用于高效非结构化数据处理的任务级联

Shreya Shankar 加州大学伯克利分校 美国
shreyashankar@berkeley.edu

Sepanta Zeighami 加州大学伯克利分校 美国
zeighami@berkeley.edu

Aditya G. Parameswaran
加州大学伯克利分校 美国
adityagp@berkeley.edu

arXiv:2601.05536v1 [cs.DB] 9 Jan 2026

摘要

现代数据库系统允许用户使用大语言模型驱动的函数查询或处理非结构化文本或文档列。用户可以用自然语言表达一项操作（例如，“识别此评论是否提到计费问题”），系统则以逐行方式对每个文档执行该操作。降低一批文档处理成本的一种方法是采用模型级联框架：由低成本的代理模型处理每个文档，只有不确定的案例才会升级交由更准确但成本更高的预言机处理。然而，模型级联错过了重要的优化机会；例如，回答查询时通常只需要文档的一部分，或者其他相关但更简单的操作（例如，“该评论的情感是否为负面？”、“该评论是否提到金钱？”）可以由低成本模型比原始操作更有效地处理，同时仍与原始操作保持相关性。

我们提出了任务级联框架。该框架将模型级联推广到不仅改变模型，还会在每个阶段改变文档片段和操作的情形。我们的框架利用大语言模型智能体生成简化、分解或以其他方式相关的操作，并选取最相关的文档片段，从而构建数百个候选任务，再据此组装任务级联。我们通过从最小和集合覆盖问题归约，证明最优级联选择难以处理，但我们的迭代方法能够构建有效的级联。我们还提供了一种扩展，可给出统计准确率保证：所生成的级联在有界失败概率下，能够达到用户定义的目标准确率（相对于预言机）。在八项真实世界的文档处理任务中，当目标准确率为 90% 时，与生产规模下的模型级联相比，任务级联平均可将端到端成本降低 36%。

CCS 概念

• **信息系统** → **数据管理系统**； • **应用计算** → 文档管理与文本处理； •**计算方法** →自然语言处理。

ACM 引用格式：
Shreya Shankar、Sepanta Zeighami 和 Aditya G. Parameswaran。2026。《面向高效非结构化数据处理的任务级联》。*ACM* 数据管理会议论文集4，1（SIGMOD），第 88 篇文章（2026 年 2 月），共 21 页。
https://doi.org/10.1145/ 3786702

作者联系信息：Shreya Shankar，加州大学伯克利分校，美国，shreyashankar@berkeley.edu；SepantaZeighami，加州大学伯克利分校，美国，zeighami@berkeley.edu；Aditya G. Parameswaran，加州大学伯克利分校，美国，adityagp@berkeley.edu。



本作品采用知识共享署名 4.0 国际许可协议授权。© 2026 版权所有者/作者所有。ACM 2836-6573/2026/2-ART88 https://doi.org/10.1145/3786702

1 引言

大语言模型（LLM）正日益作为算子集成到数据管理系统中，使用户能够更轻松地分析非结构化文本列。这一系列工作涵盖数据库的 SQL 扩展，可提供大语言模型驱动的映射 和筛选操作 [16, 21, 55, 67], ，以及基于 Python 的声明式框架，用于执行由大语言模型驱动的算子组成的流水线 [1,41, 49, 53, 65]。在所有情况下，用户都用自然语言描述操作，而系统则逐行调用大语言模型，对每个文本值执行该操作——下文将其称为文档。请看以下示例：

示例 *1.1*（最高法院判决意见分析）。假设一位法律分析师正在处理一个由大量最高法院司法意见构成的集合。该分析师的目标是判断每份司法意见是否推翻了下级法院判决。在这种情形下，每份文档都是一篇冗长而复杂的法律文本，而分析师用自然语言提出一项操作：“这份司法意见是否推翻了下级法院判决？”

GPT-4o 和 GPT-4.1 等最先进的大语言模型在定理 1.1 等任务上能够达到很高的准确率。然而，商业大语言模型 API 按输入和输出词元的数量收费，因此大规模分析的成本很高。因此，用户希望在不显著损失最佳模型所能提供的准确率的前提下降低推理成本。他们通常愿意接受准确率小幅下降——例如，以达到 GPT-4o 准确率的90%为目标——以换取大幅节省成本 [49, 52]。尽管存在更便宜的模型（例如 GPT-4o-mini，其价格几乎比 GPT-4o 便宜 17×），但它们的准确率往往低得多。

先前工作与局限性。在使用机器学习模型执行映射和筛选操作时，既降低成本又保持输出准确率的标准方法是采用模型级联 [2, 34, 36,37, 51]；近期系统也都将这一方法原样用于大语言模型驱动的算子 [9, 49, 70]。在这种方法中，系统首先对每个输入文档应用一个成本较低的代理模型。如果代理模型对其预测结果的置信度超过系统设定的置信度阈值，系统就会接受代理模型的输出。否则，系统会将输入交由一个准确率更高但成本也更高的预言机模型处理（例如，当前可用的最佳大语言模型）。用户指定一个性能目标（例如，达到预言机准确率的90%），系统则会调整置信度阈值，在满足该目标的同时最大限度地降低成本。

然而，模型级联框架只关注更换模型，未能充分利用优化机会。举例来说，如果没有更便宜的模型能够准确地执行该操作，最终文档就只能交由预言机处理。事实上，在医疗或法律任务中，较便宜模型的准确率可能仅为昂贵模型的 30% [17, 47, 63],；而在长上下文推理任务中，这一差距甚至更大 [42, 68]。此外，模型级联总是处理完整文档，尽管许多操作只需要文本中一小段相关内容。

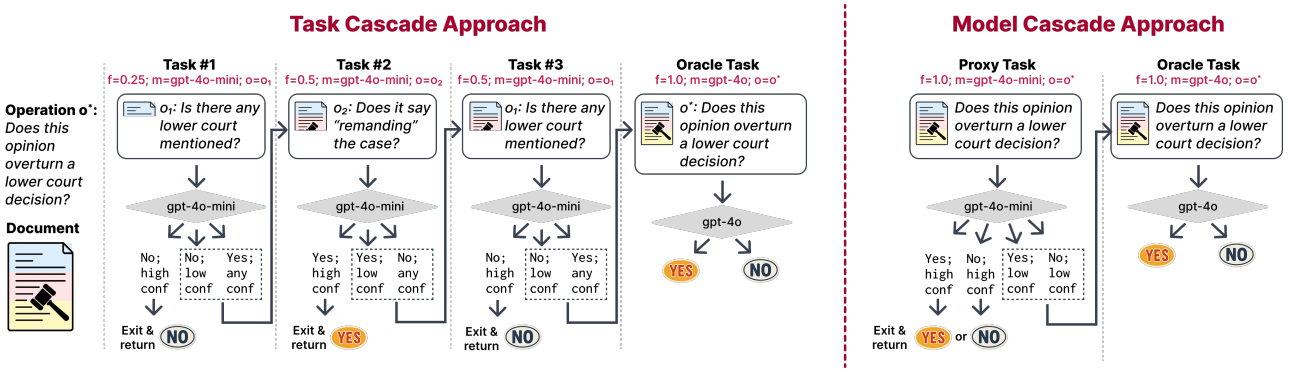

图 1：定理 1.1 中，用于判断最高法院的司法意见是否推翻下级法院判决的任务级联与模型级联对比。在任务级联（左）中，每个阶段（或任务）由文档比例（f）、模型（m）和操作（o）定义。不同文档比例下可以重复使用代理操作（如任务 1 和 3）。如果先前任务无法有把握地确定输入，最终的预言机任务便会使用预言机模型（m =gpt-4o），在完整文档（f = 1.0）上执行原始的用户指定操作（o^*）。相比之下，模型级联（右）在每个阶段都对完整文档执行相同的操作 o^*，只改变模型。

改变任务，而不只是更换模型。我们的核心洞见是，有效的级联应同时改变操作和文档中所处理的部分，而不只是更换模型。级联的早期阶段可以执行任何较便宜模型更容易完成的操作，例如前置条件检查、分解子任务，或与原始操作相关且具有相关性的操作。我们将此类操作称为替代操作：即原始操作的变体，代理模型更容易执行。在定理 1.1 中，我们不必让代理模型判断某条司法意见是否推翻了下级法院判决，而可以让它检查该司法意见是否提及下级法院。如果没有，就无需调用预言机。

另一种提高代理模型准确率的互补策略，是在推理前从每份文档中删去无关上下文。例如，我们可以不向大语言模型提供整份文档，而只提供语义上与操作相关的部分，例如包含"推翻""判决撤销"或其同义词的段落。通过消除无关文本，代理模型更有可能正确执行操作，且成本更低 [58, 69]。

本文提出了**任务级联**这一概念：由一系列有序任务组成，每个任务由一个操作、一个大语言模型以及选定的待分析文档比例定义。如图 1 所示，对于定理 1.1，任务级联可以先检查一小段摘录中是否存在下级法院，随后切换到另一种代理模型，并在之后对更多文档内容重新执行同一操作；如果仍有需要，最后再使用预言机模型，在全文档上执行原始操作。

任务级联的挑战。 然而，由于以下几个原因，设计有效的任务级联颇具挑战。首先，发现有效的代理操作非常困难，而且搜索空间近乎无限。虽然可以使用大语言模型编写新的操作，但代理操作只有同时满足以下条件才有价值：由低成本模型准确执行，并且 能够解决相当大比例的文档。质量欠佳的代理可能会使整个级联不如 省略该代理时高效。其次，为每项任务确定文档的理想部分并非易事。目标是仅提供足够准确决策所需的上下文，从而最大限度地降低令牌成本；文本过少会带来错误风险，而文本过多（例如完整文档）会抵消潜在的节省，并可能使成本更低的模型不堪重负。第三，确定应用任务的最佳顺序很困难。随着任务数量增加，可能的序列数会迅速增长，并且每个序列在成本和效果方面都可能存在显著差异。一项任务整体表现可能相对良好，但当它仅应用于先前任务未解决的较难示例时表现不佳，因而无法满足准确率约束。此外，当多个提示共享相同的输入前缀时，大语言模型 API 可提供成本节省（50–90%）。因此，优化任务顺序不仅关乎任务的准确率和选择性，还涉及构造输入以利用复用。总之，联合优化代理操作、文档比例和任务顺序会形成一个复杂的搜索空间，难以可靠地探索。

EffectiveTask 级联。我们提出一种方法，用于在给定文档集、用户指定操作以及相对于预言机的目标准确率的情况下，构建有效的任务级联。我们通过以下方式应对上述挑战：

- **任务排序**。给定一组任务，随着任务数量增加，尝试所有排序以找出最佳顺序很快就会变得不可行。我们通过从最小和集合覆盖问题归约，证明对任务进行最优排序是 NP-困难的。受该问题近似算法的启发，我们设计了一种贪心算法：依次添加能在满足准确率约束的同时最大程度降低总推理成本的任务。

- **置信度阈值选择**。 对于级联中的每项任务，我们都需要设置置信度阈值，以决定是否接受该任务的代理模型输出。一种简单的方法是为每项任务设置阈值，使得根据并合界，整个级联以所需概率达到目标准确率。然而，随着任务数量增加，这种简单方法会变得过于保守，导致难以找到可行的级联。我们基于近期的统计结果 [66], 开发了一种新的阈值调整算法，使我们能够用少得多的样本实现准确率保证。

- **代理操作生成**。 首先，要生成这组任务，一个简单的想法是提示大语言模型智能体（例如基于 OpenAI 的 o1）提出候选操作，但这些代理模型往往不适用于较小的模型，因为智能体无法判断哪些操作对成本更低的模型来说实际更容易执行。

相反，我们引入一种迭代式方法：由智能体提出代理模型，再根据代理实际执行效果良好的情况对其进行测试和改进。

• **文档剪枝**。最后，对于每项任务，我们希望在保持准确率的同时，尽量减少作为输入提供的文档比例。一种简单的无关内容剪枝方法是使用嵌入相似度，选出与用户操作最相关的文档块，这与标准的检索增强生成（RAG）技术类似；然而，对于复杂任务或长上下文任务，这种方法往往效果不佳，因此我们转而训练一个轻量级分类器，由预言机大语言模型提供监督，为每个文本块的相关性打分，并根据这些分数进行剪枝。

总的来说，我们的贡献包括：

(1) 我们提出了面向基于大语言模型的文档处理的任务级联 概念，将模型级联推广到允许每个阶段指定模型、操作（原始操作或替代操作）以及文档比例。

(2) 我们通过从最小和集合覆盖问题进行归约，证明了最优级联构建问题是NP-困难的。

(3) 我们开发了一种自动化的智能体方法，用于构建有效的任务级联——联合发现代理操作、裁剪文档，并以贪心方式组装任务序列，从而在满足准确率目标的同时最大限度地降低成本。我们表明，该方法可以扩展为提供保证。

(4) 我们在八项复杂的文档分类任务上评估了我们的方法，这些任务取自 Kaggle 和先前的基于大语言模型的数据处理研究；我们将其与标准模型级联基线进行比较，并分析各个组件（任务排序、文档裁剪、代理生成）的作用。在所有工作负载的目标准确率为 90% 时，与模型级联基线相比，我们的方法平均可将推理成本降低 ***48.5%***；与单独使用预言机模型相比，则可降低 ***86.2%***。

我们在第 2 节介绍问题并概述我们的方法。接着，在第3节，我们展示最优任务排序的难度，并描述我们的贪心级联组装算法，以及一种实现统计准确率保证的新流程。在第4节，我们介绍如何裁剪文档中的无关上下文，随后在第5节介绍我们用于发现代理操作的智能体方法。然后，在第 6 节，我们描述如何对任务级联的成本建模。最后，我们在第 7 节介绍评估，并在第8节讨论相关工作。

2 问题设置与方法

本节将任务级联的概念形式化，并概述寻找高效任务级联的方法。为方便起见，表1汇总了所有符号。

2.1 设置

问题设定。我们关注这样一种场景：商业数据库中的大语言模型驱动的映射和筛选操作 [16, 21, 55, 67]或近期系统 [41, 49, 53] 会从固定的类别集合中生成输出。形式化地说，用户提供一个文档集合 D 和一个目标操作o_{orig}，并使用自然语言对其进行描述。目标是将每个文档 $x \in D$ 分配给预定义的类别集合 C中的某个类别。用户指定一个准确率目标 $\alpha \in (0, 1]$, 要求系统的预测在至少 α比例的文档上与预言机模型 m_{oracle}的预测相匹配）。

给定一个文档 x （或者，如下文所述，文档的一部分）和一个操作 o（可以是原始操作，也可以是如下文所定义的替代操作），模型 m会为每个类别 $c \in C$生成一个分数 $p_m(c \mid x, o)$。具有最高 $p_m(c \mid x, o)$ 的类别 c 被视为模型的预测结果。对于定理 1.1， x 可以是法院意见， o可以是“该司法意见是否推翻了下级法院判决？”，而 $c \in$ 可以是{True,False}。为实现级联中的提前终止，我们使用**置信度阈值**：对于每个类别 $c \in C$，如果模型对 p_m的置信度 $p_m(c \mid x, o) \geq \tau^c$达到某个阈值 τ^c，则接受该文档 x的预测结果；否则， x继续进入下一阶段。

我们也可以不总是使用完整文档 x，而是使用文档的一部分x_f，即 x中排名前 f的部分，其中包含执行o_{orig}所需的最相关内容。相关性由评分函数确定，该函数根据文档片段对o_{orig}的效用进行排序。因此，置信度阈值现在基于$p_m(c \mid x_f, o)$。随后将一个**任务** 定义为 $T_i = (m_i, o_i, f_i, \tau_i)$，其中：

- m_i：模型（例如，代理模型或预言机）
- o_i：操作（原始操作或替代操作）
- f_i：已处理文档的比例
- $\tau_i = \{\tau_i^c\}_{c \in C}$：类别特定置信度阈值

任务级联 是由任务 $\pi = (T_1, T_2, \ldots\ldots, T_k)$组成的有序任务序列。在推理时，对于给定文档 x，我们按顺序评估每个 T_i：将模型 m_i 应用于输入x_{fi}并执行操作 o_i，得到预测类别 c和置信度分数$p_{m_i}(c \mid x_{f_i}, o_i)$。如果该置信度超过相应阈值 τ_i^c，则接受该预测，$\text{Cascade}(\pi, x) = c$，并终止级联；否则，继续处理下一个任务。如果所有 k任务都未能给出置信度足够高的预测，则级联会转交给预言机任务$T_{k+1} = (m_{\text{oracle}}, o_{\text{orig}}, 1, \varnothing)$；该任务使用预言机对完整文档执行$o_{\text{orig}}$，并将 $\text{Cascade}(\pi, x)$设为预言机的预测结果。因此，文档会在 π中首次产生置信度足够高的预测时“离开”级联；如果所有任务都推迟处理，则会在预言机处离开。传统的模型级联实际上是一种“受限”的任务级联，其中每个 T_i 都使用相同的操作$o_i = o_{\text{orig}}$，并在每个阶段处理完整文档（$f_i = 1.0$），只有模型 m_i不同。

