清华大学学报（自然科学版）
J Tsinghua Univ (Sci & Technol)

大模型的推测解码

算法小小怪下士

2025 年 12 月 19 日

摘要

LLM 缓慢的推理速度成为其实际应用的主要瓶颈。这一问题的根源在于自回归解码方式，每个新 token 的生成都严格依赖于前面所有标记，形成了一种串行依赖关系。从计算角度来看，这种解码方式导致每次前向传播只能生成一个标记，而每次前向传播都需要将完整的模型参数从内存加载到计算单元，使得整个推理过程严重受限于内存带宽而非计算能力本身。为了解决这一根本性挑战，Speculative Decoding 应运而生。其核心思路是" 先推测后验证"——使用小型、快速的" 草稿模型" 先生成一批可能的标记序列，然后让大型" 目标模型" 并行验证这些候选序列。本文主要介绍了经典的 Speculative Decoding，以及之后的针对 draft model 的改进方案。

1 基于草稿模型的经典推测解码

这种 [5] 算法很经典，如算法 1 所示。可以分为两步。首先，draft model 在已有提示序列下，自回归生成 K（前瞻步数）个 token。然后，target model 并行计算 K+1 组 logits，并进行拒绝采样。

1.1 为什么要计算 K+1 组 logits

前 K 组 logits 是 target model 检查 draft model 生成的 token 是否合理，最后一个 logits 是 target model 直接告诉 draft model 下一个 token，因为是并行解码，所以多算一个 token 也不是个事情，其次是 target model 告诉的 token，我们可以默认准确率更高，从而让并行解码的稳定性也提升了。这就好比老师批改作业：老师看你写的 K 道题，判断对错。K+1 个：老师看你 K 道题都做对了，直接把下一道题的答案也告诉你。这样你就省去了一次举手问老师的时间。

这种算法很简洁性，但是整个系统的加速效果完全维系于标记接受率。当草稿模型能够精准揣摩目标模型的“意图”时，系统便能行云流水般地快速推进；一旦二者判断出现分歧，系统不仅无法提速，额外的验证开销反而会令推理过程“雪上加霜”，造成整体延迟不降反升。

其次，引入草稿模型绝非简单的“1+1”。在分布式推理环境中，它意味着需要同时调度、维护两个模型，并妥善处理它们之间的资源分配、负载均衡与通信开销。这显著增加了系统架构的复杂性，对推理服务的工程实现提出了更高要求。

最后，对于许多主流大模型而言，并没有现成的、匹配良好的草稿模型可供使用。从头训练一个专用的草稿模型不仅成本高昂，而且其泛化能力往往有限，通常只能服务于特定的目标模型或任务领域。

1.2 为什么要用拒绝采样

这里的一个核心表达式 $\min(1, \frac{q(x)}{p(x)})$。我们分类讨论一下它的含义：情况 A：大模型觉得“这个词挺好”($q(x) \geq p(x)$) 比值 ≥ 1，那么此时该表达式的值为 1，无论均匀分布采样的阈值为多少，都一定会接受这个 token，因此接受概率为 100%。为什么要这样做呢？因为小模型居然猜到了大模型也认为高概率的词，（小模型低估了这个词）。既然大模型认可小模型输出的 token，那就直接通过，省时省力。

情况 B：大模型觉得“这个词不行”($q(x) < p(x)$) 比值 (比如是 0.3)< 1。此时，小模型盲目自信，给了这个词很高的概率，但大模型觉得它只有一点点概率。在拒绝采样的过程中，我们不能直接扔掉，而是通过拒绝的阈值 r，以 30% 的概率接受它，70% 的概率拒绝它。如果拒绝了这个 token，我们会重新从 abs（大模型的分布-小模型的分布）中采样。

那么这里就有一个问题，“既然要快，为什么不直接看小模型生成的词是不是大模型的（Top-K）？如果是就接受，不是就拒绝？”如果这样做，虽然也能筛选，但你就破坏了概率分布。大模型的生成

也并不是每次都选概率最高的那个词。如果使用简单的硬规则截断，生成内容的丰富度会大大下降下降，或者分布会偏离大模型。拒绝采样是唯一能从数学上证明：

$$P(\text{最终输出}) = P(\text{大模型输出})$$

。

这就非常神奇了，因为这里不是保持均值一样，是分布完全一样，这是一个“无损”算法。拒绝采样这里是把不符合目标分布的样本直接扔掉。而 RL 中的 off-policy 是把所有样本都留着，不符合的给低分，符合的给高分，它能保证均值一定，扭曲了分布，但方差却爆炸了。PPO 粗暴地 Clip 了重要性权重，从而强行限制了方差，虽然牺牲了一点点无偏性，但换来了训练的稳定。TRPO 则是通过 KL 散度约束，强行让新策略不要离旧策略太远。这里有点扯远了，但作者想到这件事情，就想感叹的就是拒绝采样这里设计的太妙了。

