北京大学
PEKING UNIVERSITY

智能体可读文档

——标准、实践与趋势

高志军

https://www.gaozhijun.me

2026 年 2 月

前言

AI 智能体正成为技术文档的重要使用方，这促使技术文档从“人阅读的参考资料”加速演变为“可被模型检索与执行的交互接口”。Mintlify 最新监测显示，在其托管的文档站点中，**近 48% 的访问流量来自 AI 系统/智能体**；过去一年内，相关访问量由 34.2 万次增长至 990 万次 [1]。此外，面向模型消费的入口文件 llms.txt、用于模型与工具/上下文连接的模型上下文协议（MCP），以及用于约束智能体行为的项目级指令文件 AGENTS.md 等新兴规范快速涌现，进一步强化了”文档即接口”的产业趋势。

在该趋势下，**文档质量已成为智能体性能的直接瓶颈**：研究表明，当文档足够简明且结构清晰时，能力较弱的模型在整体任务表现上亦可能超越依赖低质量文档的更强模型 [2, 3]。然而，与人类用户不同，智能体在遭遇结构混乱、信息缺失或指令不一致等文档问题时往往**静默退出**，难以形成可追踪的反馈信号，从而削弱了文档迭代的可观测性与可评估性 [1]。

本报告围绕新兴标准与结构化实践、面向智能体的写作适配、行业实施路径与学术研究进展进行系统梳理，并提出面向“人机双重受众”的文档建设要点，为构建可检索、可执行、可评估的文档体系提供参考。

目录

1 智能体可读的相关新标准

自 2024 年末以来，面向智能体的文档可消费（machine-consumable）标准呈现爆发式增长。尽管各类方案覆盖的场景各异，但其共同指向相对一致：**文档应可被机器稳定检索与解析，以 Markdown 作为天然载体，并具备清晰、可预测的语义结构** [10, 11]。

1.1 llms.txt：应用最广的 LLM 文档标准

llms.txt 由 Answer.AI 联合创始人 Jeremy Howard 于 2024 年 9 月提出 [4]，目前已成为采用最广的面向 LLM 的站点级文档约定。该规范要求在网站根路径提供 `/llms.txt` 文件，用于给出结构化目录，并为各文档页面提供可直接消费的 Markdown 版本链接。配套规范 **llms-full.txt**（由 Mintlify 与 Anthropic 合作推动）[12] 则将站点文档拼接为单一文件，便于一次性注入模型上下文。其推荐结构包括：H1 项目名称、blockquote 形式的摘要、可选的补充描述段落，以及按 H2 分类组织的链接列表（含简要说明）[4]。

```
# Acme API Documentation              % H1: 项目名称
> Acme provides a REST API for        % blockquote: 摘要
> building payment workflows.

Optional description paragraph...

## Getting Started                    % H2: 分类标题
- [Authentication](/docs/auth.md): ... % 链接 + 描述
- [Quick Start](/docs/quickstart.md): ...

## API Reference
- [Payments](/docs/payments.md): ...
- [Webhooks](/docs/webhooks.md): ...
```

项目标识

一句话摘要

分类链接列表

图 1: llms.txt 文件结构示意——H1 项目名称、blockquote 摘要、H2 分类下的 Markdown 链接列表 [4]

截至 2025 年底，BuiltWith 的追踪数据显示已有 **84.4 万 +** 网站部署该约定 [5]，覆盖 Anthropic、Stripe、Cloudflare、Vercel、Hugging Face、NVIDIA 等知名组织 [13]。值得注意的是，Google 的 John Mueller 指出，目前尚无主要 AI 平台正式宣布会专门抓取 llms.txt 文件 [5]。即便如此，llms.txt 提供的 Markdown 版本通常较 HTML **节省约 30%** 的 token 消耗，并可在一定实验设置下带来超过 7% 的准确率提升 [14]。

