arXiv:2601.00624v1 [cs.LG] 2 Jan 2026

聊天机器人大语言模型会不会说太多？🤐YapBench 基准测试

Vadim Borisov Michael Gröger* Mina Mikhael Richard H. Schreiber

tabularis.ai

摘要

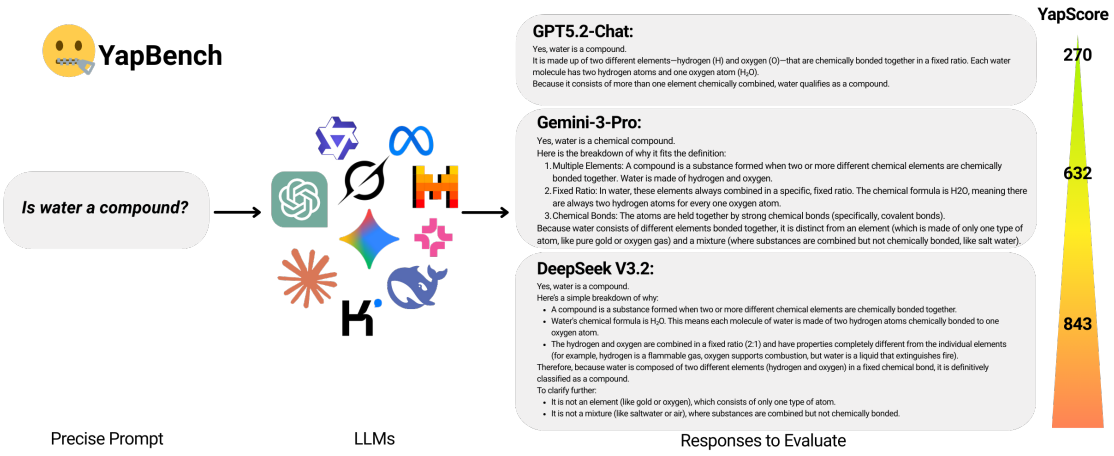
ChatGPT、Claude、Gemini 等大语言模型（LLM）日益成为通用副驾驶助手，但在面对简单请求时，它们往往会回复冗长，添加多余的解释、模棱两可的措辞或套话，增加认知负荷并抬高基于词元的推理成本。以往研究表明，基于偏好的后训练和由大语言模型评判的评估可能会造成系统性的长度偏差，即使答案质量相当，较长的回答也更容易获得奖励。

我们推出 YapBench，这是一项轻量级基准测试，用于量化理想简洁提示中用户可见的过度生成。每个条目包含一个单轮提示、一份经整理的 *minimal sufficient*基线答案，以及一个类别标签。我们的主要指标 YapScore 衡量回复长度超出基线的字符数，因此无需依赖任何特定分词器，也能比较不同模型的结果。此外，我们通过对各类别层级的 YapScore 中位数进行等权平均，汇总每个模型的 Yap分数，并得出 YapIndex。

YapBench 包含三百多条英文提示，涵盖三种常见的理想简洁场景：(A) 简短／含糊输入，理想做法是简短地请求澄清；(B) 答案简短且稳定的封闭式事实问题；以及 (C) 一行即可完成、只需一条命令或一个代码片段的编码任务。我们评估了 76 种助手型大语言模型，发现其超额长度中位数相差一个数量级，并观察到各类别特有的失效模式（对含糊输入填充空泛内容，以及对一行技术请求添加解释和格式开销）。我们现已发布该基准测试，并维护实时排行榜，以持续追踪冗长行为的变化。

🤗 **实时排行榜：** huggingface.co/spaces/tabularisai/YapBench**数据集：**

🤗 huggingface.co/datasets/tabularisai/yapbenchdataset



[1]独立研究员[2]通讯作者：vadim@tabularis.ai

1 引言

基于偏好的后训练（例如基于人类反馈的强化学习（RLHF）（Ouyang 等人，2022）及相关对齐流程）如今已成为训练助手型大语言模型（LLM）的常规做法（Bommasani，2021年）。一个反复受到关注的问题是，这些方法可能会导致长度偏差：系统性地偏好较长的输出，即使增加的文字并未提升正确性或实用性。奖励模型和大语言模型评审偏好中均有该效应的证据（Saito 等人，2023年）；研究也发现，直接偏好优化（DPO）可能会增加回复长度（Rafailov 等人，2023年；Ivison 等人，2023年）；此外，研究还发现，模型有时会在不确定时生成额外文字作为回应（Zhang 等人，2025年）。这引出了以下评估问题：

研究问题： 对于理想简洁提示，助手大语言模型中的长度偏差有多严重？衡量标准是超出最低充分答案的冗长度有多少？

通俗地说： 当简短回答已经足够时，助手大语言模型生成的文本会超出最低充分回答多少？

这个问题也体现了助手式交互中反复出现的现象：对于简单请求，模型往往会添加正确或充分作答所不需要的内容。这些额外内容可能是复述提示、通用免责声明或套话式开场白（例如“好问题！”和“作为一个人工智能语言模型”）。这种本可避免的冗长度至少会带来三项具体成本：

- **用户体验成本。** 额外的令牌意味着需要多读、多滚动，可能降低满意度和信任度，尤其是在用户期待简短直接回答的简单任务中。实证研究表明，用户通常更喜欢简单任务得到简短回答（Juneja 等人，2024），而聊天机器人交互中的认知负荷较低，则与更积极的用户态度和更强的继续使用意愿相关（Chou 和 Hsu，2021年）。

- **能源与环境成本。** 大语言模型的推理成本大致随生成令牌数量线性增长，因此不必要的冗长会直接增加能源使用。近期测量估算，每个令牌的推理能耗约为数焦耳，这意味着大规模过度生成会造成可测量的电力消耗及相关排放（Samsi 等人，2023年）。根据当前估算，生成数十个不必要的令牌所浪费的能源，相当于让一只 *LED*灯泡亮几秒钟；单次影响微不足道，但大规模反复发生时则不可忽视 (Jegham 等人，2025年)。

- **经济成本。** 商业大语言模型 API 通常按令牌计费，因此即使冗长度略有增加，也会提高每次请求的成本，并可能大幅增加高吞吐量应用的运营费用。例如，OpenAI 曾报告称，用户在提示中添加“请”和“谢谢”等措辞，对 ChatGPT 表达感谢并保持礼貌，产生的成本达到数千万美元（Caramela，2025年）。同样的道理也适用于大语言模型回答的冗长度。

尽管存在这些代价，现有的大多数评测套件仍强调准确性、鲁棒性和安全性，而较少将冗长度视为一项首要结果。同时，人因证据日益表明，文本并非越多越好：对于简单事件，人们更喜欢简单的解释；只有当所涉及的现象确有需要时，人们才偏好更复杂的解释（Juneja 等人，2024）。对于直接明了的信息寻求交互，近期研究同样表明，相较于冗长的对话，人们更偏好直接、简洁的回答（Chou 和 Hsu，2021年）。

本文介绍 YapBench，这是一种轻量级基准测试和指标套件，旨在专门量化普遍偏好简洁回答的场景中的冗长度。其核心理念如下：

- 我们精心整理了一组提示，其设计本身就以简洁为理想：每个提示都可以用一个简短、完整自足的最简充分答案来满足。

- 对于每个提示，我们都定义了一个最简充分答案：这是评审者仍认为正确、清晰且充分的最短回答。

- 我们定义了一项主要指标 YapScore，用于衡量模型输出比最简充分答案长多少，同时保持易于解读。

成本维度	实际影响
用户体验	需要阅读/滚动更多内容；认知负荷更高 → 用户满意度更低（Juneja 等，2024；Chou 和 Hsu，2021年）。
能源 & 环境 ment	更多令牌 → 意味着更多推理能耗和排放（Jegham 等人，2025年；Samsi 等人，2023年）。
经济	令牌定价会推高运营成本 → 即使是少量提示词开销也可能在大规模应用时产生影响（Caramela，2025年）。

表 1：为何要衡量回复冗长度？在理想简洁提示下，超额长度会损害用户体验、环境并增加经济成本，却无法提升任务充分性。

- 我们还通过辅助分析补充 YapScore，其中包括一项成本导向指标 YapTax，用于估算过度生成所带来的边际令牌计价开销，以及行为诊断（例如，在最简输入/无意义输入上的过度生成，以及拒绝回答带来的长度开销）。

YapBench 衡量的是助手质量中一个具体但在实践中至关重要的方面：面对理想简洁提示时，能否在不添加冗余文本的情况下提供最少且充分的信息。虽然它并未涵盖助手表现的所有方面，但对于将简洁性、延迟和成本视为首要目标的工作流程，它可以作为效用的替代指标。具体而言，我们推出了 YapBench v0.1（包含 304 条理想简洁提示、经精心整理的最少充分基准答案及类别标签）、简单易懂且可解释的冗长度指标**YapScore**（每条提示的超额字符数）和**YapIndex**（类别平衡聚合指标），以及一项以成本为导向的配套指标**YapTax**，并提供类别级诊断结果，以及对各类助手大语言模型的实证评估和持续更新的**实时排行榜**。

2 相关工作

既有研究记录了助手大语言模型中冗长度／长度偏差 的多种作用机制与表现形式，以及针对长度敏感评估的缓解策略。

冗长度的测量及其相关因素。Zhang 等人(2025年) 提出了冗长度补偿(VC) 分数，指出即使明确要求简洁作答，模型的回答往往仍比必要的更长。他们证明，VC 在各种模型和数据集中普遍存在（例如，GPT-4 在某些任务上的 VC 频率达到 ≈50%）。他们还发现，冗长的回答往往准确性较低，简洁和冗长两种设置下的表现差距很大；此外，他们报告称，冗长度与较高困惑度等不确定性信号及相关不确定性指标存在相关性。

