推测 基于语义 ID 的生成式推荐的解码 S

Harshal A. Chaudhari [1]

arXiv:2611.32855v1 [cs.IR] 24 Nov 2026

摘要

采用语义 ID（SID）的生成式推荐系统将每个项目表示为由 L 个离散词元组成的短序列，并使用序列模型根据用户的历史记录预测下一个项目的词元。工业级部署通常使用这些模型，通过约束束搜索为每个请求检索多个候选项。然而，这会导致每个请求都需要固定进行 L 次顺序模型调用；随着 SID 生成器的区分能力减弱，准确率也会下降，而且无法保证输出与精确的 top-K 有多接近。因此，生产系统无法检测单个请求的检索质量是否下降。

我们提出 SIDECAR：它使用近似草稿候选项，并通过一次树注意力遍历，在单次模型调用中为整个草稿评分（内存允许的情况下）。这次遍历会为每个尚未探索的 SID 前缀的概率设定界；如果草稿中包含精确的 top-K，则可对其进行认证。否则，分支定界回退机制只扩展仍可能改变输出的字典树部分；无论草稿如何，最多通过 L 次调用即可完成认证。此外，在每次调用的预算为 M 个词元、调用次数为 Y 的情况下，SIDECAR 会公布一个因子 $1+\delta$，保证列表中每个排名位置的结果都将在该因子范围内精确；其中，对于 $Y = L$，其资源用量与宽度为 M 的束搜索相当。根据模型的区分能力，SIDECAR 只需使用束搜索一部分调用次数或内存，就能返回精确的 top-K。即使预算过小，无法找回精确的 top-K，SIDECAR 仍会公布该因子，而束搜索不会，因此服务系统能够在达到质量目标或延迟限制时提前停止。

1. 引言

语义 ID（SID）已成为大型推荐系统表示项目的常用方式：残差

[1]Etsy Inc.，美国纽约布鲁克林。通信请联系：Harshal A. Chaudhari <hchaudhari@etsy.com>。

预印本。*2026*年*11*月*25*日。

	穷举 评分	束搜索 解码器	推测 解码器	SIDECAR （我们的方法）
精确 top-K	✓	∼	∼	✓
精确性证书	✓	✗	✗	✓
提前停止	N/A	✗	✗	✓
预算约束下的保证	N/A	✗	✗	✓
顺序模型调用	1	L	$< L$	∼1–2（最坏情况 L）
每次调用的内存	目录	W	W +草稿	M ($< W$)
草稿	—	—	训练得到的或检索得到的草稿	任意草稿候选项
示例	—	TIGER、GR4AD、UtilityGR	AtSpeed、NEZHA、SpecGR	—

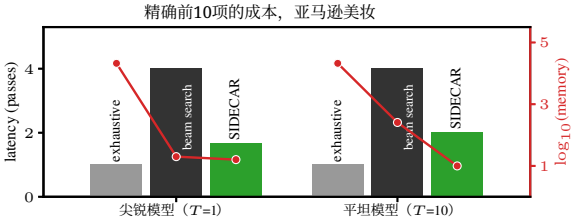


图 1. SIDECAR 返回 top-K，并提供精确性证书；与束搜索相比，所需轮次更少，且每次通行的内存更少解码器。

量化器将每个项目的内容映射为 L 离散代码（Lee 等，2022；Rajput 等，2023），序列模型则依次输出这些代码，就像生成式检索为文档生成内容一样（Tay 等，2022；Wang 等，2022）。因此，模型不需要规模与目录相同的输出层，新项目仅凭其内容就能获得标识符。继 TIGER 之后，大多数研究都专注于生成这些代码、利用协作信号学习这些代码（Jin 等，2024年；Wang 等，2024a；Liu 等，2025年），如今已有多家公司在生产环境中部署了这一设计（He 等，2026年；D'Amico 等，2026年；Deng 等，2025；Zhai 等，2024年）。

解码也受到了关注；然而，这方面的大多数工作都会修改模型以适配解码器，例如采用多词元头（Hou 等人，2025年；Zhan 等人，2026年）、训练得到的草拟器或自草拟头（Lin 等人，2025年；Wang 等人，2026b），或针对束搜索重新训练的评分器（Zhuo 等人，2020年）。训练后可直接部署的模型仍通过在有效 SID 构成的字典树上进行约束束搜索来解码（Ju 等人，2025年）；其大规模变体则会调整宽度（Xue 等人，2026年）、拆分束（Wang 等人，2026a）或重新评分前缀（Hu 等人，2026年），但每个 SID 层级仍需运行一遍有依赖关系的计算。这会带来可衡量的代价：使用束搜索解码的单头模型排名更优、内容质量更高，而多词元预测则能提供更高的吞吐量（Oh 等人，2026年）。因此，我们提出以下问题：

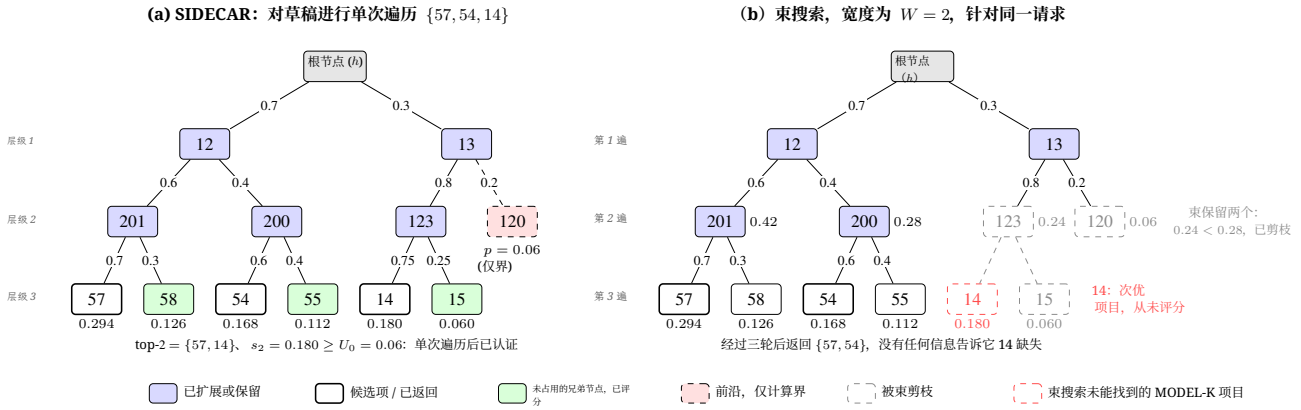


图 2. 同一请求在两种解码器下的表现 ($L = 3$, $K = 2$)。**(a)** 对草稿进行单次遍历，即可为候选项及其自由兄弟节点打分；唯一的未扩展前缀权重低于次优分数，因此第一遍获得认证。**(b)** 宽度为 K 的束搜索在每一层级保留两个最重前缀，并剪枝 (13,123)，而该前缀持有次优项。它在三轮次后返回错误列表，且其计算过程中没有任何信息揭示该错误。

我们能否设计一种解码器，在避免顺序计算成本的同时，返回单头模型的精确输出?

束搜索以僵化方式消耗内存和延迟这两项大规模生产系统的受限资源：对每一个请求，它都运行 L 轮次，每轮 W 个前缀，并且仅在最后一遍之后才输出列表。然而，这些轮次中的大多数并不会改变最终输出。SID 码本所寻址的单元远多于大多数目录规模，因此在最初几个层级之下，大多数前缀只包含单个项目。尽管如此，束搜索仍会对此类前缀剩余的每一个编码花费一次已满的顺序遍历。这种方法也制约了编码本身的设计。由于每一层级都会增加一次依赖遍历，语义 ID 通常保持较短（Hou et al.，2025年；Xieet al.，2026年），而近期的工业工作将容量从深度转移到更大的码本（Xie et al.，2026年）。然而，更大的码本会使编码空间更加远超目录。更深的树是否能更好地刻画嵌入空间，仍是开放问题。此外，束搜索无法判断其哪些输出列表不同于精确 top-K （模型的）。其输出列表为精确的，当 W 在每一层级均超过位于第 K优项目之上的前缀数量时；而束搜索无法在运行时验证这一点。例如，在 图 2(b) 中，宽度为K 的束会剪枝持有次优项的前缀，而束搜索的计算过程中没有任何信息揭示其列表因此是错误的。上述两项局限，即 L 轮次的固定成本以及缺失的证书，源于解码器而非模型本身。因此，我们保持模型及其编码不变，仅重新设计解码器。

按设计，束搜索假设在第一遍之前对最终输出一无所知。然而，工业界的大多数大规模推荐系统都能轻松提供答案草稿。如今，大多数生产流程都会在排序器运行前先运行第一阶段检索器，并且候选项通常通过批处理作业离线计算。或者，批处理作业也可以存储模型本身的前N 个项目，或者存储训练时导出的模型低成本、高度量化副本中的前几个项目。批处理作业可以根据数次交互之前的用户交互历史，为每位用户计算高质量草稿。在请求时，解码器便可以通过键值存储检索这些草稿。此外，一次使用树注意力掩码的遍历（Miao 等人，2024年；Cai 等人，2024年）即可为整个草稿打分，就像推测解码器验证草稿树一样（Leviathan 等人，2023年）。本文介绍 SIDECAR（基于近似检索草稿、带精确性证书的语义 ID 解码），这是一种能够利用此类草稿的解码器。由于解码受限于有效 SID 的字典树（De Cao 等人，2021），为草稿打分的遍也能界定其中每个项目以外所有项目的分数。因此，SIDECAR 可以证明其列表就是精确的前K （图 2(a)），现有面向 SID 的推测解码器都无法做到这一点（Lin 等人，2025年；Ding 等人，2026年）；若认证失败，回退机制只会扩展那些能够进一步改善输出列表的前缀。因此，只要每次遍历有足够的内存，无论草稿如何，输出都能保证精确。草稿质量本身只决定所需的遍数。此外，SIDECAR 利用了目录字典树的结构，而束搜索忽略了这一结构，因此最早在第一遍结束后，就能预告其输出距离精确结果还有多远。最后，由于 SIDECAR 不必在每个请求中对每个 SID 层级各执行一遍，我们认为它为构建更深的 SID 树开辟了新途径，从而能够从物品嵌入中捕获更丰富的语义结构。

我们分两个阶段发展 SIDECAR 的理论。在第2节和第3节中，我们假设每次通行具有无界内存，此时证书、回退机制以及目录字典树的作用最易于表述。在此直觉的基础上，第4节加入在线服务的内存和延迟预算；在这些约束下，BUDGET-SIDECAR 规划其通行，使其列表在预算范围内具备最坏情况保证。我们的实验（第5节）比较了两种解码器在

束搜索的宽度及其 L 轮次，并比较了平坦度不断增加的模型。在尖锐模型上，两种解码器返回相同的列表，但在内存相同的情况下，SIDECAR 所需的轮次不到束搜索的一半，并且能够为每个精确列表提供证书，而束搜索一个也无法提供。随着模型变得更加平坦，只要宽度小于整个目录，束搜索的正确性就会在每个宽度下崩溃，因为它根据概率质量为前缀排名，而平坦模型的顶级项目位于概率质量很小的簇中。相比之下，SIDECAR 只需使用束搜索一小部分的内存，仍能持续返回精确列表，因为其草稿直接指向顶级项目，而束搜索必须保留每个重簇才能找到这些项目。当宽度达到束搜索宽度的一半时，SIDECAR 还能够证明其列表是精确的。最后，首次调用后的公告会告知部署系统当前属于上述哪种情况；当两种资源均不足时，它还会说明保证因子，而束搜索无法表明其列表距离精确列表还有多远。图 1将 SIDECAR 与穷举评分、束搜索解码器和推测解码器进行了比较。附录A 给出了所有形式化结果所依据的假设和证明。

2. 解码器及其证书

本节将介绍受约束解码的设定，并给出必要的符号说明。随后，我们将介绍每次通行使用无界内存的 SIDECAR，包括其第一遍、证书和回退机制，并说明草稿的质量如何影响成本。

目录字典树上的受约束解码。 在本文中，目录中的每个项目 $e \in \mathcal{E}$ 都有一个语义 ID $\mathrm{SID}(e) = (c_1(e), \ldots, c_L(e))$，其中前 $L-1$ 个编码来自项目内容，最后一个编码用于区分内容编码相同的项目。目录中的语义 ID 构成目录字典树 $\mathcal{T}$，由服务栈持有。给定用户历史记录 h 和前缀 $c_{<\ell}$，模型的逻辑值 z 与受约束解码中的情况一样，仅定义可接受的下一个编码上的概率：

$$p(c_\ell \mid h, c_{<\ell}) = \frac{\exp z_{c_\ell}}{\sum_{c \in \mathrm{children}(c_{<\ell})} \exp z_c}, \qquad (1)$$

其中，$\mathrm{children}(c_{<\ell})$ 表示 $\mathcal{T}$ 中该前缀的可接受子节点。一个项目的概率是其 L 条件概率的乘积，而一个前缀 b 的概率则是这些条件概率中前 $\mathrm{depth}(b)$ 个的乘积。由于某个前缀下的条件概率在其子节点上之和为一，因此该前缀的概率等于其下所有项目的总概率。

我们的目标是模型精确的 top-K，即对于当前请求，在模型下概率最高的 K 个项目，我们将其记为 MODEL-K。每个解码器都会返回一个包含 K 个项目的列表；约束束搜索经过 L 轮顺序遍历后，返回其最终束中的 top-K，每个层级使用 W 个前缀，这只能近似 MODEL-K。我们的解码器返回 SIDECAR-K。我们使用 SIDECAR-K$_r$ 表示其在下述过程的第 r 轮之后的输出列表。当返回的列表等于 MODEL-K 时，我们称其为精确。当 MODEL-K 中的每个项目与其对应排名上的项目相差不超过该因子时，SIDECAR-K 处于因子 $1+\delta$ 范围内。

第一遍。 一个草稿 是任意一组 C 的 N 候选项。我们将候选项映射到其语义 ID，并将其插入字典树，以下简称候选字典树。内部节点遵循输入中的历史记录，树注意力掩码使每个节点都能关注历史记录、其祖先以及自身。使用与节点深度对应的位置编号时，每个节点的输出都等于单独遍历其自身路径所产生的输出。因此，单次遍历会返回候选字典树中每个内部节点的模型分布。我们从这些输出中读取两类数值。首先，所有深度为$(L-1)$且前缀位于字典树中的项目都会获得其精确概率，其中包括候选项及其兄弟节点，且无需额外成本。其次，每个已展开节点的可接受子节点中，未被展开的子节点都是一个前沿 前缀，单次遍历也会返回其概率，从而为其下方的每个项目提供界。 图 2 以一个小型示例展示了这一遍历过程。

在任意轮次 r之后，令 E_r 表示具有精确概率的项目集合，$s_K(r)$表示这些概率中第 K大的概率，F_r 表示前沿，并令 $U_r = \max_{b \in F_r} p(b \mid h)$表示最大的前沿概率。列表 SIDECAR-K$_r$ 是E_r的前K 项。由于 E_r 之外的每个项目都恰好位于一个前沿前缀之下，因此其概率至多为 U_r，从而得到以下证书。

定理 1（证书）。 如果 $s_K(r) \geq U_r$，则SIDECAR-K$_r$ = MODEL-K。

当定理 1 的条件成立时，我们称 SIDECAR-K已通过认证。图和表将这两种结果分别命名为精确和已认证，即 EXACT 和 CERTIFICATE。请注意，BEAM-K 在某些请求上可能是精确的，但永远不会通过认证，因为束搜索无法辨别其精确列表与其他列表。

该测试只使用已完成的轮次计算出的量，即对精确分数进行排序，并取前沿上的最大值。换句话说，草稿只决定认证测试何时通过。如果其中包含精确的前K项，测试将在第一遍之后通过。否则，回退轮次会持续进行，直到测试通过，而返回的列表仍然是精确的前K项。

回退机制。 如果第一遍后认证失败，某些前沿前缀下方可能藏着更优的项目。当某个前沿前缀的概率高于当前第 K 高的分数时，我们称其为开放 前缀，并将开放集合记为$O_r = \{b \in F_r : p(b \mid h) > s_K(r)\}$。下一轮只扩展开放前缀，在单次遍历中复用缓存的

算法 1SIDECAR

要求： 历史记录 h，草稿 C，列表大小 K

E_0，$F_0 \leftarrow$ 对候选项字典树 C 之后 h
对于 $r = 0, 1, \ldots, L-2$ **执行**
  $O_r \leftarrow \{b \in F_r : p(b \mid h) > s_K(r)\}$
  if $O_r = \emptyset$ **然后**
    返回顶部K 的 E_r
  结束条件判断
  E_{r+1}，$F_{r+1} \leftarrow$ 对 O_r with 的子节点进行单次遍历缓存的 KV
结束循环
返回 E_{L-1}的顶部-K

历史记录和祖先节点，并读取它们的子节点。 算法 1概述了该过程。

命题 1 （终止）。 无论草稿和模型如何， s_K 都不会减少， U 都不会从一个轮次到下一个轮次增加，并且 算法 *1* 至多经过 L 轮次后返回 MODEL-K。

直观地说，每个轮次都会在开放前缀下沿字典树向下遍历一层，因此经过 $L-1$ 轮次后，不会再有任何内容保持开放状态。因此，优秀的草稿可以让该过程在第一遍后停止。较差的草稿需要更多轮次，但列表仍然是精确的。

证书成立前提前停止。部署可以在证书成立前停止该过程，例如，当通行次数达到在最大服务延迟限制内所能容纳的上限时。此时的列表仍能给出较弱但有效的结论。令 $s_k(r)$ 表示第 k 高的精确分数， $p^*_{(k)}$ 表示模型排名第 k 的项目的概率，并将差距定义为 $\text{gap}_r = U_r / s_K(r) - 1$。

命题 2（逐排名界）。在任意轮次 r 之后，对每个排名 $k \le K$，

$$s_k(r) \le p^*_{(k)} \le \max\big(s_k(r), U_r\big) \le \big(1 + \max(\text{gap}_r, 0)\big)\, s_k(r).$$

因此，SIDECAR-K$_r$ 在每个排名上的结果都在 MODEL-K 的 gap$_r$ 倍范围内，而 $\text{gap}_r \le 0$ 正是证书。请注意，束搜索无法提供这两项结论，因为它不会对剪枝的项目保留任何界。

草稿质量的影响。草稿质量从不影响精确性。相反，它会影响通行次数，而影响途径只有一个量，即第一遍未评分的概率 $\varepsilon = 1 - p(E_0)$，我们称之为隐藏概率。如果 $\varepsilon < p^*_{(K)}$，也就是说，隐藏概率低于模型第 K 个最佳项目的概率，那么 SIDECAR-K$_0$ 就获得了认证。原因如下：第一遍的前沿前缀彼此不相交，且所包含的概率恰好等于未评分概率，因此每个前沿前缀的概率至多为 ε。MODEL-K 中的一个未评分项目落在其中某个前沿前缀之下的概率至少为 $p^*_{(K)} > \varepsilon$，这是不可能的。因此，MODEL-K 的每个项目都已评分， $s_K(0) = p^*_{(K)} > \varepsilon \ge U_0$，证书成立。由此可见，草稿不必包含 MODEL-K，只需包含来自正确前缀的项目，使其未覆盖的概率小于第 K 个最佳项目的概率。

3. 利用目录字典树的结构

在第 2 节解码器的基础上，本节展示目录字典树的结构为 SIDECAR 带来了什么。首先，我们观察到 SID 字典树中大多数深层前缀只对应一个项目。然后，我们利用这一结构，在每一遍中节省词元，并在单轮内解析某些开放前缀。

单项目缀。 在 SID 字典树中，大多数深层前缀仅对应一个项目，这源于编码空间的大小，而非某个特定目录或量化器。 $L-1$ 内容编码取自大小为B的码本，用于表示 B^{L-1} 个单元，其数量级远超目录所含的数千至数百万个项目（例如，TIGER 的三个码本各含 256 个编码，共有 1670 万个单元（Rajput 等，2023年））。此外，无论采用残差量化器（Lee 等，2022年；Deng 等，2025年），还是对物品嵌入进行层次聚类（Tay 等，2022年；Wang 等，2022年），所有正在使用的量化器都会在每个层级将某个前缀下的项目分散到其码本中，因此每个前缀对应的项目数量会随着深度增加而按几何级数递减。事实上，TIGER 之所以引入冲突码，正是因为仅凭内容编码，几乎每个项目都会独占一个前缀（Rajput 等，2023年）。因此，除第一个层级外，大多数前缀仅对应一个项目。在这种前缀处，模型只有一个可接受子节点，因此其受约束条件概率为 1，下方项目继承该前缀的概率。因此，这些前缀无需遍历，也无需词元。

当一个前缀有至少两个可接受子节点时，我们称它为分支前缀。前缀 b 的骨架记作 $S(b)$，由 b 以下的分支前缀，以及连接这些前缀与 b 的路径上的非分支前缀组成；我们将后者称为连接节点。请注意，连接节点是必需的，因为只有将节点与其祖先一并输入，才能输入该节点。我们用 $\sigma(b) = |S(b)|$表示骨架中的节点数，即一遍中为一次性确定 b 以下所有项目而必须输入的词元数。 $S(b)$ 和 $\sigma(b)$ 都仅取决于目录字典树，因此在任何请求到达之前就已知；而骨架上的输出决定了 b 以下每个项目的概率。由于大多数深层前缀都是链，一个簇的骨架只占其前缀的一小部分，整个目录的骨架也只占字典树的一小部分。

链规则。我们在 SIDECAR 的三个环节中利用链，并且每个环节只输入已知需要的节点。首先，在第一遍中，子树为链的候选前缀对分数没有贡献，因为其父节点处的条件概率给出了它的概率，而链上的每个节点都会将该概率原样传递给项目。因此，第一遍只输入分支候选前缀和候选路径上的连接节点。其次，出于同样的原因，无需使用词元即可确定子树为链的前沿前缀，因为它的项目会获得该前缀的概率，而这一概率已由该遍返回。第三，只有一个可接受子节点的开放前缀也会使该子节点成为开放前缀，因为子节点继承了它的概率。因此，一轮回退会一次性输入整段单分支链，直到下一个分支前缀，而不是每轮输入一个节点。这些规则不会影响证书、精确性以及 L 遍的界。