Algorithm 1 基于草稿模型的经典 SD

给定前瞻步数 (lookahead) K 和最小目标序列长度 T。
给定自回归目标模型 $q(\cdot|\cdot)$ 和自回归草稿模型 $p(\cdot|\cdot)$，以及初始提示序列 $x_0, \ldots, x_t$。
初始化 $n \leftarrow t$。
while $n < T$ **do**
  for $t = 1 : K$ **do**
    自回归地采样草稿 token：$\tilde{x}_t \sim p(x|x_1, \ldots, x_n, \tilde{x}_1, \ldots, \tilde{x}_{t-1})$
  end for
  并行计算基于草稿 $\tilde{x}_1, \ldots, \tilde{x}_K$ 的 $K+1$ 组 Logits：

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

  for $t = 1 : K$ **do**
    从均匀分布中采样 $r \sim U[0, 1]$。
    if $r < \min\left(1, \frac{q(x|x_1,\ldots,x_{n+t-1})}{p(x|x_1,\ldots,x_{n+t-1})}\right)$ **then**
      令 $x_{n+t} \leftarrow \tilde{x}_t$ 并且 $n \leftarrow n+1$。
    else
      重新采样 $x_{n+t} \sim (q(x|x_1, \ldots, x_{n+t-1}) - p(x|x_1, \ldots, x_{n+t-1}))_+$ 并退出 for 循环。
    end if
  end for
  如果所有 token $x_{n+1}, \ldots, x_{n+K}$ 都被接受，则利用 $q(x|x_1, \ldots, x_n, x_{n+K})$ 采样额外的 token x_{n+K+1} 并令 $n \leftarrow n+1$。
end while

2 Lookahead 解码：数值分析的 Jacobbi 大显神威

Motivation：Lookahead [3] 解码完全不需要辅助模型。它利用大模型自身来生成候选序列。通过利用 Jacobi 迭代的轨迹，它能在单次前向传播中并行生成多个不相交的 n-gram 候选项。这使得部署变得极简，无需针对特定任务重新训练或微调小模型。

Method：它引入了 Lookahead Branch 和 N-gram Pool。第一步，它利用 Jacobi 迭代的特性，同时在多个位置并行生成 token。虽然这些 token 可能位置不对，但它们构成了有效的 n-gram 片段。

第二步，它将生成的 n-gram 存入池中。在后续步骤中，算法直接从池中提取匹配当前上下文的 n-gram 进行验证，变废为宝，实现了零成本的“猜测”。

Effect：正如上面分析的，传统的 Speculative Decoding 的加速效果严格受限于“接受率”。即使拥有无限的计算资源，如果草稿模型猜不准，加速比就会遇到瓶颈，甚至在最坏情况下变慢。论文揭示了一个新的 Scaling Law：通过增加每一步的计算量，可以线性地减少解码步数。这意味着它是“面向未来”的算法——随着 GPU 算力增强（允许更大的并行窗口 W 和 n-gram 大小 N），它的加速效果会越来越好，而不会像投机解码那样卡在某个接受率瓶颈上。