置信度分数。 每个模型 m_i 都定义了一个关于输出类别 $c \in C$ 的分布，该分布以操作 o_i和文档片段 x_{fi}为条件，即 $p_{m_i}(c \mid x_{f_i}, o_i)$。大语言模型的输入会经过分词；词元 是子词单元，例如词语片段或标点符号。大语言模型会返回生成的输出中每个词元的对数概率，这些概率可转换为置信度分数。

成本模型。 每个任务 $T_i = (m_i, o_i, f_i, \tau_i)$都会将文档片段 x_{fi} 与操作 o_i拼接起来，作为提示发送给模型m_i。我们将词元数或大小 记作 $|\cdot|$，因此 $|x_{f_i}|$是文档片段的大小， $|o_i|$ 是操作提示的大小。我们用 λ_{in} 和 λ_{cached} 分别表示每个新输入词元和每个缓存输入词元的成本。对于分类任务，输出成本通常可以忽略不计，但如有需要，也可以轻松纳入计算。

在 x_{fi} 上运行任务 T_i 的成本取决于输入中有多少内容已通过对同一模型的先前调用缓存（大语言模型 API 支持此功能）。成本为：

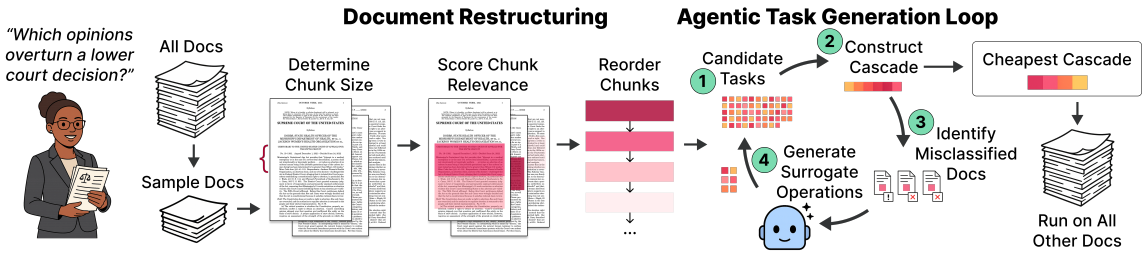

图2：方法概述。用户针对长文档提出复杂的分类查询。我们的方法会重构每个文档，以优先处理相关文本块，然后迭代提出代理操作，并组装一个具有成本效益的任务级联，以满足准确率约束。

$$\text{Cost}(T_i, x) = \begin{cases} |x_{f_i}|\,\lambda_{\text{in}} + |o_i|\,\lambda_{\text{in}} & \text{if } x_{f_i} \text{ is new} \\ |x_{f_j}|\,\lambda_{\text{cached}} + (|x_{f_i}| - |x_{f_j}|)\,\lambda_{\text{in}} + |o_i|\,\lambda_{\text{in}} & \text{if } x_{f_j} \subseteq x_{f_i} \end{cases}$$

其中，x_{f_j} 是此前处理过的、属于 x_{f_i} 前缀的最长文档。全文中，我们将文档置于提示中的操作之前，从而最大限度地提高多个任务使用同一模型时文档词元的缓存利用率。因此，文档 x 的总推理成本为：

$$\text{Cost}(\pi, x) = \sum_{i=1}^{j^*} \text{Cost}(T_i, x)$$

其中，j^* 是返回置信预测的第一个任务的索引（即确定了 Cascade(π, x) 的位置）；如果预言机在原始文档上调用，则为 $j^* = k+1$。

问题陈述。 给定一个文档集合 D、一个原始操作 o_{orig}、一组类别 C、一个预言机模型 m_{oracle}、可用的代理模型 M 以及一个准确率目标 α，我们的目标是构建一个成本最低的任务级联 $\pi = (T_1, T_2, \ldots\ldots, T_k)$，如下所示：

$$\min_{\pi} \quad \sum_{x \in D} \text{Cost}(\pi, x)$$

$$\text{s.t.} \quad \Pr\left[\frac{1}{|D|}\sum_{x \in D} \mathbb{I}\left[\text{Cascade}(\pi, x) = T_{k+1}(x)\right] \geq \alpha\right] \geq 1 - \delta$$

其中，$\mathbb{I}[\cdot]$ 是指示函数，T_{k+1} 是预言机任务；即，$T_{k+1} = (m_{\text{oracle}}, o_{\text{orig}}, 1, \varnothing)$；因此，$T_{k+1}(x)$表示预言机对 x 的预测。$\delta \in (0, 1)$ 是用户指定的上界，表示级联在 D 上的准确率低于 α 的概率。

虽然我们聚焦于具有固定输出类别的分类，但只要存在可靠的自动评估器来判断输出的正确性（即 LLM-as-a-judge [71]），并且有办法计算置信度分数，例如像 [24] 中那样计算逐词元概率的几何平均数，任务级联框架也可以扩展到开放式映射或生成操作。

2.2 方法概述

我们的方法在离线阶段构建一个高性价比的任务级联 π，使用文档集合 D中的一个小型代表性样本 D_{dev}，我们将其称为开发集。完整流程总结于算法1。构建过程包括：(i) 重构文档，以支持高效的分段处理；(ii) 通过智能体循环迭代优化一组候选任务，并组装成优化级联。我们关注双模型设置——一个成本低廉的代理模型（例如 GPT-4o-mini）

算法1： 任务级联概述

输入：文档 D、原始操作o_{orig}、预言机和代理模型 输出：任务级联 π（可选地附带统计保证） // 准备开发集 2 从 D中抽样 D_{dev}； // 文档重构（第4节） 3 对每个文档 $x \in D_{\text{dev}}$ 执行 4 训练轻量级分类器，为文本块相关性打分； 5 将 x 重新排序，把相关内容置于前面； 6 结束 // 在多个文档比例 F 下，以原始操作初始化候选任务 7 T $\leftarrow \{(m, o_{\text{orig}}, f):\ m \in \{\text{proxy, oracle}\}\ f \in \text{F}\}$;1 // 用于代理发现的智能体循环 8 迭代 n_a 次 // 根据当前候选任务组装最佳级联（第3节） 9 $\pi \leftarrow$ GreedyCascadeAssembly(T, D_{dev}); // 使用智能体生成新的代理操作（第5节） 10 $\text{O}_{\text{new}} \leftarrow$ LLMAgent(o_{orig}, π); // 将新候选任务加入任务集合 11 T $\leftarrow$ T $\cup \{(m, o, f):\ o \in \text{O}_{\text{new}}, m \in \{\text{proxy, oracle}\}\ f \in \text{F}\}$; 12 如果成本没有降低，则 13 退出； 14 结束 15 结束 16 返回 π;

以及一个更准确、成本更高的预言机（例如 GPT-4o）——与模型级联类似，但我们的方法可以推广到任意数量的模型。图2展示了我们方法的概述。

2.2.1 文档重构。我们对每份文档进行预处理，确保与原始操作 o_{orig} 最相关的内容出现在开头。重新排列后的布局使我们能够复用计算：如果在级联后续阶段需要更长的文档比例，那么模型已经处理并缓存了较短前缀中的词元，因此我们只需处理新增内容。我们使用预言机模型标注每份文档中与 o_{orig} 相关的区域，然后训练一个轻量级分类器，为不在 D_{dev} 中的新文档自动重新排序。

2.2.2 用于构建级联的智能体循环。为构建 π，我们迭代扩充候选任务配置集合 T，并反复组装满足在 D_{dev} 上的目标准确率 α 的最低成本级联。每种候选任务配置由一个模型、一项操作和一个文档比例定义——也就是按重排后文档（从开头开始）计算，提供给模型的文档比例。我们使用固定的比例集合 F（例如，F = {0.1，0.25，0.5，1.0}）。最初，我们将候选集 T 填入

表1：符号表

符号	描述
$x \in D$	集合中的文档 D
D_{dev}	用于寻找级联的文档开发集
$o_{\text{原始}}$	以自然语言描述的原始操作
$c \in C$	来自预定义类别集合的一个类别 C
α	准确率目标，其中 $\alpha \in (0, 1]$
$m;\ m_{\text{预言机}} \in \mathcal{M}$	任意模型；模型集合中的预言机模型
$p_m(c \mid x_f,\ o)$	在文档比例为 x_f、操作为 o 的情况下，m 针对类别 c 生成的分数
τ_c	类别 c 的类别特定置信度阈值
$(m_i,\ o_i,\ f_i)$	一个由 m_i、操作 o_i、文档比例 f_i 定义的任务配置
$T_i = (m_i, o_i, f_i, \tau_i)$	一个任务，包含任务配置和置信度阈值 τ_i
$\pi = (T_1, T_2, \ldots, T_k)$	任务级联，即任务的有序序列
$Cost(T_i,\ x)$	在文档 x 上运行任务 T_i 的成本
$\vdash \mid$	词元数量或大小
$\lambda_{in},\ \lambda_{cached}$	每个新输入词元（缓存输入词元）的成本
$f \in \mathrm{F}$	每个任务要考虑的文档比例集合
$n_s;\ n_a$	每次迭代中新建的代理操作数量；迭代次数
g	任务将 D_{dev} 分类后纳入级联的最低比例

原始操作o_{orig}、两种模型以及这些文档比例的所有组合。（每项任务的置信度阈值在级联组装过程中确定。）

智能体循环的每次迭代都按以下步骤进行。首先，在给定当前 T 的情况下，我们组装成本最低的级联 π_{best}，使其达到准确率目标。接下来，我们找出 π_{best} 的"失败案例"，即到达预言机或未被级联中任何任务有把握地分类的文档。随后，我们提示大语言模型智能体提出新的代理操作，并提供这些失败案例的上下文，以帮助智能体提出方案。对于每个新的代理操作，我们将其与所有模型和文档比例配对，并将生成的任务添加到 T 中。我们重复这一循环固定轮次，或直到无法进一步降低成本为止。

我们的方法包含三个主要技术组件，将在本文余下部分中加以介绍。首先，我们介绍级联组装（第3节），该组件独立于级联中任务的生成方式。接着，我们按照图2所示的方法创建候选任务：先进行文档重构（第4节），再进行代理生成（第5节）。这些组件具有高度模块化的特点，每个组件都可以单独定制或替换。

3 级联构建

我们将介绍如何从候选任务配置集合 $\mathrm{T} = \{(m_i,\ o_i,\ f_i)\}$ 中构建成本高效的级联，该集合由模型、操作和文档比例元组组成。我们的目标是组装一个带有相应置信度阈值的任务有序序列，在满足目标准确率 α的同时，最大限度地降低推理成本，具体定义见第 2.1 节。在本节中，我们假设 T 固定；第 4 节将介绍如何对每个文档中的内容重新排序，以创建分段文档；第 5 节则将说明如何生成新的代理操作。全文中，我们假设每种模型和操作（原始或代理）对应的文档比例集合固定。

我们首先证明，经过大幅简化的级联组装问题仍是 NP-困难的（第 3.1 节），这也说明了我们采用贪心方法的必要性。接着，我们详细介绍构建级联的三步流程（3.2节）：(i) 过滤掉无法满足准确率要求的候选任务；(ii) 贪心地组装一个能够维持每项任务准确率的有序级联；(iii) 调整级联以满足整体准确率保证。

3.1 最优任务级联的 NP-困难性

给定任务配置 $\mathrm{T} = \{(m_i, o_i, f_i)\}$，构建最优任务级联 πon D_{dev} 是 NP-困难的。我们证明，即使所有 $m_i = m$、只有一个代理模型且所有比例 $f_i = 1$，该问题仍然是困难的。我们的归约来自 NP-困难的最小和集合覆盖问题 [18] (MSSC)，这是更常见的集合覆盖问题的一种变体。在 MSSC 中，我们同样要选出能够覆盖全集中元素的集合子集；但输出是这些集合的一个排序，对于每个元素，我们根据覆盖它的最早集合支付成本；具体而言，如果序列中的第 i 个集合覆盖了元素 j，那么覆盖 j 的成本为 i。我们的目标是最小化覆盖所有元素的总成本。

定理 3.1。 构建最优任务级联是NP-困难。

证明。（概要）给定一个最小和集合覆盖问题实例 $(U,\ S)$，其中 U 表示元素，S 表示集合，我们按如下方式生成一个级联组装实例。对于每个元素 $u \in U$，我们创建一个文档 d_u。对于每个集合 $S_i \in \mathrm{S}$，我们创建一个候选任务 $T_i = (m,\ o_i,\ 1)$，其中 m 是一个单一代理模型，而 o_i 是一个操作：当且仅当 d_u 时，它以置信度 1 预测为 True，否则以置信度 0 返回随机答案。因此，我们级联中的每个任务的阈值都将设为 1。

我们设计成本模型，使得访问缓存的文档词元的成本为 0；每个文档在处理它的第一个任务之后即被缓存。因此，处理文档词元的总成本在所有级联中都相同，可以忽略不计。我们将每个操作 o_i的成本设为 1，因此在任意 d_u上运行任意任务 T_i都会产生 1 的成本。我们将准确率目标 $\alpha = 1$设定为，并为预言机分配无限成本，因此每个文档都必须通过某个任务退出级联。在此成本模型下，将任意 d_u通过级联 $(T_{i_1},\ T_{i_2}、\ldots\ldots)$ 进行处理时，其成本等于第一个在 d_u上具有置信度 1 的 T_{i_j} 的索引。因此，级联成本等于最小和集合覆盖问题的目标值。□

Feige 等人 [18] 描述了一种用于最小和集合覆盖问题的贪心算法，该算法在每个阶段选择能够覆盖最多（尚未覆盖）元素的集合，其近似比为 4。Feige 等人证明，进一步改进近似比十分困难。我们提出了一种类似的贪心算法，并对其进行了调整，以处理不同的模型、操作、成本、文档比例和准确率。

3.2 组装任务级联

总体而言，根据固定的任务配置候选集 T 组装任务级联包括以下步骤：

(1) 阈值选择与过滤。 针对每个候选任务，我们分别找出最小的置信度阈值，使至少 g% 的 D_{dev} 由该任务分类，且准确率不低于 α。对于任何类别，如果某个任务无法同时满足这两个条件，我们就将其移除。**(2) 级联组装**。 我们通过贪心方式逐步添加每一步中最能降低总推理成本的任务，同时确保每个任务在其分类的文档子集上达到目标准确率，从而构建级联。**(3) 阈值调整**。 我们调整已组装级联中的阈值，以确保整体处理序列达到所需的准确率。

我们将依次讨论这些步骤。

3.2.1 阈值选择与过滤。 我们的候选集 T 可能包含数百种任务配置，其中许多不可能达到目标准确率，例如基于较差代理操作的任务。我们会剔除这些配置，具体做法是舍弃任何单独来看无法以所需准确率、充满信心地对足够多文档进行分类的任务配置。对于剩余配置，我们会预先确定置信度阈值；这些阈值将在后续的级联组装中使用。

我们应用算法2中的过程：对于每个候选任务，我们检查其在 D_{dev} 上的预测，并针对每个类别，找出最低的置信度阈值，使高于该阈值的预测能够达到目标准确率 α。如果某个类别不存在这样的阈值（即无论阈值如何都无法达到所需准确率），我们就将其设为 ∞，确保该任务预测为此类别的任何文档都不会退出级联。如果这些阈值使该任务能够对 D_{dev}中至少最小比例 g 的文档进行分类，我们就保留该任务并记录其阈值；否则就将其丢弃。在我们的实现中，我们将 g 设为 10%；g 也可以通过统计界来设定（例如，使用 Hoeffding 不等式），但对于较高的准确率目标，我们需要每个任务有大量示例（例如， $\alpha = 0.95$ 和 $\delta = 0.25$对应的 > 250 ）。我们仍会确保最终级联具有统计保证，具体将在第3.2.3节讨论。

