探索了草稿长度的自适应调整。Zhang 等人（2024a）、Liu 等人（2024）、Li 等人（2024a）、Joao Gante（2023）、Brown 等人（2024）、Wang 等人（2025）、Agrawal 等人（2024）以及 Zhang 等人（2024b）的研究提出，利用草稿模型的内在输出（例如词元概率或熵）估计草稿接受率，以此确定草稿停止标准。这些方法通常依赖人工设定的阈值或更新规则，来确定何时停止草拟。尽管基于启发式的方法简单且计算效率高，但仅依赖语义不确定性指标，缺乏稳健性和一致的准确性。这主要是因为大语言模型固有的校准失准和过度自信（Quevedo 等人，2024；Valentin 等人，2024；Huo 等人，2025），导致接受率估计过于简单且不可靠

。++ SpecDec（Huang 等人， 2024）和 DISCO（Mamou 等人，2024）通过在草稿模型之上训练分类器，来判断何时停止词元生成，以此应对这一问题，但它们忽略了草稿上下文信息。与此同时，PEARL（Liu 等人，2025）引入了一种并行框架，使目标模型和草稿模型能够并行运行；然而，两种模型部署在同一位置时会相互竞争，因此该框架在应对资源约束方面面临挑战。为解决这些局限，我们提出 PACER，这是一种解码框架，利用融合上下文信息的可训练分块预验证层，高效且准确地确定最优草稿长度。

7 结论

本文提出了 PACER，这是一种有效且高效的方法，可根据上下文信息动态控制草稿长度。PACER 采用可训练的预验证层，对草稿 token 进行分块预验证。只有当数据块通过预验证后，草稿模型才会继续生成词元。这种自适应策略有效减少了推测解码中使用固定窗口大小所造成的计算浪费。我们在多种基准测试上进行了全面实验，结果表明，PACER 在不同模型架构下始终优于所有推测解码基线，展现了其处理可变草稿长度的有效性。

参考文献

Josh Achiam、Steven Adler、Sandhini Agarwal、Lama Ahmad、Ilge Akkaya、Florencia Leoni Aleman、Diogo Almeida、Janko Altenschmidt、Sam Altman、Shyamal Anadkat 等。GPT-4 技术报告。*arXiv* 预印本 *arXiv:2303.08774*，2023年。

Sudhanshu Agrawal、Wonseok Jeon 和 Mingu Lee。AdaEDL：通过基于词元接受概率的熵下界，对大语言模型的推测解码进行早期草稿停止。在*NeurIPS* 高效自然语言与语音处理研讨会，第 355–369 页。PMLR，2024。

Jacob Austin、Augustus Odena、Maxwell Nye、Maarten Bosma、Henryk Michalewski、David Dohan、Ellen Jiang、Carrie Cai、Michael Terry、Quoc Le 等。使用大语言模型进行程序合成。*arXiv* 预印本 *arXiv:2108.07732*，2021。

Oscar Brown、Zhengjie Wang、Andrea Do、Nikhil Mathew 和 Cheng Yu。动态深度解码：面向大语言模型的更快速推测解码，2024。网址 https://arxiv.org/abs/2409.00142。

Tianle Cai、Yuhong Li、Zhengyang Geng、Hongwu Peng、Jason D. Lee、Deming Chen 和 Tri Dao。Medusa：基于多个解码头的简易大语言模型推理加速框架。载于 Ruslan Salakhutdinov、Zico Kolter、Katherine Heller、Adrian Weller、Nuria Oliver、Jonathan Scarlett 和 Felix Berkenkamp（编），第四十一届国际机器学习会议论文集，第 235 卷，机器学习研究论文集，第 5209–5235 页。PMLR，2024年7月21日至27日。网址https://proceedings.mlr.press/v235/cai24b.html。

Sahil Chaudhary。Code alpaca：一个用于代码生成的指令遵循Llama模型。https://github.com/sahil280114/codealpaca，2023年。

Charlie Chen、Sebastian Borgeaud、Geoffrey Irving、Jean-Baptiste Lespiau、Laurent Sifre和John Jumper。利用推测采样加速大语言模型解码，2023年。网址https://arxiv.org/abs/2302.01318。

Mark Chen、Jerry Tworek、Heewoo Jun、Qiming Yuan、Henrique Ponde de Oliveira Pinto、Jared Kaplan、Harri Edwards、Yuri Burda、Nicholas Joseph、Greg Brockman、Alex Ray、Raul Puri、Gretchen Krueger、Michael Petrov、Heidy Khlaaf、Girish Sastry、Pamela Mishkin、Brooke Chan、Scott Gray、Nick Ryder、Mikhail Pavlov、Alethea Power、Lukasz Kaiser、Mohammad Bavarian、ClemensWinter、Philippe Tillet、FelipePetroski Such、Dave Cummings、Matthias Plappert、Fotios Chantzis、Elizabeth Barnes、Ariel Herbert-Voss、William Hebgen