骨架轮次。 一次回退轮次算法 1向开放前缀提供输入，并读取其子节点的条件概率；这些子节点将成为下一轮次的前沿。因此，回退机制每轮下降一个层级，这正是 $L-1$ 轮次这一界限的来源。当每次通行的内存不受限时，树遍历可以改为将一个开放前缀连同其整个骨架一起作为输入，就像第一遍将整个候选字典树作为输入一样。这样，每个供给节点的祖先节点都会出现在序列中，正如树注意力所要求的那样；因此，通行后，开放前缀下方的每个项目都具有精确概率。因此，如果第一轮回退将每个开放前缀 $b \in O_0$ 连同其骨架一起作为输入，代价为 $\sum_{b\in O_0}\sigma(b)$ 个词元，那么O_0 下方的内容就不再位于前沿上，而其他所有前沿前缀仍保持闭合，因为 s_K 不会减少。于是，第二遍之后开放集合为空；无论草稿和模型如何，SIDECAR-K 都得到了验证。

换言之，骨架轮次以增加词元为代价节省轮次。然而，在对每一遍的节点数设定上界时，所有开放前缀的骨架可能无法容纳在一遍中。在在线服务的内存受限场景下，处理几千个词元的一遍，其成本并不比处理五十个词元的一遍高多少；因此，当开放集合包含数量适中的簇时，这种权衡更倾向于使用骨架。即使在所有草稿、每个簇都开放的最坏情况下，该轮次也只会输入目录的骨架，而这仅占字典树穷举遍历所输入节点的一小部分。与链规则不同，骨架轮次在尚不知道节点是否开放之前就输入这些节点，因此它不属于默认解码器。

4. 预算约束下的解码

到目前为止，我们一直假设一遍可以容纳草稿和回退机制所需的节点数。然而在实践中，在线服务部署会预先配置其资源，并对每个请求应用相同的限制。它限制了

算法 2 BUDGET-SIDECAR：SIDECAR，每次遍历 M 个节点、进行 Y 轮次，并采用由部署设置的目标因子 $1+\delta_t$ 要求：历史记录 h、草稿 C、大小 K、目标因子 $1+\delta_t$（默认为 $\delta_t=0$，即精确） 1： E_0， $F_0 \leftarrow$ 对 C 的候选字典树进行第一遍遍历：最多包含 M 个骨架成员，按草稿顺序排列 2：宣布在 Y 轮次中采用因子 $1+\max(\delta^*(M,Y)\ \delta_t)$，并为精确性进行 $Y_{\min}$ 轮次遍历 3：对于 $r=0$、1、.....、$Y-2$，执行 4：如果 F_r 没有任何前缀处于开放状态，则返回 E_r 的前 K 项 5：从 F_r 重新规划：当前生效因子为 Δ_r，池、规划方案和调度为 x_r 6：$B \leftarrow$ 最重要的开放前缀，最多 M 个 7：添加池节点，直到其中 B 满足 x_r，将池外最轻的前缀移出，然后用更多节点填充空闲槽位 8：E_{r+1}，使用缓存的 KV 对 B 进行单次遍历 $F_{r+1} \leftarrow$ 9：遍历结束 10：返回 E_{Y-1} 的前 K 项

每一遍的内存，其上限由一遍最多可容纳的节点数表示，记为 M；每个请求的延迟则由其最多可运行的轮次数表示，记为 Y。注意，对于 $Y=L$ 而言，这些恰好就是宽度为 $W=M$的约束束搜索所使用的资源。在本节中，我们将讨论有关此类预算的两个问题。第一，当预算可能小到无法保证精确性时，SIDECAR 能提供何种保证？第二，我们最早何时能够提供这一保证，即在花费剩余预算之前能否提供？该保证采取近似因子的形式：在每个排名上，SIDECAR-K 都在 M 的 $1+\delta$ 因子范围内，即相对于 MODEL-K 满足 命题 2的意义。我们将第一遍之后能够陈述的最小 δ称为公告因子。我们将预算约束下的 SIDECAR 称为 BUDGET-SIDECAR（算法 2），并将其执行过程称为计划运行。首先，我们描述第一遍结束后所获得的结果（1 行，见算法 2）。其次，我们界定解析一个开放前缀可能产生的成本。第三，我们展示如何将所有开放前缀的成本转化为轮次数，以及每一轮的计划（5–7行）。最后，我们推导出预算内各轮次在最坏情况下能够保证的因子（2行）。

在本节中，我们沿用既定约定。首先，第 r 轮之后的状态带有索引 r，即E_r 是已评分项目的集合， $s_K(r)$ 是第 K 高的精确分数，而 F_r 是前沿。第 0 轮是第一遍。因此，第 r 轮之后，计划运行已使用 $1+r$ 遍，且$Y-1-r$还剩下回退轮次。下面我们给出 $r=0$ 时的构造，即公告发生的时刻；算法 2 的第5 行会在每轮之后重复执行该构造。其次，该保证是针对第一遍尚未观察到的内容给出的最坏情况保证。第一遍之后，解码器知道模型在输入节点上的输出，但不知道每个前沿前缀的概率如何分配到其下方的前缀中。我们将任何与第一遍观察结果一致的这些未观察输出分配称为

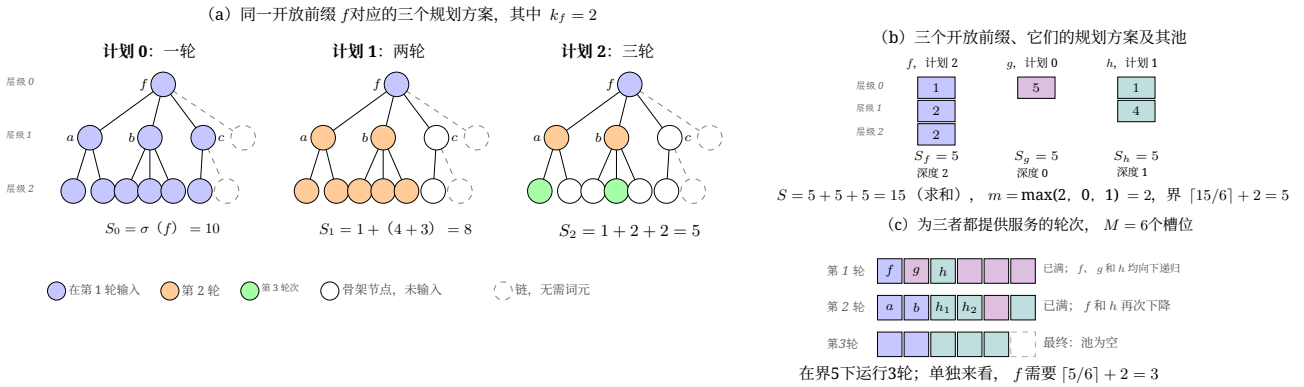


图3. 预算下的计划与轮次（命题3和4）。**（a）**一个开放前缀的三个规划方案，其中 $k_f = 2$，因此每层最多可以有两个节点处于开放状态：规划方案0 在一轮内为整个骨架提供服务，规划方案2 每轮处理一层，规划方案1 介于两者之间；对手开放骨架最大的节点。**（b）**采用不同规划方案的三个开放前缀：池大小 S 在各前缀间累加，而深度 m 取它们的最大值，因为它们同时下降。**（c）**每轮六个槽位，三轮即可处理全部三个前缀：先处理已知开放的前缀，再处理骨架节点。由于深度尚未处理完时，没有任何一轮的槽位未用满，因此这一方案优于界5。

对每个前沿前缀的概率如何分配到其下方的前缀中。我们将任何与第一遍观察结果一致的这些未观察输出分配称为一致补全，并让对手 选择对回退机制最不利的补全。公告因子对这个对手的选择仍然成立，因此对每一种一致补全也都成立。由于真实模型是一种一致补全，该因子对真实模型同样成立，而且通常还有很大的余量。

阈值与开放前缀。第一遍（第 1 行的算法 2）是 SIDECAR 的第一遍，节点数上限为 M，也就是说，它按草稿顺序输入候选字典树的骨架成员，直至达到上限。该遍返回已评分项目 E_0、其中第 K 高的分数 $s_K(0)$，以及前沿 F_0，并给出每个前沿前缀 f的概率 $p(f)$。然而，前沿以下的情况尚一无所知。根据这些量，我们可以判断当前列表是否处于 MODEL-K 的给定因子范围内。每个未评分项目都位于某个前沿前缀之下，因此其概率至多等于该前缀的概率。因此，只要每个前沿前缀 b 均满足$p(b) \le (1+\delta)\, s_K$，SIDECAR-K 就处于因子 $1+\delta$ 范围内，其中 s_K 是当前第 K 高的精确分数。令 $t = (1+\delta)\, s_K(0)$ 表示第一遍之后的这一阈值。当 $p(b) > t$时，我们称前沿前缀在 t 处开放。由于 s_K 绝不会减少，在 t 处关闭的前缀会一直保持关闭。因此，回退机制的任务是在允许的轮次和节点数内，将每个开放前缀降至 t 以下。请注意，公告在回退机制运行之前发布，因此只能使用第一遍和目录字典树返回的量。

前沿前缀 b 的骨架也会说明，在每轮一级回退机制下，需要扩展多少层才能解析它，即其最深分支前缀的深度减去 b的深度，再加一。我们将这个数称为解析深度 $d_b \le L - \text{depth}(b)$。

单个开放前缀的规划方案。固定一个开放前缀 $f \in F_0$。回退机制必须判断 f 下方是否存在某个前缀也高于 t，并持续进行判断，直到不存在这样的前缀为止。我们从 f开始计算层级： f 本身处于第 0 层，其子节点处于第 1 层，而其深度为 j 的后代节点处于第 j层。两个事实限制了第 j层中可能高于 t的前缀数量，而且这两个事实在第一遍之后便已知。首先， f 下方同一层的前缀彼此不相交，且其概率总和为 $p(f)$，因此其中至多有 $k_f = \lfloor p(f)/t \rfloor$个超过 t。其次，只有骨架节点需要被输入（第 3 节），而第j层中只有 $c_j(f)$ 个这样的节点，其中 $c_0(f) = 1$表示计入 f本身，而 $\sigma(f) = \sum_j c_j(f)$ 是 f的骨架大小。然而，在这两个界限之内，目前观察到的信息并未揭示哪些节点是开放的，因此我们让对手来决定。

回退机制可以通过 f 两种方式进行解析。第一种方式是逐层解析：回退机制输入 f，读取其子节点，输入其中开放的子节点，以此类推，每轮向下移动一个层级。通过这种方式输入的每个节点都已知是开放的。然而，每个层级都需要单独的一轮，因此解析 f 最多需要 d_f 轮，其中 d_f 是其解析深度。第二种方式是整个骨架：回退机制在一轮中输入 f 及其完整骨架。这只需要一轮，但无论这些节点最终是否开放，都要耗费 $\sigma(f)$ 个节点。注意，只输入某个层级的一部分并不能应对对手，因为对手可以将隐藏概率放在被跳过的任意部分。因此，能够提供最坏情况保证的选项，是将这两种方式结合起来：回退机制 ℓ 逐轮向下解析若干层级，然后输入层级 ℓ 的开放节点及其完整骨架。我们将 $\ell_f \in \{0, \ldots, L - 1 - \text{depth}(f)\}$ 称为规划方案 f，而图 3（a）展示了一个开放前缀的三种规划方案。

命题3（计划的最坏情况成本）。 令 f 在阈值 t 和 $k_f = \lfloor p(f) / t \rfloor$处开放。对于每个一致补全：

（i）在 f以下的 j 层级中，至多有 $\min(c_j(f),\ k_f)$ 个节点位于上方并且必须被供给；（ii）计划 ℓ至多供给

$$S_\ell(f,t) \;=\; \underbrace{\sum_{j=0}^{\ell-1} \min\big(c_j(f), k_f\big)}_{\text{levels fed one at a time}} \;+\; \underbrace{\Sigma_\ell^{\sigma}(f,t)}_{\text{skeletons at level } \ell} \qquad (2)$$

在 f以下没有任何内容开放之前的节点，其中 Σ_ℓ^{σ}（f、t）是第 ℓ 层节点中 k_f 个最大骨架大小之和。

计划 $\ell = 0$ 是骨架轮次，代价为 σ（f）个节点。最大的计划是纯逐层下降。它只馈送已知开放的节点，但每层都需要一个轮次。在这两个极端之间，规模更大的计划使用更多轮次，并馈送更少的节点。

组合所有开放前缀的规划方案。假设每个开放前缀都已分配一个规划方案，并令池表示这些规划方案可能馈送的节点集合。池的最坏情况大小为 S，即公式（2）在所有开放前缀上的总和；其深度为 $m = \max_f \ell_f$，即任一前缀逐轮下降时所经过的最大层级数。请注意，S 和 m 都由规划方案和目录在任何回退轮次运行之前确定。计划运行将发现哪些节点是开放的。然而，由于已关闭的前缀会保持关闭，之后变为开放的每个前缀都是某个 $f \in F_0$的后代节点，而 f的规划方案已经为其做好了安排。

命题 4（轮数上界与调度）。设 F_0 的开放前缀有计划，池大小为 S，深度为 m。

（i）回退机制会在以下轮次内完成，

$$\lceil S/M \rceil + m \qquad (3)$$

无论已满轮次和未填满轮次以何种顺序出现。

（ii）若 $\lceil S/M \rceil + m \le y$ 还剩 y 轮次，则

$$x_r \;=\; S - (y - 1 - m)\, M \;\le\; M, \qquad (4)$$

而一轮若至少处理 x_r 个池节点，或处理所有可用池节点，就会使池中剩下 $\lceil S'/M \rceil + m' \le y - 1$.

该界统计两类轮次。将其所有 M 槽位都填入池节点的轮次为已满轮次；由于规划方案总共最多提供 S 个池节点，因此最多有 $\lceil S/M \rceil$个已满轮次。无法填满的轮次已经提供了所有父节点已知的池节点。因此，剩余待下降的深度会减少一层，而这种情况最多发生 m 次。请注意，这两项以不同方式汇总开放前缀。节点不会在开放前缀之间共享，因此其计数相加为，而轮次是共享的，因为所有开放前缀都在相同的轮次中下降。因此，深度项是各规划方案的最大值 m，而不是它们的总和。 图3 (b) 和 (c) 展示了三个具有不同规划方案的开放前缀，以及共同服务于它们的轮次。

命题 4 的第（ii）部分命题 4 给出了单轮的规则，算法 6 和 7 行 算法 2 负责执行该规则。我们将数值 x_r 称为调度，即当前轮必须处理的池节点数，以确保轮数上界仍能满足本轮之后剩余的轮次要求。一轮中可用的池节点有两类。第一类是开放前缀本身。每个精确解码器都必须处理它们，因此第 6 行会选取最重且能放入 M 槽位的开放前缀。第二类是前缀下方计划为零的骨架节点。此时尚无法确定是否需要它们，因此处理这些节点可以为最坏情况做准备，而无需判断这种情况是否会发生。第 7 行只会按调度要求处理这类节点：不断添加，直到批次包含 x_r 个池节点；如果批次已满，则用它们替换批次中位于池外且权重最轻的前缀。我们将这些节点称为推测节点，它们是计划运行中唯一在尚不确定是否需要时就会处理的节点。最后，剩余的空闲槽位会用更多推测节点填满，因为无论槽位是否填满，一遍所需的时间大致相同，而填满的槽位可以解析开放簇的一部分。

公告因子。最佳规划方案是将公式（3）的界最小化，而这一最小化过程可以在各个开放前缀上分别进行。为了解释这一点，设定一个深度上限m。对于满足 $\ell \le m$的任何规划方案，界中的深度项至多为 m，因此每个开放前缀都可以独立选择满足 $\ell \le m$的最廉价规划方案。例如，具有较小骨架的重前缀会被整体送入处理，而具有较大骨架的轻前缀则逐层向下处理，并且二者会在同一批次中进行。令$S_m^*(t) = \sum_{f \in F_0:\ p(f)>t} \min_{\ell \le m} S_\ell(f,\ t)$ 表示所得的池大小。随后对上限进行最小化，即可得到最小的界，

$$R^*(t) \;=\; \min_{0 \le m \le L-2} \big(\lceil S_m^*(t)/M \rceil + m\big). \qquad (5)$$

注意，在第一遍之后，$R^*(t)$ 中的所有内容都已知：前沿概率给出 k_f，目录给出 $c_j(f)$ 和骨架大小，而 M 属于预算的一部分。此外，阈值越大，开启的前缀越少，其计数 k_f 也越小，因此 R^* 随 t 增大而非递增。因此，公告因子是使这些轮次能够纳入预算中的 $Y-1$ 个回退轮次的最小 δ，

$$\delta^*(M,Y) \;=\; \min\big\{\delta \ge 0 : R^*\big((1+\delta)\, s_K(0)\big) \le Y - 1\big\}, \qquad (6)$$

附录 C 说明了如何通过二分法或精确计算得出它。同样，我们将满足 $\delta^*(M,Y) = 0$ 的最小 Y 写作 $Y_{\min}(M) = 1 + R^*(s_K(0))$，也就是说，这是 SIDECAR-K 在每遍 M 个节点的条件下得到认证所需的遍数。$\delta^*(M,Y)$ 和 $Y_{\min}(M)$ 都在第一遍之后已知，算法 2 的第 2 行给出了它们。

计划运行。部署还可以设定一个目标δ_t，即对其而言足够好的因子，并以 $\delta_t = 0$ 表示精确性。随后，每一轮都由两个阈值决定。第一个是目标阈值（$1 + \delta_t$） s_K（r）：当前沿前缀高于该阈值时，它就是开放的，因为此时它仍可能使 SIDECAR-K 超出目标所允许的范围。因此，算法 2 的第 4 行仅在没有开放前缀时才停止，第 6 行则优先处理开放前缀。第二个是当前生效因子的阈值 $\Delta_r = \max(\delta_r^*, \delta_t)$，其中 δ_r^* 表示根据 F_r 和 s_K（r）重新计算的式（6），剩余 $Y - 1 - r$ 轮，因此 $\delta_0^* = \delta^*$（M, Y）就是公告。

在每一轮中，第 5 行仅针对高于 $(1 + \Delta_r)\, s_K(r)$ 的前缀计算池、规划方案和调度，因为只有这些前缀可能违反当前生效因子。当 $\Delta_r > \delta_t$ 时，处于两个阈值之间的前缀虽为开放前缀，却不在池中：第 6 行会在它们属于权重最大的前缀时处理它们，而这些正是第7 行可能挤出的前缀。当多组规划方案达到相同因子时，计划运行会采用 最小的方案，根据 式（4），这会使推测量最少。最后，第 8行是算法 1 的回退遍历，以批次代替开放集合。

与非规划运行的比较。 为了了解规划带来的变化，考虑一次不使用规划的运行。在没有规划的情况下，预算下的 SIDECAR 采用 算法 1，每次遍历M 个节点：在每一轮中，它会扩展能够容纳的最重开放前缀，每轮下降一个层级；当证书成立或所有 Y 轮次都已用完时停止。我们称这种运行方式为 非规划运行。从不会失败，意义在于，根据 命题 2，它最终持有的列表在每个排名上都处于 $1+$差距 因子以内。然而，当 Y轮次对于内存 M来说太少时，该列表并不精确，其因子就是最终得到的差距，并且在最后一轮之前无法得知这一因子的任何信息。

相比之下，第一遍之后才声明的因子必须抵御对手，因此剩余轮次必须覆盖最坏情况。具体而言，如果剩余轮次无法逐层解析一个重型开放前缀，那么在模型揭示是否需要其骨架之前，就必须立即将该骨架送入模型。因此，两次运行仅在第 7 行的 算法 2 中有所不同。只要容量允许，两者都会按照相同顺序扩展已知为开放的前缀。此外，BUDGET-SIDECAR 还会送入推测节点。它首先将这些节点放入开放前缀留下的槽位中，从而不会延迟任何开放前缀。只有当调度所需的推测节点多于空槽位时，它们才会取代池外最轻的开放前缀，而这些前缀随后要等到更晚的轮次。

由于推测节点会改变每轮结束后已知的信息，两次运行在后续轮次中扩展的节点并不相同。尽管如此，在固定预算下，它们返回的列表差异很小，因为推测节点正是对手所需要的节点，而其中很少包含真实模型所需的项目。规划增加了两点。首先，部署在第一遍之后就知道因子和 $Y_{\min}$。其次，当轮次不受上限限制时，证书会更早到达，因为将一个骨架整体输入可以在一轮中解析一个簇，而逐层下降的回退方案则需要数轮。在最坏情况下，差异最为明显，而非规划运行无法覆盖这种情况。当重簇将 MODEL-K 的项目隐藏在其骨架中时，逐层下降每轮只能下降一个层级才能到达这些项目，因此可能耗尽一个足以保证精确性的预算；而 BUDGET-SIDECAR 会将整个骨架一次性输入，并在相同预算内获得证书，正如下述命题所述。

命题5（最坏情况规划是必要的）。 存在一个目录字典树、一个草稿、一个模型和一个预算（M, Y），其中 $Y < L$在此条件下，非规划运行，即每遍处理 M 个节点的 *SIDE-CAR*（算法 *1*），返回的列表不包含MODEL-K中的任何项目；而 *BUDGET-SIDECAR* 在第一遍后公告精确性，并在相同预算内返回MODEL-K 及其证书。