算法2： 为给定任务配置查找阈值。

输入： 候选任务配置 $T = (m,\ o,\ f)$，开发集 D_{dev}，准确率目标 α
输出： 各类别阈值 $\{\tau^c\}$，或丢弃 T

初始化所有 $\tau^c \leftarrow \infty$ c；总计 $\leftarrow 0$;
对每个类别 c **执行**
  $P_c \leftarrow$ 在所有 $x \in D_{\text{dev}}$ 上由 T 分配给类别 c 的置信度分数，其中 c 是模型对 x 预测的类别（即，$\text{argmax}_{c'}\ p_m(c' \mid x_f, o) = c$）; 按置信度排序;
  对于 每个唯一的 t，在 P_c（升序）**执行**
    $S \leftarrow \{x \in D_{\text{dev}}:$ 参数最大值$_{c'}\ p_m(c' \mid x_f,\ o) = c,\ \ p_m(c \mid x_f, o) \geq t\}$;
    如果 $|S| > 0$ 且 将 S 分配给 $c \geq \alpha$ 的准确率**则**
      $\tau^c \leftarrow t$; 总计 $+= |S|$;
      中断
    end
  end
结束
返回 $\{\tau^c\}$ **如果** 总计 $\geq g \cdot |D_{\text{dev}}|$; **否则** 丢弃 T;

3.2.2 级联组装。过滤后，我们采用贪心策略构建级联（详细伪代码见算法 4）。我们从空级联 π_0 开始，该级联会对所有文档调用预言机。每一步，我们都会考虑将每个尚未使用的候选任务插入 π 的末尾，从而生成一个新的候选级联 π'。对于每个这样的候选级联 π'，我们在 D_{dev} 上执行它，以计算其成本，并衡量 π' 中每项任务在其分类的文档子集上的准确率（即到达该任务，且输出置信度高于其阈值的文档）。我们只考虑满足以下条件的 π'： 包括新添加任务在内的每项任务，在其分配的子集上都达到目标准确率 α。在满足每项任务准确率要求的候选级联中，我们选择总成本最低的一个。如果没有候选级联能比当前级联降低成本，我们就返回当前级联。

要求级联中的每项任务都在其分配的文档上独立达到目标准确率，可能会使我们构建出比严格必要的更保守的级联。然而，由于 D_{dev} 可能很小，仅满足开发集整体准确率约束的级联在推理时可能无法很好地泛化。因此，强制要求每项任务都达到相应准确率，会提高级联达到预期目标的可能性。请注意，为了简单起见，该贪心组装算法针对的是置信度阈值固定且预先确定的任务（在第 3.2.1 节中设定）。后续的阈值调整步骤（第3.2.3节）纯粹是一个后处理步骤，旨在提供统计准确率保证，并不会进一步优化级联成本。

算法 3： 阈值调整

输入： 级联阈值 $\mathcal{T}_0$ 和验证集 D_V
输出： 保证达到目标准确率的级联阈值

次 迭代 i 直到达到最大迭代次数 max_iter **执行**
  $\mathcal{T}_i \leftarrow \{\tau_j + \epsilon_j^i;\ \forall \tau_j \in \mathcal{T}_0\}$
  如果不 $\mathcal{E}(\mathcal{T}_i,\ D_V)$ **则**
    如果 $i > 1$ **则**
      返回 $\mathcal{T}_{i-1}$;
    end
    else
      返回 未找到;
    end
  end
结束
返回 $\mathcal{T}_{max\_iter}$;

3.2.3 以保证达到准确率目标。迄今为止组装的级联 π仅在开发集上表现出经验上的准确性，在新数据上可能无法达到所需的准确率 α。为获得统计保证，我们将开发集随机划分为两个大小相等的 i.i.d. 子集：一个"训练"子集D_T，用于构建级联 π；另一个"验证"子集 D_V，用于调整阈值并生成最终级联 π^*，使得 $\Pr[\text{Acc}(\pi^*) < \alpha] \leq \delta$。如果 D_T 或 D_V 过小或变化性过大，该过程可能无法找到任何在指定界限下满足目标准确率的级联；因此，我们的实验确保每个子集中至少包含 75 个文档。新的任务级联 $\pi^* = (T_1^*, ..., T_k^*)$由任务 $T_i^* = (m_i, o_i, f_i, \tau_i^*)$组成，其中对于所有任务 $T_i = (m_i, o_i, f_i, \tau_i)$，$m_i$和 o_i保持不变，但阈值 τ_i被调整为 τ_i^*。为方便起见，我们将 $\mathbf{t} = \{\tau; \tau \in \tau_i^C,\ \forall C\ \forall i\}$ 称为π中所有任务的全部阈值集合。

从整体上看，确定能够保证目标准确率的阈值，过程如下。我们首先将 $\mathbf{t}$ 中的每个阈值增加一个较大的非负偏移量 $\epsilon_j^{(1)}$，从而生成一组初始的、非常保守的阈值。然后，我们反复执行以下步骤：(1) 调整 该偏移量；(2) 估计 所得阈值是否足够。在调整阶段，每次迭代 i都会减小偏移量，以创建一个新的候选阈值集 $\mathbf{t}^{(i)} = \tau_j + \epsilon(i)j:\ \ \tau_j \in \mathbf{t}$，其中调整策略 $\epsilon_j^{(i)}$被选择为随每一步递减；即， $\epsilon_j^{(i+k)} \leq \epsilon(i)j$对于 $k \geq 0$成立。我们将在第3.2.4节讨论确切的调整策略。在估计阶段，我们使用 $\mathbf{t}^{(i)}$ 在验证划分 D_V上运行级联，并应用估计函数 $\mathbf{E}(\mathbf{t}^{(i)}, D_V)$。如果级联达到至少 α的准确率，该函数将返回 `True`； 在 D_V

失败概率至多为 δ时返回 `False`，否则返回相反结果。该过程如算法 3所示，其中 *adjust*和 *estimate* 这两个步骤会重复执行，直到达到可能的最大迭代次数；当估计函数首次确定阈值未达到目标时，我们便停止。如果没有候选集通过，则恢复为仅使用预言机的级联（这在我们的实验中从未发生）。

定理 3.2。 对于任意调整策略 ϵ 和适当的估计器 $\mathcal{E}$，算法 3 π^* π^* < 返回一个满足 $\alpha \leq \delta$ $\Pr(\mathrm{Acc}()$) 的级联。

定理 3.2 表明，对于任意有效的估计器和单调调整计划，算法 3 都能提供所需的准确率保证。有效的估计器 E 必须提供一种统计检验：给定一组候选阈值和一个验证集，仅当它能够根据观测样本证明该级联达到至少 α 的准确率且失败概率至多为 δ时，才返回`True`。这是一个经典的独立同分布的伯努利样本均值估计问题，可以使用任意可靠的集中界；例如 Hoeffding 不等式。我们采用 Waudby-Smith 和 Ramdas [66], 提出的估计器，该估计器同时使用样本均值和方差来生成更紧的界——尤其是在方差较小时——相比之下，Hoeffding 不等式仅依赖均值。定理 3.2 的证明以及 E 的完整定义见附录A。我们还考虑了调整循环中多次应用 E 的情况，确保失败的总概率仍由 δ限定。

3.2.4 阈值调整策略。 现在我们介绍一种调整策略，以确保最终级联满足所需的准确率保证。一种简单的方法是在每次迭代时将每个阈值固定降低一个量（例如，每步降低 -0.01）。然而，大语言模型生成的置信度分数往往校准不佳，且高度集中在 1 附近，因此统一递减并不起作用 [14, 29, 32]。相反，对于每个任务 $T =$ 中的每个类别 c（m、o、f、τ），我们收集训练划分 D_T 上由模型分配给类别 c 的预测中，所有高于初始阈值 τ^c 的置信度分数。然后按升序排列这些置信度分数： $\tau^c < p_1 < p_2 < \cdots < p_k$，其中 p_1 是刚好高于 τ^c 的最低置信度， p_k 是最高置信度。

我们引入一个偏移索引 s，其初始值为 $s_{\max}$，并在每次迭代时递减 1。每次迭代中，对于每个类别，我们将其阈值设为 p_s；当 s 较大时，这对应于更保守的截断值，而随着 s 减小，截断值则不那么保守。具体而言，我们从 $p_{s_{\max}}$（最保守的阈值）开始，并在之后的每次迭代中将 s 减 1，从而使阈值逐渐接近原始阈值 τ^c。一旦 $s = 0$，我们便使用原始阈值 τ^c。每次迭代按此方式更新所有阈值后，我们会在 D_V 上评估级联，并应用估计器 E 检查准确率保证是否满足。若满足，我们继续；否则，返回上一次偏移对应的阈值。完整算法见附录A.3。在我们的实现中，$s_{\max} = 5$，但取值在 3 到 10 之间通常都有效。

4 文档重构

级联中的任务可以处理每个文档的不同比例，使模型能够专注于与该

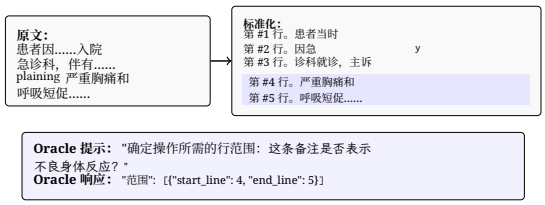

图3：标准化文档。此格式允许预言机将相关内容指定为行范围。

用户定义操作最相关的内容。为了在我们的基于前缀的成本模型（第2.1节）下降低成本，我们会重新排列每个文档，使最相关的文本出现在最前面。这一步文档重构与代理操作或级联配置的选择无关；任何按相关性对文档片段进行排序的方法都可以使用。虽然文档重构是可选的，但跳过该步骤将无法享受前缀缓存折扣，而这些折扣能够大幅降低成本。

为了有效地重构文档，我们首先确定如何对其进行分段。有些用户操作只需要较短的文本片段（例如一个句子），而另一些则需要更广泛的上下文。我们将每个文档表示为行序列，并选择分段粒度——即作为一个单元处理的连续行数。随后，每个文档都会被划分为具有该粒度的连续文本块。然后，我们训练一个轻量级分类器，根据文本块与用户定义操作的相关性，为文档中的文本块评分并排序。以下部分将介绍我们如何确定文本块粒度并训练相关性分类器。

确定文本块粒度。 为了选择适合用户指定操作o_{orig}的文本块粒度，我们使用预言机大语言模型和开发集 D_{dev}：

(1) 我们将每份文档拆分为每行 80 个字符，并在每行前加上行号（见图3）。(2) 对于每份文档，我们让预言机大语言模型返回回答 o_{orig} 所需的最小不重叠行范围集合。预言机以结构化格式返回这些行范围（例如，“起始_行”：5，“结束_行”：12）。(3) 我们合并重叠或相邻的行范围，并创建一份缩减文档，其中仅包含这些行的并集。我们在这个缩减版本上运行预言机大语言模型，并检查其答案是否与基于完整文档得出的答案一致。(4) 如果在 D_{dev}中，这份缩减文档对至少 α 比例的文档给出了正确答案，我们就进入步骤 (5)。否则，我们将每个范围的起始行向前扩展一行、结束行向后扩展一行（同时确保不超出文档边界），然后再次合并并重复检查。此过程会持续进行，直到达到目标准确率或达到最多 $e = 3$ 次扩展。

（5）我们将分块粒度设为 D_{dev}中所有文档的最终合并范围的平均长度（以行为单位）。
请考虑以下文档示例 x，以说明扩展和合并过程。假设预言机最初选择了范围

`[23,25]` `[28,30]`, $3 +$，其平 3, $2 = 3$,

均长度为 ( )/ 。如果在所选行上运行o_{orig} 得到的答案，与完整文档上的答案相比，至少有 α 比例的文档（位于 D_{dev} 中）不匹配，则我们将每个范围 x的两端各扩展一行，从而得到 `[22,26]` 和 `[27,31]`, ，其平均长度为 $(5 + 5)/2 = 5$。如果还需要进一步扩展，我们

将得到 `[21,27]` 和 `[26,32]`,，它们现在彼此重叠，因此合并为单个范围 `[21,32]`,，平均长度为 12。

按相关性为文本块评分。 确定分段粒度 s后，我们将每个文档划分为由连续s 行组成的集合，这些集合称为文本块。我们的目标是训练一个轻量级的相关性分类器，用于预测某个文本块是否包含执行o_{orig}所需的信息。

首先，我们将 D_{dev} 划分为"训练集"（D_{train}）和"留出测试集"（D_{test}），用于构建和评估分类器。在上一步中，我们已经获得了所有文档中相关文本块的起始行。我们按如下方式构建用于训练相关性分类器的预言机标注数据集：

（1）对于预言机识别出的每个起始行，我们提取从该行开始的 s 行文本块，并将其标记为相关（$r = 1$）。为构造不相关示例，我们从文档开头开始，以 s 行为步长滑动一个不重叠的窗口；仅当某个文本块不与任何相关文本块重叠时，才将其纳入不相关集合并标记为不相关（$r = 0$）。与任何相关文本块重叠的文本块都会从不相关集合中排除。

（2）对于 D_{train} 和 D_{test} 中的每个文档，我们按照上述说明收集所有带有相关性和非相关性标注的文本块。训练数据集和测试数据集分别由 D_{train} 和 D_{test} 中每个文档的所有带标注文本块的并集构成。

D_{train} 和 D_{test} 中的每个文本块 c 都使用 OpenAI 的 `text-embedding-3-small` 进行嵌入，生成成对的 (embedding(c) r)。由于相关文本块较为稀少，我们在训练期间对相关类别进行上采样。我们使用以 o_{orig} 的嵌入初始化权重的逻辑回归模型，来预测每个文本块的相关性。模型使用 D_{train} 中的文本块进行训练，以最小化二元交叉熵损失，并使用 Adam优化器；同时根据 D_{test} 中文本块的 F1分数进行提前停止。

在推理时，一个新文档会被划分为由 s 行组成的连续且互不重叠的文本块；每个文本块都会被嵌入，并由训练好的模型分配一个相关性概率。随后，文本块会按照预测概率从高到低排序，并拼接成重新排序后的文档。

5 代理操作生成

到目前为止，我们已经说明了如何从固定的候选任务配置集中组装成本最优级联。现在我们来探讨如何创建这个候选集。我们假设所有文档都已重新排序，使与原始操作最相关的内容出现在开头（详情参见第4节），并且只讨论如何生成代理操作。总体而言，我们的方法如下：初始化 T，使其包含 o_{orig}、每个模型 $m \in \mathbf{M}$ 和每个比例 $f \in \mathbf{F}$ 的所有组合。随后进入一个循环，在每一轮中：(1) 从当前的 T 中组装最佳级联；(2) 分析该级联的失败案例；(3) 使用大语言模型智能体扩充 T。接下来，我们将说明什么是有效的代理操作、如何从智能体中引出代理操作，最后介绍智能体循环的实现方式。

有效代理操作。 为了生成代理操作，我们需要某种有效性的定义。有效的代理操作是指其输出构成原始操作的类别子集的任何操作。对于分类任务，有效的代理操作就是输出空间为原始类别子集的任何操作（可选地包含一个特殊的"以上皆非"或 -1 标签）。假设原始任务是将生物医学摘要分为四个类别：0（研究论文）、1（综述论文）、2（临床试验）和 3（病例报告）。一种代理操作可以是：*"如果摘要中包含'随机试验'或'双盲'之类的短语，则输出 2（临床试验）；否则，输出 −1。"* 这类启发式规则使代理模型能够通过简单模式自信地处理某些示例，即使它们无法完成完整的多类分类。

向智能体引出代理操作。使用前文提到的代理操作概念，每当我们向大语言模型智能体查询代理模型时，都会提供以下上下文：

(1) 用户定义的操作，o_{orig}。(2)"失败案例"，即当前最佳级联中没有任何现有代理任务对其进行分类的文档示例（也就是说，会送达预言机的文档）。为确保能在智能体的提示词窗口中容纳多个文档示例，我们使用预言机模型从每份文档中找出并拼接最相关的片段——即支持或解释预言机预测的片段——而不使用整份文档。(3) 当前级联中每项任务的汇总统计信息，包括该任务是否被选入级联、其文档覆盖率（即该任务处理的文档数量），以及最多 10 个"困难样例"，即任务预测错误但置信度较高的边界案例，通常是置信度刚刚超过接受阈值的案例。(4) 明确要求提出新的代理操作，这些操作应当 (i) 比原始操作更简单，(ii) 输出原始类别的任意子集，或 (iii) 针对当前级联的弱点。提示中包含具体示例并指定了输出格式（代理提示词：...，理由：...），以便我们解析代理操作。详情请参阅 B节。

智能体循环。 现在我们已经有了一种从智能体中提取代理操作的机制，接下来将介绍完整的智能体循环。每一轮中：(1) 我们按照 3.2节中的流程，使用当前候选集组装最佳级联；(2) 我们将新一轮的失败案例和汇总统计更新到智能体的提示中；(3) 智能体提出 n_s 新的代理操作。每个代理操作都会与每个模型和文档比例配对，从而产生 $|\mathbf{M}| \times |\mathbf{F}| \times n_s$ 个新的候选任务，并将其添加到 T 中。我们重复这一过程，直到完成 n_a 轮，或不再观察到成本进一步降低。参数选择（例如 n_s、n_a、文档比例、智能体模型）具有灵活性（见 2节）；我们 $n_s = 3$、$n_a = 3$，并使用 OpenAI 的 o1-mini 作为智能体。完整算法见 B节。

6 成本模型优化

本节将推导构建任务级联的优化成本，并展示如何通过超参数选择控制优化预算。总优化成本由三个部分构成：文档重构、代理操作生成与评估，以及智能体循环成本。为简单起见，我们省略候选任务输出令牌成本（该成本是固定且可忽略的——例如，二元分类只需 1 个词元，多分类任务最多需要几个

词元）。需要注意的是，在实践中，实际优化成本可能低于我们的推导结果：大语言模型 API 提供商会采用提示缓存，在相同文档上评估多个候选任务可能产生缓存命中，从而降低成本。我们的优化实现并未显式针对缓存复用进行优化，因此，下面的公式代表对实际优化成本的保守估计。

初始标注和文档重构。我们首先在完整开发集上运行一次预言机模型，以获取用于计算代理任务准确率的"真实标签"：$C_{\text{labels}} = N(L+P)\lambda_o$，其中 N 是开发文档的数量，L 是每个文档的平均词元数，P 是每次调用的提示词元数，λ_o 是每个词元的预言机模型价格。

接下来，我们执行文档重构，为分段处理准备文档。我们再次运行预言机模型，以识别文档中的相关行范围或文本块。我们还会为所有文本块生成嵌入：

$$C_{\text{doc}} = C_{\text{labels}} + N(L+P)\lambda_o + NL\lambda_{\text{emb}} = N(L+P)(2\lambda_o) + NL\lambda_{\text{emb}}$$

where λ_{emb} is the embedding price per token.

