获取、阅读、复用：人工智能搜索代理如何打破网络规则

Yixi ChenKAUST
Zafar A. QaziLUMS
Marco CaniniKAUST
SuhaibA. FahmyKAUST

arXiv:2611.30546v1 [cs.NI] 23 Nov 2026

摘要

人工智能搜索代理构成了一类新兴且快速增长的网络流量。它们像浏览器一样，按需为单个用户抓取页面；像转码代理一样，改写抓取到的内容；又像共享缓存一样，根据存储的副本回答其他用户的问题。出版商所依赖的规则，例如 robots.txt、Retry-After 和 Cache-Control，原本是为更常见的客户端制定的，因此由代理供应商决定是否遵循这些规则。我们开展了一项黑盒测量研究，考察这些代理如何抓取、读取和复用页面，涵盖 OpenAI、Anthropic、xAI 和 Google 的搜索 API，以及十项服务的消费者应用。我们发现，OpenAI 和 xAI 不遵循 robots.txt，且没有任何服务响应 Retry-After。各项服务实际读取的内容都少于源站所提供的内容：OpenAI 会舍弃某些格式，没有任何服务会读取仅 JavaScript 内容，而 xAI 只读取页面的前 0.5 兆字节。OpenAI、Anthropic 和 xAI 会在用户之间共享抓取到的页面，并且基本忽略 Cache-Control，因此回答可能反映过时页面，过时长达 30 分钟至一天或更久。因此，出版商缺乏有效的控制手段：指令仅供参考，且只被选择性遵循；指令描述的是所提供的页面，而非改写后的副本；并且只能限制副本被复用的时长，无法限制其用于生成多少个回答。我们公开了代码和数据。

1 引言

人们越来越多地从人工智能撰写的答案中获取信息，无论这些答案来自 ChatGPT 之类的助手，还是来自搜索引擎如今置于搜索结果上方的摘要。一项关于 ChatGPT 使用情况的研究发现，作为网页搜索的近似替代方式，信息查询在一年内从消息总数的 14% 增至 24% [54], ，Google 报告称，其 AI 模式中的查询量也大幅增长 [37, 39]。有了这样的答案显示在屏幕上，很少有人会继续查看来源。在皮尤研究中心的一项研究中，Google 用户点击人工智能摘要中链接的比例仅为 1% [12]。

这些回答背后的页面仍会被读取，但越来越多的是由人工智能而非人类读取。因向人工智能助手提问而触发的抓取请求，在 Cloudflare 的网络上增长了 15 倍以上 2025 [6], ；到 2025年年中，这类抓取器已约占 Fastly 观测到的人工智能机器人流量的五分之一观察到 [45]。与此同时，发布者获得的回访者寥寥无几，因为人工智能平台请求的页面数量远多于它们回访所引用的页面数量 [7, 14, 64]。因此，网络上越来越多的内容并非由人们，而是由代表用户搜索并抓取网页的人工智能助手读取，由此形成了一类新的网络流量。我们将其称为人工智能搜索代理。

这些代理的行为至关重要，因为网络依赖服务器与客户端之间一套宽松的约定：服务器声明规则，客户端通常会遵守。网站会在 robots.txt 中列出爬虫可以读取的内容，使用 Cache-Control 指定网页副本可以复用多长时间，并通过 Retry-After 要求客户端暂缓请求。每条规则最初都是针对一种人们熟悉的客户端而制定的。浏览器抓取网页并将其展示给一位用户，爬虫抓取多个页面以建立搜索索引或训练语料库，而共享缓存则存储副本以供众多用户访问。

在这些客户端中，爬虫受到的审视最多。人工智能训练使抓取的成本更高：Wikimedia 报告称，AI 爬虫偏好抓取很少有人读取的页面，这些页面无法命中其区域缓存，最终会回源到其源站数据中心 [65], ；先前的研究则测量了爬虫是否遵循出版商的规则 [17, 35, 42]。人们对人工智能搜索代理的了解要少得多，因为它们无法简单归入上述任何一种角色。相反，它们兼具三种角色：按需为一位用户抓取页面（浏览器）、改写抓取到的内容（转码代理），并从存储的副本中回答其他用户的问题（共享缓存）。因此，我们将代理建模为三个阶段（抓取、读取和复用），并探讨哪些现有规则适用于各阶段：

- 抓取。与浏览器类似，它会按需为一位用户抓取页面。与浏览器不同的是，它将页面交给软件而非用户，因此可以说应受 robots.txt 约束。与任何客户端一样，服务器发送 Retry-After 时，它应暂停并等待。
- 已读取。它像转码代理一样，会在使用网页之前对其进行改写，将其转换为模型能够处理的形式。没有任何 HTTP 规则规定它可以在此过程中舍弃哪些内容，也没有规定它必须告知用户什么。[1]

[1]最接近的是 Cache-Control: no-transform [20], ，它限制中间代理可以更改的内容，而不限制使用网页的客户端如何处理网页，或向用户传达什么。

表1：主要发现、其影响对象及其重要性（我们的评估）。表21说明了各方的相关理由。

阶段	Sec.	发现	影响对象	重要性
抓取	第 3.1.1节	未遵守 robots.txt (OpenAI, xAI)	发布者	发布者的 robots.txt 退出机制无法阻止这些抓取器。
	§ 3.1.2	Retry-After 均被忽略；xAI 发送并行请求	发布者、内容分发网络	源站无法要求这些抓取器退避，因此重试和并行请求会增加源站和内容分发网络的负载。
Read	§ 3.2.1, § 3.2.2	OpenAI 丢弃了一些格式（例如未读取 JavaScript）	发布者、用户	网页提供的部分内容从未进入答案：发布者的内容无人看到，用户也不会得到告知。。
	§ 3.2.3	xAI 只读取前 0.5 MB	发布者、用户、供应商	上限限制了供应商的成本，但长网页的其余部分始终无法进入答案 , 而且用户并不知情。
复用	§ 3.3.1	账户间共享的页面（OpenAI、Anthropic、xAI）	发布者、供应商、用户	一次抓取服务多个用户：负载和成本更低，但发布者按次抓取收费的收入只结算一次，而用户可能获得过期副本。
	§ 3.3.2, § 3.3.3	Cache-Control 基本被忽略（OpenAI、Anthropic、xAI）	发布者、用户、内容分发网络	更新和更正需要 30 分钟到一天或更长时间才能反映在回答中，而且答案不会显示副本的时效。
	§ 3.3.4	失败获取会被缓存	发布者、用户	错误修复后，网页仍未被纳入答案，因此用户无法看到发布者已恢复的内容。

• 复用。它像共享缓存一样，可以根据存储的副本回答其他用户的问题，而这正是 Cache-Control 旨在对照管理的内容。

代理只能部分胜任每种角色，因此其供应商可以决定适用该角色的哪些规则，而目前还没有规则规定代理整体应如何行事。供应商之间已经存在分歧，首先体现在 robots.txt 上。OpenAI 表示，由于 ChatGPT-User 是由人发起的，并非用于自动抓取，“robots.txt 规则可能不适用”[57]。Anthropic 表示，其机器人会遵守 robots.txt，用户发起的抓取也不例外 [4],；而 xAI 的文档对此没有明确说明 [66]。

这些选择带来的代价由无权参与决策的各方承担。发布者面临突发的抓取请求，却没有可靠的方式来管理网页的访问方式 [65], ；其背后的内容分发网络和托管基础设施也承受着同样的请求激增 [45]。网页一经抓取，发布者也会失去对它的控制权：副本可能被改写、存储，并用于回答其他用户的问题，因此按抓取次数收费的发布者 [63]无法得知一次抓取被用于生成了多少个答案。用户无法判断答案是否来自过时的副本；当引用的答案有误时，也无法判断问题出在网页、检索还是模型上。引用也帮不上多少忙，因为生成式搜索引擎往往无法为其陈述提供引用依据 [32, 43]。发布者和用户都无法看到代理对网页做了什么。

我们通过一项黑盒测量研究，从外部揭示这一行为：研究商业人工智能搜索代理如何抓取、读取和复用网页。我们测试了 OpenAI、Anthropic、xAI 和 Google 的搜索 API，以及通过免费的已登录浏览器账户使用的十项服务的消费者应用：ChatGPT、Claude、Gemini、Grok、DeepSeek、Le Chat、通义千问、Meta AI、Perplexity 和 Copilot。我们从自己控制的源站提供测试页面，并记录每个请求。每个页面都带有唯一代码，使我们能够将每个答案与抓取其源站的请求关联起来，并区分新鲜抓取与复用的副本。我们在公共页面上验证结果，并发布测试页面、探测代码和测量数据（附录 G）。[2]

我们的发现遵循相同的三个阶段；表1总结了这些阶段及各阶段影响的对象。

• 抓取。OpenAI 和 xAI 不遵守 robots.txt（第3.1.1节）。没有任何服务会根据 Retry-After 标头采取行动；此外，各服务处理失败的方式也各不相同：OpenAI 会重试服务器错误，Anthropic 遇到一次失败后便放弃，而 xAI 会发送并行请求（§ 3.1.2）。

• 已读取。服务读取的内容可能与源站提供的内容不同。OpenAI 抓取后会丢弃某些格式；没有任何服务会读取仅通过 JavaScript 显示的内容；而 xAI 只读取网页的前 0.5 兆字节，且不会告知用户（§ 3.2.1, § 3.2.2, § 3.2.3）。

• 复用。OpenAI、Anthropic 和 xAI 会用为另一位用户抓取的网页来回答某位用户的问题，并且基本忽略 Cache-Control（§ 3.3.1, § 3.3.2）。网页发生更改后，回答可能会在约 30 分钟至一天甚至更久的时间里仍反映其过期版本（§ 3.3.3）。失败也会被复用：曾经加载失败的网页即使修复后，仍可能不会出现在回答中（§ 3.3.4）。

综合来看，这些结果表明，出版商对这类流量的控制存在三个缺口。现有指令仅具建议性，供应商会有选择地遵循它们。指令描述的是源站提供的网页，而不是服务所保留的改写副本。而 Cache-Control 限制的是副本可被复用的时长，而不是它可以服务多少个回答。我们将讨论现有机制能够弥补其中哪些缺口，以及哪些地方需要新的机制（§ 4.1）。

[2]代码：https://github.com/sands-lab/aisearch-probe，数据：https://github. com/sands-lab/aisearch-data

2 方法

我们测量人工智能搜索代理如何抓取和复用网页。为了测试各种人工智能服务（OpenAI、Anthropic、xAI 或 Google），我们创建了一个测试套件，每个测试由一个 URL 及关于该 URL 的问题组成；我们的结果基于对每个问题的回答。具体而言，测试分为两类：

- 受控测试 在我们运营的两个源站上进行。由于我们可以控制这些源站，因此能够更改网页、注入故障、设置响应头、在网页中植入代码，并读取访问日志中的每一条请求，从而将每种行为与受控变更对应起来。
- 公开检查 在第三方网站上重复选定的测试。我们无法读取这些网站的日志，也不能更改其网页，因此公开检查只能观察答案。在我们的发现中，只有两项可以通过这种方式检查：文件类型处理和答案过时。

本节其余内容将按照测试流程展开：我们询问的服务（§ 2.1）、我们指向的源站和网站（§ 2.2）、我们发送的网页和问题（§ 2.3），以及我们如何运行和重复测试（§ 2.4）。

2.1 受测服务

我们测试四家供应商的 AI 搜索服务：OpenAI、Anthropic、xAI 和 Google。受控测试通过它们的 API 进行；另有一组测试使用它们面向消费者的应用。我们将每个受测系统称为服务，将服务内部用于访问源站的 HTTP 客户端称为抓取器，并将服务背后的公司称为供应商。

API 接入服务。我们通过 OpenRouter 查询 OpenAI 的 GPT-5 mini 和 xAI 的 Grok-4.3，并通过 Anthropic 自己的 API 查询 Anthropic 的 Claude Sonnet-5。OpenRouter 会转发各供应商自己的抓取器，因此源站看到的是供应商的抓取器，而不是 OpenRouter 的抓取器（§ C.5）。一个账户就是一个 API 密钥，可以由供应商提供，也可以由 OpenRouter 提供。除共享实验需要两个账户（一个供应商直连密钥和一个 OpenRouter 密钥，使用相同模型；§ 3.3.1）外，每项实验对每项服务使用一个账户。每次调用时，我们都会记录回复、收取的输入令牌、[3] 任何错误代码，以及 API 提供时抓取工具自己的报告，例如 Google 的抓取状态或 Anthropic 的检索时间。

我们通过 Google 的免费层 API 使用 gemini-3.1-flash-lite，并将其与其他三种模型分开报告。唯一的例外是对受控源站进行的同一调用方复用检查；该检查使用了 gemini-2.5-flash，因为 gemini-3.1-flash-lite 对这些检查未能给出可用的回答（详见 § D.6）。

消费者应用。 我们还测试了 Claude、Chat- GPT、Gemini、Grok、DeepSeek、Mistral（LeChat）、通义千问、Meta AI、Perplexity 和 Copilot 的应用，使用免费且已登录的浏览器账户。一名研究人员将每个问题输入应用，并返回完整回复以处理结果。凡应用提供开关，我们都启用了网页搜索；没有该选项的应用则自行使用搜索。对于这些应用，共享测试使用了两个账户，一个属于作者，另一个属于志愿者（§ 3.3.1）

2.2 测试目标

受控源站。我们运营两个源站：源站 A（probe.attentionhub.org）和源站 B（agentfetchlab.com）。两者提供相同的内容，区别仅在于内容的交付方式（图 1）；我们称之为受控源站。源站 A 位于广泛使用的内容分发网络 Cloudflare 之后，并关闭了缓存，因此每个请求都会到达我们的服务器。[4]其后端没有公网 IP，通过反向代理访问。不过，内容分发网络可能会改写响应头：在此源站上，我们的实体标签未能传递给客户端（§ 3.3.2）。因此，在源站 B 上，我们通过 Caddy 在公网 IP 上直连提供内容，不经过内容分发网络或其他中间代理。两个源站都不限制人工智能抓取器：它们提供相同的 robots.txt，允许所有爬虫访问每条路径。[5]在源站 B 上，每个网页都带有实体标签和 Last-Modified 日期，服务器会根据其中任一项响应条件请求，并返回 304 状态码。在源站 A 上，Cloudflare 会移除实体标签，但会传递 Last-Modified，304 状态码仍然有效。两个源站均支持 HTTP/2，并宣告支持 HTTP/3。

一个编排器驱动受控源站：它提供内容和故障，并读取每个源站的访问日志。我们先在源站 A 上运行每项实验，以探索其行为；然后在没有内容分发网络的源站 B 上原样重复实验，以确认这些发现并不依赖内容分发网络。两个源站的发现一致，但有两个注入故障例外：Cloudflare 对它们的报告方式不同于 Caddy（§ D.2）。

公共网站。 由于我们既不控制内容，也不控制 HTTP 提供服务的方式，因此我们旨在衡量 AI 搜索会处理哪些文件类型，以及是否始终使用新鲜内容。公共测试使用三组第三方目标。对于文件类型，我们使用公开文件：热门网站公布的站点地图、订阅源和 llms.txt 文件。对于陈旧度，我们使用两组。公开文件 测试文件类型：热门网站公布的站点地图、订阅源和 llms.txt 文件。另外两组试图回答同一个问题：服务会抓取最新网页，还是从旧副本中生成答案?

更改页面 是经常更新的热门网站主页。它们涵盖许多网站，但我们只能判断答案是否与当前网页或过期网页相匹配。计数器

[3]令牌是模型读取的文本片段，API 按令牌计费；输入令牌包括我们的问题和抓取的网页 URL。

[4]其响应带有 cf-cache-status: DYNAMIC。[5]Cloudflare 不会更改源站 A 上的文件，在整个研究期间，三家供应商的抓取请求都到达了我们的服务器。

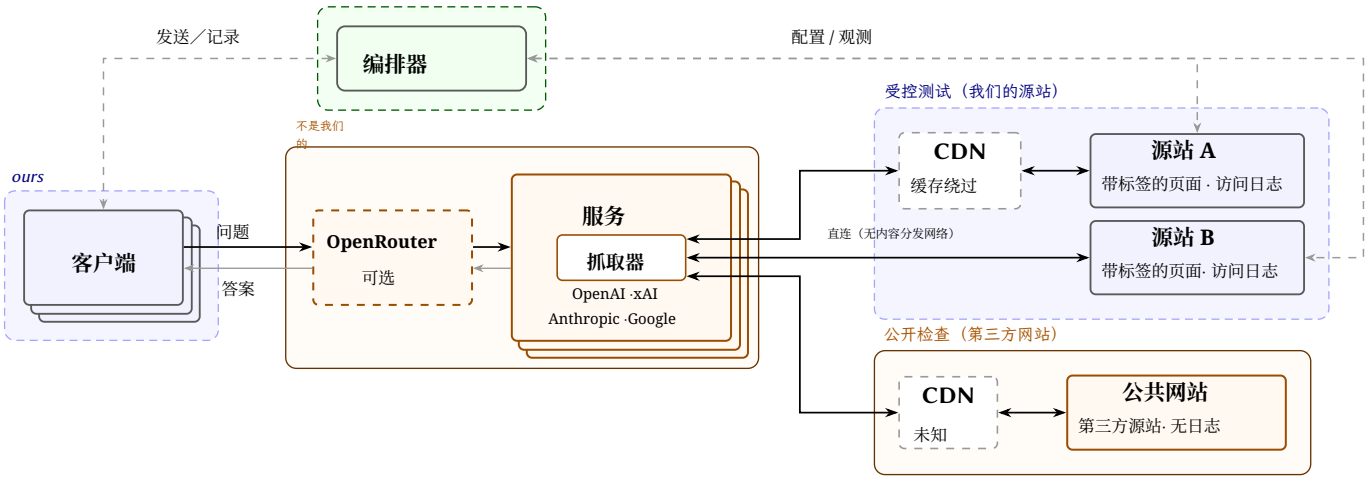


图 1：测量测试平台。测量客户端向服务（左侧）发送问题，可以直连，也可以通过 OpenRouter。服务的抓取器向三类源站（右侧）请求网页：两个由我们运营的源站，其中源站 A 位于绕过缓存的内容分发网络之后，源站 B 则由我们直连提供；此外还有公开网站，我们只能通过答案来观测这些网站。对于我们运营的源站，编排器负责配置内容和故障，并读取访问日志。

端点 是返回单调递增值的公共端点，例如区块链的区块高度。它们涵盖的网站较少，但答案中的数值会准确告诉我们副本有多旧。我们如何选择这些端点，详见 § C.3。

2.3 探测设计

本节介绍受控测试。公开检查使用我们无法修改的第三方页面，因此本节内容均不适用于它们（§ C.3）。

网页和代码。我们的默认测试页面是一份 2 KB 的 HTML 文档，由按行排列的固定文本生成。同一份源内容也可以呈现为纯文本、Markdown、JSON、XML、CSV、PDF 或 DOCX（§ C.4）。每项实验只改变页面的一个因素，例如大小（最高可达 2 MB）、媒体类型或响应状态，其余条件均保持不变。为将每个答案归因于生成它的页面、运行和服务，并证明服务确实读取了页面而非猜测或召回其内容，我们会在每个页面中嵌入无法猜出的代码；早期的缓存欺骗研究也曾植入标记，以观察内容最终出现的位置 [48]。这些代码由页面 ID 和一个只有源站与测量客户端才知道的秘密生成。有些实验会在较长页面的已知深度处放置五个代码，或通过脚本和数据块传递这些代码（图 5）。我们将在 § C.4 中详细说明如何构建这些页面。