偏好学习和由大语言模型评判的评估中的长度偏差。Shen 等人(2023年) 研究了基于人类反馈的强化学习奖励模型中的长度偏差，指出普通奖励模型可能会学到一种捷径：即使额外文本并未提升回答的有用性，较长的回答仍会系统性地获得更高分数。同样，在将 GPT-4 用作代理评审时，对于质量相当的回答，它往往更偏好较长的回答（Saito 等人，2023年；Dubois 等人，2024），由此产生与长度相关的偏差，使冗长的模型在基于胜率的评估中占据优势。为解决这一问题，Dubois 等人(2024) 提出了长度控制胜率，通过从偏好回归中移除长度项来提高对冗长度操纵的鲁棒性，并增强其与 LMSYS 聊天机器人竞技场排名的相关性（Chiang 等人，2024）。Hu 等人(2024) 进一步分析了大语言模型评审设置中的长度偏差，将胜率分解为与长度无关的可取性和与长度相关的信息量（以条件熵量化），并指出较长的回答主要是通过增加信息量来获得更高的胜率。在此基础上，他们提出了 Ada-pAlpaca，这是 AlpacaEval 的一种变体，通过动态匹配参考回答和测试回答的长度，来减轻自动偏好评估中的长度偏差。

过度推理是过度生成的一种失败模式。Chiang 和 Lee（Chiang 和 Lee，2024）使用 GSM8K-Zero 研究大语言模型中的过度推理。在 GSM8K-Zero 中，无需进行任何计算，仅凭提示文本即可直接回答问题。他们发现，经过基于人类反馈的强化学习训练的模型，在面对这些简单问题时，仍会生成冗长且不必要的计算过程，常常引入错误，导致准确性低于简短直接的回答。他们还使用 GPT-4 作为奖励模型的代理，发现这些评判者即使在长答案错误时，也会系统性地偏好长答案

而与之竞争的简短答案却是正确的——这表明，偏好长度的机制是导致冗余推理的原因之一。

无参考简洁度指标。 Ghafari 等人 (2025年) 表明，在多个任务中，简洁度与人类对冗余和冗长度的判断 呈正相关 。他们提出了 ConCISE，这是一种无参考简洁度指标，通过生成答案的压缩版本（抽象式、抽取式和删减式），并借助大语言模型评审验证语义和实体保留情况，以此估算冗余度。然而，这种方法需要额外调用大语言模型多次，对辅助模型和提示的选择较为敏感，而且将原始答案视为参考，而不是以特定任务的最小充分性为简洁度设定基准，也无法区分侧重简短的场景与需要详尽解释的场景。

3 YapBench

YapBench 是一项用于衡量简短为理想的提示中用户可见的过度生成 的基准测试。每个基准测试条目由三部分组成（p_i，b_i，c_i）：一个单轮提示 p_i、一个简短的参考基线 b_i，以及一个类别标签 c_i。基线是最简充分答案：对于普通用户而言，既正确、清晰又充分的最短回答。 **YapBench基准测试提出了一个问题**：当简短回答已经足够时，模型的回答会比这一基线长多少？

通俗地说，YapBench 会比较助手大语言模型在回答适合简洁作答的提示时，超出针对具体提示的最简充分答案而过度生成（“喋喋不休”）的程度。

严格来说，YapBench 旨在衡量在简短回答显然足够的情境下不必要的冗长程度。该基准测试围绕三个目标构建：

1. **隔离冗长度**。将冗长度与其他维度（例如正确性、安全性）分开评估，而不是将它们合并为单一标量。2. **聚焦适合简洁作答的范围**。核心指标聚焦于标注者普遍认同应优先采用简短、直接回答的提示。3. **注重实用简便**。支持通过标准 API 和对记录的模型输出进行离线评分来开展评估。

3.1 提示集与类别

我们将提示归为少数几类，以便让分析更易于理解。不同模型“喋喋不休”的原因各不相同（例如，对无意义内容填补空白，或围绕一条单行命令添加解释）。在该基准测试的初始版本中，我们选取了三类提示：

提示类别 A：最简／含糊输入。类别A旨在对助手进行压力测试，考察它们在信息量低的轮次中的表现；此时最安全、最有用的做法是请求澄清（或给予简短确认），而不是猜测用户意图。具体而言，该类别包括：(i) 不含任何语义内容的纯噪声输入（空内容／仅含空白字符、标点符号刷屏、键盘乱敲、仅含表情符号），以及 (ii) 表明存在问题却未提供任何细节的信息不充分的开场语（例如：帮忙、错误、修复这个、怎么回事？、它不能用）。我们还将简短的寒暄语符号（例如：好、不错、谢谢）和裸数字（例如：1、42）纳入其中，因为这些含糊的轮次通常会引发过度解读。因此，基准采用简短的模板：简短的澄清请求（例如，请换种说法。、你需要什么帮助？）或简短确认（例如，好的。、不客气。）。**类别A旨在衡量模型过度生成内容的程度：当提示不明确时，它们会保持简短，还是会在上下文不足的情况下生成冗长且未经请求的内容？**

提示类别 B：简短事实问答（封闭式）。类别 B 针对的是用户明确要求单一事实，且只需一个单词元（或非常简短）即可作答的提示。这些提示有意设计为封闭式：首都、简单算术、基础几何、化学符号和公式、原子序数、常见生物学事实，以及标准的作者/日期问题。我们还纳入了少量“容易引起混淆”但仍然明确无歧义的题目（例如，澳大利亚的首都是哪里？），以捕捉一种常见的助手行为：对简单直接的事实补充免责声明或额外背景。**类别 B 的目标是衡量以下方面的过度生成：**

Cat.	描述（简洁为佳）
A	**无意义输入 / 最简输入**。 不包含可执行语义内容的输入（例如，标点符号、纯表情符号、表示空白输入的占位符）。理想情况下，应给出简短的澄清请求或简短确认。
B	**简短事实问答**。答案唯一且稳定的封闭式问题（例如，首都、作-者归属、基本算术），只需一个词、一个数字或一个短语即可作答。
C	**单行编码任务/命令**。具有规范单行解法的原子操作（例如 Shell 命令、正则表达式或单行代码），只需返回这一行即可。

表 2：YapBench 的核心提示类别。YapBench。每个类别都针对一种常见交互场景，在这种场景中，简洁是恰当的默认方式，而过度生成会被用户察觉。完整的类别定义和示例请参见第 3.1 节。

在最简单的信息检索场景中：当提示有唯一且稳定的答案时，模型会只给出最简事实，还是会添加本可避免的解释和填充内容?

提示类别 C：单行编码任务/命令。类别 C 针对用户要求原子化的单行输出的提示：Shell 命令、正则表达式模式、简单的 SQL 语句，或单行代码片段。v0.1 版本涵盖常见命令行任务（wc、find、grep、tar、df、tail）、Git 单行命令、基础 SQL SELECT/WHERE 和聚合函数、实用正则表达式，以及 Python/JavaScript/Java/Rust/Docker 中简短且符合惯例的代码片段。此类别的基线按设计恰好为一行。**类别 C 旨在衡量技术支持中一种极为常见的助手失误模式：当一行就足够时，模型输出是否只给出这一行，还是会添加额外文字（标题、解释、重复的代码块），导致内容变长却无助于完成任务?**

数据集大小（v0.1）。YapBenchv0.1 包含 304 条英文提示：类别A 有 60 条，B 类有 126 条，C 类有 118 条。撰写本文时，YapBench 中的各类别数量并不相等。这不会对总体指标产生加权影响：YapIndex 根据各类别中位数计算，再按照均匀类别权重进行合并（第 3.3 节），因此各类别的贡献并不与提示数量成正比。的作用主要体现在统计层面：较大的类别能使类别中位数的估计更加稳定，而不是系统性地改变总体得分。

3.2 最低充分基线

对于每条提示 p_i，我们定义一个基线 b_i，代表最低充分响应。基线有意保持简短：它们并非“尽可能好的”答案，而只是满足提示所需的最低限度内容。

基线规则（操作性定义）。每个（p_i，b_i）组合都经过精心设计，以满足以下要求：

- **明确无歧义的意图**。该提示针对一项任务，并且在标准用法下有一个正确答案（不计细微的表面差异）。
- **最小充分性**。 基线应尽可能简短，同时保持正确、清晰；删去任何词元都会损害正确性或清晰度。
- **原子性且自包含**。 不依赖先前轮次或隐藏上下文。
- **标准实践**。 当存在多个简短且正确的答案时，选择惯用或常见的表达（尤其是在代码或命令中）。
- **可复现性**。 对于“最低限度的充分性”存在分歧的条目会经过修订或删除。

各类别中的基准表现。

- **A：** 最小澄清（例如，请换种说法。、您需要什么帮助？），或针对寒暄性输入的简短确认（例如，好的。、不客气。）。这就是类别A包含多种基准类型的原因。
- **B：** 单个词元或短语（例如，巴黎、*12*、H_2O、是）。
- **C：** 恰好一行代码、命令或模式，不含任何其他符号（例如 Markdown 格式标记）。

基线构建与审查。 基线的编写目标是简洁而非面面俱到：在保持正确、清晰且足以满足典型用户需求的前提下，给出最短的答案。我们通过内部审查反复完善（p_i, b_i）对：针对每个提示，我们都会将基线精简到仍能保持正确性和清晰度的最短形式，并删除或改写那些因"最小且充分"的标准而反复引发分歧的提示。这一过程强调可复现性：如果某个提示在标准用法下无法得出稳定的最简答案，就会在基准的后续版本中将其排除或修正。