Guss、Alex Nichol、Alex Paino、Nikolas Tezak、Jie Tang、Igor Babuschkin、Suchir Balaji、Shantanu Jain、William Saunders、Christopher Hesse、Andrew N. Carr、Jan Leike、Josh Achiam、Vedant Misra、Evan Morikawa、Alec Radford、Matthew Knight、Miles Brundage、Mira Murati、Katie Mayer、Peter Welinder、Bob McGrew、Dario Amodei、Sam McCandlish、Ilya Sutskever 和 Wojciech Zaremba。《在代码上训练的大型语言模型的评估》，2021。网址https://arxiv.org/abs/2107.03374。

Wei-Lin Chiang、Lianmin Zheng、Ying Sheng、Anastasios Nikolas Angelopoulos、Tianle Li、Dacheng Li、Banghua Zhu、Hao Zhang、Michael Jordan、Joseph E Gonzalez 等。Chatbot Arena：一个通过人类偏好评估大语言模型的开放平台。载于第四十一届国际机器学习会议，2024。

Karl Cobbe、Vineet Kosaraju、Mohammad Bavarian、Mark Chen、Heewoo Jun、Lukasz Kaiser、Matthias Plappert、Jerry Tworek、Jacob Hilton、Reiichiro Nakano、Christopher Hesse 和 John Schulman。训练验证器以解决数学文字题，2021。网址https://arxiv.org/abs/2110.14168。

Mostafa Elhoushi、Akshat Shrivastava、Diana Liskovich、Basil Hosmer、Bram Wasti、Liangzhen Lai、Anas Mahmoud、Bilge Acun、Saurabh Agarwal、Ahmed Roman 等。Layerskip：实现提前退出推理和自推测解码。载于计算语言学协会第*62*届年会论文集（第*1*卷：长篇论文），第12622–12642页，2024。

Yichao Fu、Peter Bailis、Ion Stoica 和 Hao Zhang。使用前瞻解码打破大语言模型推理的顺序依赖。载于国际机器学习会议，第14060–14079页。PMLR，2024。

Fabian Gloeckle、Badr Youbi Idrissi、Baptiste Rozi`ere、David Lopez-Paz 和 Gabriel Synnaeve。通过多标记预测实现更好、更快的大语言模型。*arXiv*预印本 *arXiv:2404.19737*，2024。

Daya Guo、Qihao Zhu、Dejian Yang、Zhenda Xie、Kai Dong、Wentao Zhang、Guanting Chen、Xiao Bi、Yu Wu、YK Li 等。DeepSeek-Coder：当大语言模型遇见编程——代码智能的兴起。*arXiv*预印本 *arXiv:2401.14196*，2024。

Daya Guo、Dejian Yang、Haowei Zhang、Junxiao Song、Ruoyu Zhang、Runxin Xu、Qihao Zhu、Shirong Ma、Peiyi Wang、Xiao Bi 等。Deepseek-r1：通过强化学习激励大语言模型的推理能力。*arXiv*预印本 *arXiv:2501.12948*，2025年。

Zhenyu He、Zexuan Zhong、Tianle Cai、Jason Lee 和 Di He。《REST：基于检索的推测解码》。载于 Kevin Duh、Helena Gomez 和 Steven Bethard（编），《*2024*年北美计算语言学协会年会论文集：人类语言技术》（第*1*卷：长篇论文），第 1582–1595 页，墨西哥城，墨西哥，2024。计算语言学协会。网址https://aclanthology.org/2024.naacl-long.88。

Kaixuan Huang、Xudong Guo 和 Mengdi Wang。Specdec++：通过自适应候选长度提升推测解码。载于基础模型高效系统研讨会 *II@ ICML 2024*，2024年。

Feiye Huo、Jianchao Tan、Kefeng Zhang、Xunliang Cai 和 Shengli Sun。C2t：一种基于分类器的推测解码树构建方法。*arXiv* 预印本 *arXiv:2502.13652*，2025年。

Sukjun Hwang、Aakash Sunil Lahoti、Ratish Puduppully、Tri Dao 和 Albert Gu。Hydra：通过广义矩阵混合器实现的双向状态空间模型。神经信息处理系统进展，37:110876–110908，2024。

Andrei Ivanov、Nikoli Dryden、Tal Ben-Nun、Shigang Li 和 Torsten Hoefler。数据移动就是你所需要的一切：优化 Transformer 的案例研究。机器学习与系统会议论文集，3:711–732，2021。

Carlos E Jimenez、John Yang、Alexander Wettig、Shunyu Yao、Kexin Pei、Ofir Press 和 Karthik R Narasimhan。SWE-bench：语言模型能否解决现实世界中的 GitHub问题？发表于第十二届国际学习表征会议，2024。网址https://openreview.net/forum?id=VTF8yNQM66。

Joao Gante。辅助生成：迈向低延迟文本生成的新方向，2023年。网址 https://huggingface.co/blog/assisted-generation。

Yaniv Leviathan、Matan Kalman 和 Yossi Matias。通过推测解码实现 Transformer 的快速推理。载于 Andreas Krause、Emma Brunskill、Kyunghyun Cho、Barbara Engelhardt、Sivan Sabato 和 Jonathan Scarlett（编），第四十届国际机器学习会议论文集，第 202 卷，机器学习研究论文集，第 19274–19286 页。PMLR，2023年7月23日至29日。网址 https://proceedings.mlr.press/v202/leviathan23a.html。