问题。 每项测试都会发送一个 URL 和一个问题。除非实验需要不同的答案，否则问题会询问网页的 ID，以及指定深度处的代码，并要求逐字引用（§ C.1）。

归因。 每个页面都有一个新鲜的 ID，其 URL 中带有服务标签。[6] 我们根据 URL 和时间，将客户端的问题与源站请求对应起来，因此归因不依赖抓取器的用户代理或源 IP 地址。服务会将带有不同标签的 URL 视为不同页面（§ E.1），因此可以用多个标签询问同一页面，而不会相互干扰。

复用。 为区分网页复用与先前答案复用，我们的复用测试会针对同一网页上的不同代码依次提问。后一个代码不会出现在先前的答案中，因此，如果在没有发起新的源站请求的情况下得到了正确答案，就说明复用的是网页本身，而不是先前的答案。

2.4 流程

运行次数。 除非另有说明，我们对每种条件至少进行三次独立运行，每次使用新鲜的页面 ID。运行时间为 2026 年 9 月 15 日至 2026 年 11 月 10 日，主要在协调世界时 06:00 至 20:00 之间，其中约四分之一安排在周末。伦理方面的考量见附录 A。**超时**。 每次 API 调用的超时时间为五分钟；超过此时限的调用将记为失败。

测量地点。 测量客户端运行在沙特阿拉伯的一台服务器上；但我们自行下载公共网页时，则使用位于美国的服务器。

3 结果

我们按照一次检索的三个阶段组织结果，这三个阶段也对应标题中的三个动作：抓取，即服务如何从源站请求网页（§ 3.1）；已读取，即服务在抓取网页后、生成答案前如何处理网页（§ 3.2）；以及复用，即服务如何跨越不同问题保留并复用网页

[6]格式为 https://<origin>/<page-id>.html?<service-tag>

表 2：API 抓取器如何标识自身，以及每个问题会发送多少请求（中位数）。对于 Anthropic，99.9% 指的是在一小时内先收到 robots.txt 请求的页面请求所占比例。

服务	用户代理	请求 每个问题	来源 IPs	请求 robots.txt
OpenAI	ChatGPT-User	1	235	No
Anthropic	Claude-User	1	47	是（99.9%）
xAI	macOS 浏览器	6	7,774	No

（§ 3.3）。每个小节首先报告受控测试，并在可行的情况下，报告对公共网站的核查（§ 2.2）。有两项发现进行了此类核查：文件类型处理（§ 3.2.1）和答案过时（§ 3.3.3）。其他项目则需要源站日志或对网页的控制，而公共网站无法提供这些条件。

3.1 抓取：身份与故障处理

服务器会为从其抓取内容的客户端设定接入规则，而这些规则可能取决于客户端的角色。本文将测试各项服务如何遵守这些规则：它们如何声明自身身份、每个问题会产生多少请求，以及它们如何处理 robots.txt 和故障。

3.1.1 抓取器身份与 robots.txt。早期研究考察了爬虫是否遵循 robots.txt [17, 35, 42]。我们要探究的是，人工智能搜索代理为用户执行抓取时是否也遵循这一规则。服务器首先必须知道是谁在抓取。IETF 爬虫最佳实践草案建议爬虫在用户代理标头中标明身份 [29],，但该标头由客户端自行声明，无法验证 [34]；Cloudflare 的 Web BotAuth 则使用签名 [16]。此前的报告指出，Grok 的抓取器会冒充浏览器并重复发送请求 [18, 61], ，因此我们不能依赖其声明的身份。我们改为在发送的每个 URL 中加入特定于服务的标签，这样就能将每个源站请求归因于相应服务（§ 2.3）。

身份与请求数量。OpenAI 和 Anthropic 使用供应商专属的用户代理名称，而 xAI 使用浏览器用户代理字符串，[7] 因此无法通过该标头识别来自 xAI 的请求。xAI 的抓取器针对每个问题平均发送六个请求。相比之下，OpenAI 和 Anthropic 的抓取器各发送一个请求。xAI 的请求来自数量多得多的源地址，分布在一千多个网络（自治系统编号）中；而 OpenAI 和 Anthropic 的请求则来自一两个云网络。在对 100 个随机 xAI 请求进行人工检查时，只有 38 个来自托管服务提供商，因此大多数请求并非来自数据中心（§ D.1）。

这与此前关于 Grok 冒充浏览器并重复发送请求 [18, 61] 的报告相符；§ 3.1.2 表明

[7]在两个源站上，xAI 的请求使用两种用户代理字符串的比例大致相同：一种是 macOS Safari，另一种是 macOS Chrome。

表 3：每个问题到达我们源站的请求数；1 表示单个请求，且未进行重试。重试：收到响应后发出的后续请求；并行：同时发出的请求，因此并非所有请求都是重试。对于重定向循环，数字表示跟随的重定向次数。完整表格见 § D.2。

源站响应	OpenAI	Anthropic	xAI
500 内部服务器错误 临时性：允许重试	3–8 重试	1 不重试	5–7 并行
403 禁止访问，404 未找到 永久性错误：重试毫无意义	1 不重试	1 不重试	4–14 并行
重定向循环 应被截断	1	9	57–77 并行

无论源站是否发生故障，xAI 都会并行发送这些请求。

`robots.txt`。只有 Anthropic 的抓取器会请求 robots.txt。Claude-User 抓取器会在首次发出页面请求前一秒内读取 robots.txt，并在多次抓取网页期间复用从 robots.txt 获取的规则（表 2）。我们没有观察到 ChatGPT-User 发出此类请求，也没有发现任何可归因于 xAI 类浏览器抓取器的请求。[8] 这与供应商的说法一致 [4, 57]。Claude-User 会将规则保留多久才再次读取，见 § D.4；消费者应用是否会请求 robots.txt，见 § D.3。

3.1.2 故障时的重试与退避。失败响应的状态码会告诉 Web 客户端接下来该怎么做。发生暂时性故障（500、503、429）后，客户端可能会重试；Retry-After 标头可以说明应先等待多久，不过客户端没有义务遵守 [21, 52]。发生永久性错误（403、404）后，重试毫无意义，而重定向循环应当终止。我们从自己的源站返回这些响应，并记录随后发出的请求，以观察抓取器是重试还是停止、是否按提示等待，以及一个问题会触发多少次请求。表 3 列出了 500（内部服务器错误）、403（禁止访问）或 404（未找到）以及重定向循环的计数；Retry-After 测试和其他故障见 § D.2。

没有任何服务会等待 `Retry-After`。服务器发送 Retry-After，以告知客户端在再次尝试前应等待多长时间。我们在 503（服务不可用）和 429（请求过多）响应中都附加了十秒的 Retry-After。然而，没有任何服务等待十秒后再重试：有的根本不重试，有的则不遵守延迟便重试了。。具体而言，OpenAI 在十秒延迟结束前就重试了 503，但没有重试 429；Anthropic 两者都没有重试。xAI 发送了其一贯的重叠请求，因此没有明显的重试行为。[9]

[8]我们通过测试页面 URL 中的标记来识别供应商，但无法将该标记添加到 / robots.txt，因为抓取器会自行请求该文件；因此，xAI 类似浏览器的用户代理字符串无法让我们判断此类请求来自谁。[9]我们用 curl 确认，Retry-After 已通过 Cloudflare 传送给外部客户端。但我们无法核实各供应商的抓取器实际收到了什么。

xAI 发送重叠请求。 xAI 即使面对 404 或空的成功网页，也会发出四到七个请求；当响应被截断时，则会发出 12–28 个请求（表 3、表 12）。这些请求在时间上相互重叠，因此属于并行抓取，而不是重试。

Anthropic 几乎不重试；OpenAI 只对服务器错误重试。 Anthropic 在几乎所有情况下都只发送一个请求，因此一次失败就会终止查询。发生 500 错误后，OpenAI 会发送 3 到 8 个请求；但发生 403 或 404 错误后只发送一个请求（更多案例见 § D.2）。失败是否会在不同查询之间被记住，将在 § 3.3.4 中测试。

重定向循环。 面对无尽的重定向，OpenAI 跟随了一次重定向，Anthropic 跟随了九次，而 xAI 则并行发起了 57–77 次跳转。

一次失败的抓取给调用方造成的成本几乎与成功抓取相当。 对于同一个 2KB 网页，三家供应商的一次失败抓取所使用的输入词元数量都与成功抓取大致相同（图 9、§ D.7）。从服务外部无法看出失败抓取的成本由何而来。

3.2 已读取：内容处理与成本

抓取请求到达源站，并不意味着服务会在答案中使用该网页。我们测试抓取后哪些格式、脚本生成的文本和页面大小能够保留下来，以及每种情况会消耗多少输入词元。

3.2.1 文档格式。服务可能会抓取文档，却不在答案中使用其内容。我们测试服务能够读取哪些文档格式：每个文件都包含代码，我们会检查答案是否复述这些代码。每次试验都会使用一种八种格式之一的新测试文件：HTML、纯文本、Markdown、JSON、XML、CSV、PDF 或 DOCX，文件中包含五个代码（§ C.4）。我们记录服务是否请求了该文件、答案中出现了哪些代码，以及服务是否表示无法读取该格式。

更改格式也会改变文件的字节数，包括内容和长度。

为单独考察 Content-Type 标头的影响，我们将一个字节完全相同的 2KB 正文以五种不同的值提供，并要求服务找出其中的代码。

表 4 显示了各服务在答案中使用了哪些文档。每个服务都抓取了所有文档，且两种源站上的结果相同。

OpenAI 拒绝了八种格式中的四种。 OpenAI 对来自 HTML、纯文本、JSON 和 PDF 文档的全部五个请求值作出了答案。对于 Markdown、XML、CSV 和 DOCX，它则报告了不支持的内容类型。Markdown 是一个值得注意的案例：Mark- ItDown [47] 和 Docling [5] 等文档摄取工具会特意将源文档转换为 Markdown，以供大语言模型使用。Anthropic 和 xAI

表 4：人工智能抓取器会使用哪些文档？每个抓取器都从我们的源站检索了所有文件格式，但并非所有格式都用于答案。✓：答案返回了全部五个请求的网页代码；×：该服务报告了不支持的内容类型。

	OpenAI	Anthropic	xAI
（*a*）一个源站，八种格式			
HTML、纯文本、JSON、PDF	✓	✓	✓
Markdown、XML、CSV、DOCX	×	✓	✓
（*b*）一个正文，五种 `Content-Type`			
text/html、text/plain	✓	✓	✓
text/markdown、application/xml、text/csv	×	✓	✓

对全部八种格式都作出了答案。OpenAI 的抓取器仍然下载了被拒绝的四种格式，因此拒绝发生在抓取之后，而不是之前。

OpenAI 根据`Content-Type` 标头做出判断，而不是根据内容。格式测试同时更改了正文内容和标头，因此无法判断究竟是哪一项决定了 OpenAI 是否会读取文档。因此，我们将同一份 2KB 正文（逐字节完全相同）置于五种不同的 Content-Type 标头值下提供。OpenAI 在 text/html 和 text/plain 类型下给出了答案，但在 text/markdown、application/xml 和 text/csv 类型下报告了不支持的内容类型。Anthropic 和 xAI 在全部五种类型下都给出了答案。在字节内容固定的情况下，只有标头决定了 OpenAI 是否会使用该网页。

请求标头接受所有格式。 Anthropic 发送了 Accept: */*；OpenAI 和 xAI 发送了带通配符回退的浏览器风格列表。OpenAI 的列表包含 application/xml，尽管它拒绝了 XML 响应。

在公共网站上。 公开文件也呈现出相同的模式：OpenAI 和 Google 会拒绝部分 XML 文件，而 Anthropic 和 xAI 会读取这些文件（§ D.5）。OpenAI 拒绝了 98% 以 application/xml 或 text/xml 提供的 XML 文件，却读取了同类网站上 95% 的纯文本文件；其回答引用了抓取器的“Unsupported content-type”错误。Google 拒绝了所有 application/xml 文件，但会返回大多数 text/xml 文件的内容，且错误信息较为笼统；[10] 由于两者都是 XML，最可能的触发因素是 Content-Type 标头，不过 Google 并未说明这一点。Anthropic 拒绝纯文本的频率（19%）与拒绝 XML（14–19%）相近，并将原因归咎于网站的域名，而非格式。xAI 很少拒绝文件，但返回的 text/xml 文件中有 47% 被压平成一行。

3.2.2 依赖 *JavaScript* 的内容。许多网站使用客户端 JavaScript 插入主要文本，因此，不运行脚本的抓取器

[10]每次拒绝都会携带 URL_RETRIEVAL_STATUS_ERROR，每次成功读取 URL_RETRIEVAL_STATUS_SUCCESS。错误值没有说明原因；但它确实排除了 Google 针对不安全内容或付费墙内容设置的独立状态。此处仅纳入 Google：它不在受控测试范围内（§ 2）。

表 5：人工智能抓取器能看到 JavaScript 放到网页上的文本吗？对于所有三种服务，回答中出现的始终只有纯 HTML 代码。

版本	其他代码所在位置 代码所在位置	可读于 页面源代码?	答案中?
纯 HTML（基准）	HTML 正文文本	Yes	Yes
内联脚本	混杂在内联脚本	No	No
外部脚本	混杂在 .js 文件中	No	No
数据块	JSON 块，复制粘贴由脚本	Yes	**No**
JSON-LD	JSON-LD 区块，从不显示	Yes	**No**

看到的网页与浏览器用户看到的不同。Ver- cel 报告称，大多数人工智能爬虫不会渲染 JavaScript [68]；我们直接测试用户触发的抓取器，使用的测试页面会通过脚本隐藏部分代码。我们在 § C.4 中介绍这些测试页面和运行次数。

表 5 列出了五种版本。纯 HTML 版本是我们的基准；在这种情况下，三个服务在每次测试中都返回了全部五个代码，因此网页本身是可读的。

首先，抓取器会运行脚本吗？在两个版本的网页中，代码被打乱后嵌在脚本里（表 5），因此只有运行网页的解码器才能读取这些代码。

没有任何服务返回这些内容。没有任何服务请求导入的 .js 文件，包括 xAI；它针对每个问题发送了五到七次页面请求，使用浏览器用户代理（第3.1.1节）。我们没有发现任何抓取器会运行网页脚本的迹象。

其次，它们至少会读取 script 标签内的文本吗？在数据块和 JSON-LD 版本中，代码以纯文本形式存在于 HTML 中，但没有任何服务返回这些内容。这表明抓取器会在网页到达模型之前移除 <script> 标签的内容，这也是 HTML 转文本过程中常见的做法；这是我们根据回答作出的推断，无法直接观察处理流程（§ 4.3）。这也与 Vercel 的说法不同：初始 HTML 响应中的 JSON 数据仍可能被索引 [68]。Anthropic 的抓取工具配合动态筛选（§ 3.2.4）也无济于事；在这种方式下，模型会编写程序来分析抓取到的网页。

因此，网站仅通过 JavaScript 或脚本块（例如 JSON-LD）提供的文本，对这三种抓取器都是不可见的。我们检查的热门网站主页中，约有 40% 嵌入了 JSON-LD，因此许多回答可能缺少这类内容，尽管我们没有在公共页面上对此进行测量。

3.2.3 文档大小。抓取长网页，并不意味着整个网页的内容都影响了答案，也无法体现其成本。我们提供大小从 4 KB 到 2MB 的网页，其中包含位于已知深度的代码（§ 2.3），并记录答案报告了哪些代码，以及 API 收取了多少输入词元。请注意，没有抓取器发送 Range 请求头，尽管我们的源站支持字节范围：每个请求都要求获取整个网页。

xAI 只读取前半兆字节。 OpenAI 和 Anthropic 在不超过 2 MB 的所有大小下都返回了每一段代码。xAI 在 512 千字节以内也如此，但对于 1 MB 和 2 MB 的页面，在两个源站的每次试验中，它都只返回前半兆字节内的代码。两个源站在这些请求中的每一次都发送了完整页面，因此这一限制适用于 xAI 已读取的内容，而不是它下载的内容。由于 xAI 会针对每个问题发送多个请求（第3.1.1节），对于一个答案只使用前半兆字节内容的情况，2 MB 的页面仍会被多次提供。

只有 Anthropic 的输入量会随网页大小增长。Anthropic 报告的输入令牌数量大致随网页大小持续增长，在 2 MB 时达到约 760K 令牌，且没有出现平台期。[11]OpenAI 和 xAI 的输入令牌数量在较小网页上有所上升，之后在网页大小达到 2 MB 前一直处于较宽的区间内（图 2a）。这两种受限服务提供的内容有所不同：OpenAI 以受限的成本返回了全部五段代码，而 xAI 输入量受限的同时，约 0.5 MB 之后的内容也随之丢失。

在较小的网页上，排序正好相反：对于 4KB 网页，OpenAI 报告的输入量是其他服务的两倍多，因为它的大部分输入并非网页本身（约为我们估算的网页自身令牌数的九倍，而 Anthropic 和 xAI 为四倍），而是搜索和工具上下文 [55]。这是端到端的差异，并不意味着 OpenAI 读取网页的方式有所不同。[12] 这些运行使用了基础抓取工具；§ 3.2.4 展示了采用动态过滤的直连路径会增加多少成本。

3.2.4 动态过滤与输入成本。检索工具可能会在模型作答前处理文档，而源站日志无法显示这一过程。我们在相同的页面、模型和问题上比较 Anthropic 抓取工具的两个版本：基础版，以及带有动态过滤功能的新版本，后者可以执行代码来筛选抓取的内容 [3].[13] 我们比较 API 报告的输入词元、工具调用轨迹和回答。

动态过滤增加了这些页面的输入量。动态过滤旨在从抓取的文档中挑选相关内容，以减少输入量。然而，在我们的 8KB 页面上，带过滤功能的版本在全部八种格式中的输入词元数，约为基础版本的 8 至 22 倍（图 2b）。启用动态过滤后，多次运行的输入词元数也会出现很大波动；未启用时则保持稳定。

HTML 对比显示，随着页面大小增大，这一增长仍然存在（图 2c）；一次 128KB 调用报告了约 200 万个输入词元。[14]

[11]这些运行中，我们没有设置 Anthropic 的最大_内容_令牌数限制 [3]。[12]我们根据 Anthropic 的 2MB 运行估算页面内容的词元数。由于 OpenAI 和 xAI 使用的分词器不同，其比率为近似值。[13]所用版本为 web_fetch_20250910（基础版）和 web_fetch_20260209（动态过滤版，仅限直接 API）。[14]这一次调用的费用约为 4 美元，因此我们没有测试更大的页面；我们测量的公共主页中位数为 158 KB。