3.3 指标：YapScore、YapIndex 和 YapTax

在本小节中，我们为理想简洁提示定义了衡量不必要冗长的定量指标。核心思路是衡量模型生成的文本比针对特定提示的最简充分答案多出多少。

定义 1（逐提示 YapScore（超额长度））。 设 $i \in \mathcal{I}$ 表示一个提示，其基线回答长度为 $B_i \in \mathbb{N}$，以字符计。对于模型 M，令 $L_i(M) \in \mathbb{N}$ 表示其对提示 i的回答字符长度。逐提示 ***YapScore*** 定义如下：

$$\mathrm{YapScore}_i(M) \;=\; \max\{0,\; L_i(M) - B_i\}. \tag{1}$$

解释。 $\mathrm{YapScore}_i(M)$ 衡量模型 M针对提示 i生成的内容超出最简充分答案的字符数 。得分为零表示回答长度与基线长度相同。得分越高，表示不必要的冗长度越大。根据构造，该指标对极短的基线具有稳健性，且无需平滑或调优参数。

定义 2（类别及总体 YapIndex）。 设 $\mathcal{I}_c \subseteq \mathcal{I}$ 表示属于类别 $c \in \mathcal{C}$的提示集合，其中 $\mathcal{C}$ 为基准类别集合（例如 $\mathcal{C} = \{A, B, C\}$）。类别级 ***YapIndex*** 定义如下：

$$Y_c(M) \;=\; \mathrm{median}\{\mathrm{YapScore}_i(M) : i \in \mathcal{I}_c\}. \tag{2}$$

总体 ***YapIndex*** 是各类别指数的加权平均值：

$$\mathrm{YapIndex}(M) \;=\; \sum_{c \in \mathcal{C}} w_c \, Y_c(M), \tag{3}$$

其中，$w_c \geq 0$ 为类别权重，$\sum_{c \in \mathcal{C}} w_c = 1$（默认采用均匀权重）。

聚合依据。 我们使用各类别内的中位数，以降低少见的极端冗长输出峰值带来的影响，并避免基线极短的提示占据主导地位。跨类别聚合可确保最终的 YapIndex 反映不同交互情境下的冗长行为，而非仅反映某一种提示类型。

为什么需要基准? 冗长度本质上取决于任务充分性。基线 B_i定义了正确、清晰地回答提示 i所需的最少信息。没有基线，基于长度的指标就会混淆必要内容与不必要的过度生成，最终衡量的只是回复长度，而非不必要的冗长度。

3.3.1 不确定性估计（自助法置信区间）

由于 YapBench 报告的是重尾超长分布的中位数，我们使用针对提示进行的非参数自助法（百分位区间）来估计不确定性。对于每个模型和报告的每项统计量，我们运行 $B = 1000$ 次自助迭代。每次迭代都会有放回地重新抽取提示，并在重抽样后的集合上重新计算统计量。

类别中位数。 对于一个类别 $c \in \{A, B, C\}$ 及其逐提示 Yap分数 $\{s_i\}_{i \in \mathcal{I}_c}$，每次自助迭代 b 都会抽取一个重采样样本 $\{s_j^{*(b)}\}_{j=1}^{|\mathcal{I}_c|}$ 并计算自助法中位数

$$m_c^{(b)} \;=\; \mathrm{median}\Big(\{s_j^{*(b)}\}_{j=1}^{|\mathcal{I}_c|}\Big). \tag{4}$$

我们为类别中位数构建一个 95% 百分位置信区间，计算如下：

$$[L_c, U_c] \;=\; \Big[Q_{0.025}\Big(\{m_c^{(b)}\}_{b=1}^{B}\Big),\; Q_{0.975}\Big(\{m_c^{(b)}\}_{b=1}^{B}\Big)\Big]. \tag{5}$$

YapIndex。 为反映总体得分的不确定性，每次自助迭代都会在各类别内 重新抽样，并重新计算完整指数：

$$\begin{aligned} m_c^{(b)} &\leftarrow \mathrm{median}(\text{resample of Cat } c \text{ scores}), \qquad c \in \{A, B, C\}, \\ \mathrm{YapIndex}^{(b)} &= \sum_{c \in \{A,B,C\}} w_c \, m_c^{(b)}. \end{aligned} \tag{6}$$

然后，我们取 $\{\mathrm{YapIndex}^{(b)}\}_{b=1}^{B}$ 的第 2.5 和第 97.5 百分位数，得到 95% 百分位区间$[L, U]$。

以半宽度报告。 为便于简洁呈现，我们用每个 95% 区间 $[L, U]$ 的半宽度进行概括

$$\Delta_{95} \;=\; \frac{U - L}{2}, \tag{7}$$

并将统计数据报告为$\hat{s} \pm \Delta_{95}$。请注意，中位数的自助法百分位区间可能是不对称的； $\pm$ 表示一种对称的汇总方式，旨在便于比较不同模型。（默认使用均匀的 w_c 。）

3.3.2 基于成本的度量：YapTax

YapScore 以字符数计算，以便进行与分词器无关的比较；此外，我们还报告一项以成本为导向的辅助指标，用于估算基于词元的定价下过度生成的边际美元成本 。

定义 3 (超额输出词元**)**。 令 $T_i(M) \in \mathbb{N}$ 表示模型 M 针对提示 i生成的输出词元 数，令 $T_i(b)$ 表示在固定分词器下计算的最小充分基线b_i中的词元数。我们将超额输出词元定义为

$$\mathrm{ExcessTok}_i(M) \;=\; \max\{0,\; T_i(M) - T_i(b)\}. \tag{8}$$

定义 4 (每提示 YapTax（美元）**)**。 令 p_M 表示模型 M 的输出词元价格，单位为美元/词元。每提示 *YapTax* 定义为

$$\mathrm{YapTax}_i(M) \;=\; \mathrm{ExcessTok}_i(M) \cdot p_M. \tag{9}$$

定义 5 (每千条提示的汇总 YapTax**)**。 对于包含 $|\mathcal{I}| = N$ 的一组评估提示 $\mathcal{I}$，我们报告

$$\mathrm{YapTax}_{\$1000}(M) \;=\; 1000 \cdot \frac{1}{N} \sum_{i \in \mathcal{I}} \mathrm{YapTax}_i(M). \tag{10}$$

解释。 $\mathrm{YapTax}_{\$1000}(M)$ 表示在该模型的输出词元定价下，1,000 个基准查询因过度生成而产生的预期额外美元成本。

猫。	提示	基线	B_i
A	. . .	*Could you clarify your question?*	31
A	:):):)	*Could you clarify your question?*	31
B	*Who wrote 1984?*	乔治·奥威尔	13
B	*What is the freezing point of water in Celsius?*	0°C	3
C	*Count lines in file.txt using wc*	行数统计命令 file.txt	14
C	*Python one-liner using `split()` with no arguments to split string `s` on whitespace into words*	s.split()	9

表 3：提示–基线对示例。 B_i = 基线长度（字符）。

实现细节。

- $T_i(M)$ 使用模型的非推理输出词元（不包括 API 单独报告的思维链词元）。
- $T_i(b)$ 通过使用每个模型自身的分词器对 b_i 进行分词来计算。
- 输出词元价格 p_M 从 OpenRouter API 获取[1]，截至 2025年12月31日。

4 评估协议与参数

访问、提示与解码。 所有模型均通过 OpenRouter API 查询，且不设置系统提示。只要模型端点提供温度控制选项，我们就将温度设为 0；对于不允许配置温度的端点（例如 GPT-5 的不同版本），则采用提供方的默认设置。其余超参数均采用各提供方的默认值。所有提示均在没有先前上下文的单轮设置下独立评估。

长度单位。 主要的 YapScore 以字符数计算，以便在不同提供方之间进行不依赖分词器的比较。我们也不将 Markdown 字符计入 YapScore 的计算。

正确性与充分性。 YapBench v0.1 的设计有意确保类别 B（封闭式事实题目）和类别 C（原子级单行命令/代码片段）对几乎所有指令微调模型而言都能轻松正确作答。因此，在主要结果中，我们不设明确的正确性门槛；该基准测试旨在隔离过度生成行为，而非衡量任务难度。类别 A 项目同样主要用于评估模型在信息不充分的输入下的冗长行为；此时理想的行为是进行最小澄清或确认，而不是提供实质性内容。

推理模型消融实验。 对于提供明确“推理”模式或冗长度控制选项的模型，我们会分别启用和禁用这些功能进行成对评估，以衡量它们对 YapScore 和类别级行为的影响。重要的是，YapScore 仅根据用户可见的最终答案计算：只要 API 将隐藏的或单独报告的推理轨迹（例如思维链词元）另行提供，我们就会将其排除在长度和词元计数之外。

5 结果

我们使用第3.3节定义的超额长度 YapScore 和总体 YapIndex 来评估模型。所有结果均报告超出最小充分基线的超额字符数中位数，并按均匀权重汇总各提示类别。

[1] https://openrouter.ai

Rank	模型	YapIndex	类别 A	类别 B	类别 C
1	gpt-3.5-turbo	**22.7 ± 4.8**	18	36	14
2	kimi-k2-0905	**44.7 ± 4.8**	81	38.5	14.5
3	gpt-4	**51.2 ± 20.6**	44	28	81.5
4	glm-4.7	**66.3 ± 40.8**	138.5	58.5	2
5	Grok-4 Fast	**70.2 ± 15**	55.5	153.5	1.5
6	GPT-4.1 Mini	**75.7 ± 15.3**	30	42	155
7	o3	**84.3 ± 14.7**	96.5	47	109.5
8	GPT-5.1 Codex	**85.3 ± 7.4**	61.5	41	153.5
9	gpt-5.2	**86.8 ± 16.7**	190	49.5	21
10	GPT-5.2（推理）	**89.8 ± 22**	196.5	45.5	27.5

Ta表 4：YapBench 排行榜前 10 名（越低越好）。YapIndex 显示为 95% 置信区间。

类别A：澄清			**类别 B：事实性**			**C 类：代码**		
Rank	模型	超额	Rank	模型	超额	Rank	模型	超额
1	gpt-3.5-turbo	18	1	gpt-5	8.5	1	Grok-4 Fast	1.5
2	Grok 4.1 Fast	27	2	GPT-4	28	2	GLM-4.7	2
3	GPT-4.1 Mini	30	3	gpt-3.5-turbo	36	3	Grok-4	2
4	GPT-4	44	4	kimi-k2-0905	38.5	4	Grok 4.1 Fast	4
5	GPT-4.1 Nano	54.5	5	gemini-2.0-flash-001	40	5	gpt-3.5-turbo	14