证明中的实例是非规划运行的最坏情况，即模型的概率集中在一个高权重簇中，并均匀分布于其下方的各个项目上。在此实例中，只有经过规划，预算才足以保证精确性。

定理 2（预算下的公告因子）。 运行 *BUDGET-SIDECAR*，每遍处理 M 个节点，最多运行 Y 轮。然后：

（i）SIDECAR-K 在每个排名上的结果都在MODEL-K 的 $1 + \delta^*$（M, Y）因子范围内；（ii） $Y_{\min}$（M）$= 1 + R^*$（s_K（0）轮即可证明SIDECAR-K，且该轮数在第一遍后即可知晓；（iii） $\delta_{r+1}^* \le \delta_r^*$ 对于每一轮 r：重新规划绝不会放宽公告。

备注 1（最坏情况保证的成本。）。 公告因子对对手成立，包括这样一种补全：将重簇的概率分散到其整个骨架上。然而，真实模型会将该概率集中在少数项目上，通常是第一遍已经评分过的项目。因此，实际差距通常远低于公告因子，而部署会同时报告这两者。换言之，公告是悲观的，而

计划运行并非如此。除了 算法 2第 7行中的推测节点外，计划运行只会输入每个精确解码器都必须输入的内容。

因此，推测节点是公告的唯一成本。当调度迫使它们进入一个已满的批次时，它们会替代真实模型可能需要的前缀；而且，由于在真实模型下，它们中的大多数最终都不会包含 MODEL-K 中的任何项目，因此，在相同预算下，计划运行与 MODEL-K 的重叠度可能比非规划运行低几个百分点。当公告因子远高于 1 时，有用的输出就是公告本身，因为它会在一次调用后告知部署：其预算不足。此外，公式（5）会告知部署具体不足多少：对于给定的内存 M，它给出精确性所需的通行次数；而对于给定的通行次数 Y，它给出精确性所需的内存。

注 2（部署参数。）。 BUDGET-SIDECAR 有两个预算和一个目标。这两个预算是 M 和 Y，对于 $Y = L$ 和 $M = W$，它们分别是宽度为 W 的束搜索资源。BUDGET-SIDECAR 的其他一切都由预算、第一遍和目录字典树决定，因此无需调节任何参数。目标 δ_t 说明了何种质量足够，并且它只能缩短计划运行：算法 2的第 4 行一旦确认 SIDECAR-K 在 $1 + \delta_t$ 以内，就会立即返回；而在$\delta_t = 0$ 的情况下，它会以精确性返回。请注意，该停止规则仅依赖证书，无需进行任何规划。

规划所带来的好处在于，部署在第一次调用后就能知道目标是否能在预算内达到；当且仅当 $\delta_t \geq \delta^*(M, Y)$ 时，答案是肯定的，同时还能知道达到精确性需要多少轮次。因此，在回退运行之前，部署可以选择提供服务、升级处理，或确定一个延迟。不过，预算和目标并不是对称的，因为 Y无条件绑定，而目标只允许计划运行提前停止。因此，保证为 $\max(\delta^*(M, Y)\ \delta_t)$：如果 $\delta_t \geq \delta^*$，预算可以达到目标，计划运行也可以提前结束；如果 $\delta_t < \delta^*$，则会使用全部 Y轮次，并且 SIDECAR-K 在 $1 + \delta^*$ 内获得认证，部署在第一遍之后即可获知这一点。最后，草稿是输入，而不是需要调节的参数。对于给定的 M 和 Y，它决定 δ^* 能有多小，但从不影响 SIDECAR-K 及其证书是否正确。

5. 实验

现在，我们将 SIDECAR 和 BUDGET-SIDECAR 与约束束搜索进行比较评估。首先，我们介绍数据集、模型以及重放流程，该流程使我们能够针对每个请求验证我们的论断。然后，我们介绍基线和草稿来源。最后，我们提出实践者所关注的研究问题，并在本节其余部分回答这些问题。

数据与模型。 本文使用 Amazon Reviews 2014 的 Beauty 类别（McAuley 等，2015年；He 与 McAuley，2016年），该类别常用于评估基于 SID 的生成式检索方法。该数据集包含 22,363 名用户和 12,101 个项目，我们采用留一法协议对其进行划分。语义 ID 来自一个包含三个码本的残差量化变分自编码器，每个码本包含 256 个编码，之后还附加一个冲突码，因此每个语义 ID 都有 $L = 4$ 个编码。我们在用户的历史记录上训练了一个四层仅解码器 Transformer，这些历史记录表示为语义 ID 编码序列。需要注意的是，SIDECAR 仅修改解码器阶段，因此我们并不专注于优化 Transformer 的超参数以获得性能最佳的模型。然而，我们训练模型的排序质量处于采用该设置的已发表工作所报告的置信区间内。

评估协议。 对于每位测试用户，我们都会对目录字典树中的全部 21,091 个内部节点进行一次树遍历，以获得模型概率质量的精确分布。随后，每个解码器都会对这些存储分布进行简单的确定性重放。因此，我们可以针对每个请求检验所有主张，并在相同分数下比较各解码器。我们使用 MODEL-K 的 $K = 10$ 作为参考目标，因为它是对模型进行穷举评分所得的输出。SIDECAR 的目标是返回穷举评分的输出；任何解码器都不可能优于其所解码的模型。因此，本节中的所有比较都将 MODEL-K 作为两个解码器的参考。

更平坦的模型。 为了在不改变解码器的情况下调整模型的尖锐程度，我们对训练模型的每个条件分布应用温度 T。训练模型对应于 $T = 1$，而更大的 T会使模型更加平坦。需要注意的是，此处温度的作用不同于大语言模型中常用的单个 Softmax。对于单个 Softmax，温度会重新缩放分布，却不会改变其结果的顺序。然而在这里，每个 L 条件概率都在各自的前缀处进行温度调整，并在该前缀的子节点上重新归一化，因此项目的排名会发生变化。随着T 增大，某个项目的概率趋近于其路径上每个前缀的子节点数倒数之积。因此，更平坦的模型对项目的排名不同于训练模型，可作为在含噪数据上训练的模型的有用代理；这类数据在大规模推荐系统中很常见，因为此类模型的条件概率分布也同样分散。

基线。 我们的基线是使用相同模型和相同历史缓存的约束束搜索，即 TIGER 所采用的设置。我们不与用于语义 ID 的推测式

解码器进行比较，因为它们各自都需要训练模型之外的额外组件。AtSpeed（Lin 等人，2025年）需要训练得到的草稿器，NEZHA（Wang 等，2026b）需要经过训练的自草稿头，而 SpecGR（Ding 等人，2026年）则需要调优接受阈值。此外，这些解码器旨在用更少的轮次复现束搜索生成的列表，但都无法保证结果正确。因此，BEAM-K 也是衡量这些解码器的参照基准。

潜在草稿来源。在请求时，SIDECAR 会从键值存储中查询草稿。这些草稿主要来自两类来源：无模型来源和基于模型的来源。无模型来源完全不使用 SID 模型。内容 k 神经网络会返回内容嵌入与历史记录中最近五个项目中任意一个最接近的项目。共现和矩阵分解（隐式交替最小二乘）基于交互日志构建。相比之下，基于模型的来源从 SID 模型生成草稿，其中主要来源是用户序列，即模型为该用户生成的前N 个项目列表，由批处理作业计算并存储。请注意，在本节中，用户序列并不指用户历史记录中的项目，而是指批处理作业使用用户过时的历史记录运行 SID 模型时，该模型自身排名最高的 N 个项目。我们考虑用户序列的三种变体。第一种是由模型副本生成并存储的序列，该模型的权重量化为 FP4，因此可以由成本更低的批处理作业通过约束束搜索运行。另外两种变体是分别早两次和五次交互的序列副本。无论来源如何，草稿都始终早于触发解码的请求。RQ2 对全部七种来源进行了比较；从 RQ4 开始，为简洁起见，每幅图都使用两个草稿，每个草稿包含 500 个条目。第一个草稿是用户序列，也是我们的默认选择。第二个草稿来自矩阵分解这一最强的无模型来源，以下简称 MF 草稿。

实验测量的内容。 我们检查了第 2 节至第 4 节中证明的每一项性质在每个实验的每个请求上是否成立，以确保没有违反任何性质。具体而言，认证列表均与 MODEL-K 一致，未超过任何公告因子，且任何一次运行的轮次均未超过其最坏情况界。既然正确性已得到确认，接下来的

以下实验侧重于分析成本。它们围绕七个研究问题（RQs）展开：前四个问题在每次通行拥有无限内存的条件下研究，另外三个则在束搜索的预算限制下研究：

RQ1. 是什么决定了通行次数？RQ2. 草稿的良好来源是什么？RQ3. 输出列表的大小 K 如何影响成本？

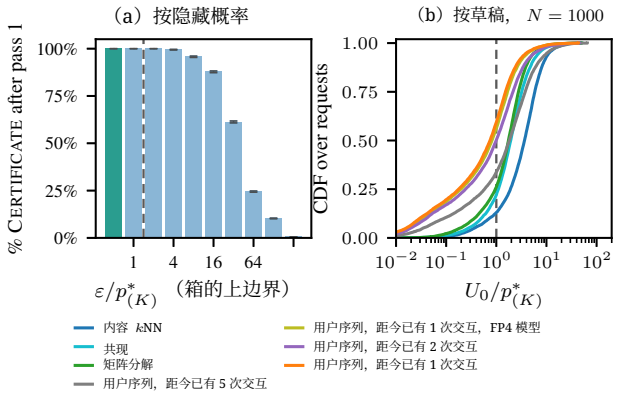


图4。**隐藏概率条件**。**（a）**在线的左侧，正如所证明的，第一次通行对每个请求都通过了认证；即使达到阈值的八倍，它对大多数请求仍然如此。**（b）**精确条件 $U_0 < p^*_{(K)}$ 按照各来源的通行次数对来源进行排序。

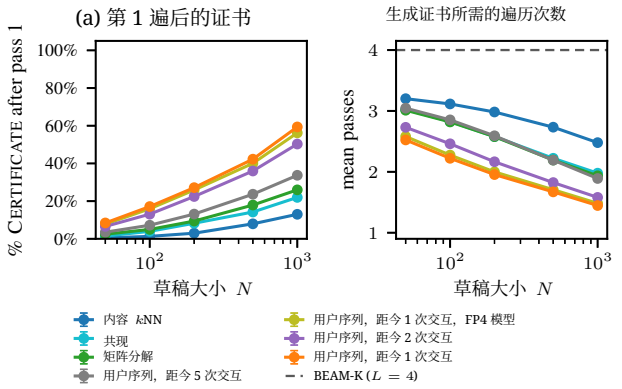


图5。**草稿来源**。每个来源都会生成一个精确列表，而通行次数取决于其与模型的对齐程度：即使来自模型的 FP4 副本，模型自身的过时排序所需通行次数也远低于两次；行为来源约为 2.2 次，内容相似度来源接近三次。束搜索始终需要四次通行。

RQ4. 目录字典树提供了哪些增量收益？RQ5. 在束搜索的资源条件下，SIDECAR-K 至少和 BEAM-K 一样好吗？RQ6. SIDECAR-K 的精确性需要付出什么代价？RQ7. BUDGET-SIDECAR 的最坏情况保证带来了什么益处，又需要付出什么代价？

最后，我们在一个预算平面上展示结果的总体景观。

图表说明。 本节中的每张图表都报告了针对请求统计得出的数据，每位用户对应一个请求。讨论每张图表的文字会说明其实验背景；凡是未提及具体设置的图表，均使用 10,000 名留出用户、包含 $N = 500$个条目的草稿以及 $K = 10$。图表中的误差条表示这些请求的 95% 置信区间，通常窄到几乎看不见。

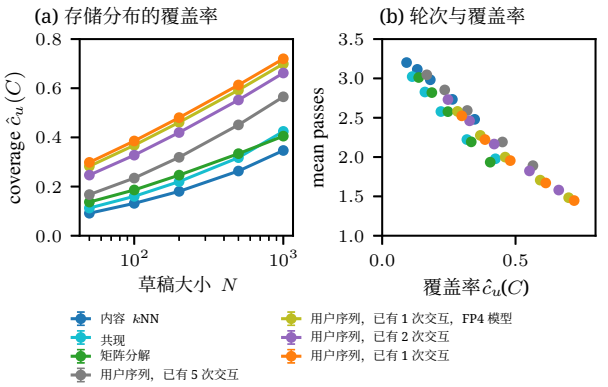


图 6。离线测量的草稿来源对齐。（a）每个草稿相对于其大小的离线覆盖率 $\hat{c}_u(C) = \sum_{e \in C} \hat{p}_u(e)$。**（b）**SIDECAR 的轮次与离线覆盖率的关系，每个来源和大小对应一个点。所有来源都落在同一条曲线上，因此，一个来源的离线覆盖率可以预测其通行次数，并可用于在来源之间进行选择。

5.1. 通行次数（RQ1）

本节中，我们探讨 RQ1：是什么决定了通行次数？直观来看，通行次数应取决于草稿未覆盖的模型概率有多少。原因在于，根据第 2 节，只要这一隐藏概率低于排名第 K的最佳项目的概率，第一遍就能得到认证。

为了验证这一直觉，我们按照隐藏概率与第K个最佳项目的概率之比对请求进行分组。我们为每遍使用无限内存，因此草稿的概率质量是唯一重要的因子。在图 4（a）中，我们观察到，阈值左侧的每个请求都能在第一遍得到证书认证，这一点已得到证明。有趣的是，对于比值达到 8× 该阈值的多数请求，证书仍然成立。这是因为隐藏概率很少集中在一个位置。相反，它分散在许多未覆盖前缀上，而这些前缀中的每一个都比第 K个最佳项目更轻。因此，第第 2 节中的充分条件远非必要条件。

面板(b)展示了精确性条件：对于每个来源在 $N = 1000$ 的请求分布，它将最重的前沿前缀 U_0 与排名第 K的项目进行比较。第一遍在位于 1 左侧的请求上可获得精确认证。草稿未覆盖的概率越小，其来源就有越多请求落在这条线的左侧：用户序列曲线最靠左，而内容 kNN 曲线最靠右。这些观察表明，只要草稿漏掉一个簇，回退机制就至少需要再进行一轮，因为草稿未覆盖的每个第1层前缀都必须被打开或排除。因此，我们得出结论：草稿未覆盖的概率，尤其是其最重的未覆盖前缀，决定了通行次数。

5.2. 草稿来源（RQ2）

接下来，我们探讨 RQ2：什么是优质的草稿来源？由于每个来源都能得到 MODEL-K，因此来源的选择只会影响通行次数。我们发现，与 SID 模型对齐的来源所需的轮次最少，而且这种对齐可以在部署前离线衡量。

图5 比较了每次遍历具有无限内存的七种来源。总体而言，它们的成本分为三个不同层级。内容 k神经网络在任何草稿大小下都需要接近三遍，其第一遍在草稿包含500个条目时仅能为不到10%的请求提供认证。读取交互日志的两种行为来源在包含500个条目时需要约2.2遍，矩阵分解略优于共现。相差一条交互的用户序列需要1.7遍，其第一遍约可为40%的请求提供认证。模型的FP4量化副本所存储的用户序列只需多三百分之一遍，尽管该副本的权重有11%的偏差，并且其前10项仅在69%的请求中与完整模型的前10项一致。相比之下，相差两条交互的用户序列还需要多六分之一遍。相差五条交互的用户序列需要2.2遍，与MF 草稿相同，但它未覆盖的概率更少，而且在一遍之后更常获得认证。因此，SIDECAR能够同时容忍成本更低的量化模型和过时历史记录。

人们可能会认为，针对留出项目的召回率能够解释这一顺序。然而，MF 草稿的召回率几乎与用户序列相同，却多需要半遍。实际上，草稿与模型的对齐程度解释了这一顺序。行为来源包含来自正确簇的项目的频率高于内容相似度，这也正是它们比内容 kNN 少需要半遍的原因。不过，它们很少包含模型本身的顶级项目。

此外，这种对齐可以在部署前测量。设 $\hat{p}_u$ 表示模型针对用户u 在一次交互之前的分布。用户序列是该分布的 top-N，因此，生成用户序列的批处理作业已经计算出 $\hat{p}_u$。我们将草稿的离线覆盖率 定义为 C 其中项目的存储概率，即 $\hat{c}_u(C) = \sum_{e \in C} \hat{p}_u(e)$。批处理作业无需运行解码器，即可从存储表中读取这一数值。离线覆盖率对应于基于陈旧用户历史记录的模型的 $1 - \varepsilon$，其计算仅针对草稿中的项目。在图 6(a)中，我们观察到，离线覆盖率对各来源的排名与其通行次数完全一致。面板(b)绘出了平均通行次数与离线覆盖率的关系，每种规模下的每个来源都落在同一条曲线上。这一结果以可离线计算的数值证实了 RQ1 的结论。因此，在部署时，只需选择在该规模下离线

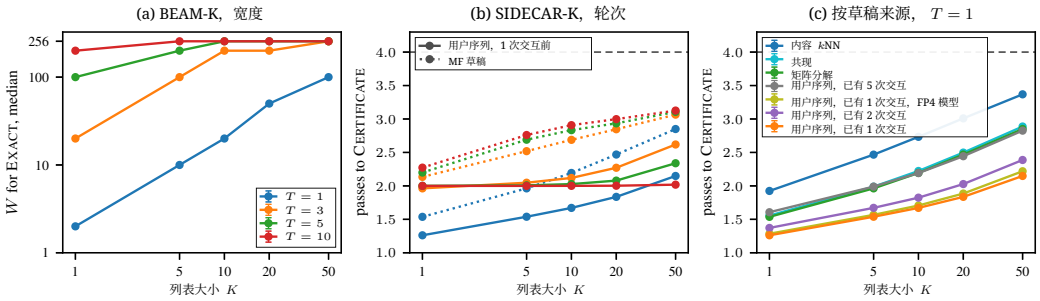


图 7。列表的大小。（a）精确列表的束宽随着训练模型中的 K 增长，并在较平坦的模型上达到最大宽度运行值。**（b）**SIDECAR 的轮次随着训练模型中的 K 缓慢增加；而在扁平模型上，无论 K 为何，随着用户序列的变化，轮次都保持为两轮。**（c）**在每个 K 下，草稿来源的顺序都保持不变。

覆盖率最高的来源即可确定最佳来源和草稿规模，而该规模受部署的内存预算限制。

5.3. 列表的大小（RQ3）

接下来，我们回答 RQ3：输出 K 列表的大小如何影响成本？与模型的尖锐程度一样，列表的大小应由应用程序而非解码器决定。我们预计，列表越长，精确性就越难保证，因为 K-第优项目的权重更低，更多前缀可能超过它。为了量化这一影响，图 7 将 K 从1 变化到50，每次通行使用无限内存。

如预期，列表越长，两个解码器都需要更多资源，但所需的资源类型不同。束搜索需要更大的束宽度（图 a）。在训练模型上，中位请求对于单个最佳项目需要宽度 2，而对于前 50 项则需要宽度 100。在更平坦的模型上，中位请求需要我们从$K = 5$ 开始运行的最大宽度。相比之下，SIDECAR 需要更多轮次，但只多出少数几轮（图 b）。在训练模型上使用用户序列时，在相同的 K范围内，通行次数从 1.2 增加到 2.2。在更平坦的模型上，每个 K都保持为两轮，因为第一遍已经包含整个列表，而一轮回退轮次即可对其进行认证，无论列表长度如何。图 (c) 显示，RQ2 中发现的来源排序在每种列表大小下都成立，MF 草稿比用户序列大约少半轮。因此，列表大小最多会使束搜索的宽度变化 50×，而对 SIDECAR 通行次数的影响约为一轮；对于 $L = 4$，SIDECAR 仍然比束搜索需要更少的通行次数。

5.4. 目录 Trie 的收益（RQ4）

本节将回答 RQ4：目录字典树能带来哪些增量收益？简而言之，目录字典树的结构使我们能够在第一遍中高效地分配词元，即内存，从而减少所需的回退遍数。目录字典树的两个属性使这一点成为可能。首先，内容编码仅需为 12,101 个项目寻址 1,670 万个单元。因此，78% 的深度 2 前缀和 98% 的深度 3 前缀仅包含一个项目，并且每个草稿中三分之二的候选前缀都位于链上。其次，目录骨架由字典树的 21,091 个内部节点中的 2,533 个组成，即每个一级簇约包含十个节点。

图 8 量化了这两个属性的影响。在面板 (a) 中，我们观察到，第 3 节的链规则在各种规模下都将第一遍削减至其词元数的三分之一，同时分数保持不变。面板 (b) 和 (c) 比较了解决开放簇的两种方式，即逐层处理和使用其整个骨架。只要内存足够，骨架轮次就不需要第三遍。它将训练模型上的 MF 草稿轮次从 2.2 降至 1.8，并将最平坦模型上的轮次从 2.9 降至 2.0。然而，骨架轮次需要输入约 7× 同样多的回退词元才能实现这一削减。此外，它几乎没有节省用户序列，因为仅凭链规则就已经使该草稿保持在接近两轮的水平。因此，只有当草稿将重簇保持开放时，完整输入一个骨架才会节省一轮，但代价是在回退轮次中增加额外词元。

5.5. 等资源比较（RQ5）

在无限内存条件下得到这些结果的基础上，我们现在比较两个解码器在资源相同的情况下的表现，并回答 RQ5：SIDECAR-K 是否至少与 BEAM-K 一样好？在图 9中，宽度为 W 的束搜索使用自身资源运行，即进行四轮，每轮使用 W 个前缀。BUDGET-SIDECAR 获得相同的资源，即每遍 $M = W$ 个节点，最多进行四遍，且所有遍（包括第一遍）都限制为 M 个节点。我们发现，在每种预算下，SIDECAR-K 至少与 BEAM-K 一样好；当模型较为平坦时，SIDECAR-K 的表现则好得多。此外，在每种预算下，SIDECAR 都会通过证书或公告因子说明其列表与 MODEL-K 有多接近，而束搜索则不会说明其列表与精确结果相差多远。