代理操作生成与评估.. 经过 n_a 次迭代后，智能体每次迭代提出 n_s 个新的代理操作，共生成 $n_s n_a$ 个代理操作。每个代理操作都与每个文档比例 $f \in \mathrm{F}$配对，并使用代理模型和预言机模型在开发集上进行评估。由于我们在 |个 |不同的比例下进行评估，因此每份文档处理的分段内容总量为 $S_f = \sum_{i=1}^{|\mathcal{F}|} f_i$（例如，对于 $\mathrm{F} = \{0.15, 0.25, 0.5, 1.0\}$，我们有 $S_f = 1.90$）。设 λ_p 为代理模型每个词元的价格，则所有代理操作评估的成本为：

$$C_{\text{eval}} = N n_s n_a \left[L S_f (\lambda_o + \lambda_p) + P|\mathcal{F}|(\lambda_o + \lambda_p) \right]$$

智能体循环。大语言模型智能体（o1-mini）经过 n_a 次迭代生成新的代理模型，每轮消耗 T_{agent} 个输入词元并生成 O_{agent} 个输出词元，价格分别为 $\lambda_{o1,\ \text{in}}$ 和 $\lambda_{o1,\ \text{out}}$。值得注意的是，此成本与开发集大小 N无关，因此为常数：$C_{\text{agent}} = n_a \left(T_{\text{agent}} \lambda_{o1,\ \text{in}} + O_{\text{agent}} \lambda_{o1,\ \text{out}} \right)$。

总优化成本。总优化成本为 $C_{\text{doc}} + C_{\text{eval}} + C_{\text{agent}}$。$C_{\text{opt}} = C_{\text{eval}}$。主要成本来自在开发集上评估候选代理任务。可通过以下方式降低此成本：(1) 减少 n_s 或 n_a，以生成更少的代理模型；(2) 使用更少的文档比例（较小的 |F|）；(3) 生成候选任务时排除预言机模型（在 C_{eval}中用 λ_p 替换 $\lambda_o + \lambda_p$）；或 (4) 减小开发集大小 N，不过这样会更难获得统计保证。在我们的实验中，我们还评估了采用策略 (3) 的低成本变体。

7 评估

我们设计实验来回答以下问题：[1]**问题一** 与模型级联相比，任务级联能否降低推理成本？我们的方法中的哪些组件对这些节省贡献最大？**问题2** 随着目标准确率变化，成本和准确率如何变化？**问题3** 任务级联成本和准确率在不同运行之间有多一致？**问题4** 构建任务级联的成本是多少？

[1]代码可在 https://github.com/ucbepic/task-cascades获取。

表2：平均词数和语料库规模的工作负载。"# Docs"报告数据集发布时文档的大致数量；后续版本中的实际语料库规模可能已有变化。

数据集	描述$^{o_{\text{orig}}}$	平均词数	# 文档
agnews	将新闻文章摘要分类为以下类别之一四个主题：世界、体育、商业或科技。	~37	~128k
court	判断美国最高法院的司法意见是否推翻了下级法院的裁决。	~3.7k	~36k
安然	识别由高管层或副总裁级别人员发送的电子邮件安然语料库中的高管。	~1.5k	~500k
FEVER 数据集	判断自然语言主张是否得到所提供证据片段的支持。	~5.1k	185k
game	判断一篇评论是否更称赞另一款不同的游戏，而非其评论对象。	~1.1k	~6.4M
法律	检测不起诉契约或知识产权不质疑许可协议中的条款。	~8.0k	510
PubMed	将生物医学文章归入以下类别之一六种研究类型：随机对照试验、观察性研究、*Meta*-分析、实验室研究、计算研究或综述。	~3.1k	~133k
wiki_talk	预测维基百科讨论页是否存在分歧——讨论最终以一次编辑回退告终。	~0.9k	~125k

在目标准确率为 90% 时，对于所有工作负载，与模型级联基线（具有可比保证）相比，我们的方法平均将推理成本降低了 ***48.5%***；与仅使用预言机模型相比，则降低了 ***86.2%***，凸显了任务级联的有效性（**问题一**）。我们方法中的每个组件（即代理操作发现、文档剪枝和级联排序）都对降低成本有所贡献。对于 **问题2**，随着目标准确率变化，任务级联始终改善成本—准确率权衡。在较低的目标准确率下，收益最大，这可能是因为更容易识别出满足宽松准确率目标的代理操作。在更具挑战性的工作负载上，当目标准确率较高时，模型级联与任务级联之间的性能差距会缩小。对于 **问题3**，我们发现任务级联和模型级联都表现出较高方差，但任务级联始终实现更低的均值和中位成本。最后，对于 **问题4**，我们表明，在规模化场景下，优化成本可以忽略不计；即使将优化成本考虑在内，任务级联仍比模型级联更便宜。

我们首先介绍实验设置（第 7.1 节）。接下来，我们针对各个研究问题展示结果：不同变体的成本效益（第 7.2.1 节）、不同准确率目标下的性能（第 7.2.2 节）、重复运行的一致性（第 7.2.3 节），以及优化成本和盈亏平衡分析（第 7.2.4 节）。最后，我们分析任务级联并总结其中的启示，包括实际部署考量和参数选择指南（第 7.3 节）。我们还在第 C 节详细介绍代理任务示例以及每个级联中的任务数量。

7.1 设置

7.1.1 数据集与工作负载。我们在八种文档分类工作负载上评估我们的方法，这些工作负载涵盖了不同领域、文档长度和任务复杂度（表2）。其中五种取自以往有关大语言模型驱动的数据处理系统的研究。为扩大覆盖范围，我们从 Kaggle 引入了另外三项需要多步骤推理或特定领域推理的任务。**源自以往研究**。工作负载 (i)–(ii) 源自以往关于模型级联的研究 [9, 49], , 而 (iii)–(v) 则来自大语言模型驱动的数据处理研究 [41, 53]：(i) 来自 [9], 的 agnews [13], 是一项四分类任务，文档极短。我们

将 agnews 作为简单工作负载的示例纳入其中，在该工作负载中，代理模型已达到准确率目标，以测试任务级联能否带来额外收益。(ii) 来自 [49], 的 fever [59], 是一项声明核验任务：给定一个自然语言主张和一组证据片段，大语言模型必须判断该主张是否得到支持。由于每篇文档都涉及不同的主张和证据，因此难以学习可复用过滤器和代理任务，这使 fever 对我们的方法尤其具有挑战性。(iii) 来自 [41], 的安然 [38], 由安然语料库中的电子邮件组成。 [41] 中的原始任务是欺诈检测，但数据集中欺诈案例极少，因此采用"始终预测无欺诈"的简单策略就能获得很高的准确率。为了创建更均衡、适合评估任务级联的分类问题，我们将原始欺诈检测任务调整为识别由高管层或副总裁级高管发送的电子邮件，从而使类别分布更加均衡。(iv) 来自 [53], 的法律 [26], 用于测试对合同中特定法律条款的检测；我们将原始跨度抽取任务转换为二元分类，重点判断是否包含某一类条款。(v) 来自 [53], 的 games [56], 经调整后成为一项更具挑战性的分类任务：判断一篇电子游戏评论是否比被评论的游戏更称赞另一款游戏。

新增工作负载。 我们从 Kaggle 和近期自然语言处理数据集中引入了三种额外的工作负载。court [22] 涉及对美国最高法院判决意见进行分类，需要对冗长而复杂的文本进行多步推理。wiki_talk [6, 12]要求模型预测维基百科讨论页上的讨论是否最终导致编辑回退，以捕捉在线话语的动态变化。最后，pubmed [11] 是一项针对长篇生物医学文章的多类分类任务，用于测试我们的方法在文档规模较大、标签空间扩展时的表现。

对于每种工作负载，我们抽取 1,000 篇文档（200 篇用于开发集，800 篇用于测试集），但法律工作负载除外，它仅包含 509 篇文档（150 篇用于开发集，359 篇用于测试集）。我们抽取数据集时确保费用不超过 5,000 美元的 API预算；如果在完整数据集上运行下文所述的所有变体，并设置多个准确率目标和试验，成本将高达数十万美元。除 agnews 和 pubmed 属于多类分类外，所有工作负载均为二元分类任务。所有工作负载的提示词见 F 节。

7.1.2 基线。 对于**Q1–Q4**中的所有实验，我们将任务级联与三种参考方法进行比较。**仅使用预言机** 在每个文档上运行预言机模型（GPT-4o），提供理想但成本高昂的上界。我们的第二个基线**双模型级联**采用 [49], 中的模型级联方法，将 GPT-4o-mini 作为代理模型、GPT-4o 作为预言机模型。这种方法利用了来自大语言模型 API 的令牌级对数概率，因此属于当前最先进的方法；相比之下，先前的模型级联论文采用了更重量级且准确率更低的方法 [9, 70]。对于每个类别，我们将代理模型阈值设置为开发集上的最小值，使得阈值以上的代理模型预测和 阈值以下的预言机模型预测的综合准确率达到目标准确率，从而积极地将成本降至最低。第三种方法**双模型级联（+ 保证）**在**双模型级联**的基础上加入我们的统计保证程序（算法 3）。尽管 SUPG 等替代方法 [35] 也可以强制提供保证，但为确保受控比较，我们采用相同的程序。

对于**问题2**，它会评估不同准确率目标下的成本—准确率权衡；我们还纳入了**LOTUS（+ 保证）** [49]。LOTUS 将模型级联嵌入筛选运算符中，但它保证的是精确率和召回率，而非整体准确率。由于 LOTUS 要求分别设定精确率和召回率目标，而不是单一的准确率目标，因此我们无法在固定准确率水平下（如**问题**一和**Q3**中）将其与我们的方法直接比较。对于**问题2**，我们使用 LOTUS 的`sem_filter` 运算符实现其基线，并在每个二元分类工作负载上评估精确率和召回率目标的所有组合：{0.60，0.80，0.90，0.95} ×{0.60，0.80，0.90，0.95}。

7.1.3 对比的变体。 与基线一样，所有方法都使用一个代理模型和预言机模型。我们评估了**任务级联**（我们的主要方法），以及**任务级联（+ 保证）**。我们还评估了**任务级联（轻量版）**，这是一种轻量级变体，它采用与**任务级联** 相同的参数来尽可能降低优化成本，但仅使用代理模型评估所有候选代理任务，不使用预言机，从而大幅降低了 C_{eval}（如第 6 节所述）。需要注意的是，我们将作为方法的**任务级联**与该方法产生的任务级联区分开来；在本节中，**任务级联** 指我们的这一方法，并视为单一实体（例如，**"任务级联**实现了更低的成本"），而"任务级联"则指该方法发现的单个级联配置。我们还纳入了任务级联的不同变体，以单独考察我们方法中各个组成部分的作用：

(1) 代理操作发现变体。 **无代理操作**会禁用代理发现，因此级联仅包含在不同文档比例下的原始用户指定操作。**单次迭代** 会在单个批次中生成所有代理操作，不进行迭代优化。

(2) 文档剪枝变体。 **无过滤** 会禁用学习式文档剪枝，因此所有任务都必须处理完整文档。**朴素 RAG 过滤器** 会用简单的检索过滤器替代学习式剪枝，该过滤器根据文本块与用户定义操作的嵌入之间的余弦相似度对文本块进行排序。

(3) 替代级联设计。 不同于采用默认的贪心策略，**选择性排序** 会优先选择（选择性 −1）/成本比最高的操作来构建级联，这与先前关于谓词排序的研究 [25],一致；其中，选择性是指未被任务分类、因而会传递到级联后续环节的文档数量。**重构（前 25%）** 会应用我们学习得到的文档重构来重新排列每份文档，然后仅允许代理模型处理重构内容的前 25%，而预言机仍像往常一样处理完整文档。此方法通过双任务级联，隔离学习式重构和剪枝的效果，同时不引入代理发现，也不增加任务数量至 2 个以上。最后，**检索增强生成 +NoSur** 仅应用朴素的基于检索的剪枝，不进行代理发现。此变体使我们能够量化简单检索带来的益处，并隔离任务级联的额外价值。

7.1.4 指标与实现细节。 我们报告以美元（$）计的平均推理成本，以及每种方法达到准确率目标 α 的工作负载占比。大多数实验使用固定的 $\alpha = 0.9$；在一项实验中，我们将 α 从 0.75 调整至 0.95。对于具有保证的方法，我们将失败概率设置为 $\delta = 0.25$。

我们使用 OpenAI 模型：GPT-4o 作为预言机模型，GPT-4o-mini 作为代理模型，o1-mini 用于代理生成，以及 文本嵌入-

3-small 用于生成嵌入。我们在第4节中使用 PyTorch 构建相关性分类器。所有大语言模型调用的温度均设为 0。我们考虑四种文档比例 F = {0.1,0.25,0.5,1.0}，并以三次迭代 ($n_a = 3$) 运行代理发现，每次迭代执行三种代理操作 ($n_s = 5$)。这些参数值在所有工作负载中保持不变。尽管增加迭代次数或代理模型数量可能会提高离线成本，但也能降低级联成本；我们发现，这些默认设置适用于各种任务，并在第 D 节中进行了更多实验，以证明任务级联对参数选择具有稳健性。

推理成本根据 OpenAI API 定价计算：GPT-4o 的输入词元为每 100 万个 2.50 美元，输出词元为每 100 万个 10.00 美元；GPT-4o-mini 的输入词元为每 100 万个 0.15 美元，输出词元为每 100 万个 0.60 美元；前缀缓存补全可享受 50% 的折扣。所有报告的推理成本还包括嵌入成本（每 100 万个词元 0.02 美元）；这约占 GPT-4o-mini 推理成本的 0.13×，与大语言模型推理总成本相比可以忽略不计。所有方法均使用 Python 实现，并在一台 2024 款 MacBook Air（M4 芯片）上运行，通过 OpenAI API 调用产生的费用超过 3,000 美元。

7.2 结果

现在，我们将介绍针对上述问题的实验结果。表3列出了主要的成本和准确率结果。我们将在第 7.3 节探讨优化成本，并将对这些成本和延迟的详细讨论留至 E节。

7.2.1 问题一：成本节省与组件重要性。**任务级联**相比**双模型级联**可将推理成本降低 41%；在考虑准确率保证时，降幅达到 48.5%。**任务级联（轻量版）**是我们降低优化成本的轻量级变体，其成本节省几乎相同（平均为 0.62× 对 0.59×），同时所需的优化投入减少 75%。在包含大量无关内容的工作负载（例如安然和法律）上，成本节省最为显著；相较于**双模型级联**，推理成本最高可降低 6×。