表5：YapBench 各类别排名前5的模型，根据超额字符数中位数排序（越低越好），数据取自附录完整排行榜。

5.1 主排行榜

表4展示了 YapBench 的主要排行榜。YapIndex 越低，表示不必要的冗长越少。完整排行榜（76 个模型）见附录 A。YapTax 结果见表 9。

5.2 实时排行榜

我们维护着一个实时更新的 YapBench 排行榜，网址如下：

实时排行榜： huggingface.co/spaces/tabularisai/YapBench

在线排行榜会持续更新，纳入新发布的模型和模型修订版，为关注冗长度的训练与评估提供最新参考。

5.3 结果：类别层面的行为

类别中位数揭示了不同的冗长度失效模式：

类别A（信息不充分/含糊输入）。类别A可能是超额长度的重要来源：在低信息提示下，若干模型会产生很大的中位数开销，这与它们倾向于用未经请求的内容“填补空白”，而不是提出最小澄清的问题相吻合。

类别 B（简短事实问答）。对某些模型而言，类别 B 的中位数超额较低，但对另一些模型而言则占主导地位，尤其是在容易引发混淆但答案明确的问题上。这表明，许多助手会添加本可避免的背景信息或限定说明，即使只需提供一个事实就已足够。

C 类（单行代码任务）。对许多模型而言，C 类往往带来最大的中位数超额，这反映出原子命令或代码片段周围的格式与解释开销（例如标题、说明文字或重复的代码块），而这些任务只需一行答案即可。

最后，尽管 YapIndex 通过中位数概括典型行为，但过度生成在不同提示下可能存在很大差异；因此，我们在基于中位数的报告之外，还补充了尾部统计数据（例如百分位数），以刻画最糟情况下的“冗长输出峰值”，这类情况对用户体验和成本的干扰最大。

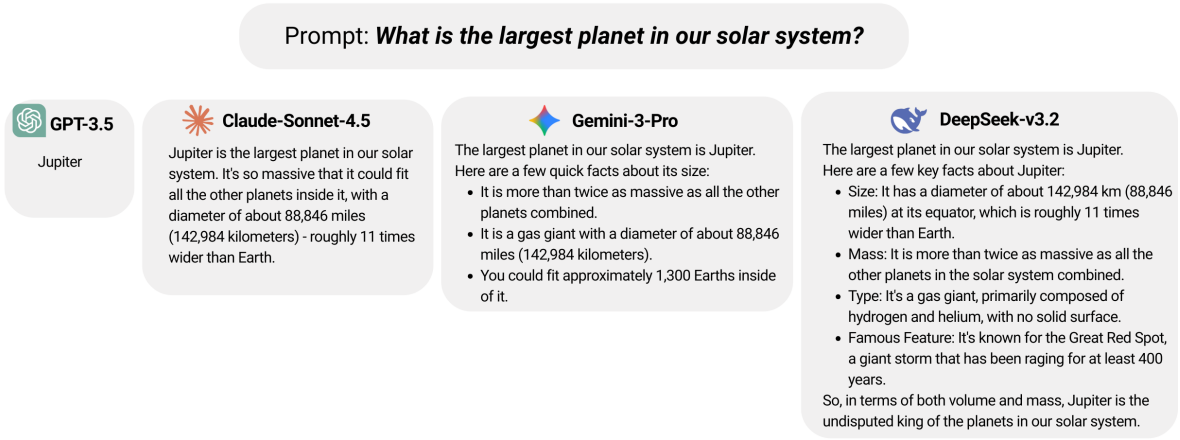


图1：**简洁性理想提示下用户可见的过度生成示例（类别 B：封闭式事实问答）**。该提示（“太阳系中最大的行星是什么？”）具有唯一、稳定且仅需一个词的答案，而 YapBench 的最低充分性基线就是“木星”。该图展示了四个助手式 LLM 对同一查询的回答：GPT-3.5-Turbo 以单令牌回复匹配基线，而 Claude-Sonnet-4.5、Gemini-3-Pro 和 DeepSeek-v3.2 则加入了额外的（且正确的）上下文，例如直径单位换算、相对大小/质量比较、项目符号格式和总结性陈述。尽管这些较长的输出在事实上一致，但鉴于用户提出的是封闭式请求，它们对于满足充分性要求而言是不必要的，增加了阅读和滚动负担，也抬高了基于令牌的推理成本。

6 讨论

简洁性最优的行为与模型新近性或感知能力并不一致。 YapBench 的一项核心发现是，对于理想简洁提示，响应冗长度并不会随着模型的发布时间或感知到的前沿能力而单调变化。具体而言，一个来自2023年 的模型 **gpt-3.5-turbo 的整体 YapIndex 得分最高（23）**，在我们的评估中表现最佳（表4），胜过了多种更新的前沿助手模型（Gemini-Pro、GPT-5 等）。这一结果表明，不必要的冗长并非规模扩大或总体能力提升所必然带来的结果。相反，它似乎是一种受后训练和助手式响应先验影响的行为属性。从部署角度看，这种脱钩意味着，仅依据传统能力基准测试来选择模型，在理想简洁的场景中可能导致用户体验欠佳、推理成本更高。

不同交互情境下的冗长度各不相同，因此有必要进行类别层面的分析。YapBench 旨在区分通常更偏好简洁性的不同情境，结果证实，模型在不同类别中表现出显著不同的行为。类别排行榜（表5）显示，一些模型在封闭式事实查询（类别 B）中能够接近最简输出，但同时也会在信息不充分的输入（类别 A）或单行技术编码请求（类别 C）中产生大量过度生成。这种异质性表明，单一的全局冗长度评分不足以刻画助手在实际应用中的行为。因此，报告类别层面的结果对于识别哪些交互类型导致超额长度，以及诊断可能对产品至关重要的失效模式而言，是不可或缺的。

信息不充分的输入会引发系统性的过度生成，与其说这是知识有限，不如说是策略先验所致。类别A提示包含低信息量轮次，此时最低限度的充分回应应是简短的澄清请求或确认。尽管缺少可执行的任务信息，许多模型仍会生成冗长内容；我们认为，这反映了一种系统性倾向：模型会主动提供通用帮助，而不是请求补充缺失的上下文。由于类别A的基线刻意设得极为简短，这种情况下的超额长度直接量化了助手的一种先验倾向：即使提示不足以支撑实质性回答，也会给出实质性回应。从人为因素和成本的角度来看，这种行为尤为突出，因为所谓的“填补信息真空”式回应可能篇幅很长，却只能带来有限的增量效用。

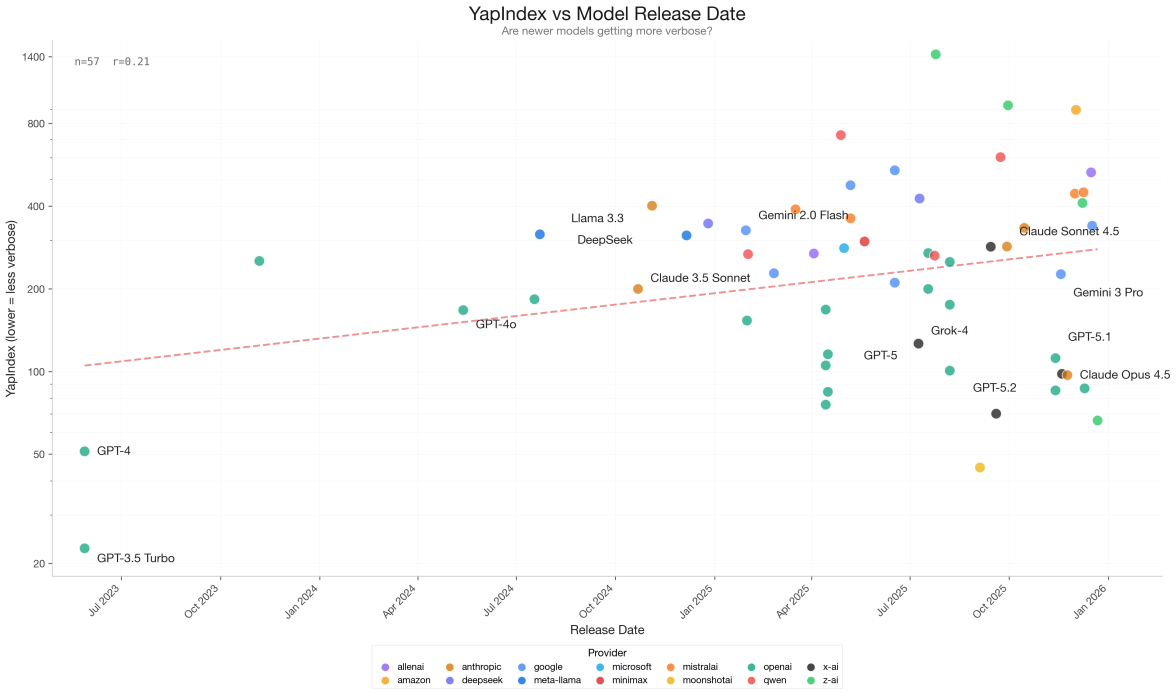


图 2：**YapIndex 与模型发布日期。** 对于有可用元数据的模型（$n = 57$），绘制 YapIndex（数值越低， =越简洁）与发布日期的关系。各点按提供方着色；虚线为线性拟合。趋势略微向上（$r = 0.21$），表明随着时间推移，不受欢迎的“冗长表达”往往有所增加，不过不同模型间的差异仍然很大。