1.2 MCP：智能体—工具交互的运行时标准

模型上下文协议（MCP）由 Anthropic 发起，并于 2025 年 12 月捐赠至 Linux 基金会旗下 Agentic AI Foundation [6]，目前已成为智能体—工具交互的重要事实标准。MCP 围绕三个核心原语组织能力：工具（可执行函数）、资源（可访问的数据上下文）与提示（可复用的模板），并以 JSON Schema 描述其结构与输入约束 [15]。客户端通过 `tools/list` 发现工具，通过 `tools/call` 发起调用并接收返回 [15]。协议支持 `stdio`、SSE 与 Streamable HTTP 等多种传输方式 [7]。

MCP 的生态扩张迅速：OpenAI [16]、Google DeepMind 及 Cursor、Zed、Replit、Sourcegraph 等工具链均已支持 [7]。其中一个关键趋势是 **OpenAPI → MCP 的桥接转换**：Speakeasy、FastMCP、

openapi-mcp-generator 等工具可将既有 OpenAPI 规范自动生成 MCP 服务器 [17]，使智能体能够以较低迁移成本访问各类既有 API 能力。

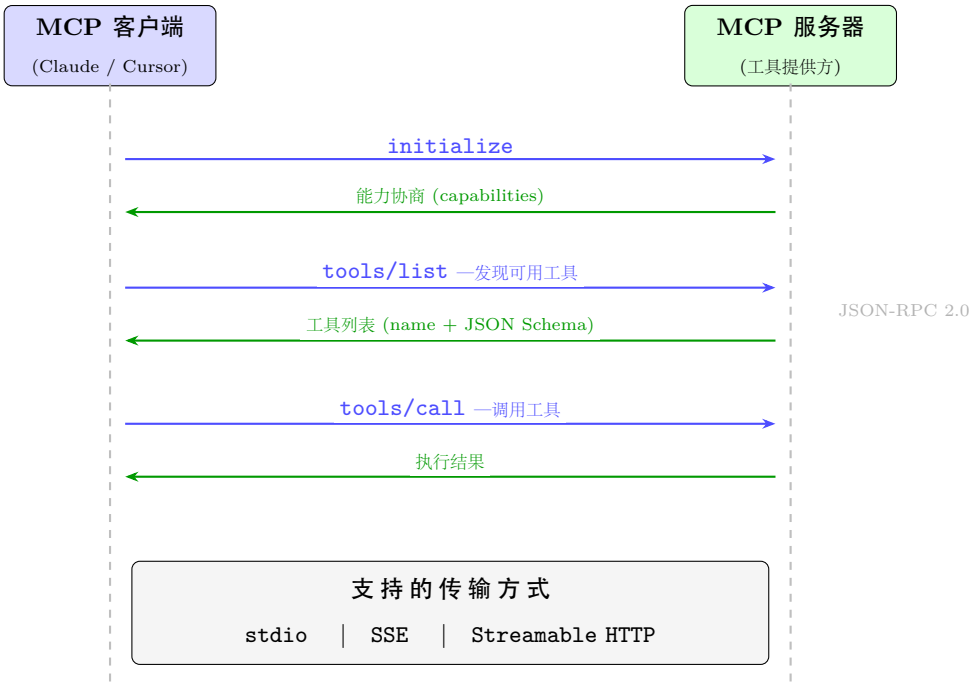

图 2: MCP 客户端—服务器交互流程：通过 `tools/list` 发现工具，并以 `tools/call` 发起调用

1.3 AGENTS.md：面向编码智能体的项目级指令

AGENTS.md 面向的核心场景是：为编码类智能体提供项目级、可执行的工作指令。该约定由 Linux 基金会 Agentic AI Foundation 维护，通常要求在代码仓库根目录提供同名 Markdown 文件，用于集中描述构建命令、架构概览、编码规范与安全注意事项等关键信息 [8]。目前已有 **6 万 +** 开源项目采用，OpenAI Codex、GitHub Copilot、Google Gemini、Cursor、Windsurf 等工具链提供支持 [9]。相近实践还包括 Claude Code 采用的 `CLAUDE.md` [18]，以及 Cursor 使用的 `.cursor/rules`（MDC 格式）[19]。这类项目级文件代表一个正在成型的新类别：**主要为 AI 消费而写、以智能体为第一读者的“执行型文档”** [8]。