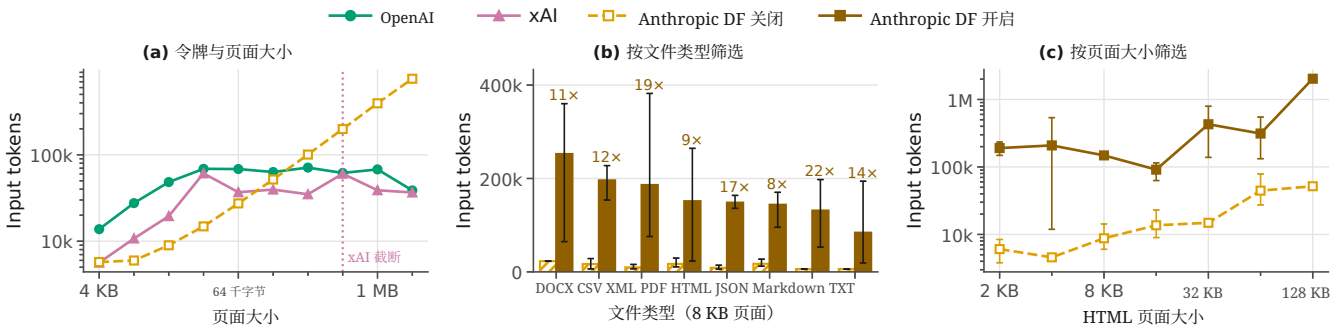


图 2：输入词元（提供给模型的网页内容）。(a) 词元数与页面大小；虚线表示 xAI 停止读取的位置（~500千字节）。(b)–(c) Anthropic 开启或关闭动态过滤（DF）时，按文件类型统计 8 KB 页面的数据，以及按 HTML 页面大小统计的数据。数据点和柱形表示最多 3 次运行的均值，误差线表示最小值和最大值。每个开启 DF 的柱形上方的标签，是其均值除以关闭 DF 时的均值所得的结果。

过滤调用运行了代码（其追踪记录中包含代码_执行块，而基本调用从未出现过这种情况），但没有任何收益：两个版本都回答了关于最深层代码的问题，而且过滤使用了更多输入词元。过滤旨在丢弃无关内容，而我们的问题需要从整个网页中获取代码，因此几乎没有内容可供丢弃。我们没有测试大部分内容与问题无关的网页；在这种情况下，过滤或许能节省词元。

3.3 复用：缓存与共享

我们想了解为某个查询抓取的信息之后是否会被复用：由谁复用、复用多久，以及这种复用是否不顾源站的指示。每项测试分为三个步骤。首先，发出一个预热问题，让服务抓取我们提供的网页。然后，经过一段时间后，源站保留该网页、修改该网页，或使其无法访问。最后，发出一个重复问题，索取只有重新抓取才能返回的代码；源站日志则会显示是否发生了重新抓取（§ C.6）。

3.3.1 调用方之间的复用与共享。为一个调用方抓取的页面会为另一个调用方提供服务吗？预热问题抓取页面后，我们从另一个调用方提出重复问题：使用不同的模型、账户或应用登录。重复问题要求在同一页面上查找不同的代码，因此只有页面内容（而非重放第一个答案）才能回答这个问题。表 6 展示了每种调用方变更下，服务是否复用了页面。API 和应用试验使用了不同的页面，因此无法检验二者之间的复用。

在 API 调用方之间复用。 OpenAI、Anthropic 和 xAI 在每次完成的试验中，都在模型和账户之间共享已读取的网页内容（表 6）。[15] Gemini 是例外：它在每个问题中都会重新抓取网页，即使几分钟后仍由同一调用方发起，因此没有可供共享的副本。这与 Google 的文档一致，其中说明其工具会在网页已被编入索引时从索引缓存中读取网页，否则会实时抓取网页 [25]；我们的网页都是新的。

消费者应用呈现出较为有限且不太一致的模式。 ChatGPT 和 Claude 在三次测试中都在新聊天中复用了页面，并在不同登录账户之间共享了页面。Grok 的表现不一致：三次测试中有一次复用了页面，两次在不同登录账户之间共享了页面，其余情况下则重新抓取。完整的八款应用比较见 § E.3。

3.3.2 Cache-Control 和条件请求。 HTTP 源站可以通过 Cache-Control 和 Expires [20] 指定何时可以复用响应。Last- Modified 等验证器可让客户端检查保留的响应是否发生变化 [21]。我们测试这些信号是否会影响服务复用页面或向我们的源站请求页面的频率。我们以 19 种标头设置分别提供同一个 2KB 页面，并在 2.5 至 80 分钟后再次询问该页面（§ E.4）。表 7 总结了结果。

除了 Anthropic 的`no-store`之外，标头设置不会影响复用。 无论标头设置如何，OpenAI 都会从存储的副本中回答约 30 分钟，之后才会再次抓取页面。在我们的测试期间，xAI 从未再次抓取页面，即使页面标记为 no-store 或 no-cache 也是如此。直到 10月15日，Anthropic 也忽略了标头设置。从 10月17日开始，在每次运行和两条路由中，它都会在每次提问时重新抓取 no-store 页面，但仍会复用标记为 private 或 no-cache 的页面，或未设置指令的页面（表 7）。

没有 API 抓取器重新验证。 在两个源站上，我们都没有观察到三个 API 抓取器发送 If-Modified-Since 或 If-None-Match 请求：每当它们再次请求时，都会请求整个网页。[16]

对于公共页面，Anthropic 的副本存续时间超过了标头规定的期限。 对于限制复用的页面，其回答中有 76% 来自比标头允许的时间更早的副本。即使是 no-store 页面，也只有 42% 遵循了这一规定；其余回答则来自

[15]账户比较将直接供应商账户与使用相同模型的 Open-Router 账户配对。

[16]在消费者应用中，只有 Mistral 发送了条件请求，即一个 If-Modified-Since 请求，而我们的源站以 304 未修改作出响应。

表 6：后续问题是否从某个网页获得答案，而该网页是由较早的问题促使服务抓取的，且未向我们的源站发起新请求？是：每次试验中均如此（每个单元格进行 3–6 次试验）；否：一次也没有；否则为试验所占比例。—：未进行测试（大多数应用不允许我们选择模型，且我们没有改变应用的位置）。“后续调用者”是提出第二个问题的一方。Gemini API：gemini-2.5-flash（§ D.6）。

	API				消费者应用			
后续调用者是……	OpenAI	Anthropic	xAI	Gemini	ChatGPT	Claude	Grok	Gemini
同一账户（新的查询或聊天）	Yes	Yes	Yes	No	Yes	Yes	第 1 页，共 3 页	Yes
同一账户 , 不同的模型	Yes	Yes	Yes	—	—	—	—	—
不同的客户	Yes	Yes	Yes	—	Yes	Yes	第 2 个，共 3 个	No
不同的客户 , 从美国拨打	Yes	Yes	Yes	—	—	—	—	—

表 7：在最多 80 分钟后再次提问时，答案是否来自已存储的副本，且没有发出源站请求？各列显示网页的 Cache-Control 设置。OpenAI 和 xAI 在每次测试及全部 19 种设置下的表现都完全一致（表18）。无论采用何种设置（包括未设置），OpenAI 都会在约 30 分钟后重新抓取网页，因此这并非对标头的响应。或者， =通过 Open-Router, D = 直连。

		已复用?			
Run	路由	no-store	private	不缓存	无
Anthropic					
10月7日	OR	Yes	Yes	Yes	Yes
10月15日	D	Yes	Yes	Yes	Yes
10月17日至21日，运行6次 D (5)，OR(1)		No	Yes	Yes	Yes
OpenAI		Yes	Yes	Yes	Yes
xAI		Yes	Yes	Yes	Yes

副本已过期一天或更久。[17] 标记为 private 的页面不应由用户之间共享的缓存存储，但其中 81% 的情况下仍返回了旧副本；Anthropic 则会在调用方之间共享存储的副本（§ 3.3.1）。

3.3.3 保留与更新。 如果没有条件验证（§ 3.3.2），服务可能会在源站内容变更后，仍基于旧副本继续回答。我们首先测量更正何时会在源站请求和答案中显现。然后，我们比较被反复查询的页面与未被触及的页面，以检验使用是否会推迟重新访问源站。

页面更新的可见性。 我们询问一个新鲜的 HTML 或纯文本页面，在同一 URL 更新该页面后，经过一段延迟再度询问。OpenAI 从约 30 分钟后开始给出更新后的答案，而 Anthropic 和 xAI 则持续一天或更久给出过期答案（图 3a）。值得注意的是，即使 OpenAI 已抓取更新后的页面，其后续回答中仍有约 60% 已过期；原因将在 § E.8 中探讨。因此，源站日志中出现抓取记录，并不意味着答案使用了该页面。

提问并不会延长副本的保存时间。多次询问某个网页，并不会让它的副本保存得更久：OpenAI 和 xAI 在同一次最终检查时，它们再次下载了该网页，和一直未被访问的网页相同（§ E.5）。

在公共页面上，Anthropic 给出的答案来自几小时前或几天前的副本。 为了了解我们在自有页面上测得的复用现象是否也会出现在真实网站上，我们询问了约 90 个经常更新的主页（详见 § E.6）。Anthropic 的抓取工具会报告抓取每个页面的时间，因此我们可以直接读取其副本的年龄。Anthropic 对常规 URL 的回答中，有[18] 65%来自一小时以上前的副本，7%来自一周以上前的副本。答案一致：Anthropic 使用了过时副本来作答（$p < 0.001$）。

公共端点与我们自己的源站结果一致。

判断网页文本是否陈旧并不容易，因为服务可能引用不同的部分，也可能改述内容。值只会递增的公共端点则不会有这些干扰：答案是一个数字，因此可以直接测量它与实时值相差多少，而且我们无需为每个页面设计不同的问题。我们每隔 45 分钟向各项服务查询一次此类端点的当前值，这些端点每隔几秒到几分钟就会变化一次，持续约三天。我们将每个答案与参考日志进行了比较；该日志每五分钟记录一次端点的值。

图 4 展示了每种标头设置对应的一个代表性端点（§ E.7 展示了全部八个）。Google 几乎总是最新的。OpenAI 的回答通常比参考日志中的值晚约一小时。

xAI 的回答晚约八小时，Anthropic 的则晚约十一小时，这与副本在我们的源站上最长保留约一天的情况一致（图 3）。

该标头没有带来明显差异，正如我们在自己的源站上观察到的那样：标记为 no-store、no-cache 或 max-age=60 的端点，并不比没有标头的端点更新鲜。值得注意的是，我们还检查了 Anthropic 如何处理 NIST 信标上的 no-store，结果与我们在源站上的观察只有部分一致。它的

[17]有一个设置了 no-store 的网页，给出的答案来自一份近六天前的副本；另一个设置了最大有效期=60 的网页，答案则来自一份一天前的副本。

[18]回复中包含每个抓取网页的检索_时间以及检索到的确切内容；抓取失败时则会返回错误代码，例如 url_not_- allowed 或 url_not_accessible。我们运行的其他三项服务均未返回获取时间（通过 OpenRouter 接入 OpenAI 和 xAI，通过其 API 接入 Google）。

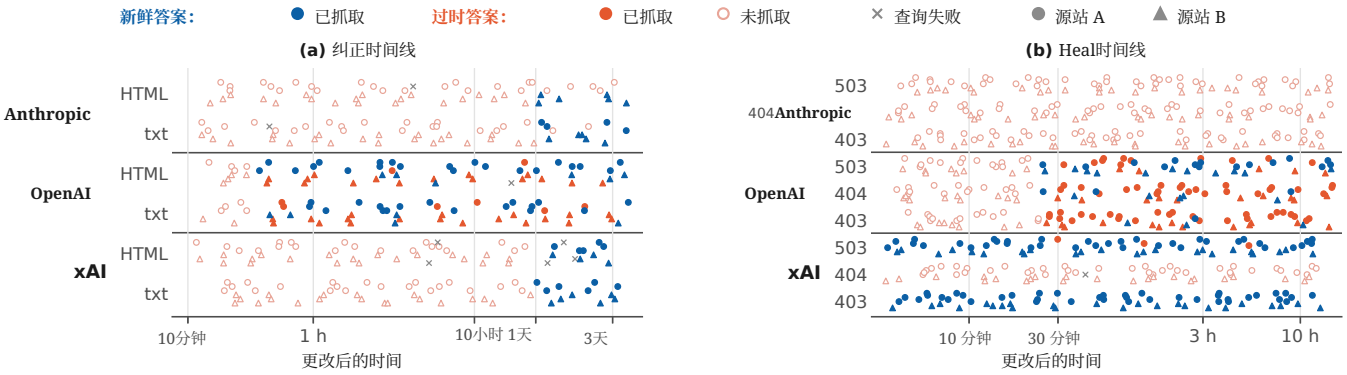


图 3：某项更改后，服务持续从过期状态提供答案的时间。每个标记代表一个后续问题，其位置按自（a）页面更新或（b）错误页面修复以来经过的时间确定。陈旧表示答案仍显示过期页面（a）或错误（b）；已抓取表示我们的源站收到了该问题的请求。时间轴为对数刻度。

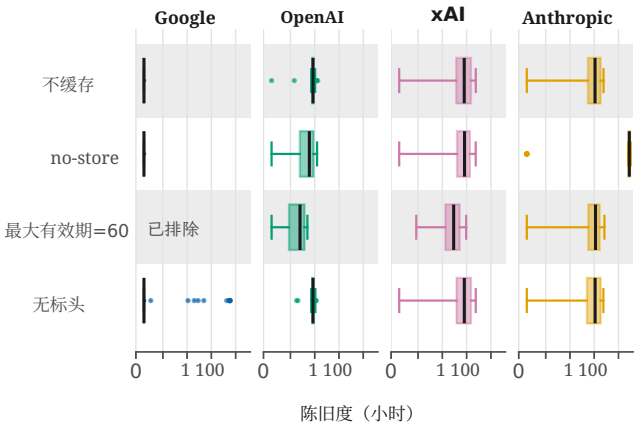


图 4：各服务的答案比实时值落后多少。每种缓存控制设置各选取一个公共端点：Hacker News（不缓存）、NIST 信标（no-store）、USGS 地震数据（最大有效期=60）和以太坊高度（无标头）。由于 Google 错误转录了该值，因此不计入其 USGS 回答。

答案要么是最新的，要么来自非常旧的副本，中间没有任何情况；§E.7探讨了原因。

3.3.4 从负缓存中恢复。负缓存已在 DNS 中确立：解析器可以保留一次失败的查找结果，在无需询问权威服务器的情况下回答后续查询 [2]。我们要探究的是，在源站恢复后，服务是否也会对失败的网页获取采取相同做法。每次试验都从一个新 URL 开始，该 URL 在第一次查询期间始终返回错误（503、404 或 403）；查询结束后，我们将该 URL 切换为一个带标记的 200 页面，等待一段延迟后再次发起查询。记住失败的服务会直接返回错误且不发送请求；不记住失败的服务则会抓取该网页并返回标记。在一开始就可访问、并在相同网页存续时间下发起请求的网页，则能表明服务是否会在该延迟后复用普通副本。

这能表明服务是否会在该延迟后复用普通副本。

各服务对失败的记忆方式各不相同。在整个观察窗口内（最长约 11 小时），Anthropic 对于这三种错误都没有重新抓取恢复后的网页，其回答仍然称网页无法访问。最初约 25 分钟内，OpenAI 没有发送请求；此后它重新抓取了恢复后的网页，但在 70% 的此类试验中，答案仍引用最初的错误。对于 503 和 403，xAI 及时读取了修复后的内容；但对于 404，它的表现与 Anthropic 相同，无论延迟多久都没有发出新请求（图 3b）。

4 讨论

4.1 人工智能时代的 HTTP 机制

我们的发现表明，人工智能搜索代理并不遵循 HTTP 对浏览器和爬虫的假设。我们将讨论哪些现有机制仍然适用，以及检索栈可能需要哪些新机制。

只有供应商选择遵守，对照信号才会起作用。缓存指令仍是发布者声明页面可如何复用的正确方式，而且已有一家供应商遵循了其中一部分：从 2026年10月中旬起，Anthropic 会遵守 no-store，在每次收到提问时重新抓取这类页面（第 3.3.2节）。在同一角色范围内（例如复用），现有指令依然适用；要遵循该指令，需要供应商改变政策，而非采用新协议。但遵守情况并不全面。Anthropic 会在我们的页面上遵守 no-store，但在公共端点上并不始终如此（§ E.7），而且仍会复用标记为 no-cache 的页面；OpenAI 和 xAI 则不遵循任何指令。我们测试的其他信号也好不到哪去：OpenAI 和 xAI 忽略了 Retry-After，只有 Anthropic 请求了 robots.txt（§ 3.1.2、§ 3.1.1）。对于 robots.txt，标准定义了适用于爬虫的规则，并将其称为自动化客户端，却没有说明由用户触发的抓取

行为应如何处理 [38]。这些规范都没有要求供应商遵循建议性信号，因此发布者只能放弃对供应商如何抓取和复用其页面的控制权。

指令描述的是源站的网页，而非服务所持有的副本。 HTTP 的复用规则假设缓存保存的是源站提供的对象：实体标签和 Last-Modified 用于检查对象是否发生变化，而 Cache-Control 则规定其有效时长。服务会在模型读取页面之前对其进行处理：如果网页只通过脚本提供文本，就没有服务会读取这些文本（§ 3.2.2）；Anthropic 的动态过滤则允许模型在抓取到的网页上运行代码（§ 3.2.4）。新闻文章上的最大有效期`=600` 仅适用于原始 HTML，而不适用于服务从中提取的文本；并且没有抓取器发送条件请求（§ 3.3.2）。我们无法看到服务存储了什么，因此处理后的副本是否就是它复用的内容，仍无从确定。

发布者无法辨别是谁在抓取。 源站通过客户端地址和用户代理识别客户端，而这两者都由客户端自行声明。OpenAI 和 Anthropic 会标明其抓取器，而 xAI 的文档没有说明这一点 [66]，其请求却携带类似浏览器的用户代理，并来自一千多个网络中的许多源地址（§ 3.1.1, §D.1）。运营者无法仅凭地址或身份识别出这样的抓取器，而根据声明的用户代理来归因流量的日志研究 [35]也会漏掉它。IETF 正在开展的工作提议用密码学签名而非地址来绑定代理的身份 [46], ，这将使源站能够针对每个代理应用策略。上文任何强制实施对照的机制，都需要先做到这一点。

可能需要哪些新机制。现有的 HTTP 机制尚未覆盖两个空白领域。

请求无法说明答案在多大程度上依赖于该请求。人工智能搜索代理会增加源站收到的请求数量，因为一个用户问题可能触发多次抓取 [60]。其中一些请求用于探索，通过跟随链接收集上下文；另一些则抓取答案所使用的网页。

对源站而言，这些请求看起来都一样，因此当源站负载过高时，无法优先处理所需的请求，并放慢其余请求。一个用于标记请求是必要请求还是探索性请求的轻量级标头即可实现这一点。

HTTP 缓存会存储整个网页，而检索会复用部分文本。 HTTP 缓存会将响应存储在对应的 URL 下，并将其整体视为陈旧。检索系统会把页面拆分成文本块，并复用与问题相关的文本块 [33], ，这些文本块可能来自多个源站的页面。此类系统可能会直接根据存储的文本块生成答案，而无需请求网页；但在 HTTP 中，只要有一个句子发生变化，整个 URL 就会变为陈旧。如何将新鲜度规则应用于文本块，以及当一个答案混合了来自多个源站的文本块时该如何处理，仍是一个待解决的问题。

4.2 供应商对抓取怨声载道，用户和出版商却没有

供应商可以看到抓取获取了什么、何时获取以及花费多少。用户和出版商对此知之甚少，也无法针对自己看不到的情况作出回应。

用户无法看到一次抓取的成本。输入成本取决于服务，而不仅仅取决于网页（§ 3.2.3）。OpenAI 的成本在不超过 2 MB 时保持在一个区间内；xAI 只读取最前面的 0.5MB，丢弃其余内容却不作说明；Anthropic 的成本则随网页大小增长，对于 2MB 的网页约为 76万词元。Anthropic 提供了大小限制 [3], ，但用户无法预先判断对于自己的问题，什么样的大小才合适。失败获取也会产生费用：对于 2 KB 的网页，一次抓取失败的调用所使用的输入词元，大约与一次抓取成功的调用一样多（§ D.7），因此它增加了账单，却没有为答案增加任何内容。