Yuhui Li、Fangyun Wei、Chao Zhang 和 Hongyang Zhang。Eagle-2：利用动态草稿树加速语言模型推理。载于*2024*年自然语言处理实证方法会议论文集，第 7421–7432 页，2024a。

Yuhui Li、Fangyun Wei、Chao Zhang 和 Hongyang Zhang。《EAGLE：推测采样需要重新思考特征不确定性》。载于 Ruslan Salakhutdinov、Zico Kolter、Kather- ine Heller、Adrian Weller、Nuria Oliver、Jonathan Scarlett 和 Felix Berkenkamp（编），第四十一届国际机器学习会议论文集，第 235 卷，机器学习研究论文集，第 28935–28948 页。PMLR，2024年7月21日至27日，2024b。网址：https://proceedings.mlr.press/v235/li24bt.html。

Fangcheng Liu、Yehui Tang、Zhenhua Liu、Yunsheng Ni、Duyu Tang、Kai Han 和 Yunhe Wang。《Kangaroo：通过双重提前退出实现大语言模型的加速无损自推测解码》。载于 神经信息处理系统进展，2024。网址：https://openreview.net/forum?id=lT3oc04mDp。

Tianyu Liu、Yun Li、Qitan Lv、Kai Liu、Jianchen Zhu、Winston Hu 和 Xiao Sun。PEARL：具有自适应草稿长度的并行推测解码。发表于 第十三届国际学习表征会议，2025年。网址https://openreview.net/forum?id=QOXrVMiHGK。

Minh-Thang Luong、Dawsen Hwang、Hoang H Nguyen、Golnaz Ghiasi、Yuri Chervonyi、Insuk Seo、Junsu Kim、Garrett Bingham、Jonathan Lee、Swaroop Mishra 等。迈向鲁棒数学推理。发表于《2025年自然语言处理实证方法会议论文集》，第 35406–35430 页，2025年。

Jonathan Mamou、Oren Pereg、Daniel Korat、Moshe Berchansky、Nadav Ti mor、Moshe Wasserblat 和 Roy Schwartz。动态推测前瞻加速大语言模型的推测解码。发表于*NeurIPS* 高效自然语言与语音处理研讨会，第 456–467 页。PMLR，2024。

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhe ngxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi 等。Specinfer：通过基于树的推测推理与验证加速大语言模型服务。载于第 *29* 届 *ACM* 编程语言与操作系统架构支持国际会议论文集，第 *3* 卷，第 932–949 页，2024。

Ernesto Quevedo、Jorge Yero、Rachel Koerner、Pablo Rivas 和 Tomas Cerny。检测大语言模型生成中的幻觉：一种词元概率方法。*arXiv* 预印本 *arXiv:2405.19648*，2024。

Andrea Santilli、Silvio Severino、Emilian Postolache、Valentino Maiorca、Michele Mancusi、Riccardo Marin 和 Emanuele Rodola。通过并行解码加速翻译的 Transformer 推理。载于 Anna Rogers、Jordan Boyd-Graber 和 Naoaki Okazaki（编），计算语言学协会第*61*届年会论文集（第*1*卷：长篇论文），第12336–12355页，多伦多，加拿大，2023年。计算语言学协会。DOI：10.18653/v1/2023.acl-long.689。网址https://aclanthology.org/2023.acl-long.689。

Abigail See、Peter J Liu 和 Christopher D Manning。直击要点：使用指针生成网络进行摘要。*arXiv*预印本 *arXiv:1704.04368*，2017年。

Noam Shazeer。《快速 Transformer 解码：只需一个写入头》。*arXiv* 预印本 *arXiv:1911.02150*，2019年。

Benjamin Spector 和 Chris Re。通过分阶段推测解码加速大语言模型推理，2023年。网址 https://arxiv.org/abs/2308.04623。

Mitchell Stern、Noam Shazeer 和 Jakob Uszkoreit。《深度自回归模型的分块并行解码》。载于 Samy Bengio、Hanna M. Wallach、Hugo Larochelle、Kristen Grauman、Nicol`o Cesa-Bianchi 和 Roman Garnett（编），《神经信息处理系统进展》第*31*卷：神经信息处理系统年度会议 *2018，NeurIPS 2018，2018*年*12*月*3*日至*8*日，蒙特´利尔，加拿大，第10107–10116页，2018年。网址 https://proceedings.neurips.cc/paper/2018/hash/c4127b9194fe8562c64dc0f5bf2c93bc-Abstract.html。

Ziteng Sun、Ananda Theertha Suresh、Jae Hun Ro、Ahmad Beirami、Himanshu Jain 和 Fe- lix X. Yu。Spectr：通过最优传输实现快速推测解码。载于 Alice Oh、Tristan Nau- mann、Amir Globerson、Kate Saenko、Moritz Hardt 和 Sergey Levine（编），神经信息处理系统进展，2023年。网址 http://papers.nips.cc/paper files/paper/2023/hash/6034a661584af6c28fd97a6f23e56c0a-Abstract-Conference.html。