达到准确率目标。**任务级联**和**任务级联（+ 保证）**在一个工作负载上的准确率均未达到 90%准确率目标，差距为 < 1%；而**双模型级联**在 8 个工作负载中的 3 个上未达标。当需要准确率保证时（+ 保证变体），两种方法都仅使用一半的可用数据来构建级联，将其余数据留作阈值调整（算法 3），这使**任务级联**的成本平均增加 1.43× 。

组件贡献。这三个核心组件——贪心任务排序、代理操作发现和文档剪枝——对于实现稳健的性能都至关重要。**选择性排序**的表现最差（比**任务级联**差 7.5× ）。**单次迭代**（比**任务级联**差 1.13× ）表明，与单轮代理生成相比，迭代式优化还能进一步降低成本。令人意外的是，**无代理操作**（1.21×）和**无过滤**（1.55×）的成本都高于双模型基线。没有代理模型时，代理模型只能执行用户定义的操作，因此仅在一半工作负载上节省成本；对于其余任务，则必须处理整个文档，从而抵消成本优势。没有文档剪枝时，每个代理操作都必须处理整个文档，因此每个代理操作的成本都与**双模型级联**的第一阶段一样高。同样，**重构（前 25%）**在 8 个工作负载中的 6 个上未能降低成本：当代理模型获得的上下文过少，无法执行原始操作时，就无法有把握地判定文档是否符合要求，导致更多请求升级至成本高昂的预言机。

同样，**重构（前 25%）**在 8 个工作负载中的 6 个上未能降低成本：当代理模型获得的上下文过少，无法执行原始操作时，就无法有把握地判定文档是否符合要求，导致更多请求升级至成本高昂的预言机。在不同工作负载中，较简单的变体可能会出现明显问题：**无过滤**比**任务级联**差至多 8×，**无代理操作**差至多 4× ，所有变体中的最大性能退化平均比**任务级联**差 12× 。通过结合多种技术，**任务级联**能够在多种工作负载下实现稳健的成本节省。

与基于检索的剪枝进行比较。 我们学习得到的文档剪枝优于简单的检索替代方案。**朴素 RAG 过滤器** 的平均成本排名第二低，但**任务级联** 还能进一步降低 11% 的成本。更值得注意的是，**检索增强生成 +NoSur** （不含代理发现的基于检索的剪枝）的成本平均比**双模型级联** 高 1.16×： 朴素的文档重排可能会让重构后的文档更难由代理模型处理，从而使更多文档被送交成本高昂的预言机。加入代理发现（**朴素 RAG 过滤器**）可解决这一性能退化，使其在所有工作负载上的成本都低于**双模型级联** 。

固定目标下的准确率-成本权衡。 在固定的 90% 准确率目标下，任务级联能够探索更广阔的配置空间，从而实现更精细的成本-准确率权衡。在某些情况下，任务级联能找到既更便宜又 比基线更准确的方案：在安然数据集上，**任务级联** 以 0.11× 的成本达到 95.5% 的准确率，而**双模型级联**以 1.0× 的成本达到 89.1% 的准确率。在其他情况下，任务级联能找到成本更低且更接近目标的方案：在 FEVER数据集上，**任务级联** 以 0.54× 的成本达到 90.6% 的准确率，而**双模型级联** 以 1.0× 的成本达到 94.4% 的准确率。总体而言，**任务级联** 能够搜索更多配置，并且往往能找到更接近目标准确率的方案。

7.2.2 问题*2*：不同目标准确率下的准确率—成本权衡。为了评估不同准确率要求下的性能，我们将目标准确率从 75% 提高到 95%，每次递增 5%，并采用与第 7.2.3 节相同的工作负载分组。图5展示了由此得到的准确率—成本权衡。我们在二元分类工作负载（除 agnews 外的所有工作负载）上纳入 **LOTUS（+ 保证）**，采用第 7.1 节所述的设置。总体而言，与所有基线相比，**任务级联**降低了推理成本，并在简单和困难任务上拓展了帕累托前沿。**LOTUS（+ 保证）**在所有工作负载上的表现与**双模型级联（+ 保证）**相当。对于较简单的 A组任务（安然、法律），**任务级联**始终占据帕累托前沿；无论目标准确率如何，它都能以低得多的成本达到或超过模型级联的准确率。对于更具挑战性的 B组工作负载（games、court），在较低的目标准确率（75%—85%）下，**任务级联**带来的成本节省最大，因为智能体可以从更多满足这些宽松准确率目标的代理操作中进行选择。随着目标准确率提高，满足更严格要求的代理模型会变少，因此成本优势也会减弱；但**任务级联**仍经常能达到或超过基线，这可能得益于文档剪枝。在 C组（agnews）中，代理模型本身已达到较高准确率，因此改进空间有限：在较高目标准确率下，**任务级联**的表现与基线相近；而在较低准确率目标下，它能够找到成本更低的方案。

表3：在每个工作负载抽取1,000份文档的样本上，以90%准确率目标进行推理的结果（适用于文档数量超过1,000份的数据集）。每个单元格显示平均准确率和成本。对于所有主要方法（预言机、双模型级联和任务级联），结果均取三次试验的平均值；变体结果基于单次试验。基线成本以美元表示。其他成本以相应双模型级联变体（普通版或 +G）的倍数表示。"平均成本"列计算的是该方法达到90%目标的各工作负载上的平均值。加粗条目标示每个工作负载中各方法的最低成本。

方法	agnews	court	安然	FEVER数据集	games	法律	PubMed	维基_talk	平均成本
仅使用预言机	$0.45	$11.05	$6.34	$13.01	$3.13	$10.20	$8.36	$2.94	$6.94
双模型级联	94.1% |$0.03	89.4% | $2.81	89.1% | $0.61	94.4% | $0.80	89.8% | $0.78	91.5% | $3.23	92.5% | $0.56	91.4% | $0.18	$1.13
双模型级联（+G）	94.7% | $0.04	93.0% |$4.06	96.2% | $1.56	96.0% | $2.22	95.2% | $1.29	94.8% | $4.12	93.6% | $0.83	95.6% | $0.49	$1.83
任务级联	95.3% | 0.66×	88.0% | 0.72×	95.5% | 0.11×	90.6% | 0.54×	92.0% | 1.09×	91.3% | 0.27×	91.6% | 0.84×	91.5% | 0.64×	0.59×
任务级联（+G）	95.4% | 0.71×	90.8% | **0。53×**	95.6% | **0。09×**	92.4% | 0.44×	89.2% | **0。49×**	91.2% | 0。26×	93.5% | 1。23×	92.3% | **0。37×**	**0。52×**
任务级联（轻量版）	94.1% | 0.78×	88.6% | 0.76×	95.5% | 0.14×	90.6% | 0.63×	90.2% | 0.70×	90.2% | 0.41×	91.4% | 0.86×	89.9% | 0.68×	0.62×
无代理操作	95.1% | 1.43×	94.5% | 1.25×	98.0% | 0.49×	92.5% | 0.86×	90.5% | 1.16×	90.0% | 0.95×	91.9% | 0.79×	96.0% | 2.77×	1.21×
单次迭代	92.4% | **0。56×**	93.8% | 1.21×	95.9% | 0.15×	90.0% | **0。30×**	89.2% | 1.21×	92.8% | 1.04×	93.4% | **0.76×**	90.6% | 0.61×	0.66×
无过滤	93.9% | 0.74×	90.0% | 1.10×	98.5% | 0.96×	95.5% | 1.82×	98.4% | 1.84×	96.1% | 0.93×	95.1% | 1.47×	98.4% | 3.51×	1.55×
朴素 RAG 过滤器	95.1% | 0.95×	90.9% | 0.94×	97.2% | 0.30×	90.9% | 0.42×	85.1% | 0.61×	88.9% | **0。12×**	90.5% | 0.88×	89.9% | 0.44×	0.65×
选择性排序	94.2% | 1.20×	90.8% | 2.33×	98.8% | 5.50×	93.1% | 5.08×	90.1% | 2.32×	92.8% | 1.86×	93.8% | 8.44×	96.9% | 8.81×	4.44×
重构（前 25%）	96.5% | 2.67×	93.9% | 1.22×	99.4% | 0.85×	90.5% | 0.37×	97.1% | 1.70×	95.3% | 1.78×	92.6% | 2.38×	98.1% | 3.49×	1.81×
检索增强生成 + NoSur	96.0% | 1.83×	92.1% | 1.10×	98.2% | 1.73×	91.2% | 0.54×	87.9% | 1.09×	90.0% | 0.77×	90.5% | 0.94×	93.4% | 1.21×	1.16×

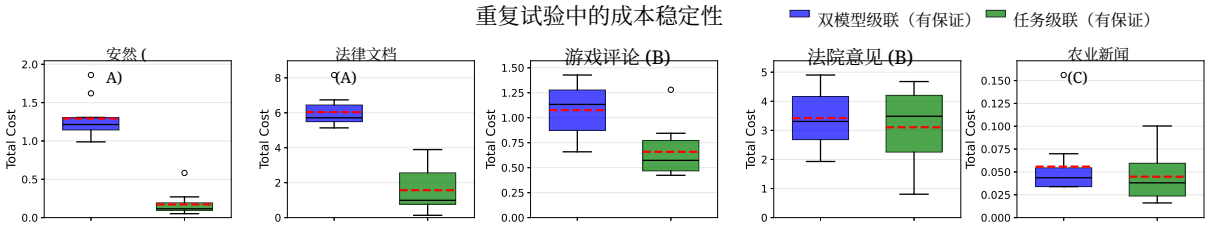

图4：以双模型级联和任务级联在工作负载上进行的 10 次独立运行中的任务级联（+ 保证）。每个箱线图展示了总推理成本的分布；红色虚线表示中位数。两种方法的方差都很高。

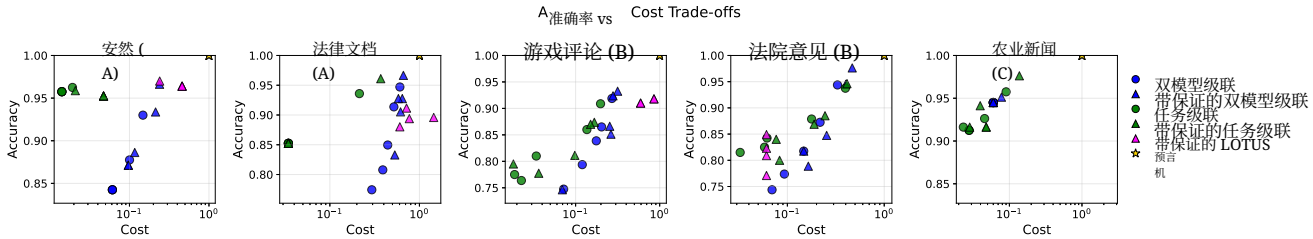

图5：目标准确率从 75% 变化到 95%（每次递增 5%）时，准确率与成本之间的权衡。任务级联（绿色）在简单工作负载上占据帕累托前沿，并在较难的任务上带来稳健的收益或新的工作点。对于 agnews 这类简单工作负载，成本改善主要出现在较低的目标准确率下，此时基线无法与之匹敌。LOTUS（+保证）（粉色）仅支持二元分类工作负载，因此我们排除了 agnews。

7.2.3 Q3：重复试验中的一致性与方差。接下来，我们评估任务级联的结果在多次重复运行中的一致性，同时考虑到基于大语言模型的代理模型发现过程本身具有随机性。为提供具有代表性的分析，同时兼顾计算资源和成本限制，我们选取了五个工作负载，涵盖不同的文档长度和任务复杂度。根据表3中观察到的结果，我们将它们分为三组：A组（安然和法律），任务级联在其中带来的成本降幅最大；B组（games 和 court），在其中最难找到低成本且准确的级联；C组（agnews），代理模型单独使用时的准确率已经很高，且文档较短，因此文档剪枝技术难以带来节省。

接下来，我们评估任务级联在重复运行中的结果一致性，因为基于大语言模型的代理模型发现过程本身具有随机性。为在控制计算和成本的同时提供具有代表性的分析，我们选取了五种工作负载，涵盖不同的文档长度和任务复杂度。我们根据观察到的难度将它们分为表3中的三组：A组（安然、法律），任务级联在这组中实现的成本降幅最大；B组（games、court），在这组中最难找到成本低且准确的级联；C组（agnews），这组中的文档较短，且代理模型已经具有很高的准确率。如图4所示，我们针对每种工作负载进行了十次独立运行，评估了**双模型级联（+ 保证）**和**任务级联（+ 保证）**。在A组工作负载中，**任务级联（+ 保证）**使安然和法律数据集的平均成本分别降低了86.5%和74.1%，而且即便是成本最高的任务级联

运行，其成本仍低于成本最低的双模型级联。在B组工作负载中，由于任务难度增加，两种方法的方差都更大，但任务级联成本的第*75*百分位数仍低于基线成本的第*25*百分位数。在C组（agnews）中，基线已经接近最优，**任务级联（+ 保证）**的成本和方差与之持平。

两种方法都表现出明显的运行间方差。**双模型级联（+ 保证）**也表现出较高的变异性，这可能是因为开发集规模较小，且用于阈值调整的数据划分流程会给级联构建带来不确定性。**任务级联（+ 保证）**在所有工作负载下的均值和中位数成本始终更低，表明这种变异性并未削弱其优势。

7.2.4 问题*4*：优化成本。迄今为止，我们一直关注推理成本，因为在生产规模部署中，这部分成本占主导地位，而离线优化成本可由在线收益抵消。为考察这项投入何时能够"收回成本"，我们报告优化成本、盈亏平衡点（即总成本低于仅使用**预言机**时的成本节点），以及处理100万份文档的估算成本（选取这一规模是为了展示规模化带来的影响）。完整详情见E节，表4中对此作了说明。

处理100万份文档时，优化成本变得微不足道，任务级联则能带来可观的成本节省。**任务级联（轻量版）**——我们在第 7.1 节介绍的轻量级变体——在所有工作负载上的成本都低于**双模型级联**，平均节省37.5%，在安然数据集上最高可达86%。**任务级联**在8种工作负载中的7种上实现了成本节省（平均节省36%，在安然数据集上最高可达87%）；对于games，**任务级联**无法节省成本，而**双模型级联**则未达到90%准确率目标（表3）。

尽管**任务级联**的优化成本高于**双模型级联**（在不同工作负载下为 7.9×–27.0× 倍，均值为 11.4× 倍），且构建时间更长（优化延迟为 ~ 14×），但仍实现了这些成本节省。我们的轻量版**任务级联（轻量版）**变体的成本是**双模型级联**基线的 2.2×–4.7× 倍（均值为 2.9× 倍），同时保留了 95% 的推理成本节省。对于**任务级联**，平均处理 2,986 份文档后即可收回优化成本；而**任务级联（轻量版）**仅需处理 678 份——相较于我们的工作负载规模（中位数为 129k 份文档，如表2所示），这只是适度的阈值。推理延迟处于同一数量级（E节中的表 5）。

总之，尽管任务级联需要较高的前期优化投入，但在实际规模下，这项成本很快就会变得微不足道，表明任务级联适用于生产部署且具有成本效益。

7.3 分析任务级联的洞见

选择任务级联方法的变体。**任务级联**方法在不同工作负载下都能持续降低成本，因此是可靠的默认选择。在特定场景中，较简单的变体可能更合适：**朴素 RAG 过滤器**适用于工程资源有限且文档中相关部分易于识别的情况；**无代理操作**适用于原始操作难度不高且代理模型能够准确执行该操作的情况；如果文档非常短（例如 agnews），则适合使用不带过滤的变体。**单次迭代**变体尤其值得关注，它在降低离线成本的同时实现了出色的性能，并且随着大语言模型智能体推理和处理复杂任务的能力提升，很可能会进一步改进。尽管个别变体偶尔会优于完整方法，但**任务级联**的平均成本比次优变体低 20%，因此推荐将其作为起点。

参数选择指南。任务级联取决于三个超参数：每次迭代的代理操作数量（n_s）、细化迭代次数（n_a）以及文档比例集合（F）。我们在两个具有代表性的工作负载上进行了敏感性分析，分别调整 $n_s \in \{3、5、10\}$、$n_a \in \{1、2、3\}$，并测试不同的 F 配置（详细结果见第 D 节）。在所有配置下，任务级联始终优于基线，表明其对参数选择具有稳健性。根据这些实验，我们建议：$n_s \geq 3$、$n_a \geq 1$，并采用从较小值（0.1 或 0.25）到完整值（1.0）的文档比例。我们发现，n_s 对降低成本的影响大于 n_a，因此从业者应优先探索多样化的代理模型，而非进行大量细化。