在束搜索的延迟下。 图 9 的顶行将通行次数固定为四，并改变每次通行的内存。在训练模型上，黑色柱和绿色柱

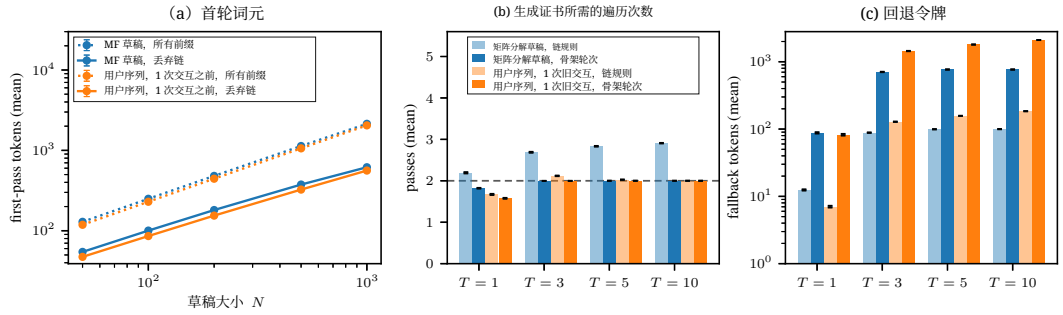


图 8. 目录字典树节省的开销。（a）舍弃链前缀后，第一遍所需的词元数减少到原来的三分之一。**（b、c）**骨架轮次从不需要第三遍，但仅对 MF 草稿最多节省一遍，并且其提供的回退令牌数量是原来的数倍。

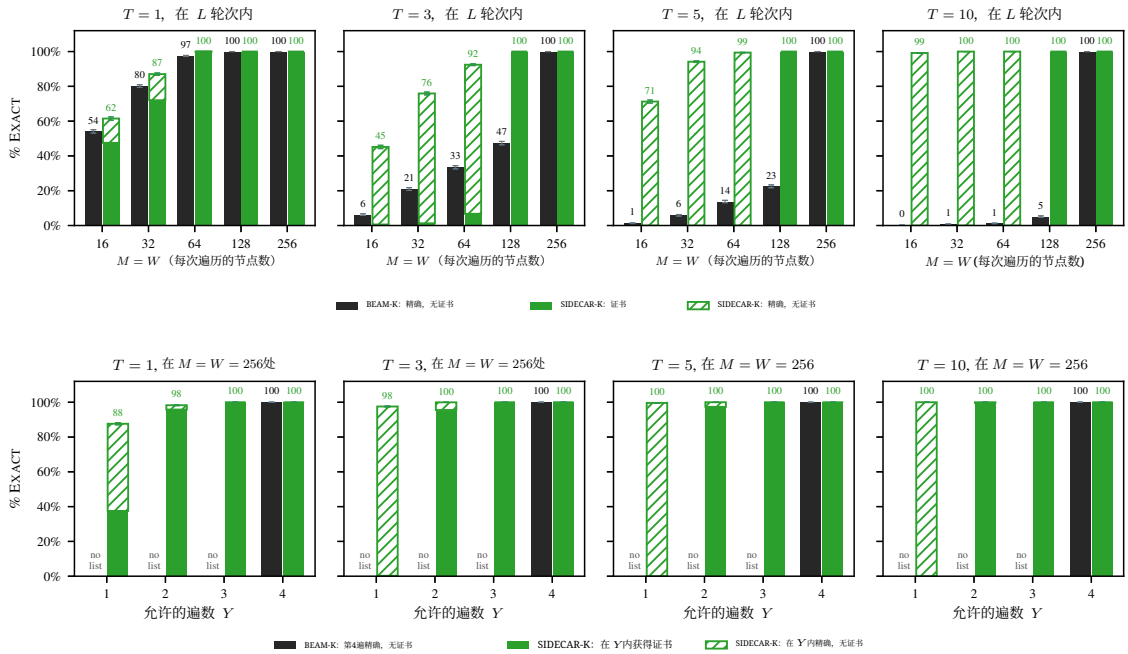


图 9. 与束搜索进行资源相等的比较。顶部：束搜索进行四轮时的延迟与内存的关系。**底部：**束搜索在 256 个节点时的内存与轮次的关系。在尖锐模型上，列表一致，且只有一个解码器知道这一点。在扁平模型上，SIDECAR-K 只需一小部分内存，并提前两轮即可实现精确。

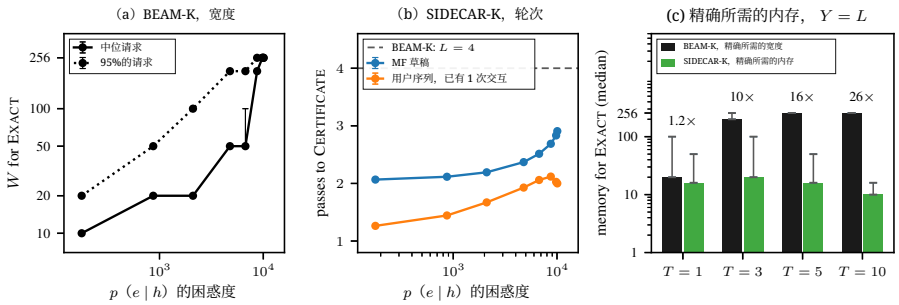


图10. 模型越趋平坦，精确性所需的代价越高。束搜索需要一个无法预先确定的束宽 (a)，而 SIDECAR 只需要有限轮次 (b)。在内存轴 (c) 上，扁平模型中的差距扩大到一个数量级以上。

二者共同上升，并在相同内存下覆盖每个请求。因此，BEAM-K 与 SIDECAR-K 重合，仅观察列表的部署方式将看不出任何差异。然而，差异在于绿色柱的实心部分，即 SIDECAR 已证明其列表精确的请求所占比例。从适中的内存开始，这一比例几乎覆盖所有请求。相比之下，束搜索在任何宽度下都没有这样的比例，因为其计算过程无法区分精确列表和错误列表。在扁平模型上，正确性本身就将两个解码器区分开来。在最小内存下，SIDECAR-K 对大多数请求都能给出精确结果，并且只需束搜索宽度的一小部分即可对每个请求都达到精确；而 BEAM-K 则需要完整宽度才能变得精确。需要注意的是，在较小内存下，SIDECAR 会先返回精确列表，然后才能证明该列表是精确的，如绿色柱中的阴影部分所示。这是因为开放簇无法在三次回退轮次内全部关闭。

束搜索的内存。 图 9 的最底行图 9将每次通行的内存固定为束搜索的宽度，并改变通行次数。单次遍历后，SIDECAR 已经能够在尖锐模型的大多数请求以及扁平模型的几乎所有请求上保留 MODEL-K。此时，束搜索尚未保留任何列表，因为它还需要再经过三轮才能输出列表。再经过一轮后，SIDECAR 已经为其几乎所有列表完成认证。在第四遍时，束搜索有一个单独的柱状条，表明在这一宽度下，其列表在这些请求上是精确的。因此，与等待束搜索完成四轮遍历的部署方案相比，在两轮遍历后提供 SIDECAR 列表的部署方案更具优势。

为什么平坦度有利于 SIDECAR-K，却会损害束搜索。 BUDGET-SIDECAR 的性能会随着底层模型平坦度的提高而改善，这看起来或许有悖直觉。下面我们解释其中的原因。这一规律源于决定平坦模型前K 项的因素。一个项目的概率是 L 个条件概率的乘积。随着模型趋于平坦，每个条件概率都趋近于可接受子节点数量的倒数，因此，一个项目的概率趋近于其路径上这些分支因子的倒数之积。于是，平坦模型的前K 项由小子树中的项目构成；这些项目与少数兄弟节点共享所在簇的概率，而与历史记录无关。事实上，在训练模型中，包含前10项的簇属于字典树中最重的簇。相比之下，在最平坦的模型中，这些簇的概率质量处于中位数水平，且它们的大小都低于中位数。这一机制有三种表现。首先，平坦模型的用户序列，也就是针对过时历史记录的排名，几乎必然包含当前历史记录对应的顶级项目，因为此时排名由目录的结构而非历史记录决定。因此，随着模型趋于平坦，陈旧性的影响越来越小，第一遍就能返回精确列表。其次，束搜索按照概率质量对前缀进行排名。因此，它会保留重簇并丢弃轻簇，尽管顶级项目所在的恰恰是轻簇。这就是为什么在平坦模型中，束搜索的柱形图会逐渐收缩，直到其宽度覆盖所有簇。第三，证书必须排除重簇。重簇的每个前沿前缀所占的概率质量都超过那些较小的顶级项目，尽管这些前缀中没有任何一个包含顶级项目；而要排除这些前缀，需要经过多个轮次。换言之，平坦度使概率质量的分布位置与字典树中最佳项目的位置彼此分离。

5.6. 精确性的代价（RQ6）

接下来，我们回答 RQ6：SIDECAR-K 的精确性需要付出什么代价？ 图10 改变模型的尖锐程度，并报告每个解码器输出精确列表所需的资源。我们发现，对于尖锐模型，稀缺资源是通行次数；而对于扁平模型，稀缺资源是每次通行的内存。在这两类模型上，SIDECAR 所需的都是较为稀缺的资源。

束搜索需要宽度，也就是为更宽的束分配内存（图 a）。平坦模型会将概率分散到许多概率质量近乎相等的前缀上，因此束必须足够宽，才能保留正确的前缀。在不同温度的扫描范围内，中位请求所需的宽度增长超过 10×。从平坦度达到中等水平开始，95%的请求所需的宽度就达到我们测试的最大宽度。此外，束搜索达到精确性所需的宽度取决于请求，而束搜索只有事后才能获知这一点，因此部署时必须按最坏情况为每个请求预留资源。相比之下，SIDECAR 需要额外的轮次才能达到精确性（图 b），且这一开销在设计上有界。对于用户序列，在扁平模型上轮次会稳定在两轮；使用 MF 草稿时，则会逐渐增至三轮。

图 (c) 从内存轴比较了两种解码器，因为每次通行的内存决定一台服务机器能同时处理多少请求，进而决定服务集群的规模。在训练模型上，两种解码器生成精确列表所需的内存大致相同。在此情况下，SIDECAR 在轮次方面更有优势：内存相同时，它所需的轮次不到束搜索的一半。然而，随着模型变得更加平坦，束搜索所需的内存会增加，而 SIDECAR 所需的内存则不会。两者的差距扩大到十倍以上；在最平坦的一端，差距为 10–26×。这对 BUDGET-SIDECAR 有利，差异显著，并会影响两种解码器的实用性。

图12 展示了同一平面上的两种资源。束搜索在内存与轮次构成的平面上是一个单点：它需要四轮，以及满足中位请求所需的宽度。相比之下，BUDGET-SIDECAR 是一条曲线，因为增加

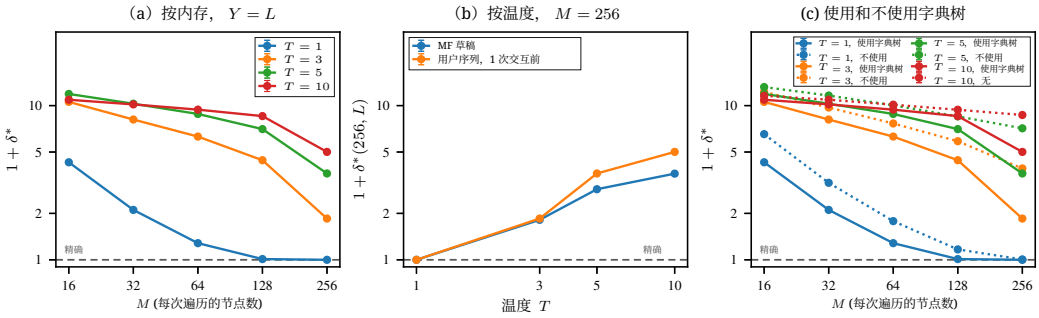


图11。公告因子：在四轮次时与内存的关系（a），以及在束搜索的内存设置下与温度的关系（b）。对于两种草稿，随着模型变得更加平坦，该因子都会上升；在束搜索的内存设置下，MF 草稿的该因子较低，因为其候选项分散在更多簇中，尽管在较小内存下该草稿达到精确的频率要低得多。所有预算下，实际差距均为零。（c）使用目录字典树（实线）时，该因子从不会比不使用目录字典树的基于计数的宣告（虚线，式（7））宽松；在束搜索宽度下，对于扁平模型，该因子约为其一半。

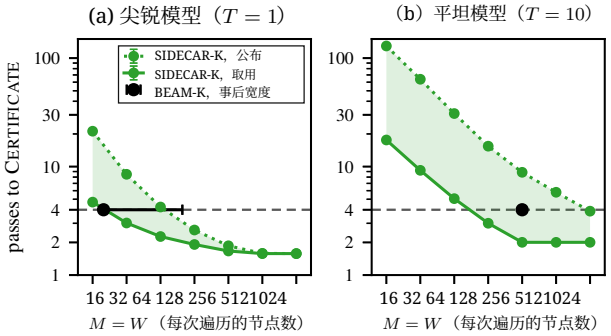


图12。认证列表的遍数与内存。 束搜索在其四遍处表现为一个点，而 BUDGET-SIDECAR 在两个模型上都是位于其左下方的一条曲线。

每次通行的内存会减少得到认证列表所需的轮次。在两个模型上，这条曲线都位于该点的左下方。因此，内存不足和延迟受限的部署都能从 SIDECAR 中获益。

5.7. 最坏情况保证的成本与收益（RQ7）

在测量了计划运行所达到的效果后，我们最后回答 RQ7：BUDGET-SIDECAR 的最坏情况保证带来了什么好处，又付出了什么代价？该保证是定理 2中的公告因子 $1 + \delta^*$，它在预算范围内对抗第4节所描述的对手时成立。非规划运行没有这样的保证：在最后一遍之前，它无法说明自己的列表距离 MODEL-K 还有多远。在最坏情况下，它可能完全返回 MODEL-K 中的任何项目，即使在计划运行时，同样的预算足以实现精确性（命题5）。我们发现，在训练模型上，该公告因子能够在束搜索的预算内达到精确性；并且只有当该因子过大而不具实用价值的预算下，这一保证才会降低与 MODEL-K 的重叠度。

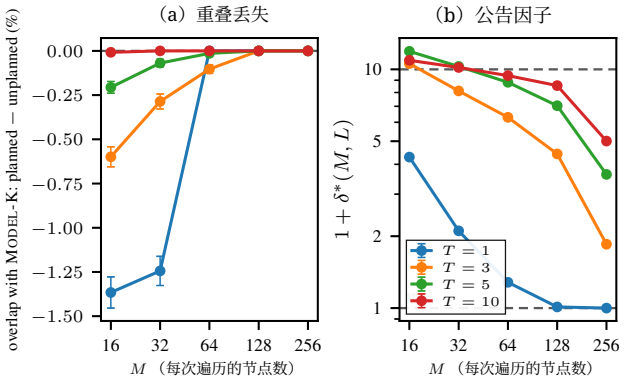


图 13。公告的代价，用户序列。**(a)**BUDGET-SIDECAR 的列表与 MODEL-K 的重叠度减去非规划运行的重叠度，以百分比表示。**(b)** 相同预算下的公告因子。代价至多为百分之一或百分之二，且仅出现在最小的内存配置下。

公告因子的大小。 图11展示了三个面板中的公告因子。在面板 (a) 中，通行次数固定为四次，与束搜索相同，而每次通行的内存会发生变化。在训练模型上，该因子在达到束搜索的宽度时降至一。因而，采用这一预算的部署在第一遍之后就知道，其列表将在四遍之内达到 MODEL-K。在扁平模型上，该因子随内存增加而下降得更慢，因为每个开放簇都承载着更多概率，对手可能将一个顶层项目隐藏在其下方的任意位置。

在面板(b)中，每次通行的内存固定为束搜索的宽度，温度则不断变化。随着模型变得更加平坦，该因子随之上升。MF 草稿的这一数值低于用户序列，因为行为草稿将其 500 个候选项分散到更多簇中，未覆盖的概率也更少。然而，如 RQ5 所示，精确性对这两种草稿的排名恰好相反。因此，公告衡量的是草稿未覆盖多少概率，而精确性取决于草稿是否包含顶级项目。对于用户序列，在上述每个预算下，实际差距均为

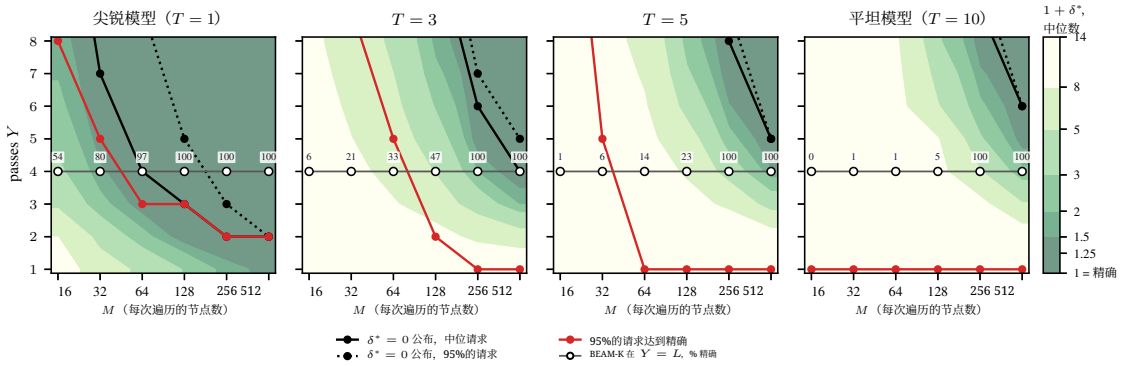


图 14. 预算图。 颜色：每种预算下第一遍后公布的因子，精确性最高时颜色最深。黑色：针对中位请求（实线）和 95%的请求（虚线），公布精确性时最小的 Y。红色：SIDECAR-K 对 95%的请求达到精确时最小的 Y。束搜索对应 $Y = L$ 这一行，并标出其精确列表所占比例。在尖锐模型上，公布的前沿与实际前沿重合；而在平坦模型上，两者则有所分离。

零，因此只有认证未能达标。

面板 (c) 将 定理 2 的公布结果与一种只使用每个前缀下项目数量的公布结果进行比较（命题 8 见附录 B）。纳入目录字典树绝不会使因子变差；在平坦模型上，使用束搜索宽度时，因子大致减半。原因在于，若不使用字典树，公布结果时必须假设每个开放簇都逐层得到解析，而这可能需要为其下的每个项目各用一个节点。使用字典树后，则可以在一轮中输入整个骨架来解析一个簇，只需大约十个节点。这是字典树带来的额外优势，除此之外还有 RQ4 中的链规则和骨架轮次。

为保证性而牺牲的重叠。 为了保证最坏情况，BUDGET-SIDECAR 在每一遍回退中都会将部分节点用于推测前缀，而真实模型可能并不需要这些前缀。非规划运行会将其全部节点用于已知处于开放状态的前缀，因此其列表与 MODEL-K 的重叠可能更多。图 13(a) 展示了这一差异。对于用户序列，从中等内存开始，两种运行的重叠程度相同；在低于该内存时，BUDGET-SIDECAR 的重叠减少了 1–2%。这种损失只发生在内存较小时，此时公告因子较大（图 13(b)）。即使在这种情况下，公告也会在第一遍之后告知部署方其内存过小，而公式（5）给出了其需要执行的回退轮次数，以便为其列表提供证明。需要注意的是，推测对重叠的帮助或损害，既取决于过程，也同样取决于模型。推测节点正是对手所需要的节点。因此，当重簇隐藏了 MODEL-K 的项目时，它们会改善列表；而当重簇不包含此类项目时，它们会略微降低重叠。在这个基准测试中，平坦模型的顶级项目位于轻簇中（RQ5），属于后一种情况。因此，部署方应公布该因子；这是它在回退运行之前能够作出的承诺，并在之后报告实际差距，该差距只会更小。

5.8. 总体景观

图14 用一张图概括了本节内容。图中每个点都代表每次遍历 M 个节点、共进行 Y 轮次的预算。该点的颜色表示 BUDGET-SIDECAR 在这一预算下完成第一遍后公告的因子。因此，对于部署设定的任何预算，该图都展示了回退机制启动前 BUDGET-SIDECAR 能够作出的承诺。图中展示了两条前沿。虚线表示 SIDECAR-K 在几乎所有请求上都能做到精确的位置，这一结果由 RQ5 和 RQ6 衡量。实线表示这种精确性还能在一次调用后得到公告的位置，这正是 RQ7 的保证。束搜索对应 $Y = L$ 这一行，并且在任何预算下都不会公告任何内容。

在尖锐模型上，两条前沿几乎重合，并且都以远低于束搜索所需宽度的比例穿过束搜索所在行。因此，在 BEAM-K 仅能做到精确的预算下，SIDECAR-K 不仅精确、会公告其精确性，而且所需轮次数也只占束搜索的一小部分。在平坦模型上，两条前沿则有所分离。对于图中显示的每种内存，实际前沿都降至单遍，而声明前沿则退至右上角。因此，在束搜索自身的预算下，保证体现为公告因子，而非精确性。处于两条前沿之间的区域内，SIDECAR-K 是精确的，但公告尚无法说明这一点。该区域在平坦模型上很宽，因为真实模型会将概率集中在草稿已经关注的位置，而公告必须考虑最坏情况，即对手将这些概率分散到其他位置。

因此，该图将实践者在部署时必须做出的两个决策区分开来。在虚线曲线以上，部署可以获得精确列表。在实线曲线以上，部署还可以在一次调用后承诺精确性。在两条曲线之间，部署会同时获得 SIDECAR-K

以及一个可以声明的因子。从实际角度来看，当模型较为尖锐时，BUDGET-SIDECAR 显然更为可取，因为它只需极少的轮次就能返回 BEAM-K 的列表并对其进行认证。当模型较为平坦时，内存是决定性资源，而 SIDECAR-K 只需束搜索一小部分内存，或只需其一半的轮次，即可实现精确。当这两种资源都不足以保证精确性时，BUDGET-SIDECAR 仍会说明 SIDECAR-K 与 MODEL-K 的接近程度。