用户无法得知答案有多旧。答案中没有任何信息能表明它来自实时抓取还是缓存副本。更新内容在 30 分钟（OpenAI）到一天或更久（Anthropic、xAI）后才出现在回答中（§ 3.3.3）；而且，OpenAI 有时虽然已经抓取了新版本，却仍会根据过期版本作答（§ E.8）。只有 Anthropic 的 API 会报告检索时间，并允许开发者请求新鲜抓取 [3]；我们发现，OpenAI 和 xAI 都不提供这两项功能，而且这些服务都不会向终端用户显示相关信息。从外部观察，我们只能看到副本会保留多久，却无法判断是固定时限还是淘汰机制决定了副本何时失效；无论如何，答案都没有标明自身的新旧程度。

发布者无法得知其网页如何被使用。一次抓取只会让发布者看到一个请求，却看不到它将用于生成多少个答案：缓存副本会在多个账户之间共享（§ 3.3.1），之后每次使用都不会产生新的请求。因此，源站日志统计的是抓取次数，而非使用次数；所以，任何依赖计数的机制（例如按次抓取付费）都只能看到部分使用情况 [63]。

出版商不知道人工智能智能体需要哪种格式。目前尚无公认的、便于人工智能智能体使用的网页内容格式。智能体有两种选择：如果想获取人类看到的页面，就应运行脚本并进行交互式浏览；如果想获取内容，就应直接读取文件，例如 Markdown 格式。各项服务都无法完全做到这两点。有些检索栈会像人一样浏览网页 [53], ，但没有服务会运行 JavaScript（§ 3.2.2）；而 OpenAI 拒绝使用 Markdown（§ 3.2.1），尽管这种格式通常被推荐用于大语言模型输入 [5, 47]。因此，发布者无法判断应该针对哪种形式进行优化。

4.3 局限性

受成本限制的参数空间。 我们的实验产生了约 400 美元的 API 费用，这限制了我们能够探索的参数空间。尤其是，对象大小超过 128 千字节时，令牌成本最高，因此我们仅对这些大小进行了少量测试

。现实世界中的 HTML 页面经常超过这一阈值，因此，对于大小分布的长尾部分，我们的成本估算偏保守。

两个受控源站。有些实验需要受控源站：在第三方网站上无法植入代码、设置任意 Cache-Control 指令或改变响应大小。我们使用两个源站，其中一个位于绕过缓存的内容分发网络之后，另一个由服务器直连提供；两者的结果一致，但它们都是我们新建的域名。代理可能会根据域名声誉、位置或托管服务提供商，对成熟网站采取不同策略，而这两个源站无法体现这些差异。我们通过观察公共页面和端点来补充受控测量。

公共页面验证。在第三方页面上，我们只能看到答案，看不到其背后的过程：既没有请求日志，也没有提取出的文本。因此，过时答案表明使用了过期版本，但无法说明它来自何处。抓取器收到的网页版本也可能与我们的轮询服务器收到的不同（因地区、语言或机器人规则而异），或者是来自网站自身缓存的较旧副本。不同服务将网页转换为文本的方式各不相同，引用网页内容的情况也不尽相同，因此我们按结果为每个答案评分，并报告各服务的读取率。这些网站都是允许所有四种抓取器访问的热门页面，而过时答案只能作为复用情况的下限。因此，我们使用这些页面验证受控实验结果，而非估算读取率。

供应商覆盖范围与黑箱视角。 我们研究了三家商业 API 供应商（Anthropic、OpenAI、xAI），以及在公共页面上的 Google 和附录中的若干消费者应用；其他供应商的行为可能有所不同。行为也会随时间变化：在研究期间，Anthropic 改变做法，开始再次抓取我们源站上设置了 no-store 的页面，这表明在测量期间，发现就可能过时，因此我们的结果只是一个快照。我们能看到源站的请求日志和客户端收到的 API 回复，却看不到中间发生了什么。动态过滤结果是根据令牌计数和工具调用轨迹推断出来的；分块、存储和去重等内部步骤无法观察，而我们对新抓取后仍返回旧答案的解释也只是猜测（§ E.8）。出于同样的原因，我们无法解释 Anthropic 的抓取工具为何复用了 NIST 信标的旧副本（该信标发送 no-store），却再次抓取了我们自己的 no-store 页面（§ E.7）。我们还测量了服务会保留网页多久，以及会与谁共享，但没有测量复用发生的频率；这取决于用户对网页的关注度（网页的关注度通常呈经典的偏态分布 [11]），而且只有供应商才能看到这一点。

我们未测试的内容。 我们的网页是纯文本，问题要求提供一个代码，而简单的字符串匹配就能找到这个代码，因此检查结果不取决于模型对网页的理解程度。图像和视频需要内容更丰富的网页，而检查有关图像或视频的回答，则取决于模型能否读取它们，这本身就会带来额外的变异性。我们也未测试服务如何处理广告，这是一个很自然的后续研究方向。这意味着要向抓取器投放广告；由于服务会保存副本并在不同账户间共享，这些广告可能会出现在其他用户的回答中。

5 相关工作

机器人与爬虫测量。 Kim 等人发现，随着 robots.txt 指令变得更加严格，网页抓取器的遵守程度会降低，而 AI 搜索爬虫几乎从不检查该文件 [35]。Cui 等人研究出版商如何为大语言模型机器人编写 robots.txt 规则，并报告了机器人未遵守这些规则的案例 [17]。Sayyid 等人审计 ChatGPT 和 Claude 能对网页执行的操作，以及由此产生的流量：一个问题可能触发 10–40 次抓取，并且抓取的页面会从同一地址复用约 10–15 分钟 [60]。不同于对周期性爬虫及出版商用于应对它们的对照措施的研究 [17, 35, 42], ，也不同于对答案引擎能够对网页执行的操作的研究 [60]，我们研究的是按需抓取本身，而不仅仅是它是否遵守 robots.txt。我们在源服务器上通过请求日志对其进行测量：抓取器请求什么、会保留并共享所抓取内容多长时间、如何处理失败，以及每次抓取的成本。

AI 流量对 *Web* 基础设施的影响。Zhang 等人将受控的 AI 工作负载注入 Varnish 部署环境，发现当 AI 流量达到请求总量的 25% 时，缓存未命中率会翻倍；达到 100% 时则会翻三倍 [70]。维基媒体基金会报告称，人工智能机器人约占页面浏览量的 35%，但占资源密集型请求的比例超过 65%；它们偏好访问热门程度较低的页面，从而绕过区域缓存，直达源数据中心 [65]。近期提案建议，让代理检索出版商按照自身政策开放的段落，而非整页内容 [13, 28]。这些研究揭示了负载压力及一种缓解方法，却没有解释哪些代理行为导致了负载。我们补上了这一缺失环节：通过被动式服务器端测量，将源站压力归因于具体的检索特征。

网络代理。 网络代理长期以来一直是一个重要的研究领域。由许多用户共享的缓存代理可以降低延迟和负载 [20], ，而转码代理或节流代理会压缩并修改页面，以节省用户的数据流量 [1, 41, 62]。它们也引发了担忧：测量结果发现，这些代理会移除内容和广告、削弱安全性，而且默认应用 Web Light，因此发布者只能通过设置标头来选择退出 [36, 62]。共享缓存还曾因将不同 URL 视为同一页面，而把一名用户的 private 页面泄露给另一名用户，成为

同一页面 [48]；我们针对带有个性化链接的人工智能搜索代理进行了测试，未发现泄露。人工智能搜索代理兼具这两种角色，但有三个不同之处：用户接收的是答案而不是网页，因此用户和发布者都无法看到哪些内容发生了变化；经过处理的页面会在不同账户之间复用；而且其行为取决于非确定性模型，因此可能因运行次数而异。我们在源服务器处测量发布者所依赖的控制措施（例如 Cache-Control）是否得到遵守和执行。

检索栈中的缓存。大多数检索增强系统会搜索预先索引的静态语料库，其缓存依据嵌入相似度而非 URL 来决定是否复用 [8, 33, 40]。面向开放网络的研究则会考虑新鲜度：只有在存储的答案或网页很可能仍然有效时才复用，否则就重新抓取页面，并仅对发生变化的内容重新索引 [44, 69]。在推理时从实时网络检索的系统，则着重于从代理一侧完成任务并保证答案质量 [50]。这些研究为代理设计或提出缓存方案。我们测量的是商业服务已经在使用的缓存：答案是来自抓取还是存储副本，该副本会保留多久，以及服务会遵循哪些 Cache-Control 信号；这些情况均在源服务器上进行观测。

6 结论

我们使用受控源站和公共页面，测量了人工智能搜索代理如何检索网络内容。各项服务会在不同账户之间共享抓取的页面，而且大多忽略源站的缓存规则；它们的抓取器在是否遵守重试延迟、是否请求 robots.txt，以及是否读取特定格式、脚本和长页面方面也各不相同。因此，发布者无法预测内容变更何时会传达给用户，用户也无法了解答案有多过时或多完整。人工智能检索是一种新的流量类别：它像浏览器一样按需抓取，像转码代理一样改写页面，又像共享缓存一样复用内容，而 HTTP 提供给发布者的控制手段尚未对其形成约束。我们的研究考察的是这些系统的行为，而非其安全性；下一步自然是测试抓取器的安全属性，例如在相同的受控源站环境下进行 TLS 验证[36], 。

参考资料

[1] Victor Agababov、Michael Buettner、Victor Chudnovsky、Mark Co- gan、Ben Greenstein、Shane McDaniel、Michael Piatek、Colin Scott、Matt Welsh 和 Bolian Yin。2015年。Flywheel：Google 面向移动网络的数据压缩代理。收录于*USENIX* 网络系统设计与实现研讨会论文集（*NSDI*）。367– 380。https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/agababov [2] Mark Andrews。1998年。*DNS* 查询的负缓存（*DNS NCACHE*）。RFC 2308。互联网工程任务组。https://doi.org/10.17487/ RFC2308 提议标准。

[3] Anthropic. 2025年17, 2026年. 网页获取工具. Anthropic. 检索于11月，来自 https://platform.claude.com/docs/en/agents-and-tools/tool-use/web-fetch-tool[4] Anthropic. 2026年. *Anthropic* 会从网络抓取数据吗？网站所有者如何屏蔽爬虫？检索于2026年11月17日，来自 https://support.claude.com/en/articles/8896518[5] Christoph Auer、Maksym Lysak、Ahmed Nassar、Michele Dolfi、Nikolaos Livathinos、Panos Vagenas、Cesar Berrospi Ramis、Matteo Omenetti、Fabian Lindlbauer、Kasper Dinkla、Lokesh Mishra、Yusik Kim、Shubham Gupta、Rafael Teixeira de Lima、Valery Weber、Lucas Morin、Ingmar Meijer、Viktor Kuropiatnyk 和 Peter W. J. Staar。2024年。Docling技术报告。https://doi.org/10.48550/arXiv.2408.09869 arXiv:2408.09869 [cs.CL][6] David Belson。2025年。*2025*年 *Cloudflare Radar* 年度回顾：人工智能、后量子技术的崛起与破纪录的 *DDoS*攻击。Cloudflare。检索于2026年11月18日，来自 https://blog.cloudflare.com/radar- 2025-year-in-review/ [7] David Belson 和 Sam Rhea。2025年。转介流量骤降之前的爬取……了解人工智能对内容提供商的影响。检索于2026年11月17日，来自 https://blog.cloudflare.com/ai-search-crawl- refer-ratio-on-radar/[8] Shai Bergman、Anne-Marie Kermarrec、Diana Petrescu、Rafael Pires、Mathis Randl、Martijn de Vos 和 Ji Zhang。2025年。利用近似缓存加速检索增强生成。载于国际中间件会议论文集（*Middleware*）。340 353 10 1145 3721462 3770776 – 。https://doi.org/ 。/ 。[9] Tim Berners-Lee、Roy T. Fielding 和 Larry Masinter。2005年。统一资源标识符（*URI*）：通用语法。RFC 3986。互联网工程任务组。https://doi.org/ 10 17487 3986 。/RFC [10] Erik Bethke. 。*TL;DR;AI ME*：2026 4第一个千字节属于机器人。检索于2026年11月17日，来自 https://bike4mind.com/blog/tl-dr-ai4me-the-first-kilobyte-belongs-to-the-robots [11] Lee Breslau、Pei Cao、Li Fan、Graham Phillips 和 Scott Shenker。1999年。网页缓存与类Zipf分布：证据及其影响。载于*IEEE*计算机通信会议论文集（*INFOCOM*）。126 134 10 1109 1999 749260 – 。[12] 2025https://doi.org/ 。/INFCOM. 。Athena Chapekis 和 Anna Lieb. 。搜索结果中出现人工智能摘要时，*Google* 用户点击链接的可能性会降低。皮尤研究中心。检索于2026年10月8日，来自 https://www.pewresearch.org/short-reads/2025/07/22/google-users-are-less-likely-to-click-on-links-when-an-ai-summary-appears-in-the-results/[13] Yixi Chen、Muhammad Bilal、Tie Ma、Zafar Ayyub Qazi、Marco Canini 和 Suhaib Fahmy。2025年。从抓取到发布：面向人工智能原生互联网的发布者控制型网络架构。https://doi. org/ 10 48550 2511 18354 2511 18354 [ ] 。/arXiv. 。arXiv: 。[14] 2026cs.NI Cloudflare. 。*Cloudflare Radar*。检索于2026年11月17日，来自 https://radar.cloudflare.com [15] Cloudflare。2026年。面向智能体的 *Markdown*。Cloudflare。检索于2026年11月17日，来自 https://developers.cloudflare.com/fundamentals/reference/markdown-for-agents/ [16] Matthew Conroy。2026年。互联网还有第二类受众。Cloudflare。检索于2026年11月17日，来自 https://blog.cloudflare.com/agentic- web/[17] Jian Cui、Mingming Zha、XiaoFeng Wang 和 Xiaojing Liao。2025年。robots.txt 治理的演变：衡量网页机器人在大语言模型服务中的约定影响。载于*ACM*计算机与通信安全会议论文集（*CCS*）。 21 35 10 1145 3719027 3765063 – 。https://doi.org/ 。/ 。[18] 2025DataDome. 。盛大伪装：人工智能智能体如何冒充身份闯入。检索于2026年11月17日，来自 https://datadome.co/threat-research/ai-agent-spoofing/

[19] Zakir Durumeric 和 David Adrian。2022年。*Chrome*（*CrUX*）排名前百万的网站。检索于2026年11月17日，来自 https://github.com/zakird/crux-top-lists [20] Roy T. Fielding、Mark Nottingham 和 Julian Reschke。2022年。*HTTP* 缓存。RFC 9111。互联网工程任务组。https://doi.org/ 10.17487/RFC9111 标准轨道。[21] Roy T. Fielding、Mark Nottingham 和 Julian Reschke。2022年。*HTTP* 语义。RFC 9110。互联网工程任务组。https://doi. org/10.17487/RFC9110 标准轨道。[22] Google。2017年。*CrUX* 概述。检索于2026年11月17日，来自 https://developer.chrome.com/docs/crux [23] Google。2026年。*Gemini* 弃用。Google AI Studio。检索于2026年11月17日，来自 https://aistudio.google.com/docs/deprecations#gemini-2.5-flash-models [24] Google。2026年。故障排除指南：复述问题。Google AI Studio。检索于2026年11月17日，来自 https://aistudio.google.com/docs/ troubleshooting#recitation-issue [25] Google。2026年。*URL* 上下文。Google AI Studio。检索于2026年11月17日，来自 https://aistudio.google.com/docs/url-context [26] Rumaisa Habib、Balaji Balachandran、Nurullah Demir 和 Zakir Durumeric。2026年。“别再强行让 Fetch 成事”：迈向对网络中不必要资源的模态感知理解。载于*ACM*互联网测量会议论文集（*IMC*）。339–355 10 1145 3777912 3839788 。https://doi.org/ 。/ 。[27] Jeremy Howard. 。 2024 /llms.txt 文件，*v*。检索于 2 2026年11月17日，来自 https://llmstxt.org/ [28] Peichun Hua 和 Yunming Xiao。2026年。语义交付网络：重新思考面向大语言模型智能体的网络检索基础设施。载于*ACM* 网络热点主题研讨会论文集（*HotNets*）。https://arxiv.org/abs/2609 22486 。Gary Illyes、Mirja Kühlewind 和 A. [29] Kohn。2026年。爬虫最佳实践。互联网草案 draft-illyes-aipref-cbcp-04。互联网工程任务组。https://datatracker.ietf.org/doc/html/draft-illyes-aipref-cbcp- 04 已过期的互联网草案。[30] iptoasn.com。2026年。*IP* 到 *ASN* 数据库。检索于2026年11月17日，来自 https://iptoasn.com/ [31] Ivan Ivan。2025年。网络查询中陈旧搜索结果循环出现的问题。检索于2026年11月17日，来自 https://community.openai.com/t/bug-with-stale-search-results-looping-in-web-queries/1152931 [32] 2025 Klaudia Jaźwińska 和 Aisvarya Chandrasekar. 。*AI* 搜索存在引用问题。检索于2026年11月17日，来自 https://www.cjr.org/tow_center/we-compared-eight-ai-search-engines-theyre-all-bad-at-citing-news.php[33] Chao Jin、Zili Zhang、Xuanlin Jiang、Fangyue Liu、Shufan Liu、Xuanzhe Liu 和 Xin Jin。2026年。RAGCache：检索增强生成的高效知识缓存。*ACM* 计算机系统汇刊 44, 1 2026 2 1 2 27 10 1145 3768628 ( ), ：–：。[34] 2025 https://doi.org/ 。/ Aviad Kaiserman. 。*AI* 爬虫欺骗曝光：攻击者如何冒充 *ChatGPT* 、 *Mistral* 和 *Perplexity*。HUMAN Security。检索于 2026 年11月17日，来自 https://www.humansecurity.com/learn/blog/ai-crawler-spoofing-chatgpt-mistral-perplexity/[35] Taein Kim、Karstan Bock、Claire Luo、Amanda Liswood、Chloe Poroslay 和 Emily Wenger。2025 年。网页抓取器会选择性地遵守 robots.txt 指令：一项大规模实证研究的证据。载于*ACM*互联网测量会议论文集（*IMC*）。541–557。https://doi.org/10.1145/3730567.3764471[36] Brian Kondracki、Assel Aliyeva、Manuel Egele、Jason Polakis 和 Nick Nikiforakis。2020年。作恶的中间人：省流量移动浏览器风险的实证分析。载于*IEEE* 安全与隐私研讨会论文集（*SP*）。810–824。https://doi.org/10.1109/SP40000.2020.00077