8 相关工作

我们涵盖三个领域：大语言模型驱动的数据处理、经济高效的大语言模型执行，以及利用大语言模型进行查询改写。

大语言模型驱动的数据处理。现代数据系统日益将大语言模型集成为一等算子，用于自然语言抽取与转换。Databricks、DuckDB、Snowflake 和 Google AlloyDB 等工业系统在 SQL 中支持大语言模型驱动的筛选和映射函数 [16, 21, 55, 67]。LOTUS [49]、、ThalamusDB [30]，和 Aryn [1] 分别提供基于 pandas、SQL 和类似 Spark 的接口。Palimpzest [41]、、DocETL [53]，和 AOP [65] 为大语言模型驱动的数据处理提供了定制的领域特定语言。然而，这些系统通常会对每份文档独立调用大语言模型，从而导致较高的推理成本。有些系统专注于支持特定查询或数据模态的大语言模型驱动的数据处理 [3, 40, 44, 61]。

大语言模型推理的成本优化。降低大语言模型推理成本的策略包括基于成本的优化器、专用模型、缓存、集成和模型级联。ABACUS [52]针对常见大语言模型驱动算子的不同"物理"实现引入了 Cascades 风格优化器 [23]，而 ELEET [62] 则用可训练模型替代大语言模型算子。其他研究会重新排序大语言模型调用，以复用 KV 缓存 [43]。有些方法会对多个大语言模型进行性能分析，并汇总其输出以降低成本。例如，ThriftLLM [27] 和 LLMBlender [28], 会集成所有已评估模型的预测，而不像模型级联那样仅使用最后一个处理阶段的输出。不过，在我们研究的主要场景中——级联模型在成本和质量上差异显著——将预言机的输出（若可用）与代理模型的输出集成，并不会带来额外收益。SpareLLM [31] 会针对一组任务对多个大语言模型进行性能分析，但最终只选择一个模型来处理剩余任务；就我们的情况而言，这相当于对所有元素采用只有一个步骤的级联，即使用代理模型或预言机，且不考虑置信度。我们的工作以模型级联 [2, 5, 9, 33–35, 45, 49, 64, 70], 为基础并将其推广，在每个阶段不仅改变模型，还改变操作和文档比例。

使用大语言模型进行查询改写。由大语言模型驱动的查询改写已被证明在检索、问答 [4, 10, 46, 50], 和

传统查询优化 [15, 39, 57, 72]中卓有成效。DocETL [53] 将复杂任务分解为逻辑等价的子任务，以提高准确率。与此不同，我们会对任务进行改写，以采用成本更低、可能不正确的操作，从而大幅扩展改写空间。任务级联可能与逻辑分解截然不同——同一操作可以在不同的文档比例下多次出现。

我们的工作也建立在近似查询处理（近似查询处理） [7, 19] 之上。尽管我们的方法专为非结构化数据和大语言模型设计，但其组件与经典近似查询处理相似：代理操作类似于近似谓词 [54], ，文档比例利用抽样，我们还使用集中界来提供准确率保证 [8, 35]。

9 结论

我们提出了任务级联，这是对模型级联框架的推广，适用于高效的大语言模型驱动的非结构化文本处理。任务级联在每个阶段调整模型、操作和文档比例，以便在满足准确率目标的同时，最大限度地降低推理成本。在八种真实世界工作负载上，与模型级联相比，任务级联平均可降低 48.5% 的推理成本（且保证水平相当）；与仅使用预言机进行推理相比，推理成本降低 86.2%。消融实验也证实了我们方法中所有组成部分的重要性。总而言之，任务级联为可扩展的非结构化数据分析提供了一种实用且易于扩展的解决方案。

致谢

我们感谢美国国家科学基金会授予的 DGE-2243822、IIS-2129008、IIS-1940759 和 IIS-1940757 项目资助、加利福尼亚州提供的资金、NDSEG奖学金、阿尔弗雷德·P·斯隆基金会提供的资金，以及 EPIC实验室赞助方 Adobe、Google、G-Research、Microsoft、PromptQL、Sigma Computing、Bridgewater 和 Snowflake。Azure、Modal、美国国家科学基金会（通过 NAIRR）和 OpenAI 提供了计算额度。

参考文献

[1] Eric Anderson、Jonathan Fritz、Austin Lee、Bohou Li、Mark Lindblad、Henry Lindeman、Alex Meyer、Parth Parmar、Tanvi Ranade、Mehul A. Shah、Benjamin Sowell、Dan Tecuci、Vinayak Thapliyal 和 Matt Welsh。2025年。《大语言模型驱动的非结构化分析系统的设计》。载于创新数据系统研究会议（*CIDR*）论文集。https://mail.vldb.org/cidrdb/papers/2025/p13-anderson.pdf[2] Michael R Anderson、Michael Cafarella、German Ros 和 Thomas F Wenisch。2019年。基于物理表示的可视分析数据库谓词优化。载于*2019*年 *IEEE* 第*35*届国际数据工程会议（*ICDE*）。IEEE，1466–1477。[3] Simran Arora、Brandon Yang、Sabri Eyuboglu、Avanika Narayan、Andrew Ho- jel、Immanuel Trummer 和 Christopher Ré。2023年。语言模型使异构数据湖生成结构化视图的简单系统成为可能。*VLDB*基金会会议录 17，2（2023年），92–105。[4] Muhammad Imam Luthfi Balaka、David Alexander、Qiming Wang、Yue Gong、Adila Krisnadhi 和 Raul Castro Fernandez。2025年。Pneuma：利用大语言模型实现表格数据表示与检索的端到端系统。*ACM* 数据管理会议录3，3（2025年），1–28。[5] Jiashen Cao、Karan Sarkar、Ramyad Hadidi、Joy Arulraj 和 Hyesoon Kim。2022年。Figo：视频分析中的细粒度查询优化。载于*2022*年 *ACM* 数据管理国际会议论文集。559–572。[6] Jonathan P Chang、Caleb Chiam、Liye Fu、Andrew Wang、Justine Zhang 和 Cristian Danescu-Niculescu-Mizil。2020年。ConvoKit：对话分析工具包。载于话语与对话特别兴趣小组第*21*届年会论文集。57–60。[7] Surajit Chaudhuri、Bolin Ding 和 Srikanth Kandula。2017年。近似查询处理：并非灵丹妙药。载于*2017*年 *ACM* 数据管理国际会议论文集。511–519。[8] Surajit Chaudhuri、Rajeev Motwani 和 Vivek Narasayya。1998年。用于直方图构建的随机采样：多少才够？*ACM SIGMOD Record* 27，2（1998年），436–447 .

[9] Lingjiao Chen、Matei Zaharia 和 James Zou。2024年。FrugalGPT：如何在降低成本的同时使用大语言模型并提升性能。《机器学习研究汇刊》（2024年）。[10] Peter Baile Chen、Yi Zhang、Mike Cafarella 和 Dan Roth。[无日期]。我们能否一次性检索所有内容？ARM：一种面向对齐的基于大语言模型的检索方法。载于 ACL 2025 表格表示学习研讨会第 4 届会议。[11] Arman Cohan、Franck Dernoncourt、Doo Soon Kim、Trung Bui、Seokhwan Kim、Walter Chang 和 Nazli Goharian。2018年。用于长文档抽象式摘要的话语感知注意力模型。载于《2018年计算语言学协会北美分会年会论文集：人类语言技术》，第 2 卷（短篇论文）。计算语言学协会，新奥尔良，路易斯安那州，615–621。https://doi. org/10 18653 1 18 2097 。/v /N - Cristian Danescu-Niculescu-Mizil、Lillian [12] Lee、Bo Pang 和 Jon Kleinberg。2012年。权力的回响：语言效应与社会互动中的权力差异。载于《第 21 届万维网国际会议论文集》。699–708。[13] Gianna M Del Corso、Antonio Gulli 和 Francesco Romani。2005年。新闻流排序。载于《第 14 届万维网国际会议论文集》。97–106。[14] Shrey Desai 和 Greg Durrett。2020年。预训练 Transformer 的校准。载于《2020年自然语言处理实证方法会议（EMNLP）论文集》。295–302。[15] Sriram Dharwada、Himanshu Devrani、Jayant Haritsa 和 Harish Doraiswamy。2025年。通过大语言模型进行查询改写。arXiv 预印本 arXiv:2502.12918（2025年）。[16] Till Döhmen。2024年。隆重介绍 prompt() 函数：借助 SQL 运用大语言模型的强大功能！网址 https://motherduck.com/blog/sql-llm-prompt-function-gpt-models/。访问日期：2025-06-22。[17] Yu Fan、Jingwei Ni、Jakob Merane、Etienne Salimbeni、Yang Tian、Yoan Hermstrüwer、Yinya Huang、Mubashara Akhtar、Florian Geering、Oliver Dreyer 等。2025年。LEXam：对 340 场法律考试中的法律推理进行基准评测。arXiv 预印本 arXiv:2505.12864（2025年）。[18] Uriel Feige、László Lovász 和 Prasad Tetali。2004年。最小和集合覆盖的近似算法。Algorithmica 40（2004年），219–234。[19] Minos N Garofalakis 和 Phillip B Gibbons。2001年。近似查询处理：驾驭太字节级数据……载于 VLDB，第 10 卷。645927–672356。[20] Google Cloud。2025年。使用 Gemini 进行批量预测。https://cloud.google.com/vertex-ai/generative-ai/docs/multimodal/batch-prediction-gemini。最后更新：2025-10-15；访问日期：2025-10-16。[21] Google Cloud。2025年。使用 AlloyDB AI 查询引擎执行智能 SQL 查询。https://cloud.google.com/alloydb/docs/ai/evaluate-semantic-queries-ai-operators。访问日期：2025-06-22；最后更新：2025-06-11。[22] gqfiddler。2022年。美国最高法院意见书。https://www.kaggle.com/datasets/gqfiddler/scotus-opinions。访问日期：2025-07-16。[23] Goetz Graefe。1995年。用于查询优化的 Cascades 框架。《IEEE 数据工程通报》18（1995年），19–29。https://api.semanticscholar.org/ CorpusID:260706023[24] Neha Gupta、Harikrishna Narasimhan、Wittawat Jitkrittum、Ankit Singh Rawat、Aditya Krishna Menon 和 Sanjiv Kumar。 [无日期]。语言模型级联：词元级不确定性及其他。载于第十二届国际学习表征会议。[25] Joseph M. Hellerstein 和 Michael Stonebraker。1993年。谓词迁移：优化包含高代价谓词的查询。载于《1993年 ACM SIGMOD 数据管理国际会议论文集》（美国华盛顿特区）（SIGMOD '93）。计算机协会，纽约，美国，267–276。https://doi.org/10 1145 170035 170078 。/ 。Dan Hendrycks、Collin Burns、[26] Anya Chen 和 Spencer Ball。2021年。CUAD：用于法律合同审查的专家标注自然语言处理数据集。NeurIPS（2021年）。[27] Keke Huang、Yimin Shi、Dujian Ding、Yifei Li、Yang Fei、Laks Lakshmanan 和 Xiaokui Xiao。2025年。ThriftLLM：针对分类查询经济高效地选择大语言模型。arXiv 预印本 arXiv:2501.04901（2025年）。[28] Dongfu Jiang、Xiang Ren 和 Bill Yuchen Lin。2023年。LLM-Blender：通过成对排序和生成式融合集成大语言模型。载于《第 61 届计算语言学协会年会论文集》（第 1 卷：长篇论文）。14165–14178。[29] Zhengbao Jiang、Jun Araki、Haibo Ding 和 Graham Neubig。2021年。我们如何知道语言模型何时掌握了知识？论用于问答的语言模型校准。《计算语言学协会汇刊》9（2021年），962–977。[30] Saehan Jo 和 Immanuel Trummer。2024年。ThalamusDB：多模态数据上的近似查询处理。《ACM 数据管理会议录》2，3（2024年），1–26。[31] Saehan Jo 和 Immanuel Trummer。2025年。SpareLLM：在等价约束下自动选择特定任务的最低成本大语言模型。《ACM 数据管理会议录》3，3，文章 219（2025年6月），26 页。https://doi.org/10.1145/3725356

[32] Saurav Kadavath、Tom Conerly、Amanda Askell、Tom Henighan、Dawn Drain、Ethan Perez、Nicholas Schiefer、Zac Hatfield-Dodds、Nova DasSarma、Eli Tran-Johnson 等。2022年。语言模型（大多时候）知道自己知道什么。arXiv 预印本 arXiv:2207.05221（2022年）。[33] Daniel Kang、Peter Bailis 和 Matei Zaharia。[无日期]。BlazeIt：优化用于基于神经网络的视频分析的声明式聚合查询和限制查询。《VLDB基金会会议录》13，4（[无日期]）。[34] Daniel Kang、John Emmons、Firas Abuzaid、Peter Bailis 和 Matei Zaharia。2017年。NoScope：大规模优化视频上的神经网络查询。《VLDB基金会会议录》10，11（2017年）。[35] Daniel Kang、Edward Gan、Peter Bailis、Tatsunori Hashimoto 和 Matei Zaharia。2020年。使用代理实现带保证的近似选择。《VLDB基金会会议录》13，12（2020年），1990–2003。[36] Daniel Kang 和 John Guibas。2021年。利用高代价谓词加速近似聚合查询。《VLDB基金会会议录》14（2021年），2341–2354。 https://api.semanticscholar.org/CorpusID:237012342 [37] Daniel Kang、John Guibas、Peter D Bailis、Tatsunori Hashimoto 和 Matei Zaharia。2022年。Tasti：面向非结构化数据上基于机器学习的查询的语义索引。载于《2022年国际数据管理会议论文集》。1934–1947。[38] Bryan Klimt 和 Yiming Yang。2004年。安然语料库：用于电子邮件分类研究的新数据集。载于《欧洲机器学习会议》。Springer，217–226。[39] Zhaodonghui Li、Haitao Yuan、Huiming Wang、Gao Cong 和 Lidong Bing。2024年。LLM-R2：用于提升查询效率的大语言模型增强型基于规则的重写系统。《VLDB基金会会议录》18，1（2024年），53–65。[40] Yiming Lin、Madelon Hulsebos、Ruiying Ma、Shreya Shankar、Sepanta Zeighami、Aditya G Parameswaran 和 Eugene Wu。2025年。使用大语言模型查询模板化文档集合。载于《2025年 IEEE 第 41 届国际数据工程会议（ICDE）》。IEEE 计算机学会，2422–2435。[41] Chunwei Liu、Matthew Russo、Michael Cafarella、Lei Cao、Peter Baile Chen、Zui Chen、Michael Franklin、Tim Kraska、Samuel Madden、Rana Shahout 等。2025年。Palimpzest：通过声明式查询处理优化人工智能驱动的分析。载于《创新数据库研究会议论文集》（CIDR）。2。[42] Nelson F Liu、Kevin Lin、John Hewitt、Ashwin Paranjape、Michele Bevilacqua、Fabio Petroni 和 Percy Liang。2024年。迷失在中间：语言模型如何利用长上下文。《计算语言学协会汇刊》12（2024年），157–173。[43] Shu Liu、Asim Biswal、Amog Kamsetty、Audrey Cheng、Luis Gaspar Schroeder、Liana Patel、Shiyi Cao、Xiangxi Mo、Ion Stoica、Joseph E Gonzalez 等。2025年。优化关系数据分析工作负载中的大语言模型查询。载于第八届机器学习与系统会议。[44] Duo Lu、Siming Feng、Jonathan Zhou、Franco Solleza、Malte Schwarzkopf 和 Uğur Çetintemel。2025年。VectraFlow：将向量集成到流处理中。CIDR。[45] Yao Lu、Aakanksha Chowdhery、Srikanth Kandula 和 Surajit Chaudhuri。2018年。利用概率谓词加速机器学习推理。载于《2018年国际数据管理会议论文集》。1493–1508。[46] Xinbei Ma、Yeyun Gong、Pengcheng He、Hai Zhao 和 Nan Duan。2023年。检索增强型大语言模型中的查询改写。载于《2023年自然语言处理实证方法会议论文集》。5303–5315。[47] Matteo Magnini、Gianluca Aguzzi 和 Sara Montagna。2025年。用于个人医疗助理聊天机器人的开源小型语言模型。《基于智能的医学》11（2025年），100197。[48] OpenAI。2025年。批量请求指南。https://platform.openai.com/docs/guides/batch。访问日期：2025-10-16。[49] Liana Patel、Siddharth Jha、Melissa Pan、Harshit Gupta、Parth Asawa、Carlos Guestrin 和 Matei Zaharia。2025年。语义算子及其优化：在 LOTUS 中实现具备准确率保证的基于大语言模型的数据处理。《VLDB基金会会议录》18，11（2025年），4171–4184。[50] Wenjun Peng、Guiyang Li、Yue Jiang、Zilong Wang、Dan Ou、Xiaoyi Zeng、Derong Xu、Tong Xu 和 Enhong Chen。2024年。淘宝搜索中的基于大语言模型的长尾查询改写。载于《ACM 万维网会议 2024 配套论文集》。20–28。[51] Matthew Russo、Tatsunori Hashimoto、Daniel Kang、Yi Sun 和 Matei Zaharia。2023年。加速非结构化数据流上的聚合查询。《VLDB基金会会议录》16，11（2023年），2897–2910。[52] Matthew Russo、Sivaprasad Sudhir、Gerardo Vitagliano、Chunwei Liu、Tim Kraska、Samuel Madden 和 Michael Cafarella。2025年。Abacus：面向语义算子系统的基于成本的优化器。arXiv 预印本 arXiv:2505.14661（2025年）。[53] Shreya Shankar、Tristan Chambers、Tarak Shah、Aditya G Parameswaran 和 Eugene Wu。2025年。DocETL：面向复杂文档处理的智能体式查询改写与评估。《VLDB基金会会议录》18，9（2025年），3035–3048。[54] N. Shivakumar、H. Garcia-Molina 和 C. Chekuri。1998年。使用近似谓词进行过滤。载于第 24 届超大型数据库国际会议（VLDB 1998）。http://ilpubs.stanford.edu:8090/351/