1.4 OpenAPI：智能体与外部 API 的关键桥梁

OpenAPI 规范仍是智能体对接外部 API 的关键桥梁 [17]。面向智能体的 OpenAPI 最佳实践通常包括：为组件与端点提供足够语义化的描述（不仅说明“做什么”，也说明“何时用/为何用”）、给出完整的请求/响应示例、保持命名与字段模式一致、尽量采用扁平化的数据结构、使用机器友好的认证方式（如 OAuth 2.0 客户端凭据或静态 API Key），并以 `application/problem+json` 形式提供可程序化解析的错误说明 [20]。此外，**OpenAPI Slimmer** 等工具可将大型规范裁剪为仅含相关端点的最小子集 [21]；LangChain 的 OpenAPI Toolkit 则采用层级化规划策略——Planner 先读取规范概览，Controller 再按需加载所选端点的细节 [22]。

1.5 其他重要标准

其他值得关注的方案包括 **Context7**（Upstash 开发）：其对 9,000+ 个库的文档进行索引与向量化，以支持语义检索，并可通过 MCP 工具向开发环境提供服务 [23, 24]——开发者只需在 Cursor 或 Claude Code 的提示中添加“use context7”。**TDMRep**（W3C）则聚焦 AI 训练阶段的版权与权利表达 [25]；但其采用规模有限，全网实现不足 300 个 [26]。

Context7 代表一种正在成型的“Doc-as-Context”范式：以 MCP Server 的形式提供版本化官方文档检索与片段注入，将“实时文档”能力标准化到工具调用接口层。因此，它更适合作为 MCP 生态中的关键实现与新类别，而非与 llms.txt/MCP/AGENTS.md 并列的独立协议标准。

1.6 标准对比总览

表 1: 主要标准对比

标准	用途	格式	发起方	采用量
llms.txt / llms-full.txt	面向 LLM 的网页文档入口与目录	Markdown（`/llms.txt`）	Jeremy Howard / Answer.AI [4]	84.4 万 + 网站 [5]
MCP	智能体—工具运行时协议	JSON-RPC（多传输）	Anthropic → Linux 基金会 [6]	主流平台与工具链支持 [7]
AGENTS.md	编码智能体的项目级指令	仓库根目录 Markdown	Linux 基金会 / 社区 [8]	6 万 + 仓库 [9]
OpenAPI	面向智能体的 API 描述与契约	YAML/JSON	OpenAPI Initiative [17]	普遍使用
Context7	“实时文档”作为上下文服务	MCP 服务器 + API	Upstash [23]	9,000+ 库已索引 [23]