提前停止。 证书和公告因子还允许部署以三种束搜索无法提供的方式提前停止。首先，解码器会在列表获得认证后立即停止。 图 9 的底部一行显示，在束搜索所需的内存下，几乎所有列表在两轮后都已获得认证，因此不会耗尽预算中剩余的轮次。其次，如果部署接受近似列表，就可以设定目标因子，并在列表获得认证且因子达到目标时立即停止。 图 14 中的颜色显示了每种预算下，中位请求经过首次调用后公告的因子，因此也显示了达到给定目标所需的预算。第三，如果部署有延迟期限，就可以在任意一轮后停止。 图 9 的底部一行显示，对于尖锐模型的大多数请求以及扁平模型几乎所有请求，仅一轮后的列表就已精确；而 图 12中的虚线则显示了首次调用后证书所需的轮数。在这三种情况下，部署也可以根据公告本身采取行动：如果首次调用后公告的因子超过目标，就可以立即停止，并按所声明的因子提供列表，或者将请求升级到更大的预算。

6. 相关工作

据我们所知，SIDECAR 是首个针对语义 ID 的解码器：它能够证明其列表是模型的精确 top-K，或者在预算约束下，在回退机制运行前给出其列表的因子保证。我们的工作与多个研究方向相关，下面将对此进行讨论。

生成式检索与语义 ID。一条研究路线将检索框架为生成文档标识符：可微搜索索引（Tay 等，2022）提出了这一思想，神经语料索引器（Wang 等，2022）和查询生成变体（Zhuang 等，2022）则将其发展应用于文档。后续研究探讨了其在语料库规模下的局限性（Pradeep 等，2023）、将其理解为基于前缀的稠密检索（Nguyen 和 Yates，2023），以及针对标识符的排名目标（Li 等，2024b；Zeng 等，2024）。TIGER（Rajput 等，2023）借助残差量化变分自编码器（Lee 等，2022）生成的编码，将语义 ID 引入推荐，其方式类似于神经编解码器中的残差量化（Zeghidour 等，2022），并通过受约束的束搜索对其进行解码。Geng 等（2022）研究了语言模型推荐系统的物品标识符；Hua 等（2023）。另一条研究路线则从协作信号中学习分词器（Jin 等，2024；Zheng 等，2024a；Wang 等，2024a；Zhu 等，2024；Liu 等，2025；Qu 等，2025；Si 等，2024；Wang 等，2024b）。此外，研究表明，语义 ID 能够提升排序模型的泛化能力（Singh 等，2024）。SID 树的深度仍是一个开放选择：增加一个层级可以在一定程度上提升推荐效果（Fang 等，2026）；更浅的 SID 提升了离线和在线指标（Xie 等，2026）；而更深的 TIGER 在束搜索下性能下降（Hou 等，2025），尽管深度会同时改变服务成本和模型。PLUM（He 等，2026）、GLIDE（D'Amico 等，2026）、OneRec（Deng 等，2025；Zhou 等，2025）、COBRA（Yang 等，2025）以及 Meta 的生成式推荐器（Zhai 等，2024；Yang 等，2024）等工业系统，均通过束搜索在 SID 字典树上进行解码；一项面向实践者的研究则总结了它们的设计选择（Ju 等，2025）。尽管这些工作改进了模型的排名对象，但它们与我们的工作正交，因为它们没有改变排名的解码过程。相比之下，我们保持模型及其分词器不变，仅改变解码器。我们的方法适用于上述任何一种分词器。

束搜索及其替代方案。Oh 等人（2026年）表明，与多词元预测（Gloeckle 等人，2024年）相比，单头配合束搜索能够获得更好的排名和内容质量；并行 SID 解码器（Hou 等人，2025年；Zhan 等人，2026年）则采用多词元预测以提高吞吐量。Wu 等人（2025年）表明，约束束搜索能够实现较高的 Top-1 精确率，但在 top-k 处对分支的召回较差。在语义 ID 出现之前，基于树的深度模型（Zhu 等人，2018年；2019年）和 Deep Retrieval（Gao 等人，2021年）通过在学习得到的节点分数上进行束搜索来完成检索，这一方法源自分层 softmax（Morin 与 Bengio，2005年）；Zhuo 等人（2020年）表明，在此类树上进行束搜索并非贝叶斯最优，并训练评分器进行补偿。这些工作都对模型进行调整以适应解码器；相比之下，我们采取相反的路径：保持模型训练完成后的状态不变，使解码器变得精确且具备认证保证。

受约束解码和精确搜索。 受约束解码将每个下一个词元限制为允许的选项。GENRE（De Cao 等人，2021年）使用实体名称的字典树实现了这一点，而 PICARD 和语法约束解码则使用语法实现了这一点（Scholak 等人，2021年；Geng 等，2023年）。解码语义 ID 是其中最简单的情况，因为每个项目在字典树中都是长度相同的一条路径，而前缀的概率就是其下方项目的总概率。因此，前缀的概率永远不会高于其父节点；精确搜索方法正是依赖这一事实。这些方法包括对单个最佳序列进行精确搜索（Stahlberg 和 Byrne，2019）以及最佳优先

束搜索（Meister 等人，2020b；a）。然而，它们搜索的是一个长度无界的序列，而保证精确性代价高昂。相比之下，我们在深度固定为 K 的字典树上搜索前 L 个项目的列表；对一个良好的草稿进行单次遍历通常就足以验证该列表，而 L 轮遍历总能做到这一点。类似的思想也出现在经典检索中，WAND 和块最大 WAND（Broder 等人，2003；Ding 和 Suel，2011）使用离线计算的分数上界来验证前 K 个项目的列表。相比之下，我们的上界来自对草稿进行评分的同一次前向传播，而回退机制只扩展违反这些上界的前缀。

推测解码。推测解码（Leviathan 等人，2023年；Chen 等人，2023年；Sun 等人，2023年）以低成本草拟词元，并通过单次遍历将其与目标模型进行核验。树注意力将这一方法扩展到整棵草稿树（Miao 等人，2024年；Cai 等人，2024年），后续研究则加入了目标模型上的草稿头（Ankner 等人，2024年；Li 等，2024a）、前瞻与自推测（Fu 等人，2024年；Zhang 等人，2024年）、分阶段的大小模型草稿器（Spector 和 Ré，2023年；Kim 等人，2023年）、基于检索的草稿（He 等人，2024年），以及在令牌预算约束下选择的草稿树（Chen 等人，2024年）。与我们的目标最为接近的，或许是用于生成式推荐的推测解码器：AtSpeed（Lin 等人，2025年）将推测解码用于束搜索，并训练了一个对齐的草拟器；NEZHA（Wang 等，2026b）增加了一个自推测头；SpecGR（Ding 等人，2026年）通过检索生成草稿，并依据似然阈值决定是否接受；STARK（Sun 等人，2026年）则让多个束共享 KV 缓存。这些解码器能够加速生成，但与我们不同，它们都无法为其列表提供认证。相比之下，我们使用现有草稿，无需训练；通过单次遍历为整份草稿评分，并对由此得到的输出列表进行认证，或使用回退机制进行修正。

预算约束下的搜索。我们的证书（定理 1）采用了 top-K 聚合（Fagin 等人，2003年）中阈值算法停止规则的形式，即，将已见到的第 K 高分与所有未见项目的得分上界进行比较。此外，隐藏概率（第 2 节）决定的请求相关成本，属于 Fagin 的实例最优性所描述的类型。类似地，任何要为其列表提供证书的解码器，都必须为概率超过第 K 高分项目得分的每个前缀评分；这与最佳优先搜索在任何可采纳启发式函数下都必须扩展的节点相呼应（Dechter 和 Pearl，1985年）。预算约束下轮数上界中的加性深度项，与探索结构事先未知的树时所需的下界形式相同（Fraigniaud 等人，2006年；Dereniowski 等人，2015年），任何探索规则都无法消除这一项。目录字典树消除了这种不确定性，因为解码器预先知道哪些前缀存在（Papadimitriou 和 Yannakakis，1991年）。因此，定理 2量化了了解字典树能带来多大帮助。最后，我们给出的因子保证与 anytime 和契约搜索中的保证属于同一类型（Likhachev 等人，2003年；Aine 等人，2010年），这类方法会返回一个解，其质量在预先声明的因子范围内接近最优；我们的计划则为该因子提供证书，而非对其进行估计。Sequoia（Chen 等人，2024年）同样在令牌预算下选择推测树，但依据的是预期接受率，而我们的规划方案针对的是最坏情况。

在线服务。我们的布局依赖于分页 KV 缓存（Kwon 等，2023年）、前缀共享（Zheng 等，2024b）、连续批处理（Yu 等，2022年），以及融合式和块稀疏注意力内核（Dao 等，2022年；Dong 等，2025年），这些技术与解码器正交，并且同样适用于 SIDECAR 和束搜索。

影响声明

本文提出了一种面向生成式推荐器的解码过程：在减少顺序模型调用成本的同时，返回与模型自身排名相同的列表，并提供一份证书，说明该列表是否精确，或其与精确结果的接近程度。它不会改变推荐内容，只会改变模型排名生成的速度和可靠性。更低的解码成本能够降低此类推荐器的在线服务成本，而证书则使其在给定预算下的行为可被审计，我们认为这是积极的影响。推荐系统本身的社会影响，包括其对曝光度和多样性的影响，应由模型及部署该模型的技术栈负责，且不会因解码器而改变。

参考文献

Aine, S., Chakrabarti, P. P., 和 Kumar, R. 契约搜索：在节点扩展约束下进行启发式搜索。收录于*ECAI 2010 – 第 19 届欧洲人工智能会议，人工智能与应用前沿*，第733–738页。IOS Press，2010年。doi: 10.3233/978-1-60750-606-5-733。

Ankner, Z., Parthasarathy, R., Nrusimha, A., Rinard, C., Ragan-Kelley, J., 和 Brandon, W. Hydra：用于 Medusa 解码的顺序依赖型草稿头。收录于 第一届语言建模会议，2024年。网址：https://openreview.net/forum?id=FbhjirzvJG。

Broder, A. Z., Carmel, D., Herscovici, M., Soffer, A., 和 Zien, J. 使用两层检索流程进行高效查询评估。收录于第十二届国际信息与知识管理会议论文集，第426–434页。ACM，2003年。doi：10.1145/956863.956944。

Cai, K.、Li, Y.、Geng, Z.、Peng, H.、Lee, J. D.、Chen, D. 和 Dao, T. Medusa：具有多个解码头的简单大语言模型推理加速框架。载于第*41*届国际机器学习会议论文集，机器学习研究论文集第*235*卷，第5209–5235页。PMLR，

2024年。网址：
https://proceedings.mlr.press/v235/cai24b.html。Chen, C.、Borgeaud, S.、Irving, G.、Lespiau, J.-B.、Sifre, L. 和 Jumper, J. 使用推测采样加速大语言模型解码。*arXiv*预印本 *arXiv:2302.01318*，2023年。Chen, Z.、May, A.、Svirschevski, R.、Huang, Y.、Ryabinin, M.、Jia, Z. 和 Chen, B. Sequoia：可扩展且稳健的推测解码。载于神经信息处理系统进展，第37卷，第129531–129563页，2024年。doi: 10.52202/079017-4116。D'Amico, E.、De Nadai, M.、Chandar, P.、Vohra, D.、Lin, S.、Lefarov, M.、Gigioli, P.、Penha, G.、Kopysitsky, I.、Senese, I. J.、Mei, D.、Fabbri, F.、Semerci, O.、Zhao, Y.、Tang, V.、St. Thomas, B.、Ranieri, A.、Smith, M. N. K.、Bernkopf, A.、Leung, B.、Fazelnia, G.、VanMiddlesworth, M.、Heath, T. C.、Skiden, P. P.、Wang, A. Y.、Cole, D. J.、Damianou, A.、Hristakeva, M.、Wilbur, R.、Chillara, T.、Radosavljevic, V.、Chitkara, P.、Adapa, S.、Elenter, J.、Huber, B.、Wood, J.、Vedantam, S.、Stypka, J.、Ghael, S.、Gould, M. D.、Murgatroyd, D.、Raimond, Y.、Lalmas, M. 和 Bennett, P. N. 从习惯到发现：在 Spotify 部署用于个性化生成式推荐的大语言模型。载于第*32*届*ACM SIGKDD*知识发现与数据挖掘会议第二卷，第7141–7150页。ACM，2026年。doi: 10.1145/3770855.3818419。Dao, T.、Fu, D. Y.、Ermon, S.、Rudra, A. 和 Ré, C. FlashAttention：具备输入输出感知能力的快速且内存高效的精确注意力机制。载于神经信息处理系统进展，第35卷，第16344–16359页。Curran Associates, Inc.，2022年。De Cao, N.、Izacard, G.、Riedel, S. 和 Petroni, F. 自回归实体检索。载于第*9*届国际学习表征会议（*ICLR 2021*），2021年。Dechter, R. 和 Pearl, J. 广义最佳优先搜索策略与 A* 的最优性。*ACM*学报，32(3)：505–536，1985年。doi: 10.1145/3828.3830。Deng, J.、Wang, S.、Cai, K.、Ren, L.、Hu, Q.、Ding, W.、Luo, Q. 和 Zhou, G. OneRec：利用生成式推荐系统和迭代式偏好对齐统一检索与排名。*arXiv*预印本 *arXiv:2502.18965*，2025年。Dereniowski, D.、Disser, Y.、Kosowski, A.、Paj.ak, D. 和 Uzna´nski, P. 快速协作式图探索。信息与计算，243：37–49，2015年。doi: 10.1016/j.ic.2014.12.005。Ding, S. 和 Suel, T. 使用块最大索引实现更快速的顶*k* 级文档检索。载于第*34*届国际*ACM SIGIR*信息检索研究与发展会议论文集，第993–1002页。ACM，2011年。doi: 10.1145/2009916.2010048。Ding, Y.、Li, J.、McAuley, J. 和 Hou, Y. 归纳式生成

通过基于检索的推测实现生成式推荐。在 *AAAI* 人工智能会议论文集，第 40 卷，第 14675–14683 页，2026年。doi: 10.1609/aaai.v40i17.38486。Dong, J.、Feng, B.、Guessous, D.、Liang, Y. 和 He, H. Flex Attention：用于生成优化注意力内核的编程模型。在机器学习与系统论文集，第 7 卷，2025年。网址：https://arxiv.org/abs/2412.05496。Fagin, R.、Lotem, A. 和 Naor, M. 面向中间件的最优聚合算法。计算机与系统科学杂志，66(4)：614–656，2003年。doi: 10.1016/ S0022-0000(03)00026-6。Fang, D.、Gao, J.、Li, Y.、Zhao, X. 和 Chang, Y. PRISM：用于生成式序列推荐的纯化表示与集成语义建模。*arXiv* 预印本 *arXiv:2601.16556*，2026 年。Fraigniaud, P. 、 G. asieniec, L.、Kowalski, D. R. 和 Pelc, A. 集体树探索。网络，48(3)：166–177，2006年。doi: 10.1002/net.20127。Fu, Y.、Bailis, P.、Stoica, I. 和 Zhang, H. 使用前瞻解码打破大语言模型推理的序列依赖。在 第 *41* 届国际机器学习会议论文集，第 235 卷，收录于 机器学习研究论文集，第 14060–14079 页。PMLR，2024年。网址：
https://proceedings.mlr.press/v235/fu24a.html。Gao, W.、Fan, X.、Wang, C.、Sun, J.、Jia, K.、Xiao, W.、Ding, R.、Bin, X.、Yang, H. 和 Liu, X. 学习大规模推荐中的端到端检索结构。在第 *30* 届 *ACM* 国际信息与知识管理会议论文集，第 524–533 页。ACM，2021年。doi: 10.1145/3459637.3482362。Geng, S.、Liu, S.、Fu, Z.、Ge, Y. 和 Zhang, Y. 将推荐视为语言处理（RLP）：统一的预训练、个性化提示与预测范式（P5）。在 第 *16* 届 *ACM* 推荐系统会议论文集，第 299–315 页。ACM，2022年。doi: 10.1145/3523227.3546767。Geng, S.、Josifoski, M.、Peyrard, M. 和 West, R. 无需微调即可用于结构化自然语言处理任务的语法约束解码。在 *2023* 年自然语言处理实证方法会议论文集，第 10932–10952 页，新加坡，2023年。计算语言学协会。doi: 10.18653/v1/2023.emnlp-main. 674。Gloeckle, F.、Idrissi, B. Y.、Rozière, B.、Lopez-Paz, D. 和 Synnaeve, G. 通过多词元预测实现更优、更快速的大语言模型。在 第 *41* 届国际机器学习会议论文集，第 235 卷，收录于机器学习研究论文集，第 15706–15734 页。PMLR，2024年。网址：https://proceedings.

mlr.press/v235/gloeckle24a.html .
He, R. 和 McAuley, J. 起伏：使用单类协同过滤建模时尚趋势的视觉演变。在第*25*届国际万维网大会论文集，第507–517 页。国际万维网会议指导委员会，2016年。doi：10.1145/2872427.2883037。

He, R., Heldt, L., Hong, L., Keshavan, R., Mao, S., Mehta, N., Su, Z., Tsai, A., Wang, Y., Wang, S.-C., Yi, X., Baugher, L., Cakici, B., Chi, E., Goodrow, C., Han, N., Ma, H., Rosales, R., Van Soest, A., Tandon, D., Wu, S.-L., Yang, W., and Zheng, Y. PLUM：为工业规模的生成式推荐改造预训练语言模型。载于 *2026*年 *ACM Web Conference*论文集，第 8093–8104 页。ACM，2026年。doi：10.1145/3774904.3792802。He, Z., Zhong, Z., Cai, T., Lee, J. D., and He, D. REST：基于检索的推测解码。载于*2024*年计算语言学协会北美分会年会论文集：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年。计算语言学协会。doi：10.18653/v1/2024.naacl-long.88。Hou, Y., Li, J., Shin, A., Jeon, J., Santhanam, A., Shao, W., Hassani, K., Yao, N., and McAuley, J. 并行生成长语义 ID 以用于推荐。载于第*31*届 *ACM SIGKDD* 知识发现与数据挖掘大会论文集（第 *2* 卷），第 956–966 页。ACM，2025年。doi：10.1145/3711896.3736979。Hu, Y., Wang, D., Li, N., Li, K., She, J., Guo, C., Cai, K., Liu, B., Liu, Y., and Chen, X. 效用感知生成式推荐：效用增强语义编码与效用引导解码。*arXiv* 预印本 *arXiv:2611.02981*，2026 年。Hua, W., Xu, S., Ge, Y., and Zhang, Y. 如何为推荐基础模型构建物品 ID 索引。载于 亚太地区年度国际 *ACM SIGIR* 信息检索研究与发展会议论文集，第 195–204 页。ACM，2023年。doi：10.1145/3624918.3625339。Jin, B., Zeng, H., Wang, G., Chen, X., Wei, T., Li, R., Wang, Z., Li, Z., Li, Y., Lu, H., Wang, S., Han, J., and Tang, X. 语言模型作为语义索引器。载于 第*41*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 22244–22259 页。PMLR，2024年。网址：
https://proceedings.mlr.press/v235/jin24h.html。
Ju, C. M., Collins, L., Neves, L., Kumar, B., Wang, L. Y., Zhao, T., and Shah, N. 基于语义 ID 的生成式推荐：从业者手册。载于 论文集

第*34*届 *ACM* 国际信息与知识管理会议论文集，第6420–6425页。ACM，2025年。doi：10.1145/3746252.3761612。Kim, S., Mangalam, K., Moon, S., Malik, J., Mahoney, M. W., Gholami, A., and Keutzer, K. 使用大小解码器进行推测解码。载于 神经信息处理系统进展，第36卷，第39236–39256页。Curran Associates, Inc.，2023年。Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. 使用分页注意力高效管理大语言模型服务的内存。载于 第*29*届操作系统原理研讨会论文集，第611–626页。ACM，2023年。doi：10.1145/3600006.3613165。Lee, D., Kim, C., Kim, S., Cho, M., and Han, W.-S. 使用残差量化进行自回归图像生成。载于 *IEEE/CVF*计算机视觉与模式识别会议（*CVPR*）论文集，第11513–11522页，2022年。doi：10.1109/CVPR52688.2022.01123。Leviathan, Y., Kalman, M., and Matias, Y. 通过推测解码实现 Transformer 的快速推理。载于第*40*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。网址：
https://proceedings.mlr.press/v202/leviathan23a.html。
Li, Y., Wei, F., Zhang, C., and Zhang, H. EAGLE：推测采样需要重新思考特征不确定性。载于第*41*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，2024a。网址：
https://proceedings.mlr.press/v235/li24bt.html。
Li, Y., Yang, N., Wang, L., Wei, F., and Li, W. 生成式检索中的学习排序。载于*AAAI*人工智能会议论文集，第38卷，第8716–8723页，2024b。doi：10.1609/aaai.v38i8.28717。Likhachev, M., Gordon, G. J., and Thrun, S. ARA*：具有次优性可证明界的任意时刻 A*。载于神经信息处理系统进展，第16卷，第767–774页。麻省理工学院出版社，2003年。Lin, X., Yang, C., Wang, W., Li, Y., Du, C., Feng, F., Ng, S.-K., a nd Chua, T.-S. 基于大语言模型的生成式推荐的高效推理。载于第十三届国际学习表征会议，2025年。Liu, E., Zheng, B., Ling, C., Hu, L., Li, H., and Zhao, W. X. 采用端到端可学习物品标记化的生成式推荐系统。载于第*48*届国际 *ACM SIGIR* 信息检索研究与发展会议论文集，第729–739页。ACM，2025年。doi：10.1145/3726302.3729989。

McAuley, J., Targett, C., Shi, Q., and van den Hengel, A. 关于风格与替代品的基于图像的推荐。载于第*38*届国际 *ACM SIGIR* 信息检索研究与发展会议论文集，第43–52页。ACM，2015年。doi：10.1145/2766462.2767755。Meister, C., Cotterell, R., and Vieira, T. 如果束搜索是答案，那么问题是什么？载于 *2020*年自然语言处理经验方法会议（*EMNLP*）论文集，第2173–2185页。计算语言学协会，2020a。doi：10.18653/v1/2020.emnlp-main.170。Meister, C., Cotterell, R., and Vieira, T. 最佳优先束搜索。计算语言学协会汇刊，8:795–809，2020b。doi：10.1162/tacl_a 00346。_Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Wang, Z., Zhang, Z., Wong, R. Y. Y., Zhu, A., Yang, L., Shi, X., Shi, C., Chen, Z., Arfeen, D., Abhyankar, R., and Jia, Z. SpecInfer：利用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* 国际编程语言与操作系统架构支持会议论文集（第*3*卷），第932–949页。ACM，2024年。doi：10.1145/3620666.3651335。Morin, F. and Bengio, Y. 分层概率神经网络语言模型。载于第十届人工智能与统计国际研讨会论文集，第R5卷，机器学习研究论文集，第246–252页。PMLR，2005年。网址：
h
t
t
p
s://
proceedings.mlr.press/r5/morin05a.html。
Nguyen, T. and Yates, A. 将生成式检索视为稠密检索。载于*Gen-IR@SIGIR 2023*：首届生成式信息检索研讨会，2023年。arXiv:2306.11397。Oh, S., Amat Gil, F., Smothers, E., Iter, D., Thornburg, M., Argaw, D. M., Medapati, S., Wang, T., Verma, R., Xu, Q., Syal, H., Gholami, E., Bhattacharya, M., and Rastogi, A. 基于语义 ID 的个性化大语言模型推荐：自回归束搜索与多词元预测。*arXiv*预印本 *arXiv:2610.15864*，2026年。已被 NeurIPS 2026 PALM 研讨会接收。Papadimitriou, C. H. and Yannakakis, M. 无需地图的最短路径。理论计算机科学，84(1): 127–150，1991年。doi：10.1016/0304-3975(91)90263-2。Pradeep, R., Hui, K., Gupta, J., Lelkes, A., Zhuang, H., Lin, J., Metzler, D., and Tran, V. 生成式检索如何扩展到数百万篇段落？载于*2023* 年自然语言处理实证方法会议论文集，第1305–1321页，新加坡，2023年。计算语言学协会。doi：10.18653/v1/2023.emnlp-main.83.
Qu, H., Fan, W., Zhao, Z., 和 Li, Q. TokenRec：学习为基于大语言模型的生成式推荐标记化 ID。*IEEE* 知识与数据工程汇刊, 37(10):6216–6231, 2025年。doi: 10.1109/TKDE.2025. 3599265.