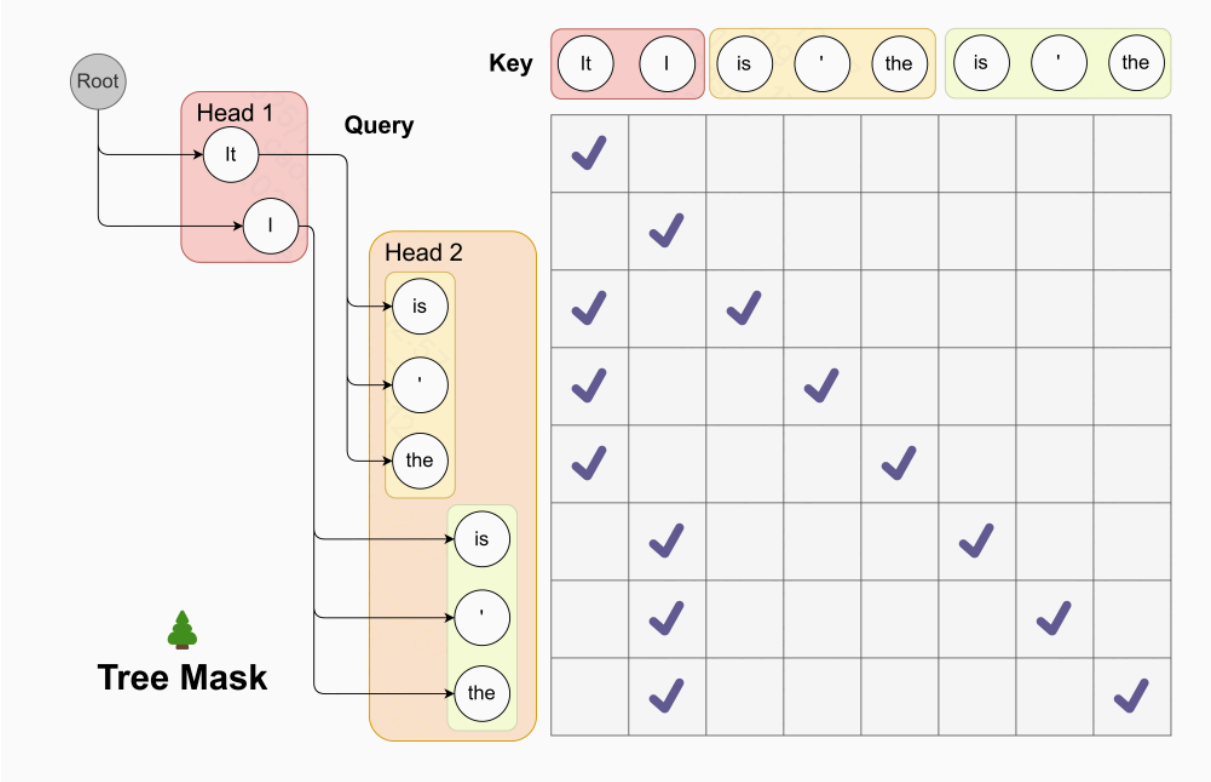

图 1 树状注意力其实就是根据输入的 draft 序列，精心设计了一下 mask。笔者写到这里非常激动，因为笔者突然想起了美团的生成式推荐工作 MTGR，它的 mask 之前让读者一直觉得很晦涩，突然发现其本质就是一个 tree attention 的 mask 的设计。在 Medusa 中，root 是所有候选路径共享的上下文（例如 Prompt），branch 是多个互不干扰的候选生成路径。分支节点可以看到根节点，但分支之间互相不可见。MTGR 中 root 是用户画像（U）和历史行为序列（$\vec{S}, \vec{R}$）组成。而 branch 就是 K 个待排序的候选商品（C, I）聚合在一个样本中。这对应于树状注意力中的多个“分支”。可见性规则是分支节点可以看到根节点，但分支之间互相不可见。两者的区别是，MTGR 的分支都是特别短，但是也可以理解成一种退化的 tree attention。通了，都通了！

论文明确指出，该方法是一种“精确的、并行的、无损的”。它虽然引入了并行猜测机制，但所有猜测出来的 n-gram 都必须经过一个严格的 Verification Branch（验证分支）才能被接受，这确保了最终输出的 token 序列与大模型自回归生成的序列完全一致。

在贪婪搜索模式下，Lookahead Decoding 的验证逻辑与 Speculative Decoding 类似。它会检查生成的 n-gram 是否与大模型在给定前缀下的输出完全匹配。如果匹配则接受，不匹配则拒绝并回退。理论上，这不会改变贪婪搜索的任何输出结果。对于非贪婪的随机采样（如 Top-K, Top-P），Lookahead Decoding 也设计了专门的验证算法来保证分布一致性。

3 Medusa：draft model 到 draft head

MEDUSA [2]：不需要独立的草稿模型。它通过在原有的 LLM 最后隐藏层上添加额外的解码头来实现预测。这些解码头是轻量级的（通常只是单层前馈网络），可以直接集成到现有模型中。

在验证时，论文构建了多个候选序列，并利用 Tree Attention 在原模型的一次前向传播中同时验证这些候选。MEDUSA 根据不同解码头的预测结果构建候选树，并通过精心设计的注意力掩码一次性处理所有分支。