Hugo Touvron、Louis Martin、Kevin Stone、Peter Albert、Amjad Almahairi、Yasmine Babaei、Nikolay Bashlykov、Soumya Batra、Prajjwal Bhargava、Shruti Bhosale、Dan Bikel、Lukas Blecher、Cristian Canton Ferrer、Moya Chen、Guillem Cucurull、David Esiobu、Jude Fernandes、Jeremy Fu、Wenyin Fu、Brian Fuller、Cynthia Gao、Vedanuj Goswami、Naman Goyal、Anthony Hartshorn、Saghar Hosseini、Rui Hou、Hakan Inan、Marcin Kardas、Viktor Kerkez、Madian Khabsa、Isabel Kloumann、Artem Korenev、Punit Singh Koura、Marie-Anne Lachaux、Thibaut Lavril、Jenya Lee、Diana Liskovich、Yinghai Lu、Yuning Mao、Xavier Martinet、Todor Mihaylov、Pushkar Mishra、I gor Molybog、Yixin Nie、Andrew Poulton、Jeremy Reizenstein、Rashi Rungta、Kalyan Saladi、Alan Schelten、Ruan Silva、Eric Michael Smith、Ranjan Subramanian、Xiaoqing Ellen Tan、Binh Tang、Ross Taylor、Adina Williams、Jian Xiang Kuan、Puxin Xu、Zheng Yan、Iliyan Zarov、Yuchen Zhang、Angela Fan、Melanie Kambadur、Sharan Narang、Aurelien Rodriguez、Robert Stojnic、Sergey Edunov 和 Thomas Scialom。Llama 2：开放基础模型与微调聊天模型，2023年。网址 https://arxiv.org/abs/2307.09288。

Simon Valentin、Jinmiao Fu、Gianluca Detommaso、ShaoyuanXu、Giovanni Zappella 和 Bryan Wang。面向大语言模型的高性价比幻觉检测。*arXiv* 预印本 *arXiv:2407.21424*，2024。

Jikai Wang、Yi Su、Juntao Li、Qingrong Xia、Zi Ye、Xinyu Duan、Zhefeng Wang 和 Min Zhang。Opt-tree：采用自适应草稿树结构的推测解码。计算语言学协会汇刊，13：188–199，2025年。

Heming Xia、Tao Ge、Peiyi Wang、Si-Qing Chen、Furu Wei 和 Zhifang Sui。推测解码：利用推测执行加速序列到序列生成。载于 Houda Bouamor、Juan Pino 和 Kalika Bali（编），计算语言学协会论文集发现：*EMNLP 2023*，第 3909–3925 页，新加坡，2023年。计算语言学协会。DOI：10.18653/v1/2023.findings-emnlp.257。网址 https://aclanthology.org/2023.findings-emnlp.257。

Heming Xia、Zhe Yang、Qingxiu Dong、Peiyi Wang、Yongqi Li、Tao Ge、Tianyu Liu、Wenjie Li 和 Zhifang Sui。《提升大语言模型推理效率：推测解码综合综述》。载于 Lun-Wei Ku、Andre Martins 和 Vivek Srikumar（编），计算语言学协会论文集发现 *ACL 2024*，第 7655–7671 页，泰国曼谷及线上会议，2024年8月。计算语言学协会。DOI：10.18653/v1/2024.findings-acl.456。网址https://aclanthology.org/2024.findings-acl.456。

Heming Xia、Zhe Yang、Qingxiu Dong、Peiyi Wang、Yongqi Li、Tao Ge、Tianyu Liu、Wenjie Li 和 Zhifang Sui。《提升大语言模型推理效率：推测解码综合综述》。载于 Lun-Wei Ku、Andre Martins 和 Vivek Srikumar（编），计算语言学协会论文集发现 *ACL 2024*，第 7655–7671 页，泰国曼谷及线上会议，2024年8月。计算语言学协会。DOI: 10.18653/v1/2024.findings-acl.456。网址https://aclanthology.org/2024.findings-acl.456。

Zilin Xiao、Hongming Zhang、Tao Ge、Siru Ouyang、Vicente Ordonez 和 Dong Yu。Parallelspec：面向高效推测解码的并行起草器。*arXiv* 预印本 *arXiv:2410.05589*，2024。

An Yang、Baosong Yang、Beichen Zhang、Binyuan Hui、Bo Zheng、Bowen Yu、Chengyuan Li、Dayiheng Liu、Fei Huang、Haoran Wei 等。通义千问2.5技术报告。*arXiv* 预印本 *arXiv:2412.15115*，2024。

An Yang、Anfeng Li、Baosong Yang、Beichen Zhang、Binyuan Hui、Bo Zheng、Bowen Yu、Chang Gao、Chengen Huang、Chenxu Lv 等。Qwen3 技术报告。*arXiv* 预印本 *arXiv:2505.09388*，2025年。

Seongjun Yang、Gibbeum Lee、Jaewoong Cho、Dimitris Papailiopoulos 和 Kangwook Lee。预测式流水线解码：实现精确大语言模型解码的计算与延迟权衡。发表于 基础模型高效系统研讨会 @ *ICML2023*，2023年。网址 https://openreview.net/forum?id=xK9FnwDMZp。