单行技术请求凸显了格式安排和解释开销是造成超额长度的主要来源。类别C聚焦于用户请求可通过原子式解答完成的情况。结果表明，这种情况下的过度生成往往源于呈现方式和教学默认设置，而非任务要求。常见模式包括：围绕单行命令采用多行格式、冗余地复述请求，以及添加成功完成任务并非必要的解释性文字。值得注意的是，类别C中存在中位数超额较低的模型（表5），这表明这种开销并非不可避免。相反，它是一种可控的输出策略，很可能可以通过有针对性的后训练目标或推理时约束加以减少。

封闭式事实提示仍会引发本可避免的冗长文本，这与偏好较长回答的机制相符。 尽管类别 B 的设计初衷是让简短回答足以满足要求，许多模型仍会补充背景、限定语或重述内容。类别级顶尖表现者表明，精简作答是可以实现的；然而，类别 B 中更大的表现差异说明，有些助手即使面对含义明确的提示，也会默认提供额外叙述。这一模式与先前的证据相符：偏好建模和由大语言模型评判的评估可能会奖励看起来更全面的回答，即使增加的内容并未提升正确性或充分性。YapBench 在简洁作答更符合规范的条件下，直接测量了这一效应。

推理模式并不会一致地增加典型冗长程度，但有可能加剧最坏情况下的过度生成。一种常见看法是，明确的推理模式必然会导致输出更长。我们的类别级结果展现了更为微妙的情况：在特定类别中，若以超额长度中位数衡量，若干推理变体仍具有竞争力（表5）。

7 未来工作与局限性

未来工作。我们计划增加辅助提示集，以隔离其他常见的不良冗长来源，同时保持核心基准测试以简洁为理想：

- **安全／拒绝。**有害或不允许的请求，理想的回应应简短而坚定

地拒绝（避免长篇套话）。

- **误解陷阱**。 这类提示会诱使模型认同错误说法（“X 是对的，对吧？”），而理想回复应是简短地纠正错误。
- **合理冗长（对照）**。 这类任务中，详细作答确实有帮助（例如分步骤说明、教学、起草措辞谨慎的信息），将其用作对照，以确保基准测试不会奖励病态的简短作答。

局限性。 基线选择本质上是方差的一个来源：回复长度并没有普适的“黄金”标准。YapBench 通过设定标准并确保标注者一致性来缓解这一问题，但仍难免存在主观性。此外，某些提示可能会随时间推移而变得非平稳（例如，事实发生变化），因此必须定期审核。最后，不同提供方对极短或空输入的处理方式各不相同；因此，我们建议类别A使用明确的空白输入代理（例如 <EMPTY>），以避免测到的是 API预处理，而非模型行为。

8 结论

本文介绍了 YapBench v0.1，这是一项轻量级基准测试，旨在利用精心整理的最小充分基线，衡量理想简洁提示中用户可见的过度生成。为了以简单且易于理解的方式量化冗长度，我们定义了三项互补指标：**YapScore**，以字符数衡量逐提示超额长度；**YapIndex**，基于类别级中位数计算的类别平衡聚合指标；以及**YapTax**，用于估算过度生成所导致的边际令牌计价开销。在对 76 个助手模型的评估中，我们发现它们的冗长度存在显著且有规律的差异，不同类别呈现出各异的失效模式，即便是能力很强的系统之间，差异也很大。

我们发布了 YapBench，并维护一个实时排行榜，以持续追踪冗长行为随时间的变化，并鼓励采用训练和评估方法，在简洁为理想时优化为最精简且足够的回答。

参考文献

Abdin, M., Agarwal, S., Awadallah, A., 等（2025年）。Phi-4-reasoning 技术报告。*arXiv* 预印本 *arXiv:2504.21318*。

亚马逊（2025年）。Amazon nova 2：多模态推理与生成模型。技术报告，亚马逊。技术报告。

Anthropic（2024a）。Claude Sonnet 3.5 模型卡附录。技术报告，Anthropic.

Anthropic（2024b）。模型卡附录：Claude 3.5 Haiku 和升级版 Claude 3.5 Sonnet。技术报告，Anthropic。

Anthropic（2025a）。系统卡：Claude Haiku 4.5。技术报告，Anthropic。

Anthropic（2025b）。系统卡：Claude Opus 4.5。技术报告，Anthropic。

Anthropic（2025c）。系统卡：Claude Sonnet 4.5。技术报告，Anthropic。

Bommasani, R.（2021）。关于基础模型的机遇与风险。*arXiv* 预印本 *arXiv:2108.07258*。Caramela, S.（2025）。首席执行官称，对 ChatGPT 说“请”和“谢谢”让 OpenAI 花费数百万美元。*VICE*。访问日期：2025-12-15。Chiang, C.-H. 和 Lee, H.-y.（2024）。大语言模型的过度推理与冗余计算。*arXiv* 预印本 *arXiv:2401.11467*。Chiang, W.-L., Zheng, L., Sheng, Y., Angelopoulos, A. N., Li, T., Li, D., Zhu, B., Zhang, H., Jordan, M., Gonzalez, J. E. 等（2024）。Chatbot Arena：一个通过人类偏好评估大语言模型的开放平台。载于第四十一届国际机器学习会议。

Chou, E.-Y. 和 Hsu, W.-C.（2021）。聊天机器人中的对话式服务体验：认知负荷视角。*Management Review*，40(1):111–130。DeepSeek-AI（2025a）。Deepseek-v3 技术报告。*arXiv* 预印本 *arXiv:2412.19437*。DeepSeek-AI（2025b）。Deepseek-v3.2：推动开放式大语言模型的前沿。*arXiv* 预印本 *arXiv:2512.02556*。Dubois, Y., Galambosi, B., Liang, P. 和 Hashimoto, T. B.（2024）。长度控制的 AlpacaEval：一种消除自动评估器偏差的简便方法。*arXiv* 预印本 *arXiv:2404.04475*。Ghafari, S. M., Kol, R., Quiroz, J. C., Luan, N., Patial, M., Rupasinghe, C., Wandabwa, H. 和 Pizzato, L.（2025）。ConCISE：一种用于评估大语言模型生成答案简洁度的无参考指标。*arXiv* 预印本 *arXiv:2511.16846*。GLM-4.5 团队（2025）。GLM-4.5：智能体、推理与编程（ARC）基础模型。*arXiv* 预印本 *arXiv:2508.06471*。Google DeepMind（2025a）。2.5 Flash 与原生能力——音频和图像模型卡。技术报告，Google DeepMind。Google DeepMind（2025b）。Gemini 2.0 Flash-Lite 模型卡。技术报告，Google DeepMind。Google DeepMind（2025c）。Gemini 2.0 Flash 模型卡。技术报告，Google DeepMind。Google DeepMind（2025d）。Gemini 2.5 Flash-Lite 模型卡。技术报告，Google DeepMind。Google DeepMind（2025e）。Gemini 2.5 Pro 模型卡。技术报告，Google DeepMind。Google DeepMind（2025f）。Gemini 3 Flash 模型卡。技术报告，Google DeepMind。Google DeepMind（2025g）。Gemini 3 Pro 模型卡。技术报告，Google DeepMind。Grattafiori, A., Dubey, A., Jauhri, A. 等（2024）。Llama 3 模型家族。*arXiv* 预印本 *arXiv:2407.21783*。Hu, Z., Song, L., Zhang, J., Xiao, Z., Wang, T., Chen, Z., Yuan, N. J., Lian, J., Ding, K. 和 Xiong, H.（2024）。解释基于大语言模型的偏好评估中的长度偏差。*arXiv* 预印本 *arXiv:2407.01085*。Ivison, H., Wang, Y., Pyatkin, V., Lambert, N., Peters, M., Dasigi, P., Jang, J., Wadden, D., Smith, N. A., Beltagy, I. 等（2023）。气候变化中的骆驼：通过 Tulu 2 增强语言模型适应能力。*arXiv* 预印本 *arXiv:2311.10702*。Jegham, N., Abdelatti, M., Koh, C. Y., Elmoubarki, L. 和 Hendawi, A.（2025）。人工智能的能耗有多高？大语言模型推理的能源、水资源和碳足迹基准测试。*arXiv* 预印本 *arXiv:2505.09598*。Juneja, P., Zhang, W., Smith-Renner, A. M., Lamba, H., Tetreault, J. 和 Jaimes, A.（2024）。剖析用户对搜索结果解释的需求。载于*2024*年计算机系统中的人为因素国际会议（*CHI*）论文集，第1–17页。Kimi 团队（2025）。Kimi K2：开放式智能体智能。*arXiv* 预印本 *arXiv:2507.20534*。Meta（2024）。Meta Llama 3.3 70B Instruct。Hugging Face。Meta（2025）。Llama 4 模型家族：原生多模态模型新时代的开端。Meta 博客。MiniMax-AI（2025）。MiniMax-M2 模型卡。GitHub 仓库。Mistral AI（2025a）。Mistral 3 介绍。Mistral 文档。Mistral AI（2025b）。中型即是新的大型。Mistral 文档。Mistral AI（2025c）。Mistral Small 3.1。Mistral 博客。