Rajput, S., Mehta, N., Singh, A., Hulikal Keshavan, R., Vu, T., Heldt, L., Hong, L., Tay, Y., Tran, V. Q., Samost, J., Kula, M., Chi, E. H. 和 Sathiamoorthy, M. 生成式检索推荐系统。在《神经信息处理系统进展》第36卷，第10299–10315页。Curran Associates, Inc.，2023年。Scholak, T., Schucher, N. 和 Bahdanau, D. PICARD：用于语言模型受约束自回归解码的增量解析。在《*2021*年自然语言处理经验方法会议论文集》中，第9895–9901页，线上及多米尼加共和国蓬塔卡纳，2021年。计算语言学协会。doi：10.18653/v1/2021.emnlp-main.779。Si, Z., Sun, Z., Chen, J., Chen, G., Zang, X., Zheng, K., Song, Y., Zhang, X., Xu, J. 和 Gai, K. 基于语义树结构标识符和对比学习的生成式检索。在《*2024*年亚太地区国际 *ACM SIGIR* 信息检索研究与发展会议论文集》中，第154–163页。ACM，2024年。doi：10.1145/3673791.3698408。Singh, A., Vu, T., Mehta, N., Keshavan, R., Sathiamoor- thy, M., Zheng, Y., Hong, L., Heldt, L., Wei, L., Tan- don, D., Chi, E. 和 Yi, X. 利用语义 ID 实现更好的泛化：推荐排名案例研究。在《第*18*届*ACM*推荐系统会议论文集》中，第1039–1044页。ACM，2024年。doi：10.1145/3640457.3688190。Spector, B. 和 Ré, C. 通过分阶段推测解码加速大语言模型推理。在《基础模型高效系统研讨会（*ES-FOMO*），第*40*届国际机器学习会议》中，2023年。arXiv：2308.04623。Stahlberg, F. 和 Byrne, B. 关于神经机器翻译搜索错误和模型错误：你舌头被猫叼走了吗？在《*2019*年自然语言处理经验方法会议和第*9*届国际自然语言处理联合会议（*EMNLP-IJCNLP*）论文集》中，第3356–3362页，中国香港，2019年。计算语言学协会。doi：10.18653/v1/D19-1331。Sun, J., Yang, J., Guo, Z., Xu, D., Yan, J., Deng, L. 和 Zeng, X. UniSGR：语义 ID 生成与排名的统一框架。《*arXiv* 预印本 *arXiv:2607.04068*》，2026年。Sun, Z., Suresh, A. T., Ro, J. H., Beirami, A., Jain, H.,

和 Yu, F. SpecTr：通过最优传输实现快速推测解码。在《神经信息处理系统进展》第36卷，第30222–30242页。Curran Associates, Inc.，2023年。doi: 10.52202/075280-1314。Tay, Y., Tran, V. Q., Dehghani, M., Ni, J., Bahri, D., Mehta, H., Qin, Z., Hui, K., Zhao, Z., Gupta, J., Schuster, T., Cohen, W. W. 和 Metzler, D. 将 Transformer 记忆作为可微搜索索引。在《神经信息处理系统进展》第35卷，第21831–21843页。Curran Associates, Inc.，2022年。Wang, W., Bao, H., Lin, X., Zhang, J., Li, Y., Feng, F., Ng, S.-K. 和 Chua, T.-S. 面向生成式推荐的可学习物品标记化。在《第*33*届 *ACM* 国际信息与知识管理会议论文集》中，第2400–2409页。ACM，2024a。doi: 10.1145/3627673.3679569。Wang, Y., Hou, Y., Wang, H., Miao, Z., Wu, S., Sun, H., Chen, Q., Xia, Y., Chi, C., Zhao, G., Liu, Z., Xie, X., Sun, H. A., Deng, W., Zhang, Q. 和 Yang, M. 用于文档检索的神经语料索引器。在《神经信息处理系统进展》第35卷，第25600–25614页。Curran Associates, Inc.，2022年。Wang, Y., Xun, J., Hong, M., Zhu, J., Jin, T., Lin, W., Li, H., Li, L., Xia, Y., Zhao, Z. 和 Dong, Z. EAGER：具有行为-语义协同的双流生成式推荐系统。在《第*30*届 *ACM SIGKDD* 知识发现与数据挖掘会议论文集》中，第3245–3254页。ACM，2024b。doi: 10.1145/3637528. 3671775。Wang, Y., Liu, Z., Tang, G., Yang, Y., Su, S., Liu, J., Zhou, N., Zhou, X., Jiang, W., Liang, J., Lv, X., Tang, R., Hong, L. 和 Ou, W. 多解码器 OneRec：面向多目标工业推荐的可控生成式检索。《*arXiv* 预印本 *arXiv:2607.26500*》，2026a。Wang, Y., Zhou, S., Lu, J., Liu, Z., Liu, L., Wang, M., Zhang, W., Li, F., Su, W., Wang, P., Xu, J. 和 Zhao, X. NEZHA：用于生成式推荐的零牺牲超高速解码架构。在《*2026*年 *ACM Web Conference* 论文集》中，第8073–8082页。ACM，2026b。doi: 10.1145/3774904.3792797。Wu, S., Ren, Z., Xin, X., Yang, J., Zhang, M., Chen, Z., de Rijke, M. 和 Ren, P. 受约束的自回归解码约束生成式检索。在《第*48*届国际 *ACM SIGIR* 信息检索研究与发展会议论文集》中，第2429–2440页。ACM，2025年。doi: 10.1145/3726302. 3729934。Xie, T., Ku, X., Sun, M., Sha, Y., Wang, L., Wang, P., Wang, Y., Wu, W., Liu, Z., Jiang, P. 和 Ou, W. 从静态多级小型语义码本到用于生成式推荐的动态单级大型语义码本。《style id='17'>arXiv 预印本 arXiv:2608.21012》，2026年。Xue, B., Liu, D., Wang, L., Sun, M., Wang, P., Zhang, P., Shi, S., Xu, T., Sha, Y., Liu, Z., Kong, B., Wang, B., Yang, H., Xue, J., Wang, J., Wang, S., Hui, S., Ye, W., Lin, X., Li, Y., Chen, Y., Yin, Z., Chen, Q., Wen, S., Wu, W., Li, H., Zhou, G., Li, C., Jiang, P., and Gai, K. 面向大规模广告的生成式推荐。In 第*32*届*ACM SIGKDD*知识发现与数据挖掘会议第*2*卷, pp. 8346– 8357, 2026. Yang, L., Paischer, F., Hassani, K., Li, J., Shao, S., Li, Z. G., He, Y., Feng, X., Noorshams, N., Park, S., Long, B., Nowak, R. D., Gao, X., and Eghbalzadeh, H. 统一序列推荐中的生成式检索与稠密检索。 *arXiv*预印本 *arXiv:2411.18814*, 2024. Yang, Y., Ji, Z., Li, Z., Li, Y., Mo, Z., Ding, Y., Chen, K., Zhang, Z., Li, J., Li, S., and Liu, L. 稀疏与稠密相结合：基于级联稀疏-稠密表示的统一生成式推荐。In神经信息处理系统进展, volume 38, pp. 104045–104069, 2025. doi: 10.52202/085713-3135. Yu, G.-I., Jeong, J. S., Kim, G.-W., Kim, S., and Chun, B.- G. Orca：面向Transformer生成模型的分布式服务系统。In 第*16*届*USENIX*操作系统设计与实现研讨会（*OSDI 22*）, pp. 521–538, 加利福尼亚州卡尔斯巴德, 2022. USENIX协会.
U
R
L
h
t
t
p
s://
w
ww.usenix.org/conference/osdi22/presentation/yu.
Zeghidour, N., Luebs, A., Omran, A., Skoglund, J., and Tagliasacchi, M. SoundStream：端到端神经音频编解码器。*IEEE/ACM*音频、语音与语言处理汇刊, 30:495–507, 2022. doi: 10. 1109/TASLP.2021.3129994. Zeng, H., Luo, C., Jin, B., Sarwar, S. M., Wei, T., and Zamani, H. 可扩展且高效的生成式信息检索。In *2024*年*ACM*网络会议论文集, pp. 1441–1452. ACM, 2024. doi: 10.1145/3589334. 3645477. Zhai, J., Liao, L., Liu, X., Wang, Y., Li, R., Cao, X., Gao, L., Gong, Z., Gu, F., He, J., Lu, Y., and Shi, Y. 行动胜于言辞：用于生成式推荐的万亿参数序列转换器。In 第*41*届国际机器学习会议, volume 235 of机器学习研究论文集, pp. 58484–58509.PMLR,2024.
U
R
L
https://proceedings.mlr.press/v235/zhai24a.html.
Zhan, T., Fu, K., Lv, C., Lv, Z., and Zhang, S. 用于生成式推荐的语义裁剪与辅助多步预测。 *arXiv*预印本 *arXiv:2604.05329*, 2026. Zhang, J., Wang, J., Li, H., Shou, L., Chen, K., Chen, G., and Mehrotra, S. 起草与验证：无损大

语言模型通过自推测解码实现加速。In 第*62*届计算语言学协会年会论文集（第*1*卷：长篇论文）, pp. 11263–11282, 泰国曼谷, 2024. 计算语言学协会. doi: 10.18653/v1/2024.acl-long.607. Zheng, B., Hou, Y., Lu, H., Chen, Y., Zhao, W. X., Chen, M., and Wen, J.-R. 通过融合协同语义使大语言模型适用于推荐。In *2024*年*IEEE*第*40*届国际数据工程会议（*ICDE*）, pp. 1435–1448. IEEE, 2024a. doi: 10.1109/ICDE60146.2024.00118. Zheng, L., Yin, L., Xie, Z., Sun, C., Huang, J., Yu, C. H., Cao, S., Kozyrakis, C., Stoica, I., Gonzalez, J. E., Bar- rett, C., and Sheng, Y. SGLang：高效执行结构化语言模型程序。In 神经信息处理系统进展, volume 37, pp. 62557–62583. Curran Associates, Inc., 2024b. doi: 10.52202/079017-2000. Zhou, G., Deng, J., Zhang, J., Cai, K., Ren, L., Luo, Q., Wang, Q., Hu, Q., Huang, R., Wang, S., Ding, W., Li, W., Luo, X., Wang, X., Cheng, Z., Zhang, Z., Zhang, B., Wang, B., Ma, C., Song, C., Wang, C., Wang, D., Meng, D., Yang, F., Zhang, F., Jiang, F., Zhang, F., Wang, G., Zhang, G., Li, H., Hu, H., Lin, H., Cheng, H., Cao, H., Wang, H., Huang, J., Chen, J., Liu, J., Jia, J., Gai, K., Hu, L., Zeng, L., Yu, L., Wang, Q., Zhou, Q., Wang, S., He, S., Yang, S., Yang, S., Huang, S., Wu, T., He, T., Gao, T., Yuan, W., Liang, X., Xu, X., Liu, X., Wang, Y., Wang, Y., Liu, Y., Song, Y., Zhang, Y., Wu, Y., Zhao, Y., and Liu, Z. OneRec技术报告。*arXiv*预印本 *arXiv:2506.13695*, 2025. Zhu, H., Li, X., Zhang, P., Li, G., He, J., Li, H., and Gai, K. 面向推荐系统的基于树的深度模型学习。I n 第*24*届*ACM SIGKDD*知识发现与数据挖掘国际会议论文集, pp. 1079–1088. ACM, 2018. doi: 10.1145/3219819.3219826. Zhu, H., Chang, D., Xu, Z., Zhang, P., Li, X., He, J., Li, H., Xu, J., and Gai, K. 面向推荐系统的基于树的索引与深度模型联合优化。In 神经信息处理系统进展, volume 32. Curran Associates, Inc., 2019. Zhu, J., Jin, M., Liu, Q., Qiu, Z., Dong, Z., and Li, X. CoST：用于生成式推荐的基于对比量化的语义标记化。In 第*18*届*ACM*推荐系统会议论文集, pp. 969–974. ACM, 2024. doi: 10.1145/3640457.3688178. Zhuang, S., Ren, H., Shou, L., Pei, J., Gong, M., Zuccon, G., and Jiang, D. 通过查询生成弥合可微搜索索引中索引与检索之间的差距。*arXiv*预印本 *arXiv:2206.10128*, 2022.

Zhuo, J., Xu, Z., Dai, W., Zhu, H., Li, H., Xu, J. 和 Gai, K. 在束搜索下学习最优树模型。载于第*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 11650–11659 页。PMLR，2020 年。网址：https://proceedings.mlr.press/v119/zhuo20a.html。

A. 假设与证明

在本附录中，我们陈述本文结果成立所依据的假设，并按照这些结果在正文中出现的顺序加以证明。每项假设都会在正文中首次需要时提出。

假设。 在整个附录中，我们作出如下假设。

A1 每个 SID 都恰好包含 L 个编码，且目录字典树是前缀闭包的。**A2** 分数仅针对某个前缀的可接受子节点计算，并且在这些子节点上的总和为一；因此，某个前缀的概率等于其下方项目的总概率。**A3** 节点处树遍历的输出等于对该节点路径单独进行一遍遍历所得的输出。**A4** 各陈述所比较的概率之间不存在并列。**A5** 第一遍之后至少会对 K 个项目进行评分，因此 $s_K(0) > 0$。**A6** 除非另有预算说明，否则每个回退轮次都会扩展所有开放前缀。

前沿。 下面的每个证明都使用前沿的以下性质，我们先陈述并证明该性质 .

引理 1 (前沿)。 经过任意轮次后，每个项目要么具有精确分数，要么位于恰好一个前沿前缀之下；在后一种情况下，其概率至多等于该前沿前缀的概率。

证明。 固定一个项目 e，从根节点开始沿其 SID 向下追踪。根节点视为已扩展，追踪至 e 中第一个未扩展的前缀。分两种情况。

(a) 如果 e 中深度不超过 $L-1$ 的所有前缀均已扩展，那么 e 的深度为 $L-1$ 的前缀也已扩展，因此 e 具有精确分数。(b) 否则， e 中第一个未扩展的前缀有一个已扩展的父节点，因此它是一个前沿前缀。根据 A2， e 的概率至多等于该前缀的概率。

此外，任何前沿前缀都不是另一个前沿前缀的祖先节点，因为前沿前缀尚未扩展，而每个前沿前缀的父节点都已扩展。因此，前沿前缀覆盖的项目集合互不相交，且情况 (b) 中找到的前沿前缀是唯一的。

□

定理 *1* 的证明定理 *1*。 设 e 是 E_r之外的一个项目。根据 引理 1， e位于某个前沿前缀 b之下，因此

$$p(e) \;\le\; p(b) \;\le\; U_r \;\le\; s_K(r),$$

其中，第一个不等式由引理 1得出，第二个不等式由 U_r 的定义得出，即 U_r 是最大的前沿概率，第三个不等式则由该定理的假设得出。因此， E_r 之外的每个项目的概率至多为 K中的第 K高分数，并且由于 A4 排除了并列情况，其排名低于 K 中的最佳项目。因此， E_r 的前K 名就是目录中的前K 名，即 SIDECAR-K$_r$ = MODEL-K。

□

命题 *1* 的证明命题 *1*。 该命题包含三个主张，我们将依次证明。

(a) s_K 永不下降。一轮只会增加已评分项目，因此 $E_{r+1} \supseteq E_r$，而更大集合中的第 K 大分数不可能更小。(b) U 永不增加。每一轮都会将所有已扩展前缀 b 从前沿中移除，并加入 b的子节点。根据 A2，这些子节点的概率之和为 $p(b)$，因此每个子节点的概率至多为 $p(b)$。所以，最大的前沿概率不会上升。(c) 最多 L 轮次。首先，某一轮之后已关闭的前沿前缀，在之后的每一轮中都会保持关闭，因为它的概率不会改变，而且根据 (a)， s_K 永不下降。其次，根据 A6，第 $r+1$ 轮会扩展所有在第 r轮之后仍开放的前缀。因此，第 $r+1$ 轮之后仍开放的每个前缀，都是第 r轮之后开放的某个前缀的子节点，而开放前缀的最小深度每轮至少增加一。由于前沿前缀的深度介于 1 和 $L-1$之间，因此经过 $L-1$ 轮回退后，不会再有开放前缀，也就是说，最多经过 L 轮次。此时 $U \le s_K$，定理 1 即可适用。

□

命题 *2* 的证明命题 *2*。 我们将从左到右证明该命题中的三个不等式。

(a) $s_k(r) \le p^*_{(k)}$。由于 E_r 是目录的子集，其第 k大分数不可能超过目录中的第 k 大概率。

(b) $p^*_{(k)} \le \max(s_k(r)\ U_r)$。我们区分两种情况。如果 $p^*_{(k)} > U_r$，那么目录中每个 k 个最佳项目的概率都高于 U_r。根据 引理 1，没有精确分数的项目的概率至多为 U_r，因此所有 k 个项目都获得了评分；结合 (a)，可得 $s_k(r) = p^*_{(k)}$。否则， $p^*_{(k)} \le U_r$ 直接成立。两种情况下，p^*

$_{(k)} \le \max(s_k(r)\ U_r)$。

(c) $\max(s_k(r)\ U_r) \le (1+\max(\text{gap}_r,\ 0))\ s_k(r)$。第一项显然满足 $s_k(r) \le (1+\max(\text{gap}_r,\ 0))\, s_k(r)$。对于第二项，根据差距的定义可得 $U_r = (1+\text{gap}_r)\, s_K(r)$；又因为 $k \le K$ 意味着 $s_k(r) \ge s_K(r)$，所以我们得到 $U_r \le (1+\max(\text{gap}_r,\ 0))\, s_k(r)$。

□

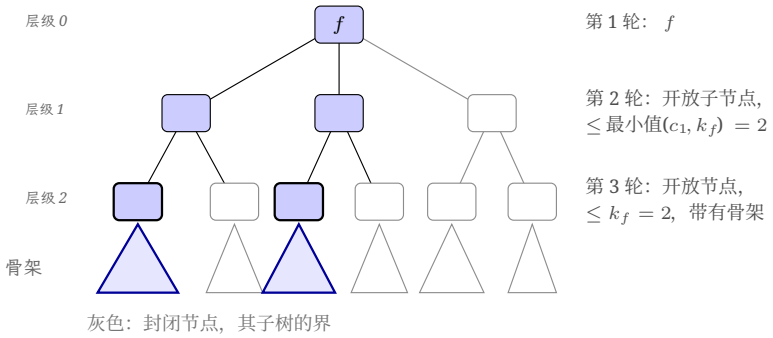


图15。计划的代价（命题3），对于开放前缀 f 和 $k_f = \lfloor p(f)/t \rfloor = 2$ 以及计划 $\ell = 2$。每层至多有 k_f 个节点高于阈值，而开放节点（蓝色）构成树的顶部区域，因为开放节点的父节点也是开放的。封闭节点（灰色）界定其整个子树，因此其下方的内容不会被馈入。计划每轮向下馈入顶部区域的一层，直至第 $\ell - 1$层；第 $\min(c_j,\ k_f)$ 层至多有 j个节点，然后馈入第 ℓ层的开放节点及其整个骨架（阴影三角形），这对应于公式（2）中的两项公式（2）。