在接受 draft model 的样本时，MEDUSA 提出了一种典型接受 (Typical Acceptance) 方案。它不强求严格匹配原分布，而是基于预测概率的熵和阈值来选择“合理”的候选 Token 。这种方法在保证生成质量的同时，比拒绝采样能获得更高的接受率和加速比。

4 EAGLE 框架：某种意义上的 Next latent prediction

4.1 EAGLE [8]

EAGLE 的提出是为了解决传统投机采样中“寻找合适的小模型难”以及“Token 级预测准确率低”的问题。

EAGLE 观察到，在特征层面（Target Model 倒数第二层的特征）进行自回归比直接预测 Token 更简单且更有规律。——》EAGLE 的 Draft Model 是一个轻量级的 Transformer Decoder 层（通常只有一个），它利用 Target Model 提取的顶层特征作为输入，预测下一个特征向量，再通过 Target Model 的 LM Head 映射为 Token。

在特征层面进行预测时，由于采样的随机性，同一个特征后面可能接不同的 Token（例如”I” 后面可能是”am” 也可能是”always”），导致特征序列存在不确定性。——》EAGLE 将“时间步提前一位的 Token 序列”作为 Draft Model 的额外输入。这意味着在预测第 $t+1$ 步的特征时，模型不仅知道第 t 步的特征，还知道第 t 步采样出来的具体 Token 是什么，从而消除了歧义

使用固定的树状结构来生成候选 Token，利用 Tree Attention 在一次前向传播中验证多个 Token

4.2 EAGLE-2 [6]

Motivation: EAGLE-1 使用的是静态的草稿树（无论上下文如何，树的形状和候选数量都是固定的）。但研究发现，草稿 Token 的接受率不仅与位置有关，还高度依赖于上下文

Method: Context-Aware Dynamic Draft Trees: 利用置信度估算接受率：EAGLE-2 发现 Draft Model 的置信度（Confidence Score）与实际的 Token 接受率呈强正相关。

动态调整策略：1）Expansion: 根据当前的置信度，动态选择最有希望的分支进行扩展，不再死板地填充固定位置 2）Reranking: 在扩展后，对所有生成的候选 Token 按照全局接受概率进行重排序，筛选出最优的 Token 集合组成验证树

Effect：在不重新训练模型的情况下，仅通过算法层面的改进，相比 EAGLE-1 提升了 20%-40% 的速度

4.3 EAGLE-3 [7]

Motivation: 随着训练数据的增加，EAGLE-1/2 的性能提升非常有限。原因是 EAGLE-1 强制 Draft Model 去拟合 Target Model 的特征，这限制了 Draft Model 的表达能力，使其难以从更多数据中获益。

Method: EAGLE-3 放弃了特征预测任务，Draft Model 不再需要输出“逼近 Target Model 的特征”，而是直接以预测正确的 Token 为目标。因此这一节的标题，next latent prediction 对 EAGLE-3 来说，可能没有那么合适了。

引入 Training-Time Test (TTT): 为了让 Draft Model 具备多步预测能力，EAGLE-3 在训练过程中模拟推理时的多步生成过程（即把上一轮自己生成的预测结果喂给下一轮），强制模型在误差累积的情况下依然能预测准确。

多层特征融合 (Multi-Layer Feature Fusion): 改进点：既然不再强制回归顶层特征，Draft Model 的输入就不再局限于 Target Model 的顶层特征。EAGLE-3 融合了 Target Model 的低层、中层和高层特征作为输入，从而获取更丰富的语义信息。

Effect: EAGLE-3 发现了推理加速的 Scaling Law，即随着训练数据量的增加，加速比线性增长。EAGLE-3 实现了最高 6.5 倍的加速比，比 EAGLE-2 快约 1.4 倍 15。

5 Multi-token-prediction：我们依然能看见 Speculative Decoding 的影子

在 Your LLM knows Future [9] 中，论文提出了 MTP 技术, 改方案可以看作是一种基于 Self-Speculation 的改进方案。它与之前的 speculative decoding（如 MEDUSA、EAGLE）有着紧密的继承关系，但在实现路线上选择了一条不同的分支：“Mask Token”路线。

简单来说，之前的 Medusa 和 EAGLE 是通过额外的 Heads 来预测未来；而 MTP 则是通过插入 Mask Token，强迫大模型利用自身内部的深层知识来预测未来。Medusa 的 Heads 只能看到大模型最后一层的特征，无法利用大模型内部的 Transformer 层来专门为“第 k 个词”进行推理。MTP 方法中追加的 Mask Token 会完整地流经大模型的所有 Transformer 层。这意味着 MTP 利用了整个大模型的深度和容量来推断未来，而不仅仅是最后一层。