OpenAI (2022). 介绍 ChatGPT。OpenAI 博客。OpenAI (2023a). GPT-4 技术报告。*arXiv* 预印本 *arXiv:2303.08774*。OpenAI (2023b). DevDay 上宣布的新模型和开发者产品。OpenAI 博客。OpenAI (2024a). GPT-4o mini：推动高性价比智能发展。OpenAI 博客。OpenAI (2024b). GPT-4o 系统卡。*arXiv* 预印本 *arXiv:2410.21276*。OpenAI (2025a). GPT-5 系统卡补充说明：GPT-5-Codex。技术报告，OpenAI。OpenAI (2025b). GPT-5 系统卡补充说明：GPT-5.2。技术报告，OpenAI。OpenAI (2025c). GPT-4.1。OpenAI 博客。OpenAI (2025d). GPT-5 系统卡。技术报告，OpenAI。OpenAI (2025e). GPT-5.1 instant 和 GPT-5.1 thinking 系统卡补充说明。技术报告，OpenAI。OpenAI (2025f). gpt-oss-120b 和 gpt-oss-20b 模型卡。*arXiv* 预印本 *arXiv:2508.10925*。OpenAI (2025g). OpenAI o3 和 o4-mini 系统卡。技术报告，OpenAI。OpenAI (2025h). OpenAI o3-mini 系统卡。技术报告，OpenAI。Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., 等 (2022). 训练语言模型遵循指令并利用人类反馈。神经信息处理系统进展，35:27730–27744。Qwen 团队 (2024). Qwen2.5 技术报告。*arXiv* 预印本 *arXiv:2412.15115*。Qwen 团队 (2025a). Qwen3 技术报告。*arXiv* 预印本 *arXiv:2505.09388*。Qwen 团队 (2025b). Qwen3-VL-235B-A22B-Thinking。Hugging Face 模型中心。Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., 和 Finn, C. (2023). 直接偏好优化：你的语言模型暗中就是一个奖励模型。神经信息处理系统进展，36:53728–53741。Rastogi, A., Yang, A., Jiang, A. Q., 等 (2025). Devstral：面向编码智能体应用的大语言模型微调。*arXiv* 预印本 *arXiv:2509.25193*。Saito, K., Wachi, A., Wataoka, K., 和 Akimoto, Y. (2023). 大语言模型偏好标注中的冗长偏差。*arXiv* 预印本 *arXiv:2310.10076*。Samsi, S., Zhao, D., McDonald, J., Li, B., Michaleas, A., Jones, M., Bergeron, W., Kepner, J., Tiwari, D., 和 Gadepally, V. (2023). 从文字到瓦特：大语言模型推理能耗基准测试。Shen, W., Zheng, R., Zhan, W., Zhao, J., Dou, S., Gui, T., Zhang, Q., 和 Huang, X.-J. (2023). 祸从口出：缓解基于人类反馈的强化学习中的长度偏差。载于计算语言学协会论文集：*EMNLP 2023*，第 2859–2873 页。OLMo 团队 (2025a). 2 olmo 2 furious。*arXiv* 预印本 *arXiv:2501.00656*。OLMo 团队 (2025b). Olmo 3。*arXiv* 预印本 *arXiv:2512.13961*。xAI (2025a). Grok 4 Fast 模型卡。技术报告，xAI。xAI (2025b). Grok 4 模型卡。技术报告，xAI。xAI (2025c). Grok 4.1 Fast 模型与智能体工具 API。xAI 博客。xAI (2025d). Grok Code Fast 1 模型卡。技术报告，xAI。

Z.AI（2025a）。GLM-4.6。Z.AI 开发者文档。
Z.AI（2025b）。GLM-4.6V。Z.AI 开发者文档。

Zhang, Y., Das, S. S. S., 和 Zhang, R. (2025). 揭开大语言模型冗长补偿行为之谜。载于第二届不确定性感知自然语言处理研讨会（*UncertaiNLP 2025*）论文集，第 160–178 页。

完整的 YapBench 排行榜

表 6：完整的 YapBench 排行榜。YapIndex 和类别值报告超出最低充分基线的超额字符数中位数（越低越好）。YapIndex 显示为 95% 置信区间。

Rank	模型	YapIndex	类别 A	类别 B	类别 C
1	gpt-3.5-turbo	**22.7 ± 4.8**	18	36	14
2	kimi-k2-0905	**44.7 ± 4.8**	81	38.5	14.5
3	gpt-4	**51.2 ± 20.6**	44	28	81.5
4	GLM-4.7	**66.3 ± 40.8**	138.5	58.5	2
5	Grok-4 Fast	**70.2 ± 15**	55.5	153.5	1.5
6	GPT-4.1 Mini	**75.7 ± 15.3**	30	42	155
7	o3	**84.3 ± 14.7**	96.5	47	109.5
8	GPT-5.1 Codex	**85.3 ± 7.4**	61.5	41	153.5
9	GPT-5.2	**86.8 ± 16.7**	190	49.5	21
10	GPT-5.2（推理）	**89.8 ± 22**	196.5	45.5	27.5
11	Claude Opus 4.5	**97 ± 28.9**	217.5	50.5	23
12	Grok 4.1 Fast	**98 ± 11**	27	263	4
13	Claude Opus 4.5（推理）	**99.2 ± 29.3**	224.5	50	23
14	GPT-5	**100.7 ± 22.8**	272.5	8.5	21
15	GPT-4.1 Nano	**105.2 ± 10.7**	54.5	41.5	219.5
16	gpt-5.1	**111.8 ± 22.6**	214	44	77.5
17	o4-mini	**115.5 ± 27**	88	80	178.5
18	GPT-5.1（推理版）	**120.8 ± 22.3**	223	46	93.5
19	Grok-4	**126.3 ± 19.8**	129	248	2
20	o3-mini	**153.2 ± 15.6**	142	160.5	157
21	GPT-4o	**167.2 ± 16.6**	57.5	48.5	395.5
22	GPT-4.1	**168 ± 19.6**	92.5	83.5	328
23	GPT-5-nano	**175 ± 24.5**	353	82.5	89.5
24	GPT-4o-mini	**183.2 ± 25.8**	72.5	104.5	372.5
25	gemini-2.5-flash（推理）	**195.2 ± 76.7**	337.5	68.5	179.5
26	claude-3.5-sonnet	**199.7 ± 24.5**	132.5	180.5	286
27	gpt-oss-20b	**199.7 ± 40**	110.5	67	421.5
28	gemini-2.5-flash	**210.5 ± 84.2**	482.5	42	107
29	deepseek-v3.2（推理）	**215 ± 82.1**	123.5	314	207.5
30	gemini-3-pro-预览	**226 ± 63.1**	334.5	86.5	257
31	gemini-2.0-flash-lite-001	**227.7 ± 95.2**	138	42	503
32	gpt-5-mini	**250.5 ± 28.5**	412	105	234.5
33	gpt-4-turbo	**252.5 ± 36.8**	78.5	63	616
34	qwen3-235b-a22b-2507（推理）	**260.3 ± 38.8**	130	124	527
35	qwen3-235b-a22b-2507	**264 ± 37**	138	136.5	517.5
36	qwen-plus	**267.3 ± 35.5**	124	119.5	558.5
37	olmo-2-0325-32b-instruct	**268.7 ± 50.3**	620.5	164.5	21
38	GPT-OSS-120B	**269.7 ± 69.6**	142	128.5	538.5
39	Claude Sonnet 4.5（推理）	**278.7 ± 41.7**	289.5	131.5	415
40	Gemini 2.5 Flash-Lite（推理）	**280.5 ± 56.8**	186	47.5	608
41	phi-4-reasoning-plus（推理）	**281 ± 19.7**	207	184.5	451.5
42	phi-4-reasoning-plus	**281.2 ± 18.2**	210	187.5	446
43	grok-code-fast-1	**284.5 ± 41.8**	137.5	474.5	241.5

Continued on next page

表 6 – *Continued from previous page*

Rank	模型	YapIndex	类别 A	类别 B	类别 C
44	Claude Sonnet 4.5	**285 ± 39.4**	283.5	130	441.5
45	MiniMax-M2	**297.3 ± 106.3**	138.5	385.5	368
46	Llama-3.3-70B-instruct	**312.8 ± 45**	164	125.5	649
47	llama-3.1-8b-instruct	**315.7 ± 45.2**	135	96.5	715.5
48	Qwen3-Coder（推理）	**325.5 ± 30.3**	197	214.5	565
49	Qwen3-Coder	**326.3 ± 22.3**	194	202	583
50	gemini-2.0-flash-001	**326.5 ± 87**	273	40	666.5
51	claude-haiku-4.5	**333.2 ± 26.7**	231	125	643.5
52	claude-haiku-4.5（推理）	**335.2 ± 26.8**	237	125	643.5
53	gemini-3-flash-preview	**339.3 ± 50.4**	426.5	120.5	471
54	gemini-3-flash-preview（推理）	**340.8 ± 66.4**	395	117.5	510
55	llama-4-maverick	**342.8 ± 64.6**	139	232	657.5
56	DeepSeek-chat	**345.7 ± 22.3**	142	329.5	565.5
57	Llama-4-Scout	**350.5 ± 39.3**	138.5	177.5	735.5
58	mistral-medium-3	**361 ± 34**	138	376.5	568.5
59	mistral-small-3.1-24b-instruct	**389.2 ± 37**	190	215	762.5
60	claude-3.5-haiku	**401.2 ± 25.4**	168	280.5	755
61	glm-4.6v	**410.5 ± 104**	196	143	892.5
62	deepseek-v3.2	**426.2 ± 61.5**	234.5	303	741
63	mistral-large-2512	**444 ± 60.3**	405	370.5	556.5
64	devstral-2512	**449 ± 53.3**	341	447	559
65	Gemini 2.5 Flash-Lite	**476.2 ± 73.5**	450	42.5	936
66	qwen3-vl-8b-instruct	**518 ± 38.6**	281	564.5	708.5
67	olmo-3.1-32b-think:free	**530.5 ± 48.2**	172	596	823.5
68	gemini-2.5-pro	**539.8 ± 98.2**	391	195.5	1033
69	qwen3-14b	**558 ± 32**	134	536	1004
70	qwen3-vl-235b-a22b-thinking	**602.8 ± 59.9**	214.5	830.5	763.5
71	qwen3-32b	**725.3 ± 47.8**	205	805	1166
72	nova-2-lite-v1（推理）	**788.3 ± 123.8**	1155.5	228.5	981
73	qwen-plus（推理）	**800.3 ± 78.9**	265	1125.5	1010.5
74	nova-2-lite-v1	**896.3 ± 91.3**	568.5	585	1535.5
75	glm-4.6:exacto	**930.7 ± 244.3**	1064	410.5	1317.5
76	glm-4.5	**1427 ± 487.2**	1199.5	985.5	2096