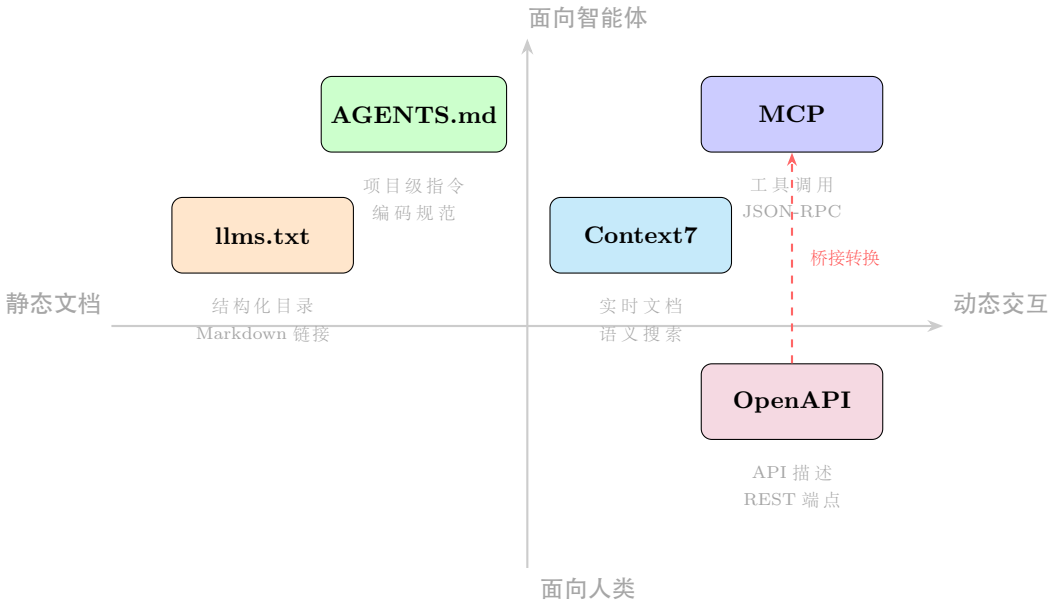

图 3: 面向智能体的文档标准生态关系图——五大标准在“静态/动态”与“人类/智能体”两个维度上的定位

2 智能体可读的文档

面向 AI 智能体写作的根本转变可概括为一条原则：**“每一页都是第一页（Everypage is Page One）”** [27]。与人类读者能够顺序阅读并逐步建立上下文不同，智能体往往通过检索获取**彼此孤立的文本片段**后再进行推理与执行。因此，每个文档页面都应具备**自洽性与可独立使用性**：关键信息不依赖隐含前文，必要的交叉引用显式给出，参数与约束条件完整说明。HumanLayer 的研究发现，在“合理一致”的前提下，先进 LLM 大致可稳定遵循 **150–200 条指令**；超出该范围后，遵循度与任务表现会出现下降 [28]。因此，面向智能体的文档应保持聚焦、避免冗余，并将规则收敛为少量高优先级的“硬约束”。

2.1 提升 LLM 理解的写作模式

以下写作模式被反复证明有助于持续提升 LLM 的理解与检索效果 [27, 29]：

- **清晰的描述性标题**：使用“WebSocket 认证流程”而非“概述”，以增强语义匹配并提升检索命中；
- **短段落与单一主题**：每段控制在 3–5 行，且每段只表达一个要点，便于分块与重组；
- **术语一致性**：同一概念在不同页面分别称为“API 密钥”“访问令牌”“凭据”会削弱跨页对齐能力，导致模型难以稳定关联相关信息 [30]；
- **可直接运行的完整示例**：示例包含 import、配置、关键边界条件与错误处理，减少模型对缺失步骤的猜测与补全；
- **图像与 UI 的文本等价物**：为截图、流程图与界面控件提供完整文字描述，因为许多 AI 系统无法可靠解析视觉内容。

此外，面向智能体的网页文档应优先确保**可静态渲染（static rendering）**：AI 爬虫通常难以稳定处理依赖 JavaScript 动态生成的主体内容 [27]。

2.2 结构化数据标注

结构化数据标注可作为高质量自然语言正文之外的“机器可读锚点”。JSON-LD 结合 Schema.org 词汇仍是 Google 推荐的结构化数据格式 [31]。常见类型包括：用于文档页面的 `TechArticle`、用于分步指南的 `HowTo`、用于问答内容的 `FAQPage` [27]。同时，较为丰富的 YAML frontmatter（如标题、摘要、分类、难度级别、前置条件、标签、版本/最后更新时间等）可在内容展开前向检索与生成系统提供**可显式利用的上下文线索**，从而提高路由、过滤与回答的一致性 [29]。

2.3 分块与 RAG 策略

分块与 RAG 领域近年来进展显著。随着上下文窗口扩展至 100 万 + token，以及嵌入模型开始支持 128K+ token（如 Cohere embed-v4.0）[32]，并非所有文档都必须分块。对于约 15 万 token 以下的语料库，直接上下文注入在部分任务设置中可能优于 RAG [33]。当确需分块时，可按复杂度由低到高采用以下策略层级：