EAGLE 也是建立一个独立的轻量级 Draft Model，它利用大模型的特征作为输入，进行特征层面的自回归。它外挂了一个小模型，主模型参数不变。而 MTP 复用主模型本身作为 Draft Model。为了不破坏主模型原本的能力，它使用了 Gated LoRA（门控低秩适应）技术。笔者以为这是非常有意思的小创新。MTP 在微调时，LoRA 参数只对 Mask Token 生效，对原始文本 Token 不生效（Gate=0）。这确保了模型既能学会预测未来（MTP 能力），又完全保留了原始的自回归能力（NTP 能力），实现了无损集成。

再扯远一点，MTP 的核心灵感实际上可以追溯到 BERT 时代的 Masked Language Modeling ，但将其适配到了生成式任务中。传统 MLM: 随机 Mask 句子中间的词，预测被遮挡的词。MTP 的 MLM: 在句子末尾追加 Mask，预测未来的词。论文发现，预训练好的自回归模型其实已经“潜意识”里知道未来几个词是什么，只是平时没地方输出。MTP 就是通过 Mask Token 给模型一个“接口”，让它把这些潜在的知识吐露出来。

在验证时，MTP 提出 Quadratic Decoding (二次解码)，这实际上就是 Tree Attention 的一种实现形式。做法：通过构造特定的 Attention Mask，让模型在验证步骤中，既能验证上一轮的猜测，又能同时为下一轮生成新的猜测（Mask Token 再次生效），保证每一步都能产生 k 个新的候选。

6 企业部署 Speculative Decoding

6.1 纯正的大模型应用

对于传统的大模型应用技术岗，笔者认为应该考虑自己的任务到底是偏向高吞吐，还是偏向低延迟，从而针对性的设计 Speculative Decoding。

- **低延迟场景：**例如实时对话（Chatbot）、实时代码补全（Code Completion）等在线服务。在这类场景下，Batch Size 通常较小，GPU 主要处于 **Memory-Bandwidth Bound**（显存带宽受限）状态，计算单元存在大量闲置。此时部署 Speculative Decoding 收益最大，因为它利用闲置的计算能力进行“投机”或“前瞻”生成，能显著降低首字延迟（TTFT）和每字延迟（Tpot），且几乎不影响显存占用（特别是 Lookahead Decoding 这类无额外模型的方案）。

- **高吞吐场景：**例如离线文档分析、批量数据处理等任务。这类场景通常会把 Batch Size 打满以追求最大的吞吐量（Tokens/s）。此时 GPU 已经处于 **Compute Bound**（计算受限）状态，计算单元接近饱和。在这种情况下，引入 Speculative Decoding 可能会带来负面效果。因为投机解码引入了额外的验证（Verification）计算开销（额外 FLOPs），在计算资源本就紧张的情况下，这反而可能导致整体吞吐量下降，甚至拖慢推理速度。

因此，如果业务目标是最大化显卡利用率和吞吐量（如离线批处理），应谨慎使用 Speculative Decoding；如果业务目标是提升用户体验和响应速度（如在线服务），则该技术是打破显存墙限制的最佳方案之一。

6.2 LLM4Rec/生成式推荐

对于生成式推荐来说，之前常常关注的是 tokenization 和 Generative Model 的设计，但其实还有一个重要环节就是部署。推荐系统那些北极星指标的好坏，不仅仅取决于算法的智能，还取决于延迟等系统延迟。未来使用 Speculative Decoding 也是一个让系统更高效的一个课题。在 LLM4Rec 中，也有很多工作已经用了 Speculative Decoding 的技术，比如 NEZHA [10]。也有一些工作，用了 MTP 的技术，比如 RPG [4]，PinRec [1]。

RPG 引入 MTP 作为一个辅助任务，目的是追求效果，让模型在训练时不仅学会预测 $t+1$，还要学会预测未来更远的“关键节点”。通过预测多个未来的 items，模型能学到更好的用户表征，从而提升推荐的准确性。RPG 使用了一个 Recurrent 的结构。它将预测出来的“未来 Goal”重新作为输入，反馈给模型，用来修正当前的预测。

PinRec 引入 MTP 是为了追求效率，实现并行推理。它希望模型在一步推理中，能同时输出接下来 K 个可能的物品（比如 $t+1, t+2, ..., t+K$），从而避免反复调用模型，显著降低推理延迟。PinRec 修改了模型的输出层（Prediction Head），使其拥有 K 个并行的 Heads，这与 Medusa 非常像。