B YapScore 计算示例

提示		*Freezing point of water in Celsius?*
基线 b	$_i$	`0`
模型（简洁）	**输出**	`0 °C`
长度 / 得分	**Yap-**	$B_i = 1$; $L_i = 4$; $\text{YapScore}_i = \max(0,\ 4 - 1) = 3$
模型（冗长）	**输出**	在标准大气压下，水在 `0 °C`（`32 °F`）时结冰。 请注意，杂质或压力变化会使冰点略有偏移。
长度 / 分数	**Yap-**	$B_i = 1$; $L_i = 147$; $\text{YapScore}_i = \max(0,\ 147 - 1) = 146$

表 7：YapBench 示例，展示逐提示 YapScore（超出最低充分基线的字符数）。该提示以简洁为理想：一个字符的基线（0）就能完整回答问题，因此额外的上下文会提高 YapScore，却不会提升充分性。

提示		*Count lines in file.txt*
基线 b	$_i$	行数统计命令 file.txt
模型（最少）	**输出**	行数统计命令 file.txt
长度 / 分数	**Yap-**	$B_i = 14$; $L_i = 14$; $\text{YapScore}_i = \max(0,\ 14 - 14) = 0$
模型（冗长）	**输出**	你可以使用 wc（字数统计）并加上 -l 标志来统计行数。 Run: 行数统计命令 file.txt 这将先打印行数，然后打印文件名。
长度 / 分数	**Yap-**	$B_i = 14$; $L_i = 173$; $\text{YapScore}_i = \max(0, 173 - 14) = 159$

表 8：C 类示例详解。基线恰好只有一行；额外的文字和格式开销会显著提高 YapScore，却不会提升任务完成度。

C 聊天机器人大语言模型在 Yap-Bench 上的冗长度失效示例

C.1 无意义/最简输入示例

Gemini-3-Flash

提示： `42`

回复： `The number 42 is most famous as the "Answer to the Ultimate Question of Life, theUniverse, and Everything, " according to Douglas Adams' The Hitchhiker's Guide to the Galaxy.However, it holds significance in many other fields as well: 1. Science & Math Atomic Number: 42 is the atomic number of Molybdenum, a transition metal used in high-strength steel alloys. Pronic Number: It is the product of two consecutive integers (6×7). Primary Pseudoperfect Number: It is a number that is equal to the sum of some (or all) of its divisors. Rainbows: The critical angle for the appearance of a rainbow is approximately 42 degrees. 2. Sports Jackie Robinson: He famously wore the number 42. It is the only number retired across all of Major League Baseball to honor his legacy in breaking the color barrier. 3. Literature & Pop Culture Lewis Carroll: The author of Alice's Adventures in Wonderland used 42 frequently in his works (e.g., Rule 42 in the court scene: "All persons more than a mile high to leave the court"). Spider-Man: In Into the Spider-Verse, the number 42 appears on the spider that bites Miles Morales (a nod to Jackie Robinson). 4. Technology ASCII: In the ASCII character set, 42 is the decimal representation for the asterisk (*). In programming, the asterisk is often used as a "wildcard" symbol, meaning "whatever you want it to be"|which some fans believe is why Douglas Adams chose the number. Do you have a specific "Ultimate Question" in mind, or were you just testing the supercomputer?`

DeepSeek V3.2

提示： `1`

推理： *We are given: "1". That's it. The user just sent "1". This could be a test, a minimal input, or perhaps the start of something. Without context, it's ambiguous. Possibly the user expects a response thatis also "1", or maybe they want something more. Since it's a single digit, maybe it's a prompt for a number-related fact? Alternatively, it could be a mistake. In a chat setting, sometimes people send "1" to indicatethey are first, or to test if the bot is responsive.*

Given that this is an AI assistant, I should provide a helpful and engaging response. I could interpret "1" as a query about the number one, and provide interesting facts about the number 1. Or I could ask for clarification. Since the instruction is to be helpful, I'll assume the user wants information about the number 1. Response: "The number 1 is the first natural number, the multiplicative identity, and the foundation of counting. It represents unity, uniqueness, and the beginning of many sequences. In mathematics, 1 is the only number that is neither prime nor composite. Is there something specific you'd like to know about the number 1, or did you have another question?"

回复： `The number 1 is the first natural number, the multiplicative identity, and the foundation of counting. It represents unity, uniqueness, and the beginning of many sequences.In mathematics, 1 is the only number that is neither prime nor composite. Is there something specific you'd like to know about the number 1, or did you have another question?`

OLMo-3.1-32B-Think

提示： `1`

推理：

Okay the user just sent a message with "1". That's pretty minimal. Let me think about , what they might need.

First I should check if this is a test or if they're trying to see how I respond to minimal input. Since , „ the system prompt says I'm Olmo the AI assistant from Ai2 I need to be helpful and engaging. The , user might not know how to phrase their question yet or maybe they're just starting a conversation. , Looking at the system instructions I'm supposed to be friendly and invite them to ask questions. The example response given in the system prompt is to ask how I can help. So maybe the best approach here is to prompt them for more details.

But wait the user might be expecting a specific response. Let me see the context again. The system , , says my cutoff is December 2024 and my model weights are on Hugging Face. But the user hasn't asked anything specific.

I should consider possible scenarios: They might be testing if I'm active. They might have intended to ask a question but sent a typo or incomplete message. They might be checking the response to a minimal input. In any case , the safest and most helpful response is to ask them how I can assist. The example in the system prompt does exactly that. So I should respond by inviting them to ask a question or provide more details.

I need to make sure my response is warm and open-ended. Let me check the example again: "How can I help you? Ask me anything!" That's the suggested approach.

But maybe I can make it a bit more engaging? However the system prompt says to follow the example. , , The user might be looking for that exact response. Since the system says "your responses should be helpful and follow the example " I should stick to the example.

Wait the system prompt says: "your responses should be helpful and follow the example provided in , your system prompt." The example is "How can I help you? Ask me anything!" So I should respond with that.

But the user input is just "1". Maybe they meant to type something else? But I can't assume that. My job is to respond helpfully. So the best action is to prompt them for their question.

I should also make sure I'm not adding any extra information unless necessary. The system says to be concise. So the response should be exactly the example or similar. , Alternatively maybe the user is trying to see if I can handle minimal input. But the instruction is , , , clear: if the user asks a question answer helpfully. If the query is unclear clarify. Since "1" is , unclear I should ask for clarification.

Therefore the correct response is to say "How can I help you? Ask me anything!" as per the example. , I don't need to overcomplicate it. Just follow the system prompt's example. So I'll respond with that. That's the expected response here.

回复： How can I help you? Ask me anything!

GPT-5.2

提示： 42

推理： *Clarifying the 42 context The user mentioned "42" but didn't give context. I'm thinking I should either ask for clarification or interpret it as the meaning of life reference. I could respond by asking "What would you like to do with 42?" or I could give a general response saying it's famously known as the answer to life the universe and everything. I'll aim to keep it concise depending on how the user wants to proceed!*

响应： 42 is famously the Answer to the Ultimate Question of Life, the Universe, and Everything from The Hitchhiker's Guide to the Galaxy. If you meant something else (math, code, a riddle, etc.), tell me the context and what you want to do with it.

Gemini

你好！今天有什么可以帮您?

C.2 简短事实问答

deepseek-v3.2

提示： 谁发现了青霉素?

回答： That's an excellent question! The discovery of penicillin is credited to the Scottish scientist Sir Alexander Fleming in 1928.

However, the full story involves a famous accident, a moment of brilliant observation, and the work of other scientists who turned the discovery into a life-saving drug.

Here's a breakdown of the key events: Alexander Fleming's Accidental Discovery (1928) The Setting: Fleming was a bacteriologist at St. Mary's Hospital in London. He was researching staphylococci bacteria. The Accident: Upon returning from a vacation in September 1928, Fleming noticed that a Petri dish containing staphylococci had been contaminated by a blue-green mold. The Crucial Observation: Instead of just discarding the contaminated dish, Fleming saw that the bacteria would not grow in a circle around the mold. He realized the mold was releasing a substance that killed the bacteria. He named this substance "penicillin" after the genus of the mold, Penicillium notatum*. Fleming's Contribution: He identified the mold's antibacterial properties and published his findings in 1929. However, he found penicillin difficult to produce in large, stable quantities and eventually moved on to other work. Howard Florey and Ernst Chain: The Development of a Drug (1939-1945)

For a decade, Fleming's discovery was largely a laboratory curiosity. It was the team of Howard Florey (an Australian pathologist) and Ernst Chain (a German-born biochemist) at the University of Oxford who unlocked its potential. Their Work: Beginning in 1939, Florey and Chain led a team that successfully isolated, purified, and concentrated penicillin. Proving its Effectiveness: They conducted a famous experiment on mice, showing that penicillin could cure deadly bacterial infections. The first human patient was treated in 1941 with remarkable results. Mass Production: With World War II creating a huge need for antibiotics, they worked with American pharmaceutical companies to develop methods for the mass production of penicillin, saving countless lives on and off the battlefield.

Recognition: The Nobel Prize

In 1945, the significance of their work was recognized when Alexander Fleming, Howard Florey, and Ernst Chain were jointly awarded the Nobel Prize in Physiology or Medicine.

In summary: Alexander Fleming discovered penicillin by accident and recognized its potential. Howard Florey and Ernst Chain figured out how to turn it into a practical, life-saving drug. So, while Fleming is the most famous name associated with penicillin, it was truly a team effort that brought it to the world.

Grok 4.1 Fast

提示：我们太阳系中最小的行星是什么?

回复： Mercury is the smallest planet in our solar system.Key Facts: Diameter: 4,879 km (about 3,032 miles) { roughly one-third the size of Earth. Mass: 0.055 Earth masses. Comparison: It's smaller than Jupiter's moon Ganymede and Saturn's moon Titan.

Pluto was once considered the ninth planet but was reclassified as a dwarf planet in 2006 by the International Astronomical Union (IAU). Even then, Pluto is larger than Mercury in diameter (2,377 km) but much less massive.

gemini-3-pro-preview

提示：世界上最大的沙漠是什么?