[55] Snowflake。2025年。推出 Cortex AISQL：将 SQL 重新构想为面向多模态数据的人工智能查询语言。https://www.snowflake.com/en/blog/ai-sql- query-language/。访问日期：2025年7月15日。[56] Antoni Sobkowicz 和 Wojciech Stokowiec。2016。Steam 评论数据集——全新大规模情感数据集。载于第十届国际语言资源与评估会议（LREC 2016）情绪与情感分析研讨会论文集。55–58。[57] Yuyang Song、Hanxu Yan、Jiale Lao、Yibo Wang、Yufei Li、Yuanchun Zhou、Jianguo Wang 和 Mingjie Tang。2025年。QUITE：一种超越规则、基于大语言模型智能体的查询重写系统。arXiv 预印本 arXiv:2506.07675（2025）。[58] Sijun Tan、Xiuyu Li、Shishir G Patil、Ziyang Wu、Tianjun Zhang、Kurt Keutzer、Joseph Gonzalez 和 Raluca Popa。2024。LLoCO：离线学习长上下文。载于2024年自然语言处理实证方法会议论文集。17605–17621。[59] James Thorne、Andreas Vlachos、Christos Christodoulopoulos 和 Arpit Mittal。2018。FEVER：用于事实抽取与验证的大规模数据集。载于2018年计算语言学协会北美分会年会：人类语言技术论文集，第1卷（长篇论文）。809–819。[60] u/ICausedAnOutage 等。2023。Azure OpenAI API 的响应速度远慢于 ChatGPT 官方版本。https://www.reddit.com/r/AZURE/comments/1c1n48j/the_response_of_the_azure_openai_api_is_much/ Reddit 帖子，访问日期：2025-10-16。[61] Matthias Urban 和 Carsten Binnig。2024。CAESURA：将语言模型用作多模态查询规划器。载于第14届创新数据系统研究会议（CIDR 2024），美国加利福尼亚州 Chaminade，2024年1月14日至17日。www.cidrdb.org。https://www.cidrdb.org/cidr2024/papers/p14-urban.pdf [62] Matthias Urban 和 Carsten Binnig。2024。ELEET：在文本与表格上高效执行学习式查询。VLDB基金会会刊17，13（2024），4867–4880。[63] Vals AI, Inc. 2024。Vals AI。https://www.vals.ai/。访问日期：2025-06-22。[64] Paul Viola 和 Michael Jones。2001。使用简单特征的提升级联实现快速目标检测。载于2001年 IEEE 计算机学会计算机视觉与模式识别会议论文集。CVPR 2001，第1卷。IEEE，I–I。[65] Jiayi Wang 和 Guoliang Li。2025年。AOP：用于回答复杂查询的自动化交互式大语言模型流水线编排。CIDR。[66] Ian Waudby-Smith 和 Aaditya Ramdas。2024。通过下注估计有界随机变量的均值。《英国皇家统计学会会刊B辑：统计方法论》86，1（2024），1–27。[67] Patrick Wendell、Eric Peter、Nicolas Pelaez、Jianwei Xie、Vinny Vi-jeyakumaar、Linhong Liu 和 Shitao Li。2023年。推出 AI 函数：将大语言模型集成到 Databricks SQL 中。https://www.databricks.com/blog/2023/04/18/introducing-ai-functions-integrating-large-language-models-databricks-sql.html。访问日期：2025-06-22。[68] Xinyi Wu、Yifei Wang、Stefanie Jegelka 和 Ali Jadbabaie。2025年。论 Transformer 中位置偏差的出现。arXiv 预印本 arXiv:2502.01951（2025）。[69] Peng Xu、Wei Ping、Xianchao Wu、Lawrence McAfee、Chen Zhu、Zihan Liu、Sandeep Subramanian、Evelina Bakhturina、Mohammad Shoeybi 和 Bryan Catanzaro。2023年。检索与长上下文大语言模型相遇。载于第十二届国际学习表征会议。[70] Murong Yue、Jie Zhao、Min Zhang、Liang Du 和 Ziyu Yao。2024。结合思维表征混合实现高性价比推理的大语言模型级联。载于第十二届国际学习表征会议（ICLR）。https://openreview.net/forum?id=60kaSfANzh [71] Lianmin Zheng、Wei-Lin Chiang、Ying Sheng、Siyuan Zhuang、Zhanghao Wu、Yonghao Zhuang、Zi Lin、Zhuohan Li、Dacheng Li、Eric Xing 等。2023。使用 MT-Bench 和 Chatbot Arena 评判 LLM-as-a-judge。《神经信息处理系统进展》36（2023），46595–46623。[72] Xuanhe Zhou、Guoliang Li、Jianming Wu、Jiesi Liu、Zhaoyan Sun 和 Xinning Zhang。2023。一种学习式查询重写系统。《VLDB基金会会刊》16，12（2023），4110–4113。

级联组装与级联准确率的统计保证 A.1 贪心级联组装算法

本节给出第3.2.2节所述贪心级联组装算法的伪代码。该算法以一组符合条件的任务（经筛选并通过算法2分配阈值）作为输入，并通过迭代选择能够最大程度降低推理成本、同时满足每项任务准确率要求的任务，贪心地构建有序级联。

算法 4： 贪心级联组装（详细版）

输入： 符合条件的任务 $\mathcal{T}^{\text{eligible}}$ 及其阈值、开发集 D_{dev}，准确率目标 α
输出： 有序级联 π

$\pi \leftarrow \emptyset$;
while *true* **do**
  best_任务 $\leftarrow$ None; best_成本 $\leftarrow$ 成本 of π;
  foreach 未使用的任务 T *in* $\mathcal{T}^{\text{eligible}}$ **do**
    $\pi' \leftarrow \pi + [T]$; // 将 T 追加到级联末尾
    在 D_{dev} 上执行 π' ;
    如果 π'中的每个任务在其分类的文档上的准确率（即，置信度高于阈值） $\geq \alpha$ 且成本(π') < 最佳_成本 **then**
      最佳_任务 $\leftarrow T$; 最佳_成本 $\leftarrow$ 成本(π');
    end
  end
  如果 最佳_任务为 *None* **则**
    中断
  end
  $\pi \leftarrow \pi$ + [最佳_任务] ; // 更新当前级联
结束
返回 π;

A.2 估计器细节与定理 3.2 的证明

估计函数。为了达到目标准确率，函数 E 需要根据采样集 D_V 正确估计阈值集合是否达到准确率目标。更正式地说，设使用阈值集合 $\mathbf{t}_i$ 时的模型级联为 π_i，并设对于 (x_j, y_j)，模型级联在真实答案为 的输入样本

x y X_j^i = 上 $\mathbb{I}[\pi_i\ x_j\ = y_j]$ $\mathbb{E}\ X_j^i = 1\ =\ _D\ \pi_i$

的输出为 $\pi(x_j)$。定义 ( )。注意，( ) Acc ( )，因此 E 的目标是利用一组观测到的独立同分布伯努利随机变量 $X^i = \{X_j^i;\ \forall j\}$，估计它们的均值是否大于目标 T。这可以通过应用 Hoeffding 不等式等常见集中界来实现。我们采用 Waudby-Smith 和 Ra mdas [66] 的最新结果，该结果提供了比 Hoeffding 不等式更紧的界，如 [66]所示。 [66]的结果在估计观测值的均值时同时考虑了观测值的均值和方差，从而得到更紧的界。下面重述 [66], 的定理 3.2，定义我们的函数 E，并证明它能够以高概率估计观测值的均值是否大于 T。

引理 A.1（由以下定理 3.2 推得的推论： [66]）。考虑第i个级联阈值集合$\mathbf{t}_i$及相应的伯努利随机变量 X^i，其中 $|X^i| = k$。若$\text{Acc}_D(\mathbf{t}_i) < T$，且置信参数为 $\alpha \in [0, 1]$,

$$\mathbb{P}(\mathcal{E}(\mathbf{t}, D_V) = True) \leq \alpha, \quad where \tag{1}$$

$$\mathcal{E}(\mathbf{t}_i, D_V) = \mathbb{I}[\exists j \in [k]\ s.t. \mathcal{K}(T, X^i[:j]) \geq \frac{1}{\alpha}]. \tag{2}$$

一组含 i 个随机变量 $X = \{X_1, ..., X_i\}$ 的$\mathcal{K}(T, X)$ 定义为

$$\mathcal{K}(T, X) = \Pi_{j=1}^{i}(1 + \min(\lambda_j, \frac{3}{4T}) \times (X_j - T)), \tag{3}$$

$$\lambda_i = \sqrt{\frac{2\log(2/\delta)}{i\log(i+1)\hat{\sigma}_{i-1}^2}},\ \hat{\sigma}_i^2 = \frac{1/4 + \sum_{j=1}^{i}(X_j - \hat{\mu}_j)^2}{i+1},\ \hat{\mu}_i = \frac{1/2 + \sum_{j=1}^{i} X_j}{i+1}.$$

定理 3.2 的证明。引理 A.1 表明，如果第 i次迭代时的阈值集合未达到目标，那么估计函数返回 True 的概率至多为 α。我们利用这一结果证明，算法 3 返回未达到目标的阈值的总概率也至多为 α。为此，令$\mathbf{t}_{i^*}$ 为未达到目标且 i^* 最低的阈值集合。注意，算法 3 会返回估计结果表明未达到目标之前的最后一个阈值。因此，如果 $\text{E}(\mathbf{t}_{i^*}, D_V) = False$，那么算法 3 返回的阈值集合会达到目标准确率（或返回未找到）。因此，只有当 $\text{E}(\mathbf{t}_{i^*}, D_V) = True$ 时，算法 3 才会返回未达到目标的阈值。根据定理 A.1，后者的概率至多为 α，因此算法 3 所选阈值未达到目标的概率也至多为 α。□

A.3 阈值调整过程

算法5： ThresholdShift

输入： 每个任务和类别的候选阈值列表 $\mathbf{p}_T^c$；验证集 D_V；估计器 $\mathcal{E}$（构造用于认证准确率 α，失败概率为 δ）
输出： 调整后的阈值 $\mathbf{t}^*$，使得 $\Pr[\text{Acc}(\pi^*) < \alpha] \leq \delta$

最佳_t ←阈值，且偏移最大（最保守）；
对于 $s = s_{\max}$ 递减至 0 **执行**
  遍历 类别 c 对于每个任务 T **执行**
    初始化$\mathbf{t}$ $\leftarrow \varnothing$;
    如果 $s <$ 长度$\mathbf{p}_T^c$ **则**
      设置 $\tau^c = p_s$; // **段落**$_T^c = (p_0, p_1, \ldots, p_{s_{\max}})$
    end
    else
      设置 $\tau^c = \infty$; // 如果移位超过可用值，则禁用类别可用值
    end
    将 τ^c 添加到 $\mathbf{t}$ ，适用于 (T, c);
  end
  如果 $\mathcal{E}(\mathbf{t}\ D_V)$ 为 `True`，**则**
    最佳_次 ← **次**;
  end
  else
    中断
  end
结束
返回最佳_次;

级联构建的最后一步（见第3.2.3节）会调整类别阈值，以确保级联在未见数据上的目标准确率 α 至少以 $1 - \delta$ 的概率得以实现。对于每个任务中的每个类别 c，我们构建一个由候选阈值组成的偏移列表 $p_1 < p_2 < \cdots < p_k$，这些阈值表示高于原始阈值 τ^c、逐渐趋于保守的截断值。对于从 k 递减至 0 的每个偏移值 s，我们将类别 c 的阈值设为 p_s（适用于 $s > 0$），并将其设为 τ^c（适用于

$s = 0$；如果 s 不存在这样的阈值，我们就设置 $\tau^c = \infty$，从而禁用该任务中该类别的预测。对于每个候选阈值集合$\mathbf{t}_s$，我们在验证集 D_V 上评估级联，并应用估计器 E 来认证是否满足准确率保证。我们返回满足保证的最不保守（即最小 s）阈值集合。如果没有阈值集合通过验证，我们就退回到仅预言机级联（在我们的实验中从未触发这种情况）。

完整算法见算法5。

算法 6： 用于替代级联细化的智能体循环-ment

输入： 原始操作o_{orig}、开发集 D_{dev}、模型 $\mathcal{M}$、文档比例 $\mathcal{F}$、准确率目标 α、失败概率 δ、轮次 n_a、每轮代理数 n_s
输出： 最终级联 π^*
// 初始化候选任务
T $\leftarrow \{(m, o_{\text{orig}}, f) : m \in \text{M}\ f \in \text{F}\}$
对于 $r = 1$ **到** n_a，**执行**
  // 使用 CascadeAssembly 组装最佳级联（算法 4）
  $\pi_{\text{current}} \leftarrow$ CascadeAssembly(T D_{dev}，α，δ)
  // 收集反馈和失败案例
  对每个任务 T，在 $\mathcal{T}$ **中执行**
    确定是否在 π_{current} 中选择了 T；记录其覆盖率（分类的文档数量）以及最多 10 个难分类错误示例
    接近阈值（置信度略高于 τ）。
  end
  识别未被任何非预言机任务分类的文档（即被路由至预言机）；使用预言机提取最精简的支持片段，以确保提示简洁。
  // 从智能体处获取新的代理模型
  向智能体提供：(i) 用户操作，(ii) 预言机提取的失败案例，(iii) 每项任务的统计数据，(iv) 明确的代理生成指令。
  接收 n_s 新的代理操作。
  // 扩展候选集
  遍历 新的代理 o_{sur} **执行**
    将 (m, o_{sur}, f) 添加到 T 对于所有 $m \in \text{M}\ f \in \text{F}$
  end
结束
返回 *CascadeAssembly*($\mathcal{T}$ D_{dev}, α, δ)

B 代理操作生成：智能体循环

算法 6 描述了代理操作生成的流程。在每次迭代中，我们都会向大语言模型智能体提供一个按图6所示结构组织的提示。

图6中的提示用于智能体循环的每次迭代，并在每次迭代时根据新的失败案例和任务统计数据进行更新。为了帮助智能体生成有用的代理操作，我们提供了一份简短的具体检测策略清单，例如实体检测、事件检测、关系抽取和上下文线索，这些内容很容易转化为简单的分类规则。我们还提供了一个上下文示例，展示如何将这些策略转化为具体指令。此示例仅用于演示，并未用于我们的实际实验。完整的提示词模板可在我们的代码库中找到。

尽管这些检测策略有助于引导智能体提出有成效的方案，但它们并非严格必需——即使没有这些指导，优秀的智能体也可能发现有效的代理模型。关于代理生成的提示工程的系统性研究，包括智能体微调的潜力，将留待未来工作开展。