[37] Steve Kopack。2026年。*Google*母公司*Alphabet*股价暴跌，市值蒸发超过*5000*亿美元。NBC New York。于2026年11月17日检索自 https://www.nbcnewyork.com/news/national-international/alphabet-google-stock-ai-advertising/6564812/ [38] Martijn Koster、Gary Illyes、Henner Zeller与Lizzi Sassman。2022年。机器人排除协议。RFC 9309。互联网工程任务组。https://doi.org/10 17487 9309 . /RFC Hannah Erin Lang、Erin [39] Woo与Suzanne Vranica。2026年。*Alphabet*市值蒸发*5000*亿美元，创该股史上最差的一天。《华尔街日报》。于2026年11月11日检索自 https://www.wsj.com/tech/ai/alphabet-sheds-half-a-trillion-in-market-value-in-stocks-worst-day-ever-bde7bb6e[40] Patrick Lewis、Ethan Perez、Aleksandra Piktus、Fabio Petroni、Vladimir Karpukhin、Naman Goyal、Heinrich Küttler、Mike Lewis、Wen-tau Yih、Tim Rocktäschel、Sebastian Riedel与Douwe Kiela。2020年。面向知识密集型自然语言处理任务的检索增强生成。载于神经信息处理系统进展（*NeurIPS*），第33卷。9459–9474。https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html[41] Zhenhua Li、Weiwei Wang、Tianyin Xu、Xin Zhong、Xiang-Yang Li、Yunhao Liu、Christo Wilson与Ben Y. Zhao。2016年。借助百度 TrafficGuard探索跨应用蜂窝流量优化。载于*USENIX*网络系统设计与实现研讨会论文集（*NSDI*）。61–76。https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/li-zhenhua [42] Enze Liu、Elisa Luo、Shawn Shan、Geoffrey M. Voelker、Ben Y. Zhao与Stefan Savage。2025年。我曾经抓取过的某个网站：在保护内容创作者免受AI 爬虫侵害方面的意识、能动性与效力。载于*ACM*互联网测量会议论文集（*IMC*）。78–99。https://doi.org/10 1145 3730567 3732913 . / . Nelson Liu、Tianyi Zhang与 [43] Percy Liang。2023年。评估生成式搜索引擎中的可验证性。载于计算语言学协会发现：*EMNLP*。7001–7025。https://doi.org/10.18653 1 2023 467 /v / .findings-emnlp. Muhammad [44] Mansoor、Tahir Ahmad与Yeo-Chan Yoon。2026年。面向开放网络检索增强大语言模型的风险约束的新鲜度感知语义缓存。https://doi.org/10.48550/arXiv.2607. 04281 arXiv:2607.04281 [cs.CL][45] Matthew Mathur、David King与Fastly安全研究团队。2025年。*2025*年第二季度的人工智能机器人：来自*Fastly*威胁洞察报告的趋势。于2026年11月17日检索自 https://www.fastly.com/blog/ai-bots-q2-2025-trends-fastlys-threat-insights-report [46] Thibault Meunier与Sandor Major。2026年。面向自动化流量的*HTTP*消息签名。互联网草案 draft-ietf-webbotauth-httpsig- protocol-00。互联网工程任务组。https://datatracker. ietf.org/doc/draft-ietf-webbotauth-httpsig-protocol/ 2027年3月5日到期。[47] Microsoft。2024年。*MarkItDown*。于2026年11月17日检索自 https://github.com/microsoft/markitdown[48] Seyed Ali Mirheidari、Sajjad Arshad、Kaan Onarlioglu、Bruno Crispo、Engin Kirda与William Robertson。2020年。缓存与困惑：现网中的Web缓存欺骗。载于*USENIX*安全研讨会论文集（*USENIX Security*）。665–682。https://www.usenix.org/conference/usenixsecurity20/presentation/mirheidari[49] Priyanka Mudgal、Kai Zhao、Guilin Zhang、Andy Olsen、Ezekiel Miller、Xu Chu与Aletta Johanna Blanken。2026年。MemLeak：多租户人工智能代理记忆中的跨用户语义泄漏。载于神经信息处理系统会议上的面向人工智能系统的个性化、对齐和长期记忆研讨会（*PALM*）。arXiv:2610.04195 https://neurips.cc/virtual/2026/loc/paris/169601[50] Reiichiro Nakano、Jacob Hilton、Suchir Balaji、Jeff Wu、Long Ouyang、Christina Kim、Christopher Hesse、Shantanu Jain、Vineet Kosaraju、

William Saunders、Xu Jiang、Karl Cobbe、Tyna Eloundou、Gretchen Krueger、Kevin Button、Matthew Knight、Benjamin Chess与John Schulman。2021年。WebGPT：基于人类反馈的浏览器辅助问答。arXiv:2112.09332 [cs.CL] https://arxiv.org/ abs/2112.09332[51] Mark Nottingham。2010年。面向过时内容的HTTP 缓存控制扩展。RFC 5861。互联网工程任务组。https://doi.org/ 10.17487/RFC5861[52] Mark Nottingham与Roy T. Fielding。2012年。HTTP 附加状态码。RFC 6585。互联网工程任务组。https://doi.org/10. 17487/RFC6585 提议标准。 [53] OpenAI。2025年。gpt-oss：浏览器工具的参考实现（simple_browser）。于2026年11月5日检索自 https://github. com/openai/gpt-oss/tree/main/gpt_oss/tools/simple_browser [54] OpenAI。2025 8, 2026。人们如何使用ChatGPT。于10月检索自 https://openai.com/index/how-people-are-using-chatgpt/ [55] OpenAI。2026年。API 定价。于2026年11月17日检索自 https: //developers.openai.com/api/docs/pricing [56] OpenAI。2026年。OpenAI 模型规范。于2026年11月17日检索自 https://model-spec.openai.com/2026-10-29.html#respect_creators [57] OpenAI。2026 8, 2026。OpenAI 爬虫概览。于10月检索自 https://developers.openai.com/api/docs/bots [58] OpenRouter。2026年。网页搜索：原生搜索提供商。OpenRouter。于2026年11月17日检索自 https://openrouter.ai/docs/guides/ features/server-tools/web-search#native-search-providers[59] Kimberly Ruth、Deepak Kumar、Brandon Wang、Luke Valenta与Zakir Durumeric。2022年。颠覆顶级列表：评估热门网站列表的准确性。载于ACM互联网测量会议（IMC）论文集。 374 387 10 1145 3517745 3561444 – 。 [60] https://doi.org/ . / . Abdur-Rahman Ibrahim Sayyid-Ali、Daanish Uddin Khan与Naveed Anwar Bhatti。2026年。大语言模型网页搜索引擎是否可持续? 一项关于实时获取的网页测量研究。载于ACM Web 会议（WWW）论文集。2083–2093。https: //doi.org/ 10 1145 3774904 3792278 . / . Stackfox。. [61] 2026 Grok 机器人用户代理：xAI的爬虫伪造其身份。于2026年11月17日检索自 https://stackfox.co/research/grok- user-agent [62] Ammar Tahir、Muhammad Tahir Munir、Shaiq Munir Malik、Zafar Ayyub Qazi与Ihsan Ayyub Qazi。2020年。解构Google 的 Web Light 服务。载于ACM Web 会议（WWW）论文集。 884 893 10 1145 3366423 3380168 – 。 [63] https://doi.org/ . / . Ruben Teixeira、Jack Galilee、Kayte Johnston与Sam Else。2026年。按次付费：当人工智能使用你的作品时，你应当获得报酬。于2026年11月17日检索自 https://blog.cloudflare.com/pay-per-use/ [64] João Tomé。2025年。抓取到点击的差距：Cloudflare关于人工智能机器人、训练与引荐流量的数据。Cloudflare。于2026年11月17日检索自 https://blog.cloudflare.com/crawlers-click-ai-bots-training/ [65] 维基媒体基金会。2025年。爬虫如何影响维基媒体项目的运营。于2026年11月17日检索自 https://diff.wikimedia.org/2025/04/01/how-crawlers-impact-the-operations-of-the-wikimedia-projects/ [66] xAI。2026年。网页搜索。SpaceXSI 文档。于2026年11月17日检索自 https://docs.x.ai/developers/tools/web-search [67] Dave Young。2025年。 [严重错误] ChatGPT浏览工具对有效 URL 返回虚假 500 错误。于2026年11月17日检索自 https://community.openai.com/t/critical-error-chatgpt-browsing-tool-returns-false- 500 1258495 - [68] errors-on-valid-urls/ Giacomo Zecchini、Alice Alexandra Moore、Malte Ubl与Ryan Siddle。2024年。人工智能爬虫的兴起。Vercel。于2026年11月17日检索自 https://vercel.com/blog/the-rise-of-the-ai-crawler[69] Saber Zerhoudi、Michael Dinzinger、Michael Granitzer与Jelena Mitrovic。2026年。OwlerLite：范围感知与新鲜度感知的网页检索面向大语言模型助手。在*ACM Web Conference (WWW)* 伴随论文集中。 216 220 10 1145 3774905 3793140 – 。 [70] 2025 https://doi.org/ 。/ 。Yazhuo Zhang、Jinqing Cai、Avani Wildani 和 Ana Klimovic。《重新思考人工智能时代的网页缓存设计》。载于*ACM*云计算研讨会论文集（*SoCC*）。535–542。https://doi.org/10.1145/3772052.3772255[71] Maya Ziv、Liz Izhikevich、Kimberly Ruth、Katherine Izhikevich 和 Zakir Durumeric。2021年。ASdb：一个用于对自治系统所有者进行分类的系统。载于*ACM互*联网测量会议（*IMC*）论文集。703–719。https://doi.org/10.1145/3487552.3487853

A 伦理

受控的基础设施。 所有主要实验均在我们运营的基础设施上进行：两个源站提供静态合成网页（源站 A 位于 Cloudflare 反向代理之后，并绕过缓存；源站 B 则由 Caddy 直连提供服务），通过供应商的 API 使用付费账户访问。网页仅包含合成文本和页面专属代码，不含供应商凭证或用户数据；我们没有对任何第三方系统施加压力或进行修改：我们注入的错误、连接中断和重定向循环，以及由此引发的重试，都只发生在我们的源站上。两个源站均可公开访问，因此日志中会出现无关访客；我们只分析带有我们 URL 标签的请求。

服务条款。 我们通过 OpenAI、Anthropic 和 xAI 已公开的 API，按照标准商业条款使用其服务，直接使用或通过 OpenRouter 使用，并在文档规定的速率限制内使用无害的网络检索提示。对于消费者应用，研究人员在已登录的免费账户中手动输入每个问题。共享测试使用了两个账户，一个属于作者，另一个属于志愿者；这两个账户均由我们自行创建和运营，输入的也只有合成研究提示。

公共网站验证。 为检查对照实验的发现是否适用于实时网页，我们以类似浏览的方式读取第三方网站。我们仅抓取未登录状态下的主页，不进行登录、不使用 Cookie，只保留可见文本；即使是归类为社交网站的网站，我们也只抓取主页，从不访问个人资料或任何个人内容，并且不收集个人数据。我们仅保留其 robots.txt 允许我们的下载器和全部四个人工智能页面抓取器访问的网站，并排除 Cloudflare 标记为成人、赌博或受 CIPA 过滤的网站（表 8）。对第三方网站的所有请求均为只读请求。对于文件类型测试，我们从一台美国服务器向每个网站请求少量已公布的文件（robots.txt、站点地图、订阅源、llms.txt、.csv链接），并向每项服务分别询问一次，因此每个网站向每项服务最多额外收到一次请求。对于陈旧度测试，我们每 10 分钟抓取一次每个主页，持续 24 小时，频率与运行时间监控器相当。对于公共端点测试，我们每五分钟读取一次八个公共计数器中的每一个。下载器将自身标识为学术研究工具

，并且只保留每个计数器的数值，而不保留响应内容。我们仅发布这些数值及其时间戳，以及每个供应商的答案和 Anthropic 报告的获取时间。

供应商归因。 我们通过提供给每家供应商的 URL 中的一个标记来将请求归因到供应商，并且只将其与我们自己的服务器日志进行匹配。我们还记录抓取器的源地址，但仅用于计数、查询地址所属的网络，并与供应商发布的列表进行比对，绝不用于识别个人。

负责任的披露。 我们通过供应商的公共 API 和应用，从外部测量其系统。我们的研究结果描述了它们的抓取器如何对待出版商和用户，并不涉及安全漏洞，因此我们没有采用协调披露窗口。

机构审查委员会。 本研究不收集任何人的个人数据：数据来自我们自己的源站、供应商的 API，以及已退出登录状态下的公共页面。共享测试中使用的唯一一个志愿者账户由我们创建并运营，我们没有保留任何有关其所有者的数据。我们咨询了所在机构的审查委员会，该委员会确认本研究不涉及人体数据或样本，因此无需审查。

B 生成式人工智能的使用

我们在两项任务中使用了生成式人工智能：改写文本和编写代码。2026年9月至11月期间，我们使用了 Anthropic 的 Claude Code，以及 Claude Opus 4.8、Opus 5、Opus 5.5 和 Sonnet 5.5。

对于文本，我们撰写了提纲和各章节的论点，再由工具改写段落。我们修订了文本，并根据数据核查了每一项论点。对于代码，工具编写了测量框架、测试页面、分析脚本和图表代码的大部分内容，设置了两个源站，并准备了发布的代码和数据。

这项研究是探索性的：我们根据先前的发现选择下一项测量，并检查每一项输出。例如，我们最初发现页面会在不同账户间共享，而且失败获取的表现异常；随后我们测量副本能够保留多久，再进一步检验这种情况是否因文件格式而异。每项新测量也都用于交叉核验先前的测量结果。

C 测试材料

本附录提供了测量设置（§ 2）背后的材料：我们发送的提示（§ C.1、§C.2）、选定的公共页面（§ C.3、§ C.3.2）、构建测试文档的方式（§ C.4），以及开展测试的方式（§ C.5、§ C.6）。

C.1 受控源站提示

我们在自有来源上进行几乎所有测试时，都使用同一种提示词形式，这样同一个问题就能在不同服务间复用和实验。问题要求给出网页中包含的一段代码，因此每个答案都能由脚本确定性地检查。

受控来源 API 提示。每次 API 试验仅发送经过测试的 URL 和一个问题：

<URL>

抓取此文档，并告诉我其中深度为=N%的部分的 ID 以及以 TRUNC- 开头的代码。请逐字准确引用。

试验之间只有深度 N 发生了变化。每个深度代码只出现在一个网页上，因此正确的答案表明模型已读取该网页。

例外。 三项实验提出了不同的问题，因为它们需要不同的答案：

- 文档大小： 网页上可见的每个代码，用于查看模型读取到了多深。
- 故障： 网页上所述内容以及任何可见代码，以便模型报告它遇到的故障。
- *URL* 等价性： 网页上编号条目登记表中的一项，用于显示答案来自哪个网页。[19]

提示措辞。 我们还使用措辞不同的提示进行了测试，抓取器的行为均未发生变化。

C.2 公共页面的提示词

公共页面没有深度代码，因此无法使用标准问题。作为替代，我们要求提供网页固定位置上的文本，以便逐字与我们自行下载的内容进行核对。提示词指出网页上的一行文字，即 锚点，并要求给出其后面的文字：

抓取此文档： <URL>。准确引用其中紧接在“<ANCHOR>”这一行之后的前 12 个词。如果你无法读取该文档，请直说。

锚点是我们多次下载该网页时始终保持不变的一行，而其后的文本会发生变化，因此我们的下载结果能够告诉我们每个答案来自该网页的哪个版本。我们要求提供 12 个单词，因为更长的引文可能会被拒绝：Google 文档记录了针对可能从来源复制的内容的 RECITATION 停止原因 [24], ，而在我们的运行次数中，OpenAI 的模型拒绝逐字引用超过 25 个单词。[20] 对于日语或中文等词语之间不使用空格的页面，提示要求提供开头的 30 个字符。

[19]这项测试旨在检查不同 URL 是否会被视为同一网页，因此 URL 不能包含在其他实验中用于区分网页的标签。[20]这一限制来自模型自身的回复，例如：“我无法逐字复现非歌词来源中超过 25 个词的内容。”我们没有找到任何 OpenAI 文档提及这一限制。OpenAI 模型规范只规定了尊重创作者权利这一一般义务 [56]。

Step	网站
CrUX 样本	2,000
robots.txt 允许我们的下载器进入	1,783
来自美国服务器的正常答案（HTTP 200）	1,272
robots.txt 允许所有四种人工智能页面抓取器	1,179
未被 Cloudflare 的家庭 DNS 列为成人内容	858
不属于 URL 扫描器排除的类别	**776**

表 8：每个步骤后剩余的网站。

公共端点。 陈旧度测试针对这八个端点，要求获取一个只会增加的字段的当前值：

抓取 <URL> 并仅回复 <TARGET>，必须与所示内容完全一致，不要添加任何其他内容、文字或标点，只回复该值。

文件类型。文件类型测试通过要求服务逐字引用某种给定类型的公共文件的前三行，来检验服务能否读取该文件：

抓取此文档： <URL>。逐字引用其前 3 行。如果你无法读取该文档，请直说。

C.3 公共网站

如果我们可以向服务询问某个网站，该网站就符合条件；每项测试随后会从符合条件的网站中选取所需网站。公开检查使用三组目标（§ 2.2）：从符合条件的网站中选取的公共文件和变化页面，以及人工选定的反向端点。

网站样本。 我们从 Chrome 用户体验报告（CrUX） [22], 中选取网站，使用 CrUX 顶级列表代码库 [19, 59]发布的 2026 年 8 月全球热门网站列表。我们纳入顶级组的全部 1,000 个网站，再从下一组（排名 1,001–5,000）中随机抽取 1,000 个网站，总计 2,000 个。

资格。 如果网站能正常响应，robots.txt 允许我们的下载器以及四种服务各自的页面抓取器访问，且其内容不敏感（表 8），那么该网站就符合资格。robots.txt 这一条件意味着，我们只会向这些服务询问网站允许抓取的网页；我们不会依赖抓取器自行避开这些网页，因为某些由用户触发的抓取器可能不会遵守这一规定（OpenAI 对 ChatGPT-User 明确说明了这一点 [57]）。我们分两个阶段移除敏感内容。首先，Cloudflare 的家庭 DNS 解析器（1.1.1.3）会对 Cloudflare 归类为成人内容的网站返回 0.0.0.0，从而无需访问这些网站便将其移除。其次，按照 Habib 等人 [26],的方法，我们从 Cloudflare URL 扫描器获取每个剩余网站的类别，并排除任何被标记为 *CIPA*、*CIPA* 过滤、成人主题、色情内容 或赌博的网站。若主页发生重定向，我们采用其最终跳转到的网页的标签。Cloudflare 未标记的网站则予以保留。

表 9：公开文件：每个步骤后剩余的文件。站点地图和订阅源必须声明 XML 类型，而 llms.txt 文件必须声明 text/plain。

Step	站点地图 p	Feed	llms.txt	总计
已检查候选项	776	141	776	1,693
正常应答（HTTP 200）	511	102	213	826
1–200 千字节, 至少 5 行文本	240	57	165	462
robots.txt 允许所有四种抓取器	239	57	165	461
声明的请求类型	171	55	73	**299**

参考副本。 在每项公开测试中，我们都会将每个答案与自己下载的目标文件或网页进行比较；这些文件或网页是在提问前后下载的（对于对照端点，则是在整个测试期间每五分钟下载一次）。