证明命题3。 图 15 展示了证明的两个部分。

(i) 对于低于 f、高于 t 且必须输入的 j 层级节点，两个界成立。

(a) 概率界。 </style}低于 f 一个层级的前缀彼此不相交，并且根据 A2，它们的总概率为 $p(f)$。由于每个高于 t 的前缀对该总和的贡献都大于 t，因此其中超过 t 的前缀至多有 $\lfloor p(f)/t \rfloor = k_f$ 个。

(b) 骨架界。 </style}骨架 f 之外的每个节点都位于一条链上，即其子树只包含一个项目，因此根据 A2，其概率等于该项目的概率。链顶端节点的父节点处的输出已经给出了这一概率，因此无论该节点是否高于 t，都不需要词元。因此，在 j 层级必须输入的节点，属于该层级的 $c_j(f)$ 个骨架节点之中。

综合这两个界，在 j 层级这样的节点至多有 $\min(c_j(f)\ k_f)$ 个。

(ii) 固定任意一致补全（第4节）。在此补全下， t 上方的节点构成一个对取祖先封闭的集合，因为根据 A2，一个节点的概率至少不小于其任一后代的概率。计划 ℓ 分两个阶段处理 f。

(a) 下降。 对于层级 $j = 0$、.....、 $\ell - 1$，每轮处理一个层级，计划输入层级 j的开放节点，其数量至多为 $\min(c_j(f)$ $k_f)$（根据 (i)）。这给出了公式（2）的第一项。(b) 骨架轮次。 随后，计划输入层级 ℓ的开放节点，其数量至多为 k_f，并连同其骨架一起输入；这些骨架包含节点本身。这些骨架的总大小至多为层级 k_f 中最大的 ℓ个骨架大小之和，即 $\Sigma^{\sigma}_{\ell}(f,t)$，这就是公式（2）的第二项。

经过两个阶段后， f 下方的每个节点要么已被馈入，要么位于某个封闭节点下方。封闭节点将其子树中的每个项目界定为不超过 t，因此 f 下方不再有开放节点。 □

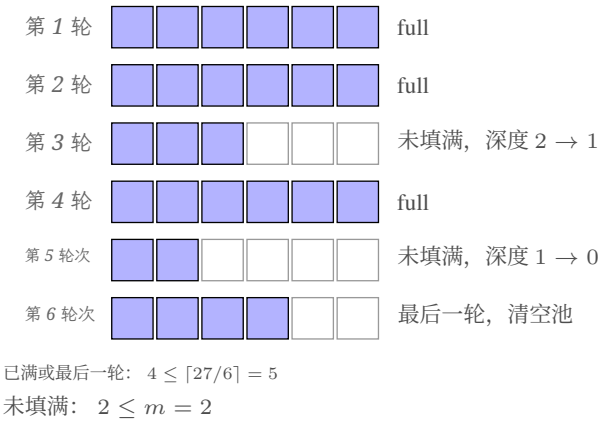


图16。轮数上界 $\lceil S/M \rceil + m$ （命题 4 (i)），此处每轮有 $M = 6$ 个槽位，一个包含 $S = 27$ 个节点且深度为 $m = 2$的池。每轮要么用池节点填满其 M 个槽位，要么耗尽可用的池节点。第一种轮次加上最后一轮，最多有 $\lceil S/M \rceil$轮，因为每个池节点只被馈送一次。第二种轮次已经馈送了所有可用的池节点，因此剩余深度会减少一层，这种情况最多发生 m 次。无论两种轮次以何种顺序出现，这两个数量相加即可。

命题 *4* 的证明命题 *4*。 (i) 我们根据每个回退轮次所馈入的池节点数量对其进行分类：当一个轮次馈入 M 个池节点时，称其为已满；否则称为未满（图 16）。一旦其父节点的输出已知，我们就称该池节点为可用，并分别界定这两类轮次。

（a）满载轮次。 每个池节点至多被馈送一次，而池中最多有 S 个节点，因此最多有 $\lceil S/M \rceil$ 个满载轮次。（b）未满载轮次。 只要还有空闲槽位，过程就不会让可用的池节点处于未被馈送状态，因此未满载轮次会馈送所有可用的池节点。在这样的轮次之后，每个剩余池节点都至少比之前深一层，因此，剩余计划仍能逐轮向下延伸的最大层级数会减少一层。因此，在这些层级仍然存在时，最多会出现 m 个未满载轮次。

(c) 最后一轮次。 一旦不存在这样的层级，剩余的每个池节点都可用，因此除非最后一轮次清空池，否则每一轮次都是已满的；最后一轮次可能未满载。如果这一最后一轮次为 $a \ge 1$ 个节点提供输入，则已满轮次的数量至多为$\lfloor (S-a)$ $/M \rfloor \le \lceil S/M \rceil - 1$，因此最后一轮次已计入 $\lceil S/M \rceil$。
无论已满轮次和未填满轮次以何种顺序出现，这两个计数之和都相同，由此得到界 $\lceil S/M \rceil + m$。
(ii) 令 $B = \lceil S/M \rceil + m$，假设 $B \le y$，并用撇号表示轮次之后的各个量。证明分为三个步骤。

(a) 调度　　安排在单次遍历中。 由于 $\lceil S/M \rceil \le y - m$，我们有 $y - 1 - m \ge \lceil S/M \rceil - 1$，因此　　e

$$x_r \;=\; S - (y-1-m)\,M \;\le\; S - \big(\lceil S/M \rceil - 1\big)\,M \;\le\; M.$$

(b) 界少一轮仍然成立。令 x 表示该轮次处理的池节点数量，并暂且假设池不会增长，且没有计划变得更深，即 $S' \le S - x$ 和 $m' \le m$；步骤 (c) 将证明这一点。我们区分两种情况。

- 如果 $x \ge x_r$，那么 $S' \le S - x_r = (y-1-m)\,M$，因此 $\lceil S'/M \rceil \le y - 1 - m$，结合 $m' \le m$可得$B' \le y - 1$。

- 如果 $x < x_r$，这一轮已处理所有可用的池节点。t 之上的所有剩余节点，都是那些被处理时未连同其骨架一起处理的节点的子节点，也就是说，它们的计划至少还会向下延伸一个层级。因此，$m' \le m - 1$，而$S' \le S$，再次得到 $B' \le y - 1$。如果 $m = 0$，那么所有池节点从一开始就都可用，因此这一轮处理了整个池，之后池便为空。

（c）重新规划不会扩大池。轮次结束后，池和规划方案会重新计算，且有三个事实确保这不会增加 S 或 m。

- 阈值在轮次之间只会升高，因为 s_K 从不降低，而更高的阈值只会缩小高于该阈值的节点集合，进而缩小池。

- 对于每个供给节点 g，其 g 子节点的分配量绝不会超过 g 减去该供给节点后的分配量。事实上，它们的计数 k_c 满足 $\sum_c k_c \le k_g$，因为它们的概率之和为 $p(g)$，下取整之和不超过总和的下取整，并且阈值没有下降。此外，不等式 $\min(a,b) + \min(a',b') \le \min(a+a', b+b')$ 将它们的层级计数限制在父节点相应计数之内，而它们的 Σ^σ 项在低一级层级中至多选择 k_g 个节点 g。

- 重新-在轮次结束后选择规划方案只会降低 B'，因为之前的规划方案仍然可用 .

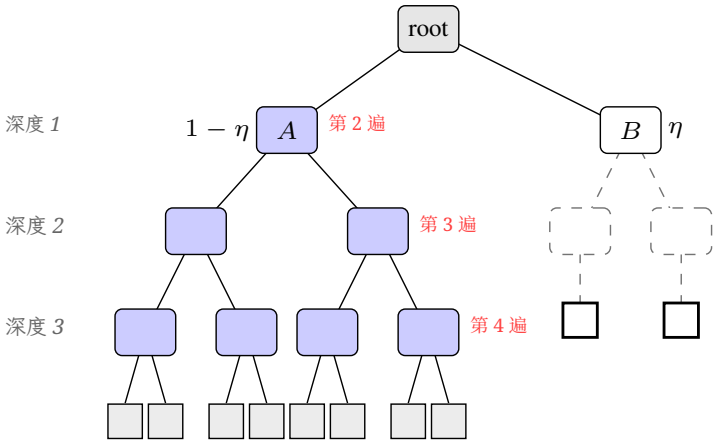
□

定理 *2* 的证明定理 *2*。 （i）令 Δ 表示公告因子，并令 $t = (1+\Delta)\ s_K(r)$ 表示第 r轮后的相应阈值。

（a）根据 δ^* 在式（6）中的定义，第一遍之后，一些规划方案满足 $\lceil S/M \rceil + m \le y$，其中 $y = Y - 1$ 是剩余轮次。（b）BUDGET-SIDECAR 的每一轮都会馈送至少 x_r 个池节点，或馈送所有可用节点（算法 2的第6行和第7行）。因此，根据命题 4（ii），每轮结束后，界在少一轮的情况下再次成立，即它是计划运行的不变量。（c）当不再有剩余轮次时，该界变为 $\lceil S/M \rceil + m \le 0$，这迫使 $S = 0$，即没有前沿节点高于 t。因此，在最后一轮 R 之后，我们有 $U_R \le (1+\Delta)\ s_K(R)$，且命题 2给出（i）。

(ii) 经过 $Y = Y_{\min}(M)$ 遍后，公告因子为 $\delta^*(M,\ Y) = 0$，因此根据 (i)，取 $\Delta = 0$ 时，经过 $Y_{\min}(M)$ 轮后没有前沿节点处于开放状态。因此，定理 1 适用，SIDECAR-K 得到认证。
(iii) 将 命题 4(ii) 应用于第 r轮后公告的因子。它表明，在该轮之后，一些规划方案满足$\lceil S'/M \rceil + m' \le y - 1$ 处于新的阈值。由于 δ^*_{r+1} 是使这类规划方案在少一轮的情况下存在的最小因子，因此它至多为 δ^*_r。

□



逐层：在第 A遍评分的项目 $L = 4$
BUDGET-SIDECAR：第 2 遍中的全部 A，因为 $\sigma(A) = 7 = M$
草稿 =两个项目 B（白色），在第 1 遍中单独输入

图17。命题5的实例，其中 $b = 2$、$L = 4$、$K = 2$ 和 $M = 7$。重簇 A 包含概率 $1-\eta$，均匀分布在其项目上；其下方的每个前缀都会分支，因此其骨架是整棵树，恰有 $\sigma(A)$ 个节点。轻簇 B 在两条链上包含 η，其两个项目就是草稿，第一遍会单独输入它们，从而留下 A 未处理。逐层下降（红色）只有在第 $L = 4$遍才会到达 A 的项目，因此轮次较少时，它会返回 B的项目。BUDGET-SIDECAR 在每个阈值下都找到 $R^*(t) = \lceil 7/7 \rceil = 1$，在第一遍后宣布精确性，并在第二遍输入 A 的完整骨架。

命题*5*的证明命题*5*。 对于任意 $L \ge 3$、$K \ge 1$和 $b \ge 2$，且满足 $b^{L-1} \ge K$ 的情况，我们构造该实例，其中每次遍历包含 $M = (b^{L-1}-1)/(b-1)$ 个节点，并进行任意 $2 \le Y \le L-1$ 轮次。随后我们证明，非规划运行，即每次遍历包含 M 个节点的算法 1，在经过 Y 轮次后返回一个不包含任何 MODEL-K 项目的列表，且只有经过 L 轮次后才达到精确；而 BUDGET-SIDECAR 在第一遍后就宣布达到精确性，并在两遍后返回 MODEL-K 及其证书。证明分为四个步骤。

步骤 *1*：该实例（图 17）。

- 字典树。 根节点有两个子节点，A 和 B。在 A之下，字典树是一棵完整的 b 叉树：深度 1 到 $L-1$的 A 之下，每个前缀都有 b 个子节点，而深度为 L 的前缀就是项目。因此，A 包含 b^{L-1} 个项目，A 之下的每个前缀都

有分支，而 A 的骨架是由所有内部节点构成的整棵树，$\sigma(A) = 1+b+\cdots+b^{L-2} = (b^{L-1}-1)/(b-1) = M$。在 B之下有 K 个子节点，每个子节点都是一条链，向下延伸至单个项目，因此 B 包含 K 个项目。

- 模型。 模型将概率 η 分配给 B，并在其 K 个项目之间均匀分配；将概率 $1-\eta$ 分配给 A，并在其项目之间分配，且这些项目的概率两两不同，彼此相差不超过两倍。例如，经过轻微扰动的均匀条件概率满足 A2 和 A4。我们选取足够小的 η，使得 A 以下的每个项目的概率都高于 η/K。那么，MODEL-K 由 A以下的 K 个项目组成；由于 A 以下的条件概率近乎均匀，这些项目分别位于不同的深度为 $(L-1)$ 的前缀之下。

- 草稿。 草稿由 B以下的 K 个项目组成。

步骤 *2*：第一遍，这是两次运行的共同步骤。候选字典树为 B，其子节点是 K 条链，因此，应用链规则进行第一遍时只输入 B，链规则则精确解析了 B 以下的 K 个项目。因此，E_0 是草稿，$s_K(0) = \eta/K$，前沿是 $F_0 = \{A\}$，其中有 p个 $(A) = 1-\eta > s_K(0)$，所以 A 处于开放状态，证书不成立。

步骤 *3*：非规划运行。

- 下面的每个前缀 A 都是分支，因此不适用链规则，并且每个回退轮次都会将开放前缀推进到更深一层。经过 r 轮次后，下面被推进的前缀深度因此为 1、.....、r，每轮最多有 $b^{L-2} \le M$ 个，因此预算不会构成限制。

- 只有当深度为（$L-1$）的前缀被推进时，下面的项目 A 才会被评分；这最早发生在经过 $L-1$ 轮次后，也就是经过 L遍后。
- 因此，经过 $Y \le L-1$ 遍后，评分集合仍是草稿，所以返回的列表由下面的 K 个项目 B 组成，且不包含 MODEL-K 中的任何项目。此外，前沿中还保留着概率约为（$1-\eta$）$/b^{Y-1} > s_K$的前缀，因此证书也不成立。

- 经过 L 遍后，下面的每个项目 A 都已评分，列表与 MODEL-K 相同，且前沿为空。

步骤 *4*：计划运行（*BUDGET-SIDECAR*）。

- 公告。在任意阈值 $t = (1+\delta)\, s_K(0) < 1-\eta$ 下，唯一的开放前缀是 A，其计划 $\ell = 0$ 成本为$S_0(A,t) = \sigma(A) = M$，依据公式（2）。因此，对于每个 $Y \ge 2$，$S_0^*(t) \le M$且 $R^*(t) \le \lceil M/M \rceil + 0 = 1 \le Y-1$。由此可得 $\delta^*(M,Y) = 0$且 $Y\min(M) = 2$，也就是说，第一遍之后便公告精确性。
- 第一回退轮次。池包含其骨架，因此 $S = M$ 和 $m = 0$，而调度要求$x_1 = S - (Y-2)\, M \le M$ 个池节点。因此，批次先容纳 A，再容纳其余的 $M-1$ 骨架节点，这些节点可以装入这一遍。

- 该结果。 经过此轮后，A 下方深度为（$L-1$）的每个前缀都已输入，A 下方的每个项目都已评分，并且不再有前沿前缀。因此，$U_1 = 0$，证书成立，且 SIDECAR-K_1 = MODEL-K。由此，计划运行在两轮后为每个 $Y \ge 2$ 返回带有其证书的 MODEL-K。

□

B. 进一步保证

在以下结果的基础上第 2 节 至 4，本附录陈述了进一步的保证，以对这些结果加以强化。正文并未使用这些保证。

按深度对齐。 令 μ_j 表示其第一个 j 编码与草稿中的任何候选项都不匹配的项目的概率，因此 $\mu_0 = 0 \le \mu_1 \le \cdots \le \mu_{L-1} = \varepsilon$。这些量衡量了草稿与模型在各个深度上的对齐程度，并细化了第 2 节中的隐藏概率条件。

命题 6（按深度对齐）。 (i) 当 $U_0 < p^*_{(K)}$ 时，经过一遍后证书成立，当且仅当 $U_0 \le p^*_{(K)}$。
(ii) 经过 r 个回退轮次后，SIDECAR-K$_r$ 最多遗漏 MODEL-K 的 $\lfloor \mu_{L-1-r}/p^*_{(K)} \rfloor$ 个项目。
(iii) 每个回退轮次最多扩展 $\lfloor \varepsilon/s_K(0) \rfloor$ 个前缀。

第 (i) 部分是该条件的精确形式：当最重的未覆盖前缀比第 K 个最佳项目更轻时，第一遍恰好给出认证；而 图 4 展示了基准测试中的这一条件。第 (ii) 部分说明，每个回退轮次都会修复一层错位。在深度 j 处草稿遗漏的项目会在 $L-j$ 轮次后被访问，而该深度尚未覆盖的概率则界定了此类项目的数量上限。第 (iii) 部分界定了一轮所输入的词元数量。

证明。 (iii) 第 r 轮次之后的每个开放前缀的概率都高于 $s_K(r) \geq s_K(0)$。开放前缀彼此不相交，因为它们是前沿前缀；它们的总概率至多为 ε，因为前沿前缀下的每个项目在第一遍之后都未被评分，并且在未扩展前缀下方仍然未被评分。因此，这类开放前缀少于 $\varepsilon/s_K(0)$ 个。

（ii）设 e 是 MODEL-K 中缺失于 SIDECAR-K$_r$ 的一个项目。那么 e 未被评分，因此它位于第一遍的某个深度为 d 的前沿前缀 b之下，并且对于每个轮次 r'，都有 $p(b) \geq p(e) \geq p^*_{(K)} \geq s_K(r')$；当 e 未被 A4 评分时，该关系为严格不等式。因此，b 在第一遍之后处于开放状态，第一轮将其扩展（A6）。随后，e 的深度为 $d+1$ 的前缀处于开放状态，第二轮将其扩展，依此类推，直到 e 的深度为 $L-1$ 的前缀被扩展并且 e 获得评分。这需要 $L-d$ 轮次。由于经过 r 轮次后 e 仍然缺失，因此 $d \leq L-1-r$，这意味着 e 的前 $L-1-r$ 个编码均不匹配任何候选项。因此，这些缺失项目是在总概率为 μ_{L-1-r} 的集合（A2）中，概率至少为 $p^*_{(K)}$ 的不同项目，所以它们最多有 $\lfloor \mu_{L-1-r}/p^*_{(K)} \rfloor$ 个。

(i) 如果 $U_0 < p^*_{(K)}$，精确前-K 中的每个项目都会被评分，因为未评分的项目必定位于概率至少为 $p^*_{(K)}$ 的前沿前缀之下。因此，$s_K(0) = p^*_{(K)} > U_0$，且证书成立。反之，证书给出$U_0 \leq s_K(0) \leq p^*_{(K)}$。

□

提前停止的列表的精确最高排名。 命题 2 界定了在证书成立之前返回的列表中的每个排名。事实上，此类列表的最高排名已经是精确的。

命题 7 （精确的最高排名）。 在任意轮次 r之后，如果对于某个 $k \leq K$，$s_k(r) \geq U_r$，则返回的前 k 个项目恰好是模型的前-k。

SIDECAR-K$_r$ 因此与 MODEL-K 一致，直到第一个得分低于 U_r 的排名；只有其后的排名由命题 2的因子界定，而不是精确确定。

证明。 如果 $s_k(r) \geq U_r$，则每个未评分项目的概率至多为 $s_k(r)$，因此，k 个最佳已评分项目就是目录中 k 最佳的项目（A4）。 □

不使用字典树的公告。 了解目录字典树对公告 第4节的贡献会有所帮助。没有目录字典树时，关于开放前沿前缀 f 下方节点的唯一信息是 命题3（i）给出的概率界，最多有 $k_f = \lfloor p(f)/t \rfloor$ 个开放节点处于任意层级，以及 n_f下方项目的数量 f，该数量同样限制了任意层级的节点数。于是，回退机制只能逐层进行，并且开放前缀在每个层级都必须适合单个轮次。这给出的计数为

$$\widehat{N}(t) = \sum_{f \in F_0 :\, p(f) > t} \Big(1 + (L - 1 - \mathrm{depth}(f))\, \min\big(n_f, k_f\big)\Big) \tag{7}$$

个可能曾经开放的前缀，因此公告 $\delta^*(M)$ =最小值为： $N((1+\delta)\, s_K(0)) < M\}$轮次。字典树在两个方面进行了改进： $c_j(f)$取代 n_f 成为结构上限，而骨架计划增加了另一种在预算内解决开放前缀的方式，即在一个轮次中输入其整个骨架。下面的命题表明，字典树从不会放宽公告。

命题 8 （不含字典树的公告）。 （i）每次遍历包含 M 个节点、共进行 $Y = L$ 遍时，逐层回退在每个排名位置返回的列表与精确的前 K 名列表相差不超过因子 $1+\delta^*(M)$。
(ii) $\delta^*(M, L) \leq \delta^*(M)$：包含字典树的公告始终不会比不包含字典树的公告宽松。

图11（第 5.7 节中的（c）比较了基准测试中的两种公告。

证明。 （i）固定阈值 $t > 0$ 和第一遍中包含 $p(f) > t$ 的一个前沿前缀 f。在 f 以下的每个层级，前缀彼此不相交，总概率为 $p(f)$（A2），因此其中至多有 $\lfloor p(f)/t \rfloor$ 个超过 t，且至多存在 n_f 个。任何曾在容差 δ 下开放的前缀，其概率都高于 $(1+\delta)\, s_K(r) \geq (1+\delta)\, s_K(0)$，并且要么是第一遍的前沿前缀，要么是其中一个的后代节点。因此，曾经开放的前缀至多有 $N((1+\delta)\, s_K(0))$ 个，即公式（7）所示。如果该数量低于 M，则超过容差的前缀始终能在一轮次中容纳；它们是权重最大的开放前缀，因此该轮会扩展所有这些前缀，并且它们如命题 1 的证明所述，每轮下降一个层级。经过 $L-1$ 轮后，没有前缀保持开放，因此 $U_R \leq (1+\delta)\, s_K(R)$，命题 2 给出因子 $1+\delta$。这就是公告 $\delta^*(M)$。