C 代理操作示例

为了了解任务级联如何节省成本，我们检查了智能体生成的代理指令，并确定了四种主要类型。**基于关键词或短语的代理**依赖预测性术语或领域术语的出现；例如，在维基_talk 中，一个代理会检查“WP:3RR”或“三次回退规则”，而在 games 中，则检查“better than”或“prefer over”等比较短语。**特定类别代理**针对与单个类别相关的特征：对于 agnews，一个代理通过检测企业并购或财务指标的相关提及来识别商业文章；对于 PubMed，另一个代理会在文档含有 “ 体外 ” 或 “ 体内 ” 等短语时，将其标记为“研究”。**基于语义模式的代理**捕捉与目标相关的更高层次线索；例如，在安然数据集中，一个代理会查找包含职称和联系信息的正式签名，以识别高管电子邮件。第四，**序列分解代理**将原始操作拆分为更简单的步骤；在 games 中，一个代理只检查是否提到了其他游戏——这是评估比较情感的必要前提。在某些情况下，智能体没有找到有帮助的代理；例如，在 FEVER数据集中，最佳级联只是将原始用户指令重新应用于文档的一个子集。智能体有时还会在多个文档比例上重复使用同一个代理，例如在法律数据集中，它会在 0.1 和 1.0 的文档比例下检查“同意不提起诉讼或质疑知识产权”的内容。

D 参数敏感性分析

任务级联依赖三个主要超参数：(1) 每次迭代提出的代理操作数量 (n_s)，(2) 智能体优化迭代的次数 (n_a)，以及 (3) 要考虑的文档比例集合 (F)。为评估这些选择对鲁棒性的影响，我们针对两个具有代表性的工作负载进行敏感性分析：安然（任务级联在此工作负载上带来了显著收益）和 games（收益较小）。对于每个工作负载，我们每次改变一个参数，同时将其他参数保持为默认值：$n_s \in$，{5，10}，并固定 $n_a = 3$；$n_a \in$ {1，2}，并固定 $n_s = 5$；以及 F $\in$ {{0。25，1。0}，{0。25，0。5，1。0}}，并固定 $n_s = 5$，$n_a = 1$。所有实验均采用 90%准确率目标，并报告相对于**双模型级联**基线的成本。

图 7 展示了结果。**任务级联**在两种工作负载的所有参数配置下都始终优于基线。在安然数据集上，所有设置在达到准确率目标的同时，相较于**双模型级联**基线都实现了显著的成本节省。在 games 数据集上，**任务级联**相较于**双模型级联**仅表现出 modest 的改进，但参数变化仍然保持稳定，展现出即使在具有挑战性的场景中也具备良好的鲁棒性。

敏感性分析揭示了若干实用见解。首先，n_s（要探索的代理操作数量）对成本的影响比 n_a（迭代优化的迭代次数）更显著。这表明，与其过度依赖迭代优化，不如在每次迭代中生成多个不同的代理候选项。其次，所有测试过的文档比例集合都取得了相似的性能，表明该方法对这一选择具有稳健性。不过，务必在 F 中纳入完整文档比例（1.0），因为某些任务或代理模型可能会受益于完整上下文。我们建议涵盖从某一比例开始的范围——

你的任务是为以下分类任务提出 [n_s] 个简单的代理操作。每个代理操作都必须针对以下列表中的不同检测类型：- 实体检测（检查是否存在特定实体）- 事件检

[插入最小相关片段]

任务统计数据：对于每个候选任务：- 已选择： [是/否]- 覆盖率： [分类的文档数量]- 困难样例：最多提供 10 个在此任务中被错误分类的文档，仅展示每个文档中

对于每个代理模型，请提供：

提示： <简洁的分类指令，允许的输出应与原始任务一致>理由： <它检测什么、使用哪种检测类型，以及它为何比之前的代理模型更简单或能与之互补>

示例（针对二元任务；判断医学文章是否描述药物不良反应）：

任务：这篇文章是否
cle 是否描述了对某种药物的不良反应？如果存在药物不良反应，则输出 True；如果不彩争霸 ot.

示例代理操作：

提示：文章是否描述了皮疹、恶心或毒性等患者结局？如果是，则输出 True . 否则，输出 False .
理由：事件检测：识别负面结果，无需明确归因于某种药物。

提示：这篇文章是病例报告吗？如果是，输出 True；否则，输出 False。理由：上下文检测：病例报告通常会重点介绍个别患

请在此处填写答案：

图6：用于从智能体中提取代理模型的提示词模板。

从较小比例（例如 0.1 或 0.25）到完整文档，以便在保持灵活性的同时有效进行文档剪枝。总体而言，这些发现表明，适度的参数值（$n_s \geq 3$、 $n_a \geq 1$，以及从较小范围到完整文档的 F）足以实现良好性能；不过，若有更多优化预算，用户可以增加 n_s。

E 优化成本与延迟分析

本节详细测量了**任务级联**和**双模型级联**的离线优化成本与延迟，并分析了何时可通过推理成本节省收回前期优化投入，以及规模化后的总成本。

E.1 优化成本与盈亏平衡分析

表4报告了各项工作负载的优化成本、盈亏平衡点，以及处理100万份文档的预估端到端成本。我们将盈亏平衡点定义为：使某种方法的总成本（一次性优化成本加上逐文档推理成本）低于对每份文档都直接运行预言机模型所需的推理文档数量。

优化成本。如第 6 节所述，**任务级联**会产生更高的优化成本，因为它还需要执行两个额外的步骤：*(i)* 训练一个轻量级相关性分类器，以重构文档；以及*(ii)* 在级联组装期间，在开发集上评估管道的多个候选重写方案。相比之下，**双模型级联**只评估代理模型和预言机

表4：离线优化成本、盈亏平衡分析，以及处理100万份文档时的估算成本。左侧列（优化成本）列出构建各方法（TC、TC（Lite）、2MC）所需的美元金额，条件是 $D_{\text{dev}} = 200$（法律领域除外，为150）。中间列（盈亏平衡）显示，在总成本低于对每份文档都运行仅使用预言机之前，必须处理多少份推理文档。最右侧列估算处理100万份文档的端到端总成本：离线优化 + 逐文档推理。括号中列出每种TC变体相对于2MC基线的比例。

	优化成本（美元）			盈亏平衡（文档数）			估算总成本（处理100万份文档，美元）		
工作负载	TC	TC（Lite）	2MC	TC	TC（Lite）	2MC	TC	TC（Lite）	2MC
agnews	$3.32	0.58美元	0.12美元	6,173.87 (27.0×)	1,080.54 (4.7×)	228.57	28.07美元 (0.75×)	29.83美元 (0.79×)	$37.62
court	$29.51	$7.59	$2.70	2,615.32 (10.0×)	680.85（2.6×）	262.14	$2,558.51（0.73×）	$2,677.09（0.76×）	$3,515.20
安然	$13.78	$3.38	$1.15	1,757.40 (10.9×)	431.84 (2.7×)	160.56	$97.65 (0.13×)	$110.13 (0.14×)	$763.65
FEVER数据集	$39.52	$10.27	$3.69	2,513.60 (10.4×)	656.72 (2.7×)	241.77	$579.52 (0.58×)	$640.27 (0.64×)	$1,003.69
games	$10.92	$2.61	$0.87	3,831.92 (12.9×)	808.24 (2.7×)	296.17	$1,073.67 (1.10×)	$685.11 (0.70×)	$975.87
法律	$45.22	$11.89	$4.30	1,745.22（7.9×）	482.16（2.2×）	222.09	$1,748.54（0.28×）	$2,598.41（0.41×）	$6,312.89
PubMed	$25.22	$6.44	$2.28	2,557.29（10.9×）	653.78（2.8×）	233.85	$613.22（0.87×）	$608.44（0.87×）	$702.28
维基讨论_	$9.49	$2.23	$0.72	2,687.62 (12.9×)	632.54 (3.0×)	208.70	$153.49 (0.68×)	$155.23 (0.69×)	$225.72

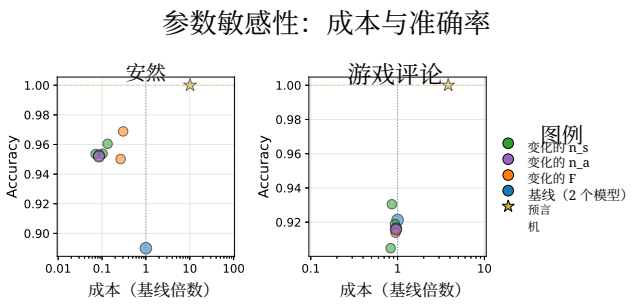

图 7：对代表不同性能区间的两个工作负载进行参数敏感性分析。每个点表示不同参数设置下的成本（相对于双模型级联基线）和准确率。任务级联 在所有参数配置下都表现出较低的方差，即使是在基线性能较强的 games 上也是如此。

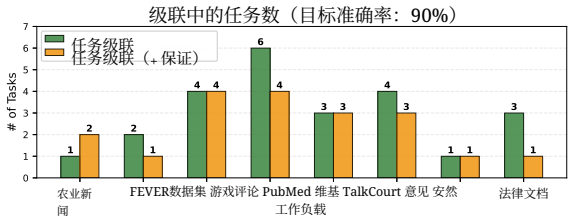

图 8：在 $\alpha = 90\%$时，任务级联和任务级联 (+ 保证) 方法中每个级联的任务数量。

模型以设定置信度阈值。这项成本的大小取决于文档长度和任务复杂度：较长的文档（FEVER数据集、法律）会增加预言机标注成本，而难度较高的任务则需要更多代理生成轮次。

我们还纳入了**任务级联（轻量版）**，这是一种轻量级变体，不考虑利用预言机模型的候选任务，如第 7.1 节所述。此配置可将优化成本降低约 70–80%（相较于**双模型级联** 基线，均值为 2.9×），同时保持相似的推理行为。

表 5：优化和推理（inf.）的延迟测量值（均值 ± 标准差，单位：秒）。由于可用额度有限，2MC、TC（Lite）和仅使用预言机 使用了 Azure OpenAI API，而TC 使用了标准 OpenAI 端点（Azure 慢了~2.5× [60]）。由于 API 尾延迟，所有方法的标准差都很高。最右侧一列估算了处理 100 万份文档的 API 请求所需的总推理延迟，单位为小时。由于未计入 API 尾延迟，这些估算值代表下限。

方法	优化（秒）	~1K 文档的推理时间（秒）	100 万文档的预计推理时间（小时）
2MC	64.93 ± 57.60	280.64 ± 291.36	3.0 ± 0.1
TC	911.45 ± 381.23	201.19 ± 149.03	2.2 ± 0.0
TC（Lite）	1253.0 ± 947.3	175.4 ± 90.6	1.9 ± 0.0
仅使用预言机	–	26.14 ± 56.80	0.3 ± 0.0

盈亏平衡分析。 在各种工作负载下，**任务级联**的盈亏平衡点介于1,745份文档（法律）和6,174份文档（agnews）之间，均值为2,986份。由于优化成本较低，**任务级联（轻量版）**的盈亏平衡阈值也相应较低（480–1,080份文档，均值为678份）。**任务级联**与**双模型级联**的盈亏平衡点之比介于7.9×（法律）和27.0×（agnews）之间，整体均值为11.4×——这反映了推理成本节省幅度的差异。

规模化成本。 表4最右侧几列估算了处理100万份文档的端到端总成本（优化 + 推理）。在此规模下，优化成本变得微不足道。**任务级联（轻量版）**在全部8种工作负载上的成本均低于**双模型级联**，每100万份文档的中位数节省为327美元，在法律工作负载上最多可节省3,714美元。**任务级联**在8种工作负载中的7种上实现成本节省（games除外），中位数节省为257美元，在法律工作负载上最多可节省4,564美元。在games工作负载上，**双模型级联**未达到90%准确率目标（表3），而两个**任务级联**变体均达到了目标。

实践意义。 我们评估中的大多数工作负载都包含数万至数百万份文档（表2），因此这些优化成本在大规模场景下可以忽略不计。例如，在完整的安然语料库（~500k 份文档）上运行，只需处理 0.35% 的文档即可达到盈亏平衡；而较小的法院数据集（~36k 份文档）在处理其

语料库的 7.3% 后即可达到盈亏平衡。这些结果表明，在真实的生产环境中，**任务级联**——尤其是其`Lite`变体——能够迅速收回投入。

E.2 延迟测量

表 5 报告了所有方法的优化和推理延迟，以及在 32 路并发下处理 100 万份文档的估计总推理延迟。测量在一台 2024 款 MacBook Air（M4 芯片）上进行，使用 32 个并行 API 线程，并通过指数退避休眠来应对速率限制。请注意，API 延迟会因服务提供商和一天中的时段而显著变化。**任务级联** 优化所需时间比**双模型级联**长 ~ 14×，主要是因为代理生成（o1-mini）以及评估大量候选配置。不过，不同服务提供商之间的运行时间无法直接比较：如表 5 所示，**双模型级联**、**仅使用预言机**和**任务级联（轻量版）**是在 Azure OpenAI 上测得的，而**任务级联** 使用的是标准 OpenAI 端点；Azure 可能会慢 ~2.5× [60]。这种服务提供商的影响或许可以解释为何**任务级联（轻量版）**的优化运行时间均值更高（1253 秒），尽管其令牌成本更低。

在百万规模下进行推理时，我们对每份文档的测量结果表明，**任务级联** 和 **任务级联（轻量版）**的延迟（小时）与**双模型级联**处于同一数量级。不过，对这些估算应谨慎解读：它们未考虑 API 尾延迟或速率限制，而且测量是在已知延迟存在差异的不同服务提供商（Azure 与标准版 OpenAI）上进行的。在批处理场景中——这是大规模文档处理的主要用例——如果能显著节省成本，较高的优化延迟是可以接受的；批处理 API 得到广泛采用便证明了这一点，其完成时间可长达 24 至 72 小时 [20, 48]。

F 工作负载提示模板

本节列出我们在实验（第 7 节）中为每种分类工作负载使用的确切提示词模板。工作负载的选择方式如第 7 节所述。

为确保任务规范清晰一致，我们使用 Claude 3.5 Sonnet 生成每条指令，并明确提示它“将其设为一个没有歧义、适用于大语言模型的分类任务”。每个模板都包含一个 {document_text} 占位符，推理时会将其替换为输入文档。

所有工作负载的完整提示词模板如下所示。

农业新闻

```
我将给你一篇新闻报道。报道内容如下：{document_text}

请将文章归入以下类别之一，并且只返回对应的编号：- 0 = 世界：文章主
- 3 =科学/技术：文章涉及科学、技术、研究、发现、创新，或科学与技

你必须只回复最符合该报道主题的数字（0、1、2 或 3）。
```

列表 1：agnews 分类提示

安然

```
我会给你一封电子邮件。邮件内容如下：{document_text}

你的任务是判断这封电子邮件是否由安然的一名高级管理人员或其他高层人士

- 如果邮件由高级管理人员（首席执行官、总裁、副总裁、董事等）或其他高

重要提示：请查找能够表明资历的职称、职位和签名。请同时考虑正式职称，

你必须仅回复 True 或 False。
```

列表 2：安然发件人分类提示

法院意见

```
我将给你一份最高法院的司法意见。司法意见如下：{ document_text} 你

你必须只回答 True 或 False .
```

列表 3：法院意见推翻提示

PubMed

```
我将向你提供一篇来自 PubMed 的完整生物医学研究文章。文章如下：
```

```
- 2 = 荟萃分析或系统综述：使用系统化方法整合并分析多项既往研究的结果。-
```

列表 4：PubMed 研究类型提示

游戏评论

```
我会给你一篇游戏评论。评论如下：{ document_text} 你的任务是仔细阅读以
```

列表 5：游戏评论比较提示

法律文件

```
我将提供一份法律文件。以下是该文档：{
        文档_文本}

你的任务是判断该文档是否包含任何形式的
        不起诉契约或不提出异议协议，涉及知识产权
        权利。这包括直接承诺和
        间接限制。

- 如果包含以下任何内容，则为真：
  - 同意不对知识产权的有效性或所有权提出异议或质疑
  - 承诺不质疑、攻击或否定知识产权
  - 同意不采取与知识产权所有权相矛盾的行动
  - 承诺不提起与知识产权相关的主张或诉讼
  - 更一般地说，任何可能被解释为
          对未来知识产权质疑的限制
- 如果不包含上述任何内容，则为 False

你必须仅回答 True 或 False。
```

清单 6：法律知识产权条款提示

维基讨论

```
我将向你提供一段维基百科讨论页讨论。以下是讨论内容：{document_
```

```
你的任务是仔细阅读以下讨论，并判断相关编辑的结果。请考虑讨论是否导致
```

列表7：维基_讨论编辑撤销提示

FEVER数据集

```
我将提供一条主张和一系列可能明确支持、也可能不明确支
```

收稿于2025年7月；修订于2025年10月；接受于2025年11月