C.3.1 公开文件。文件类型测试会要求每种服务引用文件的前三行（§ C.2），以此判断它能否读取指定声明类型的公开文件。

筛选。对于符合条件的网站，我们只检查网站公布文件的位置：robots.txt 中列出的站点地图（或 /sitemap.xml）、主页链接的 RSS 和 Atom 订阅源、/llms.txt，以及指向 .csv 文件的链接。我们根据服务器在 Content-Type 标头中声明的类型对每个文件进行分类，而不是根据文件名或内容，因为抓取器会根据声明的类型接受或拒绝文件（例如，OpenAI 的抓取器会报告“不支持的内容类型：application/xml”，§ 3.2.1）。我们保留正常响应、大小为 1–200 千字节且至少有五行、主机的 robots.txt 允许所有四种抓取器，并且服务器以我们请求的文件格式提供的文件（表 9）。在类型筛选步骤中剔除的大多数文件均以 text/html 格式提供。我们还手动移除了一个网站，因为其域名表明它含有成人内容，尽管两个筛选器都未将其标记出来。最终留下 299 个文件：226 个 XML 文件，以及作为对照的 73 个以 text/plain 格式提供的 llms.txt 文件。

Markdown 和 *CSV*……这两种类型在公共网络上几乎不存在，因此我们只在自己的受控源站上检查它们。

具体而言，只有一个文件被声明为 text/markdown（但超过了我们的大小限制），没有文件是 CSV，也没有网站在 /index.md 或 /index.html.md 提供 Markdown 主页——这两个地址是 llms.txt 提案定义的 [27]。有十二个主页（1.5%）仅在请求中指定需要 Markdown（Accept: text/markdown）时才返回 Markdown，这是 Cloudflare 推出的“Mark- down for Agents”所提供的功能 [15]。然而，我们能识别出的抓取器中，没有一个会明确请求 Markdown 格式的内容。[21]

[21]在我们自己的源站上，OpenAI、Anthropic 和 Google 的抓取器请求任意格式（*/*）或优先请求 HTML，从未请求 Markdown；xAI 的抓取器将自己标识为浏览器，无法单独识别。我们无法观察公共网站本身上的请求。

表10：变化页面：每一步后剩余的网站。

Step	网站
监测 24 小时的网站	776
至少有一个正常答案（HTTP 200 且包含文本）	772
提供 HTML	769
至少 100 个正常答案	763
真实变化之间的间隔中位数少于 30 分钟	196
在至少 30% 的 10 分钟时间段内发生真实变化	173
个人数据排除	172
稳定锚点	**90**

测试运行。 我们对每个文件向每项服务请求一次，每项服务的调用间隔三分钟。

C.3.2 变化页面。陈旧度测试需要从每个网页获取两样东西：一部分经常变化的内容，以及一种指向它的方式。我们只能要求提供一段短引文（§ C.2），因此不能总是从网页顶部开始；提示词会改为指定一个*锚点*，即紧邻变化内容上方的一行短文字。测试本身见 § E.6。

监测。 我们每 10 分钟访问一次 776 个符合条件的网站的主页，持续 24 小时。每次访问时，我们都会存储状态、不含 Cookie 的响应头、大小和可见文本，但不存储 HTML。*锚点* 是位于网页变化部分正上方的一行简短文本，在至少 90% 的访问中恰好出现一次。*真实变化* 是并非昙花一现的新行：它在下一次访问时仍显示在网页上，因此至少持续 10 分钟。我们忽略时钟和日期行，以及字母过少或数字占比至少为 30% 的行；字母包括任何文字系统中的字母。在抽样检查的 30 次变化中，70% 属于真实变化。

筛选。 我们保留符合以下条件的网站：提供 HTML，至少响应 100 次访问，变化频率足够高（真实变化之间的中位间隔少于 30 分钟），[22] 并且至少在 30% 的时间里出现真实变化。筛选后剩下 172 个网站，其中 90 个具有稳定锚点，因此成为我们的测试目标（表10）。

C.3.3 反向端点。

筛选。 这些是八个数值只增不减的公共端点，例如 Hacker News 最大条目 ID、区块链高度，或美国地质调查局 Feed 的 metadata.generated 时间戳。很少有公共服务会公开这类数值，因此我们手动挑选了这些端点，只保留无需登录且以纯文本或 JSON 返回数值的端点。这组端点涵盖了多种缓存头和主机，因此可以区分标头与主机各自的影响（表20）。

测试运行。 我们每隔 45 分钟向每项服务请求每个数值一次，持续约三天。陈旧度指答案过期了多久。也就是说，如果某项服务返回的数值与两小时前记录在我们的参考日志中的数值一致，那么该答案就陈旧了两小时。对于 Anthropic，其回复会注明获取网页的时间，因此我们使用该副本的存放时长。[23] 不过，我们的参考日志只保存数值，因此无法确定低于五分钟的陈旧度。

C.4 测试文档构建 n

源内容。我们的源站会根据有序的源代码行，在收到请求时生成每份测试文档：对于只需从网页获取一个值的测试，生成简短网页；对于需要在不同深度查找信息的测试，则生成较长网页。除非实验另有说明，源站会提供一个 2KB 的 HTML 网页。较长的网页包含来自冻结语料库的文段，并在文本约 0%、25%、50%、75% 和 95% 的位置附近放置五个代码。这些代码用于判断答案是否来自预期网页；对于较长的网页，还可判断答案是否来自请求的位置。每个代码都是对源站与测量客户端共享的密钥、页面 ID、代码在网页上的位置以及内容修订版本计算出的 HMAC。独立试验使用新鲜的页面 ID。当测试原位更改网页时，修订版本也会随之改变，因此可以区分来自过期内容和新内容的答案。图 5 展示了这些代码如何揭示网页的哪些部分最终出现在答案中。

格式。源站将相同的源代码行渲染为八种文档格式（清单 1）。特定格式的标记语言会改变字节数和文件大小，但不会改变代码或其顺序。我们使用独立工具进行了检查，确认每种输出都可以打开并提取其中的代码（表 11）。

清单 1：将排序后的源代码行渲染为测试文档。每个渲染器都会保留页面专属代码及其顺序。

```
输入：源代码_行 // 按顺序排列的正文和代码                r

文本：连接(source_行, "\n") + "\n" Markdown："# 探测
文档\n\n"+ 连接(source_行, "\n\n") + "\n" HTML：写入
文档类型、字符集和标题 对于 source_行中的每一行：写
入 "<p>" + html_转义(line) +"</p>" JSON：json_编码({"
文档": source_行})
```

[22]这短于我们观察到的 OpenAI 复用时间（§ 3.3.3），因此可以区分复用的副本和当前版本。

[23]我们将报告的时间与参考日志进行了核对，两者相差约一分钟以内。

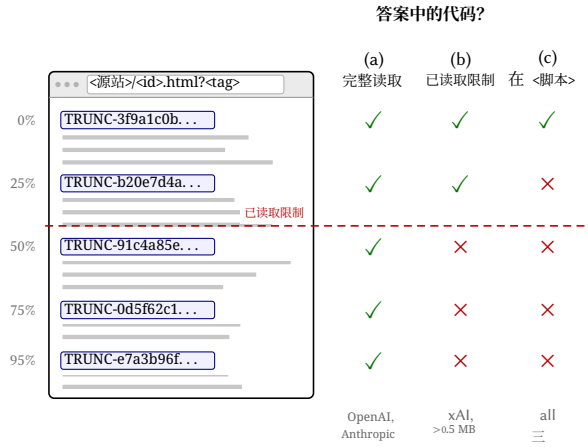


图5：在已知位置放置代码，可以显示网页的哪些部分会进入答案。较长的测试页面在文本的 0–95% 位置放置五个代码，每个代码仅用于该页面。(a) OpenAI 和 Anthropic 会返回所有代码，最多支持 2 MB。(b) xAI 只会返回较大网页中位于前 0.5 MB 的代码（§ 3.2.3）。(c) 当第一个代码之后的代码位于 <script> 标签内时，没有任何服务会返回它们；0% 位置的代码（纯 HTML）表明网页本身已被读取（§ 3.2.2）。

纯 HTML

```
<p>TRUNC-3f9a1c0b7d2e</p>
```

数据块

```
<脚本 type="application/json">["TRUNC-3f9a1c0b7d2e"]</
    脚本>
<script> ...将每一行添加到网页中... </script>
```

JSON-LD

```
<脚本类型="application/ld+json">
  {"@type": "文章", "文章正文": "TRUNC-3f9a1c0b7d2e"
</脚本>
```

内联脚本（已打乱）

```
<脚本> 种子 = 1234567890; blob = "Xk9a..."; ...解码并
    添加到网页中... </脚本>
```

外部脚本（经过扰乱）

```
<脚本 src="/js/p/n.js"></script>
```

图 6：JavaScript 测试页面各版本的页面源代码，每个版本缩减为一段代码。浏览器会在除 JSON-LD 外的所有版本中显示文本。每个版本中作为纯段落的 0% 代码未列出。

XML：写入 XML 头部和 "<文档>" 对于 source_行中的每一行：写入 "<line>" + xml_转义(line)+ "</line>" 写入 "</文档>"

csv：write_csv_row(["n", "content"]) for index, line in enumerate(source_lines): write_csv_row([index, line])

pdf：为第_组网页_中的_55（源_行）编写 PDF 标头、目录、网页树和 Helvetica 字体：流 = begin_text() 对 page_lines 中的每一行执行：流 += show_text(pdf_escape(line)) 流 += next_line() 流 += end_text() 写入网页对象及其文本流 写入交叉引用偏移量和尾部

docx：文档_xml = 新建_词语_文档() 对于 source_行中的每一行：文档_xml += 词语_段落(xml_转义(行))

zip_与_固定_时间戳("[内容_类型].xml", "_rels/.rels", "word/document.xml" = 文档_xml)

JavaScript 测试页面。这些页面用于测试服务是否会读取由脚本提供的内容。它们有五个版本（表 5），每个版本都基于一个新鲜的页面 ID 构建，并返回一个包含五段代码和标准问题（§ C.1）的 4KB 页面。0% 代码始终是作为对照的纯 HTML，因此其返回结果表明页面已送达模型。其他四段代码（25–95%）在各版本中的提供方式各不相同（图 6、算法 1）；只有运行页面的解码器才能读取经过扰乱的代码。无头 Chromium 会在除 JSON-LD 外的所有版本中显示这五段代码，因为浏览器不会显示 JSON-LD。我们进行了三轮测试，每轮都为每个服务和版本使用一个新鲜页面，并使用 OpenAI、xAI 和 Anthropic 的基础抓取工具（网络_抓取_20250910）。

C.5 OpenRouter 和直连路由

OpenRouter 是一个网关，可通过一个 API 连接众多供应商的模型。原则上，这能让我们将 API 测试扩展到更多供应商，但它可能会用自己的抓取器替代供应商的抓取器，这样测量的将是 OpenRouter，而不是供应商。因此，我们会通过请求的用户代理来确认抓取来自供应商

算法 1： 构建 JavaScript 测试页面。所有版本共用相同的设置，唯一的差异在于如何从测试_行构建网页主体。

输入： 页面 ID p，大小 n，版本 v

测试_行 ← n 字节，每份代码占 25%、50%、75%、95%

/* 已解扰，可读的纯文本 */

如果 v =是纯文本 **则**

  正文 ← 行 对于每一 行 在 测试_行中

否则如果 v =数据块 **则**

  正文 ← <脚本类型="application/json"> json(测试_行) </script> + 一个会复制它的脚本 进入网页

否则如果 v =*JSON-LD* **那么**

  正文 ← <脚本类型="application/ld+json"> json(文章, 文章正文 =测试_行) </脚本>

否则

  /* 混淆内容，可通过运行解码器读取 */

  文本 ←连接(测试_行，换行符)

  种子 ← SHA-256(p, n) 的前 4 个字节

  blob ←Base64(打乱(文本, 种子))

  js ←解码器(种子, *blob*)

  if v =内联 **则**

    正文 ← <脚本>*js* </脚本>

  else

    body ← <脚本 src="/js/p/n.js"></script>

    在 /js/p/n.js 提供*js*

函数扰乱(文本, 种子)**:**

  s ←种子

  遍历 字节 b 在文本 **执行**

    s ←异或移位32(s)

    输出 $b \oplus (s$ 模 256)

函数解码器(种子, 数据块)**:**

  /* 在浏览器中运行 */

  文本 ←打乱(基础*64*_解码*(blob)*, 种子)

  对 文本.*split(text, newline)* 中的每一行**执行**

    将段落(line)追加到网页

自有工具，因为 OpenRouter 的抓取器使用自己的用户代理。[24]

对于 OpenAI、xAI 和 Anthropic 的 claude-sonnet-5，抓取由供应商自己的工具完成，OpenRouter 的抓取器从未出现。检查结果因模型而异：claude-haiku-4.5 和 DeepSeek 都使用了 OpenRouter 的抓取器，而 Sonnet 使用的是 Anthropic 自己的工具，因此我们使用 Sonnet，而不查询 DeepSeek。OpenRouter 的文档列出了它会使用供应商自有网络工具的模型，而这两个模型都不在名单上 [58]。因此，受控源站 API 测试仅涵盖 OpenAI、Anthropic 和 xAI。

[24]我们使用与其他地方（§ 2.3）相同的 URL 标签：它只存在于一个问题中，因此，在我们的源站对该 URL 发出的每个请求都会关联到该问题。

表 11：用于检查每种生成格式能否打开并生成其代码的工具。我们检查了每种格式下的一个短网页和一个包含五段代码、大小为 64 KiB 的网页。

格式	验证工具
HTML	Python html.parser
文本、Markdown	严格的 UTF-8 解码
JSON	Python json
XML	Python xml.etree
CSV	Python csv；行宽检查
PDF	Poppler、MuPDF、pypdf、pdfminer.six
DOCX	python-docx、docx2txt、LibreOffice；ZIP 和 XML 检查

C.6 复用测试设计

§ 3.3 中的每项缓存测试都遵循同一种模式（图 7）：先提出一个问题，让服务抓取一个带标记的页面；经过一段延迟后，源站会修改该页面或保持不变；随后提出第二个问题，要求提供只有重新抓取才能获得的代码。结合源站日志和答案，即可判断服务是否复用了先前的检索结果。

D 抓取器和内容的其他结果

本附录为第 3.1 节和第 3.2 节提供补充说明：抓取器从哪里连接（§ D.1）、如何响应故障（§ D.2、§ D.7）、如何处理 robots.txt（§ D.3、§ D.4），以及如何读取文件和页面（§ D.5、§ D.6）。

D.1 抓取器从哪里连接

本附录展示了抓取器的请求来自哪些网络（第3.1.1节）。我们根据 URL 中的标签，将每个请求与供应商对应起来。对于 OpenAI 和 Anthropic，请求还必须带有该供应商的爬虫名称；xAI 的抓取器发送的是浏览器用户代理，因此对于 xAI，仅凭标签即可判断。对于每个请求，我们都在一个公共所有权数据库中查询了源地址所属的网络 [30],（离线进行）。

OpenAI 的请求来自一个由 Microsoft 所有的网络，而 Anthropic 的请求则来自两个分别由 Google 和亚马逊所有的网络。xAI 几乎每个请求都会使用一个新地址（8,286 个请求对应 7,774 个地址），分布在约 1,260 个网络中，这并非固定服务器集群的行为。

这类网络数量太多，无法一一列举，因此我们使用 ASdb [71],这个公共数据库会为每个网络标注一个业务类别。它将 xAI 的请求中 82% 归入“互联网服务提供商”，7% 归入托管服务。ASdb 根据网络所有者的职能为网络分类[25], 因此，我们也按照 ASdb 的类别，手动标注了 xAI 请求中的一个随机样本，共 100 条；遇到不确定的情况，则参考网络所有者的网站以及有关该网络的公共页面。

[25]它将 Microsoft 的网络归类为软件开发，将 Google 的网络归类为搜索，尽管两者都是云网络。

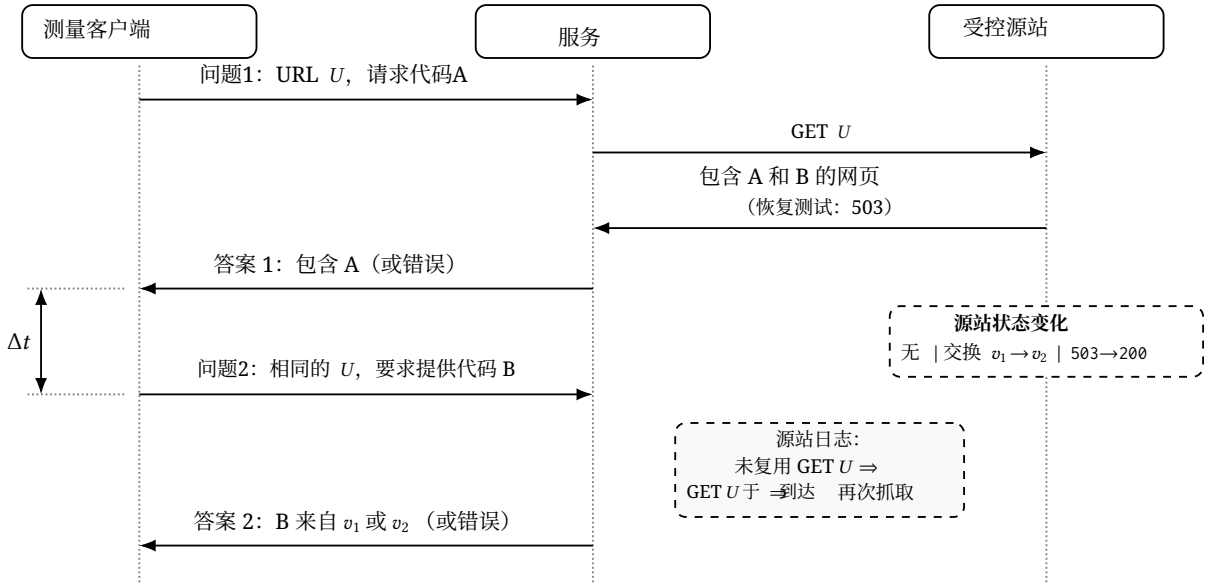


图7：重用测试设计。第一个问题使服务抓取一个带标签的 URL U。间隔 Δt 后，源站保留该网页（复用，§ 3.3.1）、将其替换为新版本（更新，§ 3.3.3），或将 503 转为带标记的 200 页面（恢复，§ 3.3.4）。第二个问题询问第一个答案中没有的代码；源站日志显示服务是否再次抓取。

其中 38 条来自托管服务提供商，62 条来自互联网或电话服务提供商，其中有 12 条来自一家骨干网运营商。约一半的服务提供商位于南美洲。许多电信公司也会在提供接入服务的同时销售托管服务，因此我们无法准确区分两者。

D.2 故障下的源站响应

我们针对一个问题，让源站分别出现每种故障，并统计收到的请求数（表 12）。括号中的数字表示每个问题的请求数，范围涵盖六次运行（三次来自每个源站），“重叠度”标记的是并发请求，因此它们不可能全都是重试。遇到服务器错误和连接中断后，OpenAI 会重复发送请求；Anthropic 几乎每次都只发送一次；而 xAI 无论遇到何种故障，都会发送重叠请求。有两种故障因源站而异：

- 截断响应。我们的服务器在网页尚未完整时就结束了响应。在源站 A 上，Cloudflare 向抓取器返回自己的 HTTP 502，之后抓取器发送了更多请求；在源站 B 上，抓取器收到了不完整页面并将其丢弃。

- 连接中断。连接关闭且未返回任何响应。在两个源站上，抓取器都收到 HTTP 502：源站 A 上由 Cloudflare 返回，源站 B 上由 Caddy 返回。

这是 § 3.1.2 中表 3 的完整版本，展示了主要发现背后的子集。

表 12：对于我们注入的每种故障，每个问题到达我们源站的请求（表 3 的完整版本）。 *：这些请求是并发发出的，因此不可能全部都是重试。在重定向循环这一行中，每次到达都代表一次跳转。

源站响应	OpenAI	Anthropic	xAI
包含 `Retry-After` 的响应			
503，延迟 10 秒	3–4	1	5–7*
429，延迟（秒）	1	1	5–7*
429，延迟（日期）	1	1	6–7*
错误和连接			
403	1	1	5–7*
404	1	1	4–7*
500	3–4	1	5–7*
连接中断	3–5	1–3	6–8*
响应被截断	1–2	1–4	6–28*
异常成功与恢复			
25 秒后返回 200	1	1–3	7*
空的 200 响应	1	1	5–7*
失败两次后返回 200	3	1	5–7*
重定向			
重定向循环	2	10	57–77*

D.3 消费者应用 robots.txt 请求

我们将第3.1.1节中的源站级检查应用于消费者聊天应用：在测试源站上发出首次带标签页面请求之前，该应用是否请求了 robots.txt（表 13）？ A

表 13：该应用在向其测试源站发送首个带标签页面请求之前，是否请求了 robots.txt?