7 手撕 SD 源码

参考了下面这个仓库https://github.com/feifeibear/LLMSpeculativeSampling/blob/main/ 这里实现的是没有 kvcache，相比于原仓库的代码，本部分仅保留核心算法的代码逻辑。

```python
import torch
from sampling.utils import norm_logits, sample, max_fn

@torch.no_grad()
def speculative_sampling_v2(prefix: torch.Tensor, approx_model: torch.nn.Module, target_model: torch.nn.Module, max_len: int, gamma: int = 4,temperature: float = 1, top_k: int = 0, top_p: float = 0) -> torch.Tensor:
    """
    Standard Speculative Sampling (DeepMind version) without KV Cache.
    """
    seq_len = prefix.shape[1]
    T = seq_len + max_len

    assert prefix.shape[0] == 1, "input batch size must be 1"

    while prefix.shape[1] < T:
        x = prefix
        prefix_len = prefix.shape[1]

        # 1. Draft: Autoregressive generation by small model
        for _ in range(gamma):
            q = approx_model(x).logits
            next_tok = sample(norm_logits(q[:, -1, :], temperature, top_k, top_p))
            x = torch.cat((x, next_tok), dim=1)

        # 2. Verify: Parallel forward pass by large model
        # Calculate logits for the whole sequence at once
        q = approx_model(x).logits
        p = target_model(x).logits

        # Normalize logits
        for i in range(q.shape[1]):
            q[:, i, :] = norm_logits(q[:, i, :], temperature, top_k, top_p)
            p[:, i, :] = norm_logits(p[:, i, :], temperature, top_k, top_p)

        # 3. Rejection Sampling
        n = prefix_len - 1
        is_all_accept = True

        for i in range(gamma):
            r = torch.rand(1, device=p.device)
            j = x[:, prefix_len + i]

            # Acceptance criteria: min(1, p(x)/q(x))
            if r < torch.min(torch.tensor([1], device=q.device), p[:, prefix_len + i - 1, j] / q[:, prefix_len + i - 1, j]):
                n += 1
            else:
                # Reject and Resample from difference distribution
                t = sample(max_fn(p[:, n, :] - q[:, n, :]))
                is_all_accept = False
                break

        prefix = x[:, :n + 1]

        # 4. If all accepted, sample one extra token
        if is_all_accept:
            t = sample(p[:, -1, :])

        prefix = torch.cat((prefix, t), dim=1)

    return prefix
```

Listing 1 手撕代码

参考文献

[1] Agarwal, P., Badrinath, A., Bhasin, L., Yang, J., Botta, E., Xu, J., and Rosenberg, C. Pinrec: Outcome-conditioned, multi-token generative retrieval for industry-scale recommendation systems, 2025.

[2] Cai, T., Li, Y., Geng, Z., Peng, H., Lee, J. D., Chen, D., and Dao, T. Medusa: Simple llm inference acceleration framework with multiple decoding heads, 2024.

[3] Fu, Y., Bailis, P., Stoica, I., and Zhang, H. Break the sequential dependency of llm inference using lookahead decoding, 2024.

[4] Hou, Y., Li, J., Shin, A., Jeon, J., Santhanam, A., Shao, W., Hassani, K., Yao, N., and McAuley, J. Generating long semantic ids in parallel for recommendation, 2025.

[5] Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding, 2023.

[6] Li, Y., Wei, F., Zhang, C., and Zhang, H. Eagle-2: Faster inference of language models with dynamic draft trees, 2024.

[7] Li, Y., Wei, F., Zhang, C., and Zhang, H. Eagle-3: Scaling up inference acceleration of large language models via training-time test, 2025.

[8] Li, Y., Wei, F., Zhang, C., and Zhang, H. Eagle: Speculative sampling requires rethinking feature uncertainty, 2025.

[9] Samragh, M., Kundu, A., Harrison, D., Nishu, K., Naik, D., Cho, M., and Farajtabar, M. Your llm knows the future: Uncovering its multi-token prediction potential, 2025.

[10] Wang, Y., Zhou, S., Lu, J., Liu, Z., Liu, L., Wang, M., Zhang, W., Li, F., Su, W., Wang, P., Xu, J., and Zhao, X. Nezha: A zero-sacrifice and hyperspeed decoding architecture for generative recommendations, 2025.