Hanling Yi、Feng Lin、Hongbin Li、Ning Peiyang、Xiaotian Yu 和 Rong Xiao。生成与验证相遇：利用智能并行自动纠错解码加速大语言模型推理。编者：Lun-Wei Ku、Andre Martins 和 Vivek Srikumar，计算语言学协会论文集发现：*ACL 2024*，第 5285–5299 页，泰国曼谷，2024年8月。计算语言学协会。DOI：10.18653/v1/2024. findings-acl.313。网址 https://aclanthology.org/2024.findings-acl.313/。

Jun Zhang、Jue Wang、Huan Li、Lidan Shou、Ke Chen、Gang Chen 和 Sharad Mehrotra。草稿与验证：通过自推测解码实现无损大语言模型加速。载于 Lun-Wei Ku、Andre Martins 和 Vivek Srikumar（编），第*62*届计算语言学协会年会论文集（第*1*卷：长篇论文），第11263–11282页，泰国曼谷，2024年8月a。计算语言学协会。DOI: 10.18653/v1/2024.acl-long.607。网址：https://aclanthology.org/2024.acl-long.607。

Ziyin Zhang、Jiahao Xu、Tian Liang、Xingyu Chen、Zhiwei He、Rui Wang 和 Zhaopeng Tu。草稿模型知道何时停止：推测解码的自验证长度策略。*arXiv* 预印本 *arXiv:2411.18462*，2024b。

Weilin Zhao、Yuxiang Huang、Xu Han、Wang Xu、Chaojun Xiao、Xinrong Zhang、Yewei Fang、Kaihuo Zhang、Zhiyuan Liu 和 Maosong Sun。《Ouroboros：逐短语生成更长的草稿，以加快推测解码》。载于*2024*年自然语言处理实证方法会议论文集，第 13378–13393 页，2024 年。

Yongchao Zhou、Kaifeng Lyu、Ankit Singh Rawat、Aditya Krishna Menon、Afshin Ros- tamizadeh、Sanjiv Kumar、Jean-Franc,ois Kagy 和 Rishabh Agarwal。DistillSpec：通过知识蒸馏改进推测解码。载于 第十二届国际学习表征会议，2024 年。网址：https://openreview.net/forum?id=rsY6J3ZaTF。

一种解码算法

算法 1 使用 PACER 的推测解码

要求： 草稿模型 M_D、目标模型 M_T、预验证层 M_B、输入前缀 $\mathbf{x}$，
最大词元长度 L，块大小 b，初始阈值 t，增长因子 ρ

当len($\mathbf{x}$) $< L$ **时执行**
  ▷ 自适应草稿生成
  $\mathbf{y}, \mathbf{q} \leftarrow [], []$ ▷ 草稿令牌和草稿概率
  $i \leftarrow 1$
  当True**时执行**
    $(y_1, q_1, h_1), \ldots, (y_b, q_b, h_b) \leftarrow M_D^b(\mathbf{x})$
    $e_1, \ldots, e_b \leftarrow$ 位置嵌入$(i, \ldots, i+b-1)$
    ▷ 使用过去的 KV 缓存进行分块预验证
    $\hat{\alpha}_1, \ldots, \hat{\alpha}_b \leftarrow M_B([h_1 + e_1], \ldots, [h_b + e_b])$
    $i \leftarrow i + b$
    $\mathbf{x} \leftarrow \mathbf{x} + [y_1, \ldots, y_b]$
    $\mathbf{y}$.追加$([y_1, \ldots, y_b])$
    $\mathbf{q}$.追加$([q_1, \ldots, q_b])$
    如果 均值$(\hat{\alpha}_1, \ldots, \hat{\alpha}_b) \leq t$ **则**
      跳出
    结束如果
    $t \leftarrow t \cdot \rho$
  结束 while
  $\mathbf{x}, \mathbf{y} \leftarrow \mathbf{x}$ ▷ 拆分以获取原始输入前缀
  $y_1, \ldots, y_\gamma \leftarrow \mathbf{y}$
  $q_1, \ldots, q_\gamma \leftarrow \mathbf{q}$
  ▷ 目标模型验证
  $p_1, \ldots, p_{\gamma+1} \leftarrow M_T(y_1, \ldots, y_\gamma; \mathbf{x})$
  采样 $r_1, \ldots, r_\gamma \sim U(0, 1)$
  $n \leftarrow$ 最小$\left(\{i-1 \mid 1 \leq i \leq \gamma, r_i > \frac{p_i(y_i)}{q_i(y_i)}\} \cup \{\gamma\}\right)$
  ▷ 如果草稿被拒绝，则调整目标分布
  如果$n < \gamma$**则**
    × 拒绝草稿 token y_n
    $p' \leftarrow$ 范数(最大值$(0, p_{n+1} - q_{n+1})$)
  else
    ✓接受所有草稿令牌
    $p' \leftarrow p_{n+1}$
  结束如果
  采样下一个词元 $t \sim p'$
  $\mathbf{x} \leftarrow \mathbf{x} + [y_1, \ldots, y_n, t]$
结束 while