- **固定大小分割**：以 200–500 token、约 10% 重叠作为起点；
- **递归分割**：优先按节标题切分，再向段落级递归细分；

- **语义分块**：利用嵌入相似度或语义边界识别确定切分点；
- **父子方法**：小块用于检索命中，大块父文档用于提供生成所需上下文 [34, 35]。

更进一步的**智能体分块**方法，则由 AI 智能体根据任务与文档结构动态推断最优边界与粒度 [35]。

图 4: 分块策略从简到复杂的层级——复杂度递增，适应性递增

Databricks 建议以 **200–500 token/块**作为经验起点 [36]。Pinecone 的 Roie Schwaber-Cohen 指出，将块大小与预期查询粒度相匹配，有助于提升相似度检索得分与命中质量 [34]。在 RAG 管道中，元数据设计通常服务于四类关键功能 [37, 36]：

- **过滤**：将搜索限定于特定文档类型/版本；
- **排序**：优先返回更新内容；
- **上下文丰富**：在提示中携带标题、章节与日期等线索；
- **审计**：追溯特定回答所依据的来源与版本。

3 知名企业的文档重构实践

最具指导意义的案例来自那些不仅“添加 llms.txt”，而是进一步从**信息架构、交互形态与智能体接入方式**上系统性重构文档体系的企业。

3.1 Stripe：多渠道交付的典范

Stripe 采取“多渠道交付（multi-channel delivery）”策略：除传统 HTML 文档与 llms.txt 外，还提供 **Agent Toolkit**（开源 SDK，支持 OpenAI Agent SDK、LangChain、CrewAI、Vercel AI SDK）[38]、位于 `mcp.stripe.com` 的远程 MCP 服务器（基于 OAuth 授权访问），以及可在 Copilot 聊天中通过 `@stripe` 调用的 VS Code 扩展。其专门页面 `docs.stripe.com/agents` 进一步将“计费工作流”以智能体视角组织为可执行的指导路径 [39]。Stripe Sessions 演讲中，Emily Sands 指出：“我们长期投入优秀文档建设……如今，LLM 正基于这些文档，使信息查找更加便捷”[40]。

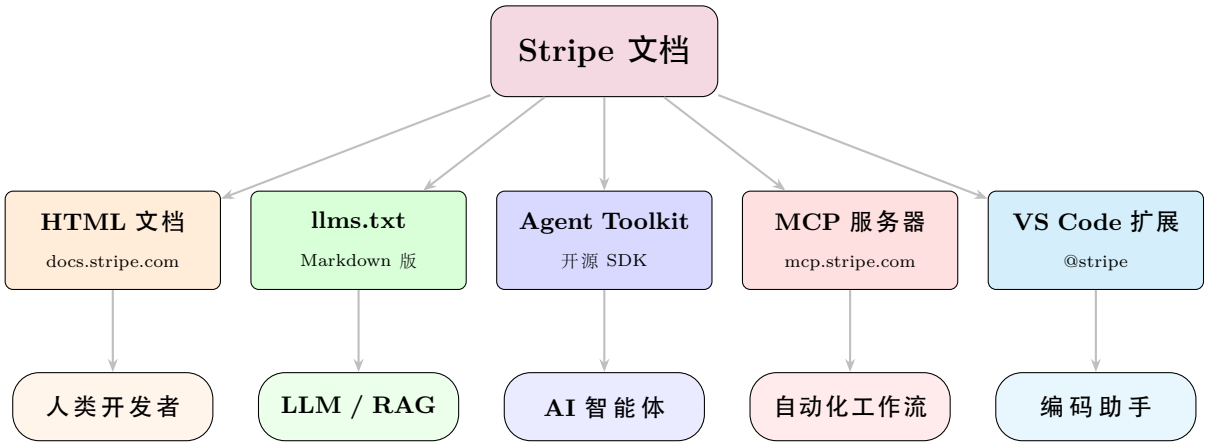

图 5: Stripe 多通道文档架构——通过五个通道服务不同类型的消费者 [38, 39]

3.2 Anthropic：工具描述优化的开创性工作

Anthropic 发布里程碑式文章“Writing effective tools for agents”，并迅速成为行业参考 [41]。其核心结论是：**工具描述的细微改进也可能带来显著的端到端性能收益**。通过对工具命名、描述语义与响应结构的精细优化，Claude Sonnet 3.5 在 SWE-bench 上达到最先进水平 [41]。