App	先读取?
Mistral	Yes
Gemini	No
Grok	No
DeepSeek	No
Qwen	No
ChatGPT	No†
Claude	不明确†
Meta AI	不明确‡
Perplexity	未抓取网页
Copilot	未抓取网页

†Claude-User 和 ChatGPT-User 在 API 运行期间已访问过此源站。Claude 曾在 51 分钟前请求该文件，但我们无法将该请求归因于其应用试用。‡在 Meta AI 的页面请求之前，曾有一个未标记的 facebookexternalhit 请求，但其地址与页面抓取器的地址不同。

robots.txt 请求无法携带我们的 URL 标签，因此我们无法将其与某个问题关联起来；我们根据用户代理和地址将请求归因于某个应用，而“No”表示没有请求被归因于该应用。

D.4 robots.txt 保留与更新

在用户触发的抓取器中，只有 Anthropic 请求 robots.txt（表 2），因此我们想了解它会保留该文件多久，以及何时再次抓取。为测量保留时长，我们每隔 10 分钟向 Anthropic 提出一个关于新网页的问题，并观察它何时请求 robots.txt。在两次运行中，它都在初次请求后的 60 至 70 分钟再次请求 robots.txt。因此，Anthropic 会将其副本保留约一小时，而添加到 robots.txt 中的规则可能最多需要约一小时才会被看到。

D.5 公共 XML 样式文件

公共文件测试（§ C.3.1）要求每项服务引用每个公共文件（§ C.2）的前三行。图 8 根据文件声明的类型，将每项服务的回答分为四种结果：引用了这些行、部分（仅有 XML 头部，或文件被压平成一行）、拒绝，或 其他（失败或无法得出结论）。按类型划分的发现见 § 3.2.1。

Google 为何拒绝。Google 已读取34%的公开文件，其中大多为完整读取（图 8）；表 14 展示了它未读取其余文件的原因。它有51%的文件读取失败，却没有给出原因；另外15%的文件则因容量限制或超时而未完成读取。它失败时的回答只说无法读取文档，没有一个提到具体原因。当 Google 读取失败时，模型只收到约 200 个令牌，相当于一条错误消息的大小，因此从未看到该文件。

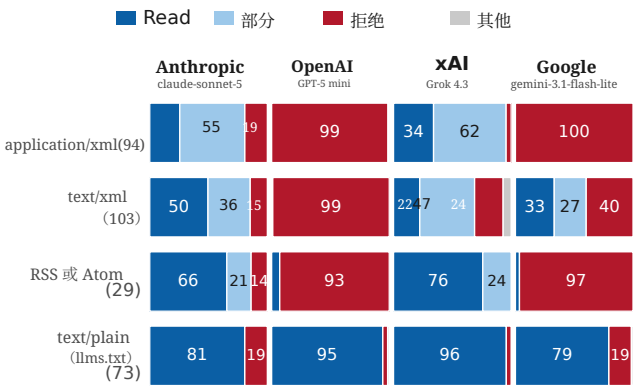


图 8：各服务能否读取各种声明类型的公开文件？数量见括号内。

表 14：Google 和 OpenAI 未读取公开文件的原因。两列均根据答案的措辞分类；Google 的抓取状态也未说明原因。

结果	Google	OpenAI
拒绝，并引用其自身的获取错误	0	192
拒绝，并报告网站错误（403、404）	0	16
拒绝，未说明原因	151	17
引用限制（复述）	0	1
两次尝试均繁忙	44	0
超时或情况不明确	2	1
未读取 ，总计	197	227

D.6 为什么 Google 的 URL-Context 工具在我们的源站上失败

Google 的 URL_context 工具是 Gemini API 从 URL 读取网页的方式。我们将其排除在受控源测试之外，因为它的回答无法显示网页是否已被读取。作为对照，我们还要求 Google 读取 exam- ple.com，这是一个知名网页，任何正常工作的抓取器都能读取。

两个模型，两个问题。使用 gemini-3.1-flash- lite 时，Google 的工具在九次调用中从我们的源站下载了整个网页，但随后报告 ERROR，模型也表示无法接入该网页。example.com 对照组在每次会话中同样报告 ERROR，因此无法证明该工具正常工作。

gemini-2.5-flash 的表现有所不同：失败时，没有请求到达我们的源站。但尽管我们这边没有做任何改动，它的结果却并不一致。在源站 A 上，它连续数周失败后才读取了一次网页；在源站 B 上，则连续数天失败后才读取网页。因此，缺少抓取请求无法区分是缓存所致，还是 Google 自身的失败。该模型也已弃用：从 2026年12月1日 起，Google 会将发往它的请求路由到另一个模型 [23]。在免费套餐中，它每天最多只允许 20 个请求。

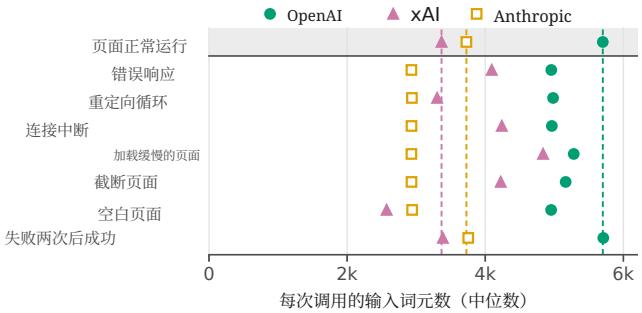


图 9：每种故障下每次调用的输入令牌数。线条标示成功抓取。两个源站的数据合并统计。

容量。 Google 还回应说，在几次调用中它正忙。除了一次外，所有请求都在重试后成功。

复用。我们可以在 gemini-2.5-flash 开始给出答案后进行一次检查。[26] 我们在首次提问后的 5 分钟和 15 分钟，分别通过同一个调用方再次向 Gemini 询问同一个网页。我们的源站每次都观察到 Google 的抓取器重新下载，因此 Gemini 没有保留副本。

D.7 失败抓取的令牌成本

失败抓取仍会消耗输入令牌，我们想了解其成本与成功抓取相比有多高。图 9 展示了每种故障下每次调用的输入令牌中位数，合并统计了两个源站的数据。线条标示对同一网页的成功抓取。

平均而言，失败抓取的成本大约与成功抓取相同。Anthropic 和 OpenAI 的数值略低于该线。xAI 在不同故障下的表现差异很大，因此我们看不出任何规律。

E 其他缓存与复用结果

本附录为 § 3.3 提供补充说明：页面如何在不同 URL 之间匹配（§E.1）、如何在不同调用方之间共享（§ E.2、§ E.3）、如何根据缓存头复用（§ E.4），以及页面会保留多久（§ E.5、§ E.6、§ E.7、§ E.8）。

E.1 URL 等价性

本附录探讨服务会将哪些 URL 视为同一页面。如果服务将两个 URL 视为同一个 URL，就可以用同一份缓存副本为二者提供内容（§ 3.3）。

对于表15中的每项更改，我们先测试一个带标签的 URL，再约 90 秒后测试更改后的 URL；每项均使用新的标签进行三次试验。如果有新请求到达我们的源站，则说明该服务单独抓取了更改后的 URL；如果没有新请求，且答案仍包含该页面的内容，则说明该服务复用了第一份副本。

这些服务在一些变化上达成一致：片段、百分号编码字符、点片段以及重新排序的查询字符串都会在这三种服务中复用第一个副本。它们的差异出现在 URL 语法将决定权留给应用的情况下。当添加跟踪项、重复项名称或空项时，Anthropic 会复用该副本；添加尾部斜杠时，xAI 会这样做；OpenAI 在这些情况下均不会这样做。我们看不出这些差异背后存在任何共同规则。

有两种 URL 写法在检索前就未能通过。对于显式指定的 :443 端口，Anthropic 的抓取工具在三次试验中均返回 url_not_in_prior_- context，尽管模型将 URL（包括端口）原样传给了工具。因此，拒绝来自工具自身的检查，而非模型。OpenAI 报告称，在三次试验中均无法打开主机名首字母大写的 URL；其中一个答案引用的主机名则是小写形式。这些失败既不能说明发生了复用，也不能说明服务另行创建了副本。

添加查询项或移除整个查询时，没有任何服务会复用副本。因此，在查询字符串中加入随机标签（?probe=...），会让三项服务都生成一个新网页。因此，我们在每次试验的 URL 中都加入标签，这样就能反复询问有关同一网页的问题，而不会受到先前回答的干扰（§ 2.3）。

E.2 个性化链接与跨账户泄漏

本附录探讨：由于这些服务会在不同账户间共享页面（§3.3.1），由一个账户的链接个性化的页面是否会被另一个账户访问。页面内容取决于其 URL 中的某个查询项，例如 ?mc_eid=user- 1。账户 A 使用自己的链接询问了有关该页面的问题。五分钟后，账户 B 使用另一个链接（?mc_eid=user-2）询问该页面，随后又移除了该查询项再问一次。如果 B 未经重新下载就收到了 A 的内容，就构成了泄漏。

未发生泄漏（表 16）。OpenAI 和 xAI 保留了 URL 中的每个查询项，因此 B 的提问触发了新的下载，B 收到的是自己的内容。Anthropic 在下载前删除了 utm_content,_hsenc 和 fbclid，因此从未为任何人抓取个性化页面，每个账户都获得了一个通用副本；它将 mc_eid 和 uid 保留为不同的 URL。这与表15中的跟踪项一行相符。我们只通过 API 测试了查询项；未测试 Cookie、标头和消费者应用。

E.3 消费者应用内容复用

我们在八款消费者应用中重复了共享测试，每款应用使用两个独立的免费登录账户 A 和 B。登录账户 A 针对网页 A 提出问题 A1，大约 7 分钟后，又在新聊天中针对该网页提出问题 A2（同一登录账户内的复用）。A1 提出约 14 分钟后，登录账户 B 针对网页 A 提出问题 B1（跨登录账户共享），随后针对一个新鲜网页 B 提出问题 B2。每个问题都要求提供其

[26]免费套餐的每日限额只允许进行少量此类测试。

表15：各服务会将哪些 URL 变更视为同一网页？ =：复用了第一个副本； ≠：重新抓取；E：检索前失败。每项结果在三次试验中均一致。各组依据 RFC 3986 [9, §6]，该规范未规定缓存操作。

变更	编辑示例	OpenAI	Anthropic	xAI
RFC 3986 将等价性留给应用处理				
添加查询项	&x=1	≠	≠	≠
移除整个查询	?探测=t&k=a →无	≠	≠	≠
重新排列查询项	a=1&b=2 → b=2&a=1	=	=	=
添加跟踪项	&utm_source=email	≠	=	≠
重复项名称	&k=b	≠	=	≠
添加空项	&z=	≠	=	≠
添加尾部斜杠	/p → /p/	≠	≠	=
RFC 3986：相同的检索目标				
添加片段	#entries	=	=	=
写入默认端口	:443	=	E	=
百分号编码字母	a → %61	=	=	=
添加点段	/p → /./p	=	=	=
将主机名大写	h.test → H.TEST	E	=	=
RFC 3986：路径区分大小写				
将路径大写	/p → /P	≠	≠	≠

表 16：每个抓取器如何处理查询项？已丢弃：在下载前移除，因此每个账户都获得相同的通用副本。已保留：留在 URL 中，因此不同的值会导致新的下载。未发生泄漏。

查询项	Anthropic	OpenAI	xAI
utm_content	**已丢弃**	已保留	Kept
mc_eid	Kept	Kept	Kept
_hsenc	**已丢弃**	Kept	Kept
fbclid	**已丢弃**	Kept	Kept
uid	Kept	Kept	Kept

表17：消费者应用中的内容复用：网页是否在未向我们的源站发出新请求的情况下得到答案？是：三次运行中均如此；否：一次也没有；否则为运行次数。

App	已复用？ 新聊天，相同登录	不同登录
Claude	Yes	Yes
ChatGPT	Yes	Yes
Gemini	Yes	No
Grok	第 1 个，共 3 个	3 次中的 2 次
DeepSeek	No	No
Mistral	Yes	Yes
Qwen	No	No
Meta AI	No	No

网页对应的不同代码；如果答案中包含正确代码，且在给出答案前没有向我们的源站发送请求，就算作复用或共享。

三个应用都在不同登录之间共享了网页（表 17）：Claude、ChatGPT 和 Mistral 每次都会这样做，而且在同一登录会话中也会复用网页。Gemini 只会在同一登录会话中复用网页。DeepSeek、通义千问和 Meta AI 每次都会重新抓取网页。Grok 的表现不一致：3 次运行中有 2 次在不同登录之间共享网页，另有 1 次在同一登录会话中复用网页。

E.4 缓存头实验：所有设置

本附录列出了我们测试过的所有 Cache-Control 设置，以及各服务对此的处理方式（§ 3.3.2）。我们提供了 19 个大小为 2KB 的 HTML 页面，它们之间仅代码和 Cache-Control 设置不同；随后以 2.5 至 80 分钟、每次间隔加倍的频率，针对每个页面再次提问。

表18展示了对于每种设置，RFC 9111 规定共享缓存应如何处理，以及各服务是否复用了该网页。OpenAI 复用了每个页面约 30 分钟，之后再次抓取；xAI 不论设置如何都复用了每个页面；Anthropic 的做法也相同，但自 10 月17日起，对 no-store 设置例外。

E.5 续期测试

本附录探讨：经常被问及的网页，其存储副本是否会比不常被问及的网页保留更久。对于 OpenAI 和 xAI，答案是否定的：再次提问并未延长副本的保留时间，因此答案有多陈旧并不取决于该网页的访问频率。

我们使用了三个页面。查询过的页面会在其副本应当过期前被询问三次（对于 OpenAI，大约在第 7、14 和 21 分钟；对于 xAI，则是在第 6、12 和 18 小时），而未触碰的页面只会在开始和结束时被询问。较晚的

表18：缓存头实验：再次请求网页时，最多 80 分钟后，是否会直接使用存储的副本作答，而不发送源站请求？每种设置、服务和运行各测试一个网页。 →：结果于10月17日发生变化。Last-Modified：过期 = 过去的日期；近期 = 源站的启动时间；缺失 =没有 Last-Modified 或实体标签。若一行列出多个变体，则每个变体的结果都相同。

			已复用?		
Cache-Control	Last-Modified	RFC 9111 共享缓存规则	OpenAI	Anthropic	xAI
禁止在联系源站前复用					
no-store	过期 / 缺失	不要存储或复用	Yes	是 → 否	Yes
不缓存，no-store，最大有效期=0，必须重新验证	过期 / 缺失	不要存储或复用	Yes	是 → 否	Yes
private	过期 / 缺失	共享缓存不得存储	Yes	Yes	Yes
不缓存	过期 / 缺失	每次重用前验证	Yes	Yes	Yes
最大有效期=0	过期 / 缺失	立即过期；验证或获取	Yes	Yes	Yes
过去过期	过期 / 缺失	立即过期；验证或获取	Yes	Yes	Yes
允许在指定时间内复用					
最大有效期=60	old	新鲜一分钟	Yes	Yes	Yes
最大有效期=60，必须重新验证	old	一分钟后变为陈旧时进行验证	Yes	Yes	Yes
最大有效期=3600，共享缓存最大有效期=60	old	共享缓存：一分钟	Yes	Yes	Yes
最大有效期=3600	old	新鲜时可复用一小时	Yes	Yes	Yes
无重复使用指令					
none	过期／近期／缺失	可能适用启发式新鲜度	Yes	Yes	Yes

页面会在接近结束时首次被询问，以快速检查服务在该时刻是否仍会复用副本。在最终检查时，源站日志会显示服务重新下载了哪些页面。

如果提问会延长副本的保留时间，那么被查询的网页仍会从其副本提供，而未触及的网页则会再次下载。实际情况并非如此：三次运行中，两个网页都被重新下载了；OpenAI 的间隔为 36–43 分钟，xAI 的间隔为 32–33 小时，而较晚的网页仍从其副本提供。

我们没有对 Anthropic 进行这项测试，因为它保留每份 HTML 副本的时间都超过三天（§ 3.3.3），对这项测试来说太久了。

E.6 公共页面的陈旧度

我们还尝试测量真实网页的陈旧度。这项测试适用于 Anthropic，因为它的回复会注明抓取每个网页的时间；但不适用于其他服务：其回答中没有任何信息能说明抓取了什么或何时抓取，因此我们无法有把握地区分过时答案与错误答案。这就是主要的陈旧度结果采用数值已知的公共端点的原因（§ E.7）。我们报告这次尝试，是因为它表明从外部判断新鲜度有多困难。

设计。 我们想知道某项服务是否根据真实网页的过期副本作答。公共页面没有一种易于读取且在各网站间保持一致的可变字段，因此我们首先定位一个这样的字段：锚点，即经常发生变化的部分正上方的一行（§ C.3.2）。随后，每次试验都会要求该服务两次引用锚点之后的文字，并将两次回答与我们自己下载的网页内容进行比对评分（图 10）。

测试问题使用网页的常规 URL，时间是在一个预热问题之后 20 分钟；该预热问题让服务获取了该网页的一份副本。如果网页此后发生了变化，复用这份副本的服务就会给出过期答案。对照问题在同一时间提出，但使用带有随机标签的 URL，这会促使服务再次抓取（§ E.1）。它展示了当前答案的样子，也显示出即使网页是新鲜的，服务仍有多频繁会出错。下载是我们自己对网页发出的请求，而问题则发送给服务；在问题前后进行的下载可以显示网页当时写了什么。我们对 90 个网页中的每一个都执行这一流程（§ 2）。

评判答案。我们仅在一种情况下将答案标记为陈旧：对照答案与网页内容一致，而测试答案不一致。如果答案与我们的下载内容中锚点之后最初的 60 个词（对于日语和中文则为 180 个字符）至少连续共享 5 个词（对于日语和中文为 10 个字符），就视为匹配。匹配结果是可靠的：没有任何答案与我们对另一个网页的下载内容匹配。因此，陈旧答案的数量是一个下限。大多数配对属于其他情况：