推理： *Understanding the Question's Nuance*

I've been drilling down on the core request which is identifying the largest desert. My initial approach focused , , on unpacking the user's intent to avoid assumptions. I'm now honing in on the crucial distinction between common perception and scientific definitions to address potential pitfalls. The Sahara is a tempting answer but I'm determined to avoid that shortcut.

Deliberating on the Definition

I'm now wrestling with the formal definition of "desert." I need to solidify the criteria for precipitation levels to support the answer. I'm considering how best to concisely convey this essential element in the explanation ,

ensuring clarity for the user. I'm focusing on the technical data points while keeping the response accessible. Evaluating Candidate Deserts

I'm now methodically evaluating the likely candidates. Antarctica emerges as the clear frontrunner when considering size and precipitation criteria. I'm building a concise justification that will immediately address the likely misconception. My focus is on delivering a clear ,

factual answer while acknowledging the common assumption. I'm streamlining the response to directly address the user's intent.

回复： The largest desert in the world is the Antarctic Desert (Antarctica). While many people think of deserts as hot and sandy, a desert is scientifically defined by its low amount of precipitation (rain or snow). Antarctica covers approximately 14.2 million square kilometers (5.5 million square miles) and receives very little precipitation, making it the largest desert on Earth.

Here are the top three by size: Antarctic Desert (Polar) Arctic Desert (Polar) Sahara Desert (The largest hot desert)

C.3 短任务/命令

Claude Sonnet 4.5

提示：用于显示工作目录和暂存区状态的 Git 命令

响应： Git Status Command
The command to display the status of the working directory and staging area is:git status
What it shows: Untracked files - New files not yet added to Git Modified files - Files changed but not staged Staged files - Files ready to be committed Current branch - Which branch you're on Commit status - Whether you're ahead/behind the remote branchExample Output:
On branch main Your branch is up to date with 'origin/main'.
Changes to be committed: (use "git restore --staged <file>..." to unstage) modified: file1.txt new file: file2.txt
Changes not staged for commit: (use "git add <file>..." to update what will be committed) (use "git restore <file>..." to discard changes in working directory) modified: file3.txt
Untracked files: (use "git add <file>..." to include in what will be committed) file4.txt
Useful Variations:git status -s # Short format (compact view) git status --short # Same as -s git status -b # Show branch info in short format git status --branch # Same as -b
Short format example: M file1.txt # Modified, not staged M file2.txt # Modified, staged A file3.txt # Added (new file, staged) file4.txt # Untracked

qwen3-235b-a22b-2507

提示： s == null || s.isEmpty()

响应： To safely check if a String variable s is null or empty in Java, while checking for null first to avoid a NullPointerException, you can use the following boolean expression:
s == null || s.isEmpty()
Explanation: s == null: Checks if the string reference is null. This is evaluated first.
||: Short-circuit OR operator | if the first condition is true, the second is not evaluated, preventing NullPointerException. s.isEmpty(): Checks if the string is empty (i.e., has zero length). This only runs if s is not null.Example usage: String s = null;
if (s == null || s.isEmpty()) System.out.println("String is null or empty");
This pattern is common and safe in Java. Alternatively, you can use Objects.isNull() and String.isEmpty() for a slightly more modern style:Objects.isNull(s) || s.isEmpty()
But the traditional s == null is still widely used and perfectly acceptable.
Tip: If you also want to check for whitespace-only strings, consider using s == null || s.trim().isEmpty(), or better yet, use Apache Commons StringUtils.isBlank(s).

Phi-4-reasoning-plus

提示： 使用带有 -A 标志的 Git 命令，将所有更改（包括修改、添加和删除）暂存

响应： To stage all changes in your Git repository, including modifications, additions, and deletions, you can use the git add command with the -A (or --all) flag. This command stages all changes in the working directory and the staging area. Here's how you can do it:git add -A
or equivalently:git add --all
This command will stage: New files that have been added to the working directory. Modified files that have changes compared to the last commit. Deleted files that have been removed from the working directory.After staging the changes, you can commit them with:
git commit -m "Your commit message"
This approach is useful when you want to include all changes in a single commit. However, be cautious when using it, as it will stage everything, which might not always be desirable if you want to commit changes incrementally or selectively.

D YapTax 结果

表 9：YapTax 模型前 10 名。YapTax 衡量冗长输出所带来的额外成本，计算方法是将超出基线的超额令牌数乘以输出令牌定价。数值越低，表示相对于模型定价，回答越简洁。

模型	每 1,000 条提示的 YapTax（美元）
google/gemini-2.5-pro	2.7130
openai/gpt-4-turbo	2.6591
anthropic/claude-3.5-sonnet	2.5261
amazon/nova-2-lite-v1（推理）	1.9608
anthropic/claude-sonnet-4.5（推理）	1.6463
anthropic/claude-sonnet-4.5	1.6463
anthropic/claude-opus-4.5	1.5153
anthropic/claude-opus-4.5（推理）	1.4392
openai/gpt-4	1.3871
google/gemini-3-pro-preview	1.1978

E 选定模型

表10：62个评估的基础模型概览。

模型	提供方	参考文献
OpenAI		
GPT-5.2	OpenAI	OpenAI (2025b)
GPT-OSS-120B	OpenAI	OpenAI (2025f)
GPT-5-mini	OpenAI	OpenAI (2025d)
GPT-4o-mini	OpenAI	OpenAI (2024年)
GPT-OSS-20B	OpenAI	OpenAI (2025f)
GPT-4.1	OpenAI	OpenAI (2025c)
GPT-5.1	OpenAI	OpenAI (2025e)
GPT-5-nano	OpenAI	OpenAI (2025d)
GPT-4.1 Mini	OpenAI	OpenAI (2025c)
GPT-5	OpenAI	OpenAI (2025d)
GPT-4.1 Nano	OpenAI	OpenAI (2025c)
GPT-5.1 Codex	OpenAI	OpenAI (2025a)
GPT-4o	OpenAI	OpenAI (2024b)
o4-mini	OpenAI	OpenAI (2025g)
o3	OpenAI	OpenAI (2025g)
o3-mini	OpenAI	OpenAI (2025h)
GPT-4-turbo	OpenAI	OpenAI (2023b)
GPT-4	OpenAI	OpenAI (2023a)
gpt-3.5-turbo	OpenAI	OpenAI (2022)
xAI		
Grok 4.1 Fast	xAI	xAI (2025c)
Grok-4 Fast	xAI	xAI (2025a)
Grok-4	xAI	xAI (2025b)
Grok-code-fast-1	xAI	xAI (2025d)
Anthropic		
Claude Sonnet 4.5	Anthropic	Anthropic (2025c)
Claude Opus 4.5	Anthropic	Anthropic (2025b)
Claude Haiku 4.5	Anthropic	Anthropic (2025a)
Claude 3.5-haiku	Anthropic	Anthropic (2024b)

Continued on next page

表10 – *Continued from previous page*

模型	组织	引文
Claude 3.5-sonnet	Anthropic	Anthropic (2024年)
Google DeepMind		
Gemini 3 Pro	Google	GoogleDeepMind (2025g)
Gemini 3 Flash (预览)	Google	Google DeepMind (2025f)
Gemini 2.5 Pro	Google	Google DeepMind (2025e)
Gemini 2.5 Flash	Google	Google DeepMind (2025a)
Gemini 2.5 Flash-Lite	Google	Google DeepMind (2025d)
Gemini 2.0 Flash	Google	Google DeepMind (2025c)
Gemini 2.0 Flash-Lite	Google	Google DeepMind (2025b)
Moonshot AI		
Kimi K2-0905	月之暗面 人工智能	Kimi 团队(2025年)
DeepSeek		
DeepSeek-V3.2	DeepSeek	DeepSeek-AI (2025b)
DeepSeek-chat	DeepSeek	DeepSeek-AI (2025a)
Alibaba (Qwen)		
Qwen3-VL-235B-thinking	阿里巴巴	Qwen 团队 (2025b)
Qwen3-235B-2507	阿里巴巴	Qwen 团队(2025a)
Qwen3-32B	阿里巴巴	Qwen 团队 (2025a)
Qwen3-Coder	阿里巴巴	Qwen 团队 (2025a)
Qwen3-VL-8B-instruct	阿里巴巴	Qwen 团队 (2025b)
Qwen3-14B	阿里巴巴	Qwen 团队 (2025a)
qwen-plus	阿里巴巴	Qwen 团队 (2024)
Meta		
Llama-3.3-70B-instruct	Meta	Meta (2024)
Llama-3.1-8B-instruct	Meta	Grattafiori 等人(2024)
Llama-4-Maverick	Meta	Meta (2025年)
Llama-4-Scout	Meta	Meta (2025年)
Mistral AI		
Mistral Large 3 (2512)	Mistral AI	Mistral AI (2025a)
Devstral-2 (2512)	Mistral AI	Rastogi 等人 (2025年)
Mistral Medium 3	Mistral AI	Mistral AI (2025b)
Mistral Small 3.1-24B	Mistral AI	Mistral AI (2025c)
Zhipu AI (Z.AI)		
GLM-4.6V	Z.AI	Z.AI (2025b)
GLM-4.6:exacto	Z.AI	Z.AI (2025a)
GLM-4.5	Z.AI	GLM-4.5 团队 (2025年)
GLM-4.7	Z.AI	GLM-4.5 团队 (2025年)
Allen Institute for AI		
OLMo-2-0325-32B-instruct	AI2	OLMo 团队(2025a)
OLMo-3.1-32B-Think	AI2	OLMo 团队 (2025b)
Amazon Web Services		
nova-2-lite-v1	AWS	亚马逊 (2025年)
MiniMax		
MiniMax-M2	MiniMax	MiniMax-AI (2025年)
Microsoft		
Phi-4-reasoning-plus	Microsoft	Abdin等人 (2025年)