B 案例研究

我们提供了一个示例，用于比较 PACER 与标准推测解码的差异，如图 4所示。在此示例中，配置 (a) 使用固定窗口大小 $\gamma = 6$，而配置 (b) 应用具有动态草稿窗口且块大小为$b = 3$的 PACER。在固定窗口设置下，第一轮中全部六个草稿令牌均被接受；然而在第二轮中，全部六个草稿令牌均被拒绝，导致不必要的草稿计算，并且总共需要三次目标模型前向传播。相比之下，当草稿预测变得不可靠时，PACER 会提前停止，避免无效草稿，仅通过两次目标前向传播完成解码过程。延迟分解清楚地展示了自适应草稿控制的优势：在此示例中，PACER 将总解码时间从 498.9 ms 降至 339.8 ms，凸显了其在降低计算开销和提升端到端效率方面的有效性。

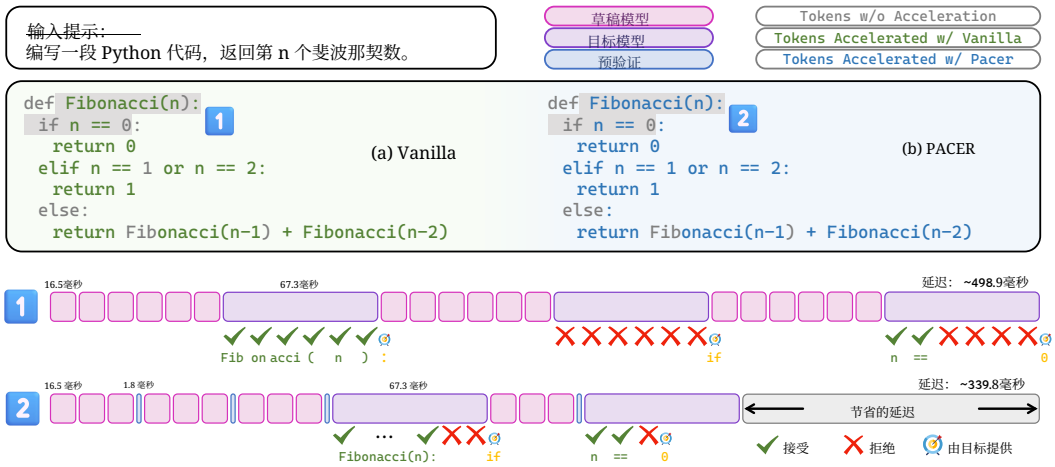


图 4：原始推测解码采用固定窗口大小（$\gamma = 6$）与 PACER 采用块大小为 $b = 3$的对比示意图。图的上半部分展示了各方法实现的加速令牌生成效果。下半部分展示了生成代码片段 `Fibonacci(n): if n == 0` 时的延迟分解

C 实验详情

C.1 评估设置

数据集。我们在多种具有代表性的文本生成任务上评估我们的方法，包括用于代码生成的 HumanEval（Chen 等人，2021）和 MBPP（Austin 等人，2021）、用于算术推理的 GSM8K（Cobbe 等人，2021），以及用于文档摘要的 CNN/DailyMail（CNN/DM）（See 等人，2017年）。HumanEval 包含 164 个手工编写的示例，每个示例由一段文本提示和一个 Python 函数前缀组成。MBPP 最初包含 500 道多样化的编程题及相应的测试用例；我们随机抽取 150 道题进行评估。CNN/DM 数据集包含与人工撰写摘要配对的新闻文章，而 GSM8K 则包含小学水平的数学文字题。对于 CNN/DM 和 GSM8K，我们按照（Zhao 等人，2024）随机抽取 100 个示例进行评估。

硬件设置。为验证我们的方法在资源受限和资源充足两种条件下的有效性，DeepSeek-Coder 1.3B/33B 和通义千问-2.5 1.5B/32B 的实验均在 **1 × NVIDIA 80GB A800 GPU** 上进行；DeepSeek-Coder 6.7B/33B 的实验在 **2 × NVIDIA 80GB A800 GPU** 上进行；Llama2-chat 7B/70B 的实验则在 **4 × NVIDIA80GB A800GPU** 上进行。

推测解码。在第4节中，我们报告了原始推测解码在各种固定窗口大小下取得的最佳解码速度所对应的性能。完整结果见第F.3节。

C.2 训练参数

对于每个模型系列，我们都会独立训练一个分块预验证层。我们使用 AdamW 优化器，其参数为 $(\beta_1, \beta_2) = (0.9, 0.999)$。所有训练均在 **8× NVIDIA A800 GPU** 上进行。表 9 总结了模型规模和训练配置。根据模型大小的不同，训练时间为 18 至 47 分钟。

模型系列	预验证层大小	学习率	训练轮数
DeepSeek-Coder 1.3B/33B	0.18B	5e-4	5
DeepSeek-Coder 6.6B/33B	0.47B	2e-4	5
Qwen2.5 1.5B/32B	0.28B	5e-4	5
LLama2-Chat 7B/70B	0.46B	1e-4	5