建议包括：工具命名采用命名空间（如 `asana_search`、`jira_search`），优先构建**少量但能力更强**的工具而非大量薄封装，并优化工具响应以提升 token 效率 [41]。其 **Tool Search Tool** 机制通过将工具标记为 `defer_loading: true` 实现按需发现，使 token 使用量降低 **85%**，同时将 Opus 4 的准确率从 49% 提升至 74% [42]。

3.3 Mintlify：AI 原生文档平台的引领者

Mintlify 已成为领先的 AI 原生文档平台，为 Anthropic、Coinbase、HubSpot 等托管文档 [43]。平台可为托管站点自动生成 llms.txt、llms-full.txt 以及 MCP 服务器 [44]，并执行段落级索引以支持向量相似度检索；其 AI Assistant 已处理超过 **100 万次**即时查询 [45]。同时，AI Writing Agent 监控代码库，在代码变更时通过 Pull Request 提示并建议文档更新 [11]。客户 **Vercel** 报告称 **10% 的注册来自 ChatGPT** [11]，表明“面向 AI 优化文档”可转化为可量化的商业结果。

Mintlify 于 2026 年 2 月披露的数据进一步显示该趋势正在加速：其平台上**近 48% 的文档流量已来自 AI 智能体** [1]。如图 6 所示，在 2024 年 12 月至 2025 年 12 月期间，AI 访问量从 34.2 万次增至 **990 万次**（约 29 倍增长），同期总访问量则由 230 万增至 2,080 万 [1]。该文特别强调一个关键差异：**“当智能体失败时，你不会收到任何反馈——没有跳出率警报，也没有愤怒的支持工单”** [1]。智能体在无法定位所需信息时往往静默离开，甚至可能将用户导向竞争对手；因此，企业需要将 AI 智能体视为文档体系的**“一等公民”（first-class citizen）**，并据此调整信息架构、可观测性与质量保障机制 [1]。

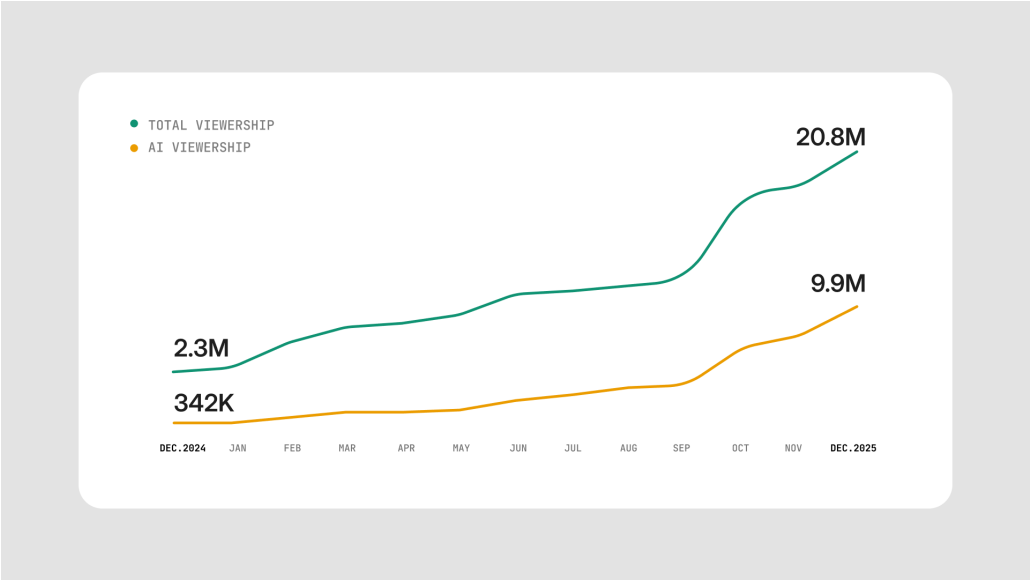

图 6: Mintlify 平台文档流量趋势（2024.12–2025.12）：AI 访问量（橙色）从 34.2 万增至 990 万，占总流量近 48%。来源：Mintlify [1]