(ii) 对每个前沿前缀取最大的计划，采用纯逐层下降。由于 $c_0(f) = 1$，每个骨架节点下方都有一个项目，因此 $c_j(f) \leq n_f$，且 f 下方至多有 $L - 1 - \mathrm{depth}(f)$ 个层级，该计划的成本至多为 $\min\big(\sigma(f)\ 1 + (L - 1 - \mathrm{depth}(f))\ \min(n_f, k_f)\big)$，这至多是公式 (7) 中 f 的项。求和可得，$S \leq \widehat{N}(t)$，因此

$\sim N(t) < M$给出 $\lceil S/M \rceil + m \le 1 + (L-2) = L-1$：不含字典树的计数条件蕴含我们的条件，因此$\delta^*(M, L) \le \delta^*(M)$。 □

公告的单调性。

命题9 (单调性)。 (i) $\delta^*(M, Y)$ 随 M 和 Y 增加而非递增。
(ii) $\delta^*(M, 2)$ 是最小的 δ，使得在 δ 下开放的前缀的骨架能够在单次遍历中装入 M 个节点；这就是内存预算下骨架轮次的两遍界（第 3 节）。

第 (i) 部分表明，增加内存或遍数绝不会使保证变差。第 (ii) 部分讨论 $Y = 2$ 的情况，此时仍有一轮回退，因此只有计划 $\ell = 0$适用，池是开放前缀的骨架的并集。

证明。 (i) 命题 4 $\lceil S/M \rceil + m$的界 命题 4 随 M非递增，而条件 $B \le Y - 1$会随着 Y增大而放宽，因此最小可行的 δ 在任一情况下都不会增加。
（ii）还剩下 $Y = 2$轮，因此 $m = 0$和 $\lceil S/M \rceil \le 1$与 $S_0(f, t) = \sigma(f)$。因此，上述前缀的骨架 t 必须适合 M，而满足条件的最小 δ 就是公告。 □

C. 计算公告因子

公式（6）将公告因子定义为 其轮数上界适合预算的最小 δ。本附录解释了该计算取决于请求的哪些内容、哪些信息可以提前存储，以及如何找到最小值。 t
该计算取决于 请求、可以提前存储的内容，以及如何找到最小值。

预计算量与每个请求对应的量。公式（5）的轮数上界 $R^*(t)$由两类量构成。计数 $c_j(f)$以及前缀 f 下各层的骨架大小仅取决于目录字典树。它们针对每个内部节点计算一次；每层的骨架大小按降序排序并以前缀和的形式存储，因此最大的 k 个值之和只需通过一次数组查找即可得到。前沿 F_0、其概率 $p(f)$以及分数 $s_K(0)$取决于请求，并在第一遍之后即可知晓。阈值 t 仅在两处影响 $R^*(t)$：它决定哪些前缀是开放的，$p(f) > t$；并为每个开放前缀设定整数 $k_f = \lfloor p(f)/t \rfloor$。对于固定的 f，公式（2）的代价 $S_\ell(f, t)$因此仅是整数 k_f 的函数。此外，当 k_f 足够大时，该函数便不再变化。一旦 k_f达到 $K_f = \max_j c_j(f)$，每一项 $\min(c_j(f), k_f)$都等于 $c_j(f)$，最大的 k_f个骨架大小之和也等于整层之和，因此更大的 k_f 不会带来任何变化。对 $R^*(t)$求值，只需从存储表中为每个开放前缀提取一行，然后对上限 $m \le L-2$ 取最小值，所需操作次数为 $O(|F_0| \cdot L)$。如有需要，也可以按节点存储表 $T_f[m][k] = \min_{\ell \le m} S_\ell(f, k)$（针对 $k \le K_f$），从而将求值简化为查找；其总大小至多为 $(L-1)\sum_f \sigma(f)$个条目。

二分法。 $R^*(t)$关于 t单调不增，因此符合预算的阈值构成一个向右延伸的区间。实现会在 t 上对 $s_K(0)$ 与 $\max_{f \in F_0} p(f)$ 之间进行二分，保留区间的可行上界，并在固定的迭代次数后停止；在我们的运行中，这个次数为 20 次。返回的阈值始终是可行的，因此它产生的公告有效，并且其与上述区间精确下确界之间的距离不超过 2^{-20}。

在断点上进行精确搜索。 由于 t 仅通过开放集合和整数 k_f发挥作用，函数 $R^*(t)$ 是一个阶梯函数。它只能在某个 k_f 发生变化或某个前缀闭合的阈值处发生变化，即在断点处。

$$B = \{\, p(f)/k : f \in F_0,\ 1 \le k \le K_f \,\} \cup \{\, p(f) : f \in F_0 \,\}, \tag{8}$$

在连续断点之间，它保持不变，因为对于 $t \in (p(f)/(k+1),\ p(f)/k]$，$k_f(t) = k$。其中，具有 $k > K_f$ 的断点会被省略，因为成本在那里已经达到饱和。因此，$|B| \le \sum_{f \in F_0}(K_f + 1)$，在该基准测试中约为几千。对 B 进行排序后，二分搜索会在 $\lceil \log_2 |B| \rceil$ 次评估内找到第一个满足 $Y - 1$ 的断点，其 R^* 符合要求，所得结果是精确的。根据开放前缀定义中的严格不等式，还可得到一个细节：符合要求的阈值构成一个下端开放的区间，因此，式（6）中的 δ^* 是下确界，而计划运行使用断点上方第一个可表示的阈值，这会使因子改变一个舍入单位。

一次扫描处理所有预算。 断点还可以在一次扫描中给出每个预算对应的公告因子。按递增顺序遍历 B 时，每一步要么将一个 k_f 降低 1，要么关闭一个前缀，因此每个 $S^*_m(t)$ 都会针对每个上限 m 通过一次表差分进行更新，而 $R^*(t)$ 则作为 m 上的最小值重新计算。这样，在排序之后，一次扫描即可在 $O(|B| \cdot L)$ 时间内得到每个断点处的 $R^*(t)$。从这条曲线中，可以读取每个 Y 对应的 $\delta^*(M, Y)$、数值 $Y_{\min}(M)$，并且，由于只有

中的上限 公式（5） 取决于 M，因此无需重新计算，即可得到多个内存预算对应的相同量。每轮之后的重新规划也使用同一次扫描，并将其应用于前沿 F_r。

计算成本。 在该基准测试中，前沿包含数百个前缀和 B 几千个条目，因此，无论采用哪种方法，在 CPU 上耗时都为微秒至毫秒级；相比之下，模型在 GPU 上运行一遍需要数毫秒。因此，选择二分法还是精确搜索，取决于对精确性的要求，而非延迟。

D. 实验详情

数据、语义 ID 和模型。 我们使用 Amazon Reviews 2014 的美妆类别数据，采用 5-core 筛选和留一划分，得到 22,363 位用户、12,101 个项目和 198,502 次交互。我们使用句子编码器对项目文本（标题、品牌、类别、描述）进行嵌入，并采用三个码本、每个码本含 256 个编码的残差量化变分自编码器（RQ-VAE）；该模型使用 $\beta = 0.25$ 和 k-均值初始化进行训练，为每个项目分配前三个编码，并通过冲突码区分内容编码相同的项目。目录字典树在深度 1 至 3 分别有 256、9,043 和 11,792 个内部节点。其中，深度为 2 和 3 的节点分别有 78% 和 98% 只有一个子节点，因此目录骨架共有 2,533 个节点。生成器是一个四层仅解码器 Transformer（$d = 128$、四个头、可学习位置编码，参数量约为一百万），使用最多含 20 个项目的展平 SID 历史序列，通过下一词元损失进行训练。采用精确解码时，其 Hit@10 为 0.055，与该设置下已发表的结果相当。

评估协议。 对于每位测试用户，在目录字典树的全部 21,091 个内部节点上进行一次树遍历，即可得到模型在每个节点处的精确条件分布。随后，论文中的每个解码器、草稿和指标，均基于这些表执行确定性重放，因此能够在每个请求上检查每一条已证明的陈述。我们还将重放结果与真实模型进行了验证。对普通序列进行树遍历时，最大 logit 差异均在 10^{-5} 以内；在真实模型上运行算法 1 时，每位测试用户都返回 MODEL-K，且所需的遍数与重放预测的结果一致。

草稿。 无模型草稿基于序列的留一训练部分构建，并使用陈旧一轮交互的历史记录进行查询。内容 kNN 以 L2 归一化的内容嵌入为基础，计算每个项目与最近五个历史项目中任一项目的最大余弦相似度，并据此为每个项目评分。对于训练序列中相距至多五个位置的每个有序项目对，共现计数统计第二个项目紧跟第一个项目出现的次数，对计数进行按行归一化，并根据最近五个历史项目上的这些比率的按时间衰减加权和为项目评分。矩阵分解采用隐式反馈加权交替最小二乘，使用 64 个因子、置信度权重 40，并在二元用户–物品矩阵上进行 15 轮迭代；然后用用户学习到的向量对每个项目进行评分。所有方法都会排除历史项目本身，并保留前 N 项。量化用户序列是一个模型副本的前 N 项；该副本的二维权重按输出行四舍五入为 FP4（E2M1），并使用同一份提前一轮交互的历史记录运行。其权重的相对 Frobenius 范数误差为 11%，且其前10项与完整模型在 69% 的请求中一致。一个 FP8（E4M3）副本的权重误差为 2.6%，前10项一致率为 92%；它生成的序列在所有指标上都与完整模型的序列无法区分，在每种规模下的差异均不超过 0.01 遍，因此未在图中展示。将用户的序列写作 s_1，……，s_{n-1}，s_n 其中 s_n 为留出目标，则请求的历史记录（模型据此进行解码）为 $s_1 \dots s_{n-1}$。用户序列是模型针对 $s_1 \dots s_{n-2}$生成的前 N项，这正是一个在最新交互 s_{n-1} 之前运行的批处理作业本应存储的内容，因此它陈旧一轮交互。若不在该次交互之后为该用户调用模型，用户级列表就不能包含 s_{n-1}；而这正是该方法所避免的在线成本。因此，一轮交互是任何用户级批处理产物所能达到的最低陈旧程度。最后，所有草稿都不包含关于留出物品 s_n的信息。更平坦的模型会对训练模型的每个条件分布应用温度，而更平坦模型所存储的序列则根据该模型重新计算。

预算。 在每次遍历 M 个节点的预算下，第一遍会按草稿顺序将适合的候选字典树骨架成员送入处理流程。由于大多数候选项位于一级簇下方的链上，只有当其簇或路径上的分支前缀为新增内容时，候选项才会消耗一个词元。因此，$M = 16$ 可容纳用户序列中的约 90 个候选项，$M = 64$ 约 210 个，而 $M = 256$ 可容纳其 500 个候选项中的约 440 个。消融实验中的非规划运行采用 算法 1，其每次遍历包含 M 个节点，即对每个前缀采用最大计划的 BUDGET-SIDECAR，并省略 算法 2 的第 7 行，这相当于在没有调度和推测的情况下逐层下降。第 第 5 节 中的每幅图都使用前 10,000 名用户。

与 MF 草稿进行等资源比较。 图 18 重复了 RQ5 中的等资源比较（图 9），针对的是 MF 草稿。在训练模型上，MF 草稿达到精确结果的频率比 BEAM-K 低几个百分点，尤其是在内存最小的情况下，因为其首批候选项所指向的簇，不如模型最佳的第 1 层编码作为起点理想。从中等内存开始，它几乎对每个请求都能给出精确且经过认证的结果，与用户序列的表现相似。然而，在扁平模型上，MF 草稿并未享受到扁平度为用户序列带来的优势（第5.5节）。扁平模型排名最高的小子树项

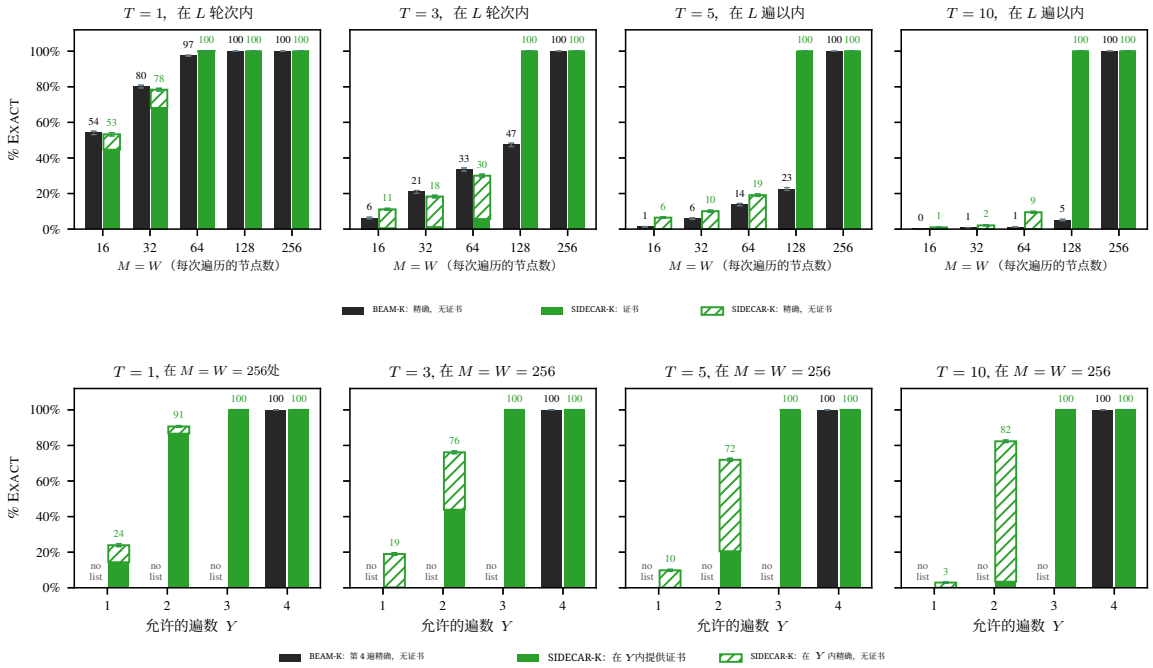


图 18. 图 9 **对应 MF 草稿。** MF 草稿在束搜索的内存预算下经过三遍后即可获得认证，但在内存较小时，在训练模型上以及 $T = 3$ 处，其达到精确结果的频率略低于 BEAM-K。因此，即使草稿忽略了模型，仍能在束搜索的预算内得到认证列表，但它无法享有平坦度带给用户序列的优势。

并不是具有相同历史记录的用户后来购买的项目，因此 MF 草稿很少包含这些项目，而回退机制必须找到它们。

E. 常见问题

1. **草稿必须包含排名前 K 的项目吗？** 不必。 第 2 节 表明，当草稿未覆盖该项目的概率低于排名第 K 的项目的概率时，第一遍即可获得认证；其更精确的形式 命题 6(i) 指出，当最重的未覆盖前缀低于该项目时，证书恰好成立。因此，草稿只需落在正确的前缀中，因为第一遍还会对候选项的兄弟节点进行评分，而无需额外词元，回退机制则会找到其余项目。这也是为什么用留出物品计算召回率并非衡量草稿的恰当指标。召回率统计是否包含某个项目，而成本取决于草稿未覆盖的概率有多大。

2. **如果草稿中的项目少于 K （即列表的大小），会怎样？** 无论是在 TIGER 中，还是在第 6 节介绍的工业系统中，约束束搜索都需要至少为 K 的宽度，才能返回 K 个项目；而在实践中，其运行宽度远高于 K。因此，假设草稿至少包含 K 个项目并不苛刻；第一阶段检索器或批处理作业无需在请求时调用模型即可生成这些项目。不过，保证条件并不要求 $N \geq K$，我们也测试了极端情况 $N < K$，在这种情况下，有两个性质可以确保解码器表现良好。首先，第一遍的评分项目数多于草稿所含项目数，因为输入某个候选项的分支前缀后，会对其下方链上的每个项目进行评分；在我们的基准测试中，除最平坦模型上的用户序列外，几乎每个请求中，单个候选项在第一遍后就已足以填满包含十个项目的列表。其次，对于第一遍评分项目数仍少于 K个项目的请求，附录A 中的假设 A5 不成立；此时第 K名的最高分为零，因此所有前沿前缀都处于开放状态，回退机制会先填满列表。因此，对于任何草稿，列表在 L 轮内仍保持精确（命题 1）；而公告因子只有从完成列表的那一轮开始才是有限值。

图 19 将非规划运行与包含 1 到 10 个项目的草稿以及作为参考的包含 500 个项目的草稿进行比较，并在资源相同的情况下与束搜索进行对比。在训练模型上，草稿的大小几乎无关紧要：即使只有一个候选项，在所有内存配置下也几乎与 BEAM-K 一样好，并且在最小内存配置下更频繁地达到精确结果。然而，在更平坦的模型上，由于少量候选项无法指向包含前10项的所有簇，包含少量项目的草稿在小内存下会放弃存储 500 个条目的序列所带来的大部分优势；而在中等内存配置下，其达到精确结果的频率略低于 BEAM-K。从 $M = 128$开始，每个草稿在所有温度下几乎对每个请求都能得到精确结果，而束搜索则需要宽度为 256。因此，部署时应将 N 保持在远高于 K的水平，但短草稿只会在内存较小时降低精确列表的占比

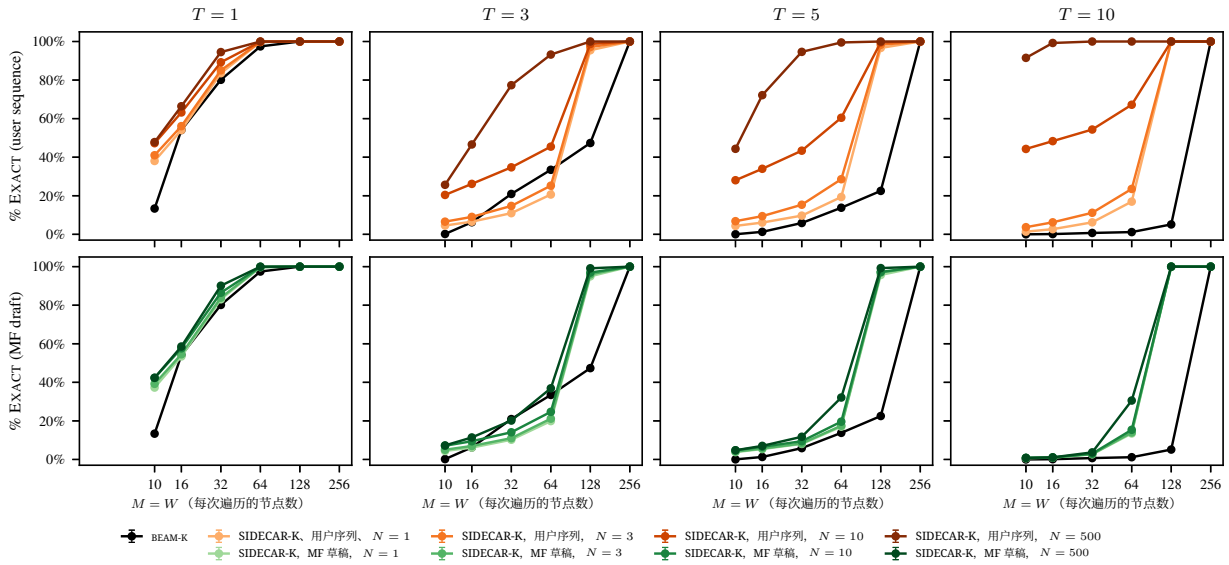


图 19. 小于列表的草稿，在束搜索的资源预算下（$K = 10, Y = L$）。即使只有一个候选项，在训练模型上也能匹配 BEAM-K；从 $M = 128$ 开始，每份草稿在几乎所有请求上都能达到精确结果，而束搜索则需要完整宽度。在低于该内存用量时，短草稿会放弃已存储序列在更平坦的模型上带来的大部分优势。

，并且绝不会导致证书错误。

3. **如果推测会降低典型情况下的准确率，为什么过程还要进行推测？**正是推测使最坏情况下的保证成为可能。没有推测，计划运行就会变成非规划运行，即算法 1 带有M 每次遍历的节点数，而在最后一遍之前，没有任何东西能界定其差距。例如，在命题5的实例中，非规划运行不会返回 MODEL-K 中的任何项目，尽管预算足以保证精确性。这是因为，当重型开放前缀的解析深度超过剩余轮次时，无法逐层解析它。只有在模型尚未生成这些层级的输出之前，就先输入其骨架，才能及时解析它。公式（4）的调度恰好强制执行最坏情况所需的部分，不多不少；过程则用已知为开放的前缀填满其他所有空位。作为交换，因子 $1 + \delta^*$ 在预算范围内能够抵御第4节中的对手，而典型情况下重叠度所付出的代价很小（第 5.7 节）。

4. **这是否为推荐任务所特有？**该方法依赖三个特性：输出长度固定、词汇表前缀闭包、约束分数的前缀概率为下方项目提供界，以及一份包含合理输出的草稿。任何基于文档标识符的生成式检索器都具备前两个特性，而且大多数检索器都有一个能提供第三个特性的第一阶段检索器。那么，SID 的独特之处在于其较小的 L，这使得经过 L 轮次后的回退界具有很强的保证；此外，RQ-VAE 字典树的单分支结构也使骨架较小。

5. **在生产环境规模下会有什么变化？**目录字典树将包含数千个一级簇，其骨架也远大于一遍所能容纳的规模，因此无限内存的结果不再适用，而 第4节中的预算结果则描述了解码器。届时，草稿必须足够准确，能够覆盖模型的概率分布；按构造，存储的序列可以做到这一点，而每遍可用的内存则必须足以容纳开放簇。公告会在第一遍之后说明是否满足这一要求。