表 9：不同模型系列中 PACER 的训练超参数。

C.3 评估参数

我们实验的超参数如表 10 所示。

Task	模型	块大小 b	阈值 t	增长因子 ρ
HumanEval	Deepseek 1.3B/33B	4	0.70	1.05
	Deepseek 6.7B/33B	4	0.70	1.05
	LLaMA2 7B/70B	4	0.65	1.02
MBPP	Deepseek 1.3B/33B	3	0.70	1.05
	Deepseek 6.7B/33B	3	0.70	1.05
	LLaMA2 7B/70B	3	0.70	1.05
CNN/DM	Deepseek 1.3B/33B	2	0.60	1.05
	Deepseek 6.7B/33B	2	0.60	1.05
	LLaMA2 7B/70B	3	0.65	1.05
GSM8K	Deepseek 1.3B/33B	4	0.65	1.05
	Deepseek 6.7B/33B	3	0.65	1.05
	LLaMA2 7B/70B	3	0.65	1.05

表 10：在我们的实验中，PACER 在不同任务和模型中使用的块大小 b、阈值 t 和增长因子 ρ。

D 训练细节

如第3.3节所述，我们采用与推理精确对齐的相同推测解码（SD）流程来构造带标签的训练数据。在此设置中，每个草稿步骤都会生成一个训练样本，其中包含输入前缀，以及标记为接受或拒绝的草稿 token。然而，这种方法效率很低，因为它会产生约 **Len(dataset)** × **Average draft steps** 个训练样本。

我们发现，来自同一提示和响应的许多草稿步骤共享相同的前缀，造成了冗余。为解决这一问题，我们提出将对应于同一提示的所有草稿步骤的训练数据打包成一个训练序列。训练期间，我们使用定制注意力掩码，如图5所示。具体而言，草稿 token d_i^i、……、$d^\gamma{}_i$ 表示从前缀x_1, x_2、……、y_{i-1}开始的一个草稿步骤，随后是γ 个按顺序生成、与响应位置i、……、 γ对齐的草稿。每个草稿 token d_i^m只能关注前缀x_1, x_2、……、y_{i-1} 以及前面的草稿 token d_i^i、……、d_i^m。

E 额外消融研究

我们使用 HumanEval数据集和 DeepSeek-Coder 1.3B/33B，对块大小 b、阈值 t和增长因子 ρ 进行了额外消融研究。我们报告了每种配置的解码速度，单位为每秒词元数（tokens/s）。

块大小 b。 如表11所示，当块大小 b 过小时，误差的累积可能导致预验证提前终止。反之，当块大小 b 过大时，模型可能无法通过预验证触发提前终止，进而因生成过多草稿而产生不必要的计算开销。

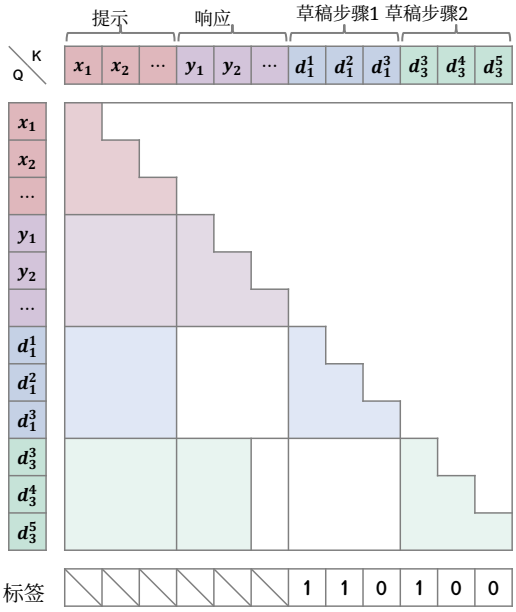


图5：用于将多个草稿步骤打包到单个序列中以实现高效训练的自定义注意力掩码。

b	1	2	3	4	5	6	7
令牌/秒	30.00	38.07	39.28	41.80	38.95	38.85	37.21

表 11：关于块大小 b 的消融实验，其中 $b \in [1,7]$

增长因子 ρ。我们观察到，大多数训练样本都涉及较短的草稿，因此预验证层难以很好地泛化到较长的位置。在这种情况下，随着位置增加，模型往往会低估接受率。我们通过消融研究证明，增长因子对解码速度有积极影响。如表 12所示，启发式增长因子（例如 1.05）作为一种轻量级归纳偏置，可鼓励随时间推移进行更积极的草拟，同时避免草稿长度不受限制。

ρ	1.00	1.01	1.02	1.03	1.04	1.05	1.06	1.07	1.08	1.09
令牌/秒	40.05	40.12	40.40	40.57	40.73	41.80	41.60	40.52	39.48	39.37

表 12：关于增长因子 ρ 的消融实验，其中 $\rho \in [1.00、1.09]$

注。针对 $\rho = 1.00$ 在表 12中以及针对 PACER 无增长因子在主论文消融研究中的结果在理论上等价，但在实现上略有差异。对于 PACER 无增长因子，阈值在解码过程中保持不变。对于表 12($\rho = 1.00$)，阈值乘以 $\rho = 1.00$（每个步骤）。尽管这种缩放不应产生任何影响，两种设置之间细微的控制流或系统级优化仍会导致轻微差异。