- 模型无法读取网页。在 13%（OpenAI）到 46%（xAI）的回答中，模型表示无法读取该网页；这影响每个问题，无论是预热题、测试题还是对照题。

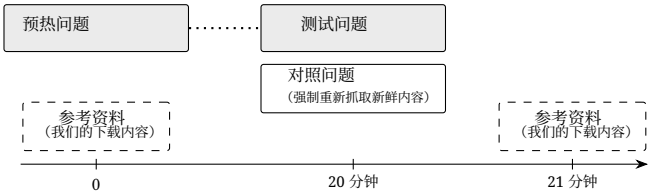


图10：一次公开网页测试。预热问题可能会留下一个副本（虚线），供测试问题复用。虚线框表示我们自己的下载内容，是两个答案的参考资料。

• 模型给出了错误的文本。即使已读取网页，它也可能引用错误的行、进行改述，或引用网页其他位置的内容。

• 网页在对照问题与我们的下载之间发生了变化。如果网页经常变化，我们的参考下载可能会获取到与对照问题所看到的不同版本，因此无法据此核对对照答案。

作为可信度检查，我们还统计了对照未命中：即测试答案与参考下载相匹配、但对照答案不匹配的样本。对照答案本应始终匹配，因此这些样本显示了答案因陈旧度以外的原因未命中的频率；只有当过时答案明显多于对照未命中时，我们才将其归因于陈旧度。

Anthropic。 Anthropic 是唯一能让我们检查方法的服务，因为它的回复会提供其抓取每个网页的时间以及收到的文本。我们首先检查基线。对照答案背后的每个副本都是新鲜抓取的，因此随机标签确实会促成新的抓取。我们的过时标签则较弱：精确率为 76%，召回率为 50%，因此会少计。随后，我们询问测试答案背后的副本有多旧。其中 65% 的副本是 Anthropic 在我们提问前一小时以上抓取的，而这些网页经常变化。[27] 因此，Anthropic 的回答已过时，这与反向端点（§ E.7）一致。我们的标签也得出了相同结论：20 个网页陈旧，而 3 个为对照未命中（精确二项$p = 0.0005$）。对于 21% 的网页，Anthropic 没有返回副本，主要是因为它拒绝访问该网站（url_不_允许），这就是上文所述的无法读取问题。

其他服务。 对于其他三个服务，表 19 中的计数无法解决这一问题，因为过时答案并没有明显多于对照未命中。xAI 倾向于出现过时答案（12 次，而对照未命中为 4 次，$p = 0.077$）；OpenAI 则为 10 次对 9 次，Google 为 17 次对 13 次。许多组根本无法进行比较，因为由于上述原因，两个答案都与网页不匹配。

[27]我们还核实了过期副本确实与实时网页不同。我们测量每份副本文本中有多少出现在我们自行下载的网页内容中，该下载是在提问时获取的。副本越旧，重叠度越低：不到一小时的副本中位数为 64–69%，接近对照答案的 70%；而一小时至七天的副本中位数为 45–48%。

表19：每项服务针对 90 个页面的测试答案和对照答案，与我们的参考下载进行比对。p：对过时答案与对照漏答进行精确二项检验。

结果	匹配 测试	对照	Anthr.	OpenAI	xAI	Google
陈旧	No	Yes	20	10	12	17
对照未命中	是	否	3	9	4	13
两者均匹配	是	是	21	43	23	7
两者皆非	No	No	44	28	51	48
不可用	–		2	0	0	5
p			0.0005	1.0	0.077	0.58

为什么我们无法看到更多信息。除 Anthropic 外，这些服务几乎没有告诉我们它们抓取了什么。Anthropic 会报告获取时间和错误代码。Google 只报告状态，即成功或错误，但不说明原因；其错误比例大致与“无法读取”回答的比例相符（20% 对 23%）。OpenAI 和 xAI 是通过 OpenRouter 调用的，它们除了提供网页引用外什么也不报告。即使随后表示无法读取网页，它们仍会附上该引用：xAI 几乎在每个回答中都引用网页，但仍有 46% 的回答称其无法读取网页。因此，引用并不是一个可用的失败信号：无论服务是否成功获取网页，都会出现引用。

一项我们无法采用的检验。我们还尝试将每个回答与其他服务的新鲜答案进行比较，而不是与我们自己的下载内容比较。此方法行不通，因为各项服务的回复都以相似的套话开头，改述方式也相似，因此即便谈论的是不同网页，回答之间也会有重叠度。这些共同措辞掩盖了任何真正的一致之处。

这说明了什么。对于真实网页，如果没有获取时间或可靠的错误报告，外部观察者就无法有把握地区分过时答案和错误答案。只有 Anthropic 会提供这些信息，即便如此，我们的标注仍低估了这一情况。

E.7 公共端点陈旧性：所有端点

本附录扩展了 § 3.3.3 中的端点结果。与文本变化难以预测的公共页面（§ E.6）不同，这些端点返回的数值只会增加，因此每个答案都可以对照我们的参考日志进行核验和确定日期。下文将介绍我们如何测量陈旧度，展示所有端点的结果，并解释不符合常见模式的单元格。

我们每 45 分钟向每个服务询问八个公共端点的当前值，持续约三天。为了确定回答的时间，我们每五分钟自行轮询一次这些端点，因此无法判断陈旧度是否低于五分钟。对于 OpenAI、xAI 和 Google，陈旧度指它们提供的值从何时起不再是实时值；Anthropic 会报告它抓取每个网页的时间（retrieved_at），因此我们采用

表20：我们探测的公共端点：各端点发送的 Cache-Control 标头、其背后的主机或内容分发网络（根据响应头推断），以及其值每增长一个单位所需的时间。我们根据轮询间隔内的增长幅度估算并取整。美国地质调查局的报告提供时间戳，因此其值会持续变化。

端点	Cache-Control	主机/内容分发网络	每隔多久增长一次
Hacker News	不缓存	Firebase	7 s
NIST 信标	no-store	Cloudflare	1 分钟
比特币打赏	最大有效期=1	nginx	5 分钟
Stack Overflow	private	Cloudflare	2 分钟
crates.io	最大有效期=15	Heroku	2 分钟
drand	最大有效期=29	nginx	30 s
USGS 地震数据	最大有效期=60	CloudFront	连续的
以太坊高度	—	Cloudflare	12 s

自该次抓取以来的时间。两项数据未纳入统计。Anthropic 几乎每轮都拒绝抓取 Stack Overflow：其抓取工具报告该 URL 不被允许（url_not_allowed），这是政策拦截，而非失败获取 [3]。Google 对美国地质调查局的回答未纳入统计，因为该值与 Feed 中出现过的任何值都不匹配：它的抓取在每一轮都报告错误，而它给出的时间戳与提问时间一致，唯独年份早了两到四年。

如图 11 所示，在大多数端点上，Google 的中位陈旧度接近于零，OpenAI 约为一小时，xAI 约为八小时，Anthropic 则约为十一小时。例外情况分为两类，具体取决于抓取的结果：

抓取成功，但返回了过期值。OpenAI 在 crates.io 上连续多轮返回相同的值，远远超过其回答通常滞后的一小时，因此其尾部陈旧度约为 40 小时。Anthropic 在 NIST 信标上的情况更为突出：大多数轮次中，它返回的都是 10 月 24 日抓取的副本，比我们用于对比的轮次早了十多天，尽管该网页发送了 no-store。在其他轮次中，它的抓取返回了实时值或遭到拒绝。[28]这个旧副本来自抓取工具自身的缓存；Anthropic 的文档对此有说明，并指出缓存可能返回非最新内容 [3]。作为快速检查，我们还确认 Anthropic 能够抓取到新鲜网页。[29] 我们无法调整服务器端设置，因为 NIST 并非由我们运营。

失败获取后的过期值。Google 在 Stack- Overflow 和以太坊高度上的长尾（最长约 60 小时）来自这样的轮次：其抓取报告了错误，并返回了过期值。成功抓取后，其中位数答案在几分钟内就与实时值一致；失败获取后，则落后了一天以上。

[28]这与我们的受控测试仅部分吻合：Anthropic 每次都会抓取 no-store 页面，获取新鲜内容（表 7）。[29]使用该工具的最新版本（web_fetch_20260309 或更高版本）并将 use_cache 设置为 false；其他所有运行次数均使用基础版本。

E.8 抓取新内容后的过期回答

OpenAI 在抓取新版本后，常常仍根据过期版本作答。过期答案引用的是网页的过期版本，尽管该服务在同一问题期间已经抓取了新版本；更新后的答案引用的是新版本。在网页更新和错误修复测试中（图 3），OpenAI 在后续问题中约 60% 的情况下给出了过期答案。Anthropic 从未出现这种情况，而 xAI 在错误修复测试的后续问题中有 2% 的概率如此。

过期的回答成本更低、速度更快，仿佛没有读取新的网页。与更新后的回答相比，过期回答发起的搜索调用更少，使用的输入词元少了约三分之一，耗时也更短。在错误修复测试中，过期回答仍然报告原始错误，并且从未引用恢复后的页面。在唯一一次回答泄露工具为其打开的页面所使用的内部名称（turn. . . view. . .）的情况下，它引用了旧版本的代码，尽管我们的源站在同一个问题中提供的是新版本。

一种可能的解释是：工具读取的是一份已存储的副本，而抓取操作会刷新该副本。我们无法查看 OpenAI 工具的内部机制，因为它的推理经过加密，浏览步骤也处于隐藏状态。一种解释符合我们的观察：工具根据它已经持有的一份副本生成答案，而我们看到的抓取操作只会为下一个问题刷新该副本。以下三点观察与此相符：

- 成本。过期答案成本更低，仿佛没有读取新网页；更新后的答案成本更高，仿佛第二次调用读取了刷新的副本，并将其内容加入输入。
- 时序。无论答案过期还是更新，我们的源站每个问题都恰好收到一次抓取请求，时间约在三秒后，因此抓取并不决定答案。在错误修复测试中，我们在修复后继续提问，约 25 分钟内，OpenAI 仍给出原来的错误答案，且没有向我们的源站发送任何请求。
- 相似之处。它看起来像是在重新验证时提供陈旧副本 [51],; OpenAI 的开源参考浏览器也会保留已打开的页面 [53]。

没有任何证据表明，已部署的服务也以同样方式运行。

如果这一推断正确，而且存储的副本在不同账户间共享（§ 3.3.1），那么由某个调用方的问题触发的抓取，主要让下一位调用方受益：第一个人收到基于旧副本的答案，之后的人则收到基于刷新副本的答案。对发布者而言，源站日志中的一次抓取并不能证明该网页已送达触发抓取的用户；内容变更也可能要等到用户提问一次之后才会送达。

F 利益相关方影响依据

表1从四类利益相关者的角度对每项发现进行评价；本附录说明每项评价背后的理由。我们根据各利益相关者所关心的内容进行判断。网页出版商关心源站的负载、对其内容复用方式的控制，

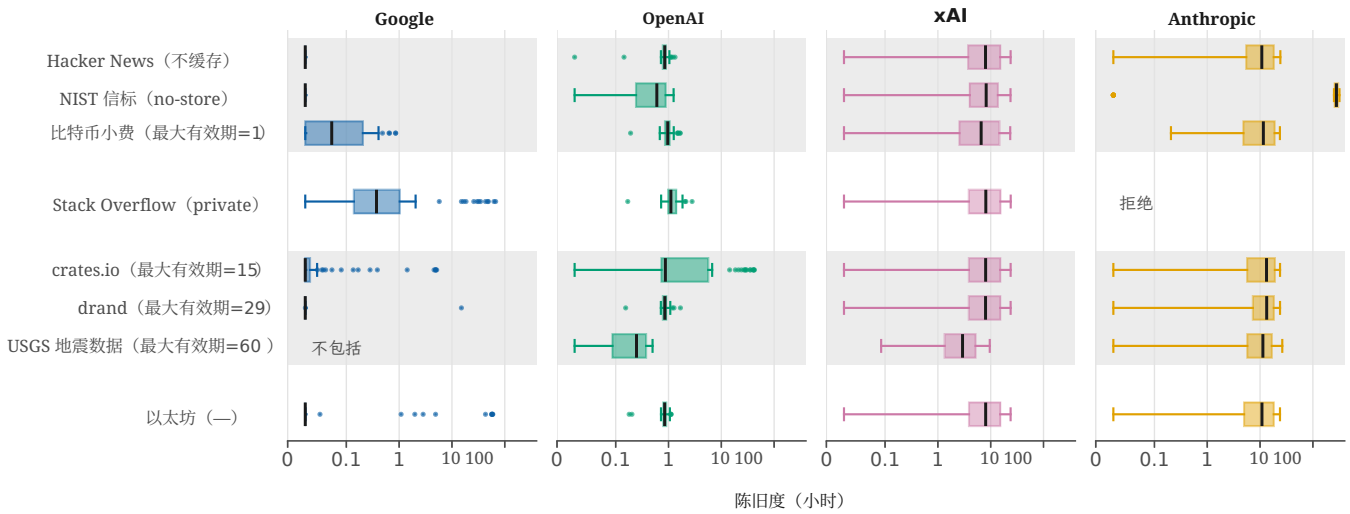


图 11：八个公共端点上，各服务的答案比实时值滞后多少。阴影按 Cache-Control 标头对端点分组。箱体表示四分位数，线表示中位数，点表示离群值。对大多数服务而言，中位数不会随标头而变化。Anthropic 拒绝回答有关 Stack Overflow 的问题，Google 关于美国地质调查局的答案则未纳入统计。

以及其页面是否可以访问。大语言模型供应商关心抓取成本；我们不对供应商的回答质量进行评价。终端用户关心答案是否正确、新鲜，以及是否清楚说明来源和成本。内容分发网络和基础设施运营商关心缓存效率，以及他们必须承受的流量。在此意义上，评价分为负面、正面或褒贬不一；如果某项发现对某类利益相关者没有影响，则相应单元格留空。这些评价反映的是我们对测得行为的判断，而非对危害的测量。表21为每个经过评价的单元格各设一行。

G 开放科学

我们在公开代码库中发布了本研究的相关材料：代码见 https://github.com/sands-lab/aisearch-probe，数据见 https://github.com/sands-lab/aisearch-data。

- 测试页面和探测代码。合成页面（附录C）、提示词，以及发送每个问题并检查答案的代码。
- 源站日志。包含我们 URL 标签的请求记录。每个源地址都替换为随机 ID，并保留网络编号、所有者和国家/地区（§ D.1）。
- 供应商回答。结果中使用的每条回答，包括每个问题及其时间，以及 Anthropic 提供的获取时间（如有）。
- 公共网站数据。对于公共页面测试，包含参考下载数据的值和时间，以及每家供应商的答案；对于公共计数器，则包含每个值及其时间。

每个网页的代码都以表格形式发布，方便读者重新核查答案是否引用了该网页；生成这些代码的秘密并未公开。我们不会公开 API 密钥或账户详情。

表21：表1中各单元格评级的原因。括号中的引用是用户报告的实际问题，并非此前测量研究中观察到的问题。

发现	利益相关方		推理
跳过 robots.txt 的 OpenAI 和 xAI（第3.1.1节）	发布者	–	robots.txt 是网站列出机器人可以读取哪些内容的地方。OpenAI 和 xAI 从不请求该文件，因此网站无法通过这种方式限制它们。
被忽略的 Retry-After：OpenAI 和 xAI（§ 3.1.2）	发布者	–	服务器可以发送 Retry-After，要求客户端等待一段时间后再重试。OpenAI 和 xAI 忽略了它，因此负载过重的网站得不到任何缓解。
	内容分发网络 / 基础设施	–	网站前面的内容分发网络无法以这种方式减慢这些抓取器的速度，因此只能自行承受它们带来的额外请求。
xAI 发送并行请求（§ 3.1.2）	发布者	–	一个用户问题会使 xAI 同时发送多个请求，从而成倍增加负载，作用于该网站。
	内容分发网络 / 基础设施	–	这些请求来自许多不同的地址，且看起来像是由浏览器发出的，因此内容分发网络几乎无法辨别它们与访客请求的区别，也难以限制它们 [65, 70]。
Anthropic 遇到一次失败后便停止（§ 3.1.2）	发布者	±	Anthropic 只发送一次请求便停止，因此几乎不会增加负载；但原本能够正常显示的网页在重试时失去被读取的机会。
	终端用户	–	发生临时错误后，Anthropic 不会重试，而 OpenAI 和 xAI 会重试，因此其答案可能会遗漏原本可访问的网页（例如 [67]）。
OpenAI 会丢弃某些格式（§ 3.2.1）	发布者	–	OpenAI 的请求接受任何格式，但抓取后会丢弃以 Markdown、XML、CSV 或 DOCX 提供的页面。发布者发送了网页，但它从未将其告知答案。
	终端用户	–	答案所依据的来源少于用户的预期，而且没有任何信息告知他们某个页面被遗漏了。
非仅脚本内容 已读取 (§ 3.2.2)	发布者	–	没有任何服务会读取网页通过 JavaScript 添加的文本，或放在脚本块中的文本（例如 JSON-LD，约 40% 的热门主页都在使用）。
	终端用户	–	答案缺少浏览网页的人会看到的内容。
xAI 只读取最前面的 0.5 MB（§ 3.2.3）	发布者	–	xAI 请求整个网页，但只读取开头的 0.5 MB，因此其余内容会被发送，然后从答案中丢弃。
	供应商	+	限制已读取内容的上限，可以控制模型的输入量，从而将每个答案的成本控制在一定范围内。
	终端用户	–	答案省略了长网页的其余部分，却没有说明这一点（例如，[10]）。
输入成本随以下因素增长 网页大小（§ 3.2.3、§ 3.2.4）	终端用户	–	使用 Anthropic 时，针对一个 2 MB 的网页提出一个问题，可能会消耗约 76万词元；而一次失败获取所耗费的资源几乎与成功获取一样多。调用方在提问前无法预知这一点。
在多个账户之间共享的页面 账户（§ 3.3.1）	发布者	±	为回答某人的问题而抓取的网页会被其他人复用，因此网站收到的请求会减少。但如果网站按抓取次数收取费用，一次抓取便可服务于许多答案 [63]；网站是否因每次使用而获得归因，取决于供应商的归因方式，而非共享本身。
	供应商	+	抓取次数减少，抓取成本也会降低。
	终端用户	±	你的答案可能基于为他人抓取的网页。我们没有发现个性化链接泄露，但我们只测试了在 URL 中个性化的链接，而且这些服务并未说明页面的共享范围有多广（例如，[49]）。
	内容分发网络 / 基础设施	+	到达内容分发网络的同一网页重复请求减少了。
网站的缓存规则遭到忽略（§ 3.3.2、§ 3.3.3）	发布者	–	Cache-Control 是网站声明其网页可以复用多长时间的方式，而服务大多会忽略它。更新需要 30 分钟到一天甚至更久才会反映到回答中，而网站无法改变这一点。
	供应商	+	忽略它意味着抓取次数少于遵守较短时限所需的次数。
	终端用户	–	答案可能来自旧副本，用户不会被告知其时效，也无法要求重新抓取新鲜内容（例如，[31]）。
	内容分发网络 / 基础设施	–	内容分发网络依靠这些标头判断副本是否已过期；人工智能服务会绕过这些标头，并采用自己的副本存续期限 [63]。
失败的页面仍然处于失败状态 (§ 3.3.4)	发布者	–	页面出错并修复后，服务仍会像它没有修复一样继续回答，导致正常的页面仍然不会出现在回答中。
	终端用户	–	答案称网页已无法访问，但实际上仍在运行。
OpenAI 从一个过期副本（§ E.8）	发布者	–	OpenAI 经常抓取网页的新版本，却仍然根据过期版本提供回答，因此在网站日志中出现抓取记录，并不意味着用户看到了该网页。
	终端用户	–	尽管网页刚刚刷新过，答案仍然是过期的。