3.4 16 个 AI 智能体的文档重构实践

最详尽的公开案例研究来自 Eugene Petrenko（2026 年 1 月）：其使用 **16 个 AI 智能体**协同重构了 2,648 行文档 [46]。其中 7 个独立评估智能体通过结构化访谈诊断文档质量，发现最关键的命令被埋在第 91 行，前置内容包含约 90 行引言。重构后文档**缩短 39%**，质量评分**提高 15%**，关键命令的导航速度**提升 5 倍** [46]。其核心启示可概括为：**人类可凭经验消解的歧义，对智能体而言往往直接转化为幻觉与执行风险** [46]。

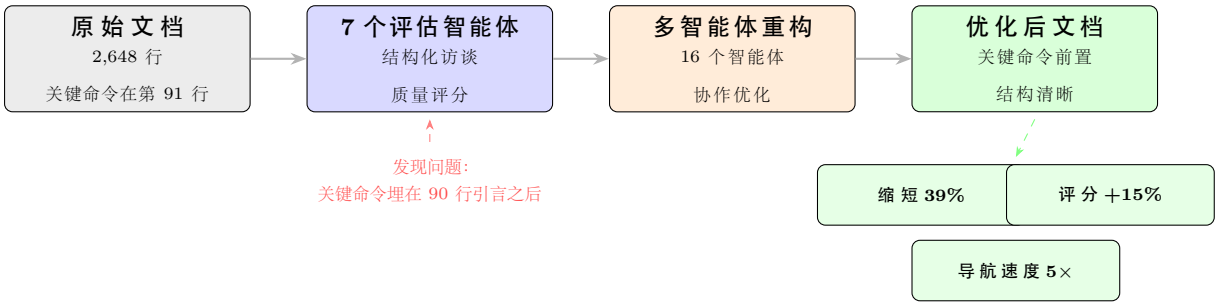

图 7: 16 个 AI 智能体文档重构工作流——从问题发现到量化改进 [46]

4 文档质量与智能体性能的学术研究

日益增多的学术研究一致表明：**文档质量是 AI 智能体性能的关键、且长期被低估的瓶颈。**

4.1 基础研究

Hsieh 等人（2023）的里程碑式研究表明，仅依赖工具文档的零样本提示，在 6 个任务上可达到与少样本提示**持平甚至更优**的性能 [2]。在包含数百个工具的大规模 API 数据集上，文档提供的信号**显著优于**示例（few-shot demonstrations）。但当文档长度超过约 **600 词**后，性能开始下降，提示存在一个近似的最优文档长度区间 [2]。该研究据此确立：文档不仅是"有用的上下文"，更是工具使用能力的**关键决定因素**。

4.2 迭代文档优化

DRAFT 框架（Qu 等，ICLR 2025）提出通过三阶段实现**迭代文档优化**：经验收集、经验学习与文档改写 [56]。其核心结论——"以人类为中心的工具文档与 LLM 的理解方式之间存在系统性鸿沟"——已被多项后续研究独立印证。更重要的是，为某一模型优化后的文档往往表现出**良好的跨模型泛化**能力，即在不同模型上也能稳定带来性能提升 [56]。

4.3 简洁工具指令

EASYTOOL（Yuan 等，NAACL 2025）进一步证明，将风格与粒度不一的工具文档转换为统一、简洁的指令格式，可在**显著降低 token 消耗**的同时提升工具调用性能 [3]。其中一个最具冲击力的结果是：使用 EASYTOOL 指令的 ChatGPT 在成功率上**超越了**使用原始文档的 GPT-4；而在原始文档条件下几乎完全失败的小型模型，在标准化指令下也获得了可观成功率 [3]。这提示：在工具使用场景中，**文档的结构与表达方式可能在一定程度上比模型规模更具决定性**。