阈值t。我们在不同阈值t设置下评估了一系列预验证层，如表13所示。阈值设为 0.7 时，整体性能最佳。阈值过高可能导致预验证提前终止，而阈值过低则会因生成不必要的草稿令牌而产生额外开销。

t	0.50	0.55	0.60	0.65	0.70	0.75	0.80
令牌/秒	37.64	38.09	40.34	39.89	41.80	40.44	39.93

表 13：阈值消融实验 t 其中 $t \in [0.50, 0.80]$

F 补充结果

F.1 前向延迟

如表14所示，分块预验证层、草稿模型和目标模型（DeepSeek-Coder 1.3B/33B）的平均前向传播延迟分别为 1.81 ms、16.52 ms 和 67.31 ms。分块预验证带来的额外开销较小，与草稿模型前向传播相比约为 0。11×，与目标模型前向传播相比约为 0。027×。

模型 / 层	**延迟（毫秒）**
分块预验证层	1.81
DeepSeek-Coder 1.3B	16.52
DeepSeek-Coder 33B	67.31

表14：使用 PACER 时，DeepSeek-Coder 1.3B/33B 中不同模型/层单次前向传播的延迟。

F.2 PACER 运行时间分解

我们报告了DeepSeek-Coder 1.3B/33B与Llama-2 7B/70B模型在HumanEval数据集上各组件的实际运行时分解，汇总于表15。如图所示，预验证层仅分别贡献**2.10%** 和**1.30%** 的总推理时间。鉴于通过动态草稿长度取得的显著加速比（72秒和221秒），该开销在实践中较为适中。

组件	**DeepSeek-Coder 1.3B/33B**		**Llama-2 7B/70B**	
	时间（秒）	**占比（%）**	**时间（秒）**	**占比（%）**
预验证层	12.86	**2.10**	22.80	**1.30**
草稿模型	434.55	71.24	1080.06	61.51
目标模型	162.59	26.66	653.14	37.19
总计（PACER）	**610**	**100**	**1756**	**100**
总计（推测）	682	—	1977	—

表15：DeepSeek-Coder 和 Llama-2 模型的时间与比例分析。

F.3 推测解码的详细结果

模型系列	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
DeepSeek-Coder 1.3B/33B	–	–	–	34.99	36.11	37.27	37.41	36.69	**37.46**	37.11
DeepSeek-Coder 6.7B/33B	26.83	28.81	29.35	29.57	**29.77**	29.31	29.70	29.49	29.41	–
Qwen2.5 1.5B/32B	24.07	27.29	28.14	**28.46**	27.78	27.77	27.35	26.97	26.32	–
LLaMA2 7B/70B	–	–	–	18.59	20.95	21.15	21.21	**21.23**	21.10	21.12

表16：不同模型系列在不同固定窗口大小（$\gamma = 1$ 到 $\gamma = 10$）下的推测解码速度（令牌/秒）。

对于原始推测解码，我们测试了不同数量的草稿令牌，如表16所示，并选择最优结果作为我们方法的基线。

F.4 不同草稿位置的接受率

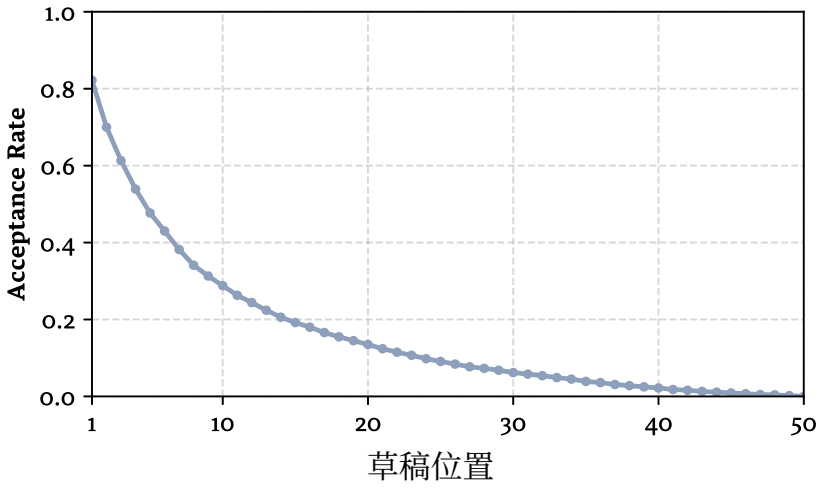


图6：不同草稿位置下的接受率。

G 局限性

尽管所提出的 PACER 能够有效地动态控制草稿长度并加速推测解码，但仍存在一定局限性。其性能取决于草稿模型所生成草稿的质量。对于代码生成等任务（例如 HumanEval），较长的平均接受长度可带来显著加速；然而，对于摘要任务（例如 CNN/DM），最优草稿往往较短，从而限制了潜在的性能提升。同样，当草稿模型和目标模型都相对较小时，最优草稿长度较短，预验证层的相对开销可能更加明显。尽管如此，PACER 能够补充旨在提升草稿质量的推测解码技术。将 PACER 与 Ouroboros（Zhao 等人，2024）相结合，可进一步将解码速度最高提升至 3.09× 倍（相比自回归解码）。我们预计，未来的推测解码研究将持续提升草稿质量。随着这些进展，我们的方法有望取得更为显著的性能提升。