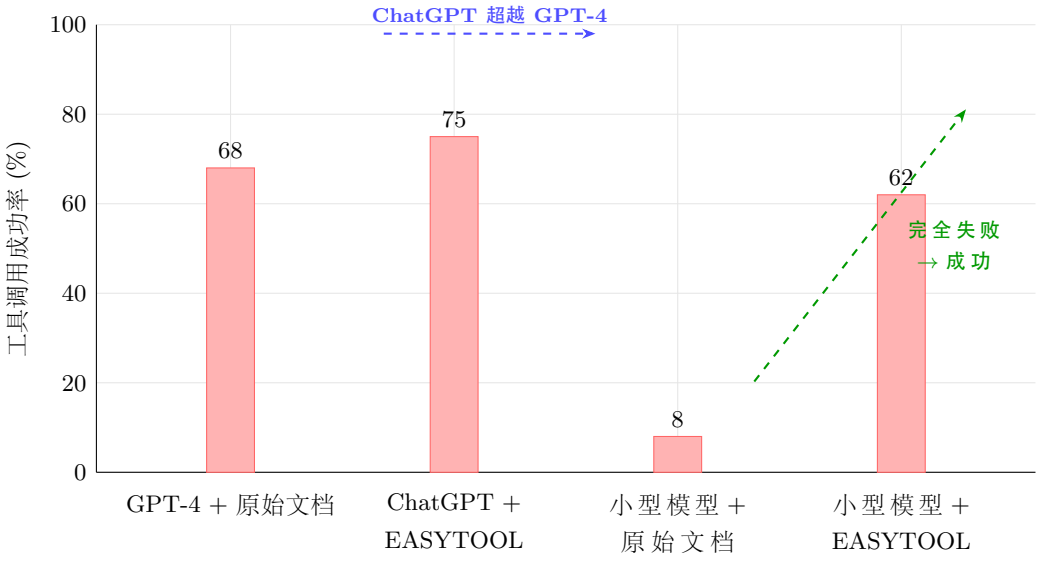

图 8: 文档质量对模型工具调用性能的影响——基于 EASYTOOL [3] 的研究发现。使用标准化指令的 ChatGPT 超越了使用原始文档的 GPT-4；使用原始文档几乎完全失败的小型模型在标准化指令下取得成功。

4.4 API 文档与幻觉

Gorilla 项目（Patil 等，UC Berkeley，NeurIPS 2024）显示，在缺乏适当文档约束时，GPT-4 与 GPT-3.5 在生成 API 调用过程中会**大量产生幻觉**（例如虚构端点/参数）[57]。相较之下，经微调的 LLaMA-7B 在结合文档检索后，于 API 调用准确性上超越 GPT-4；该工作提出了检索感知训练（Retriever Aware Training），并构建包含 11,000+ 指令—API 对的 APIBench 基准 [57]。**ToolLLM**（Qin 等，ICLR 2024 Spotlight）进一步以 ToolBench 将评测扩展至 49 个类别的 16,464 个真实 API [58]。

4.5 文档格式与结构

围绕文档的表述方式与结构组织，相关研究揭示了更细粒度的规律。Dang 等人（2025）发现，自由形式的思维链往往**不充分且有时适得其反**；他们提出受课程设计启发的模板化结构，通过显式推理步骤实现 3–12% 的提升 [59]。He（2024）仅通过提示工程与结构化文档注入，即在工具调用格式遵循上达到 **100%** 的成功率 [60]。Multi-Field Tool Retrieval（MFTR）进一步指出：**将工具文档视作纯文本存在结构性上限**——将名称、描述、参数、返回类型等字段拆解并赋予独立权重，可显著提升检索与选择的准确性 [61]。

4.6 评估指标

上述研究共同指向一组面向“AI 消费”的文档质量评估指标：通过率（给定文档条件下成功完成任务的比例）、幻觉率（如以 AST 子树匹配等方式衡量）、token 效率（性能与 token 消耗的比率）、跨模型泛化能力，以及最优文档长度区间 [2, 3, 57]。RAGAS 框架从相关性、上下文质量与忠实度三个维度评估 RAG 系统 [62]；Berkeley Function Calling Leaderboard（BFCL）则从简单、多重、并行与复合四类场景评估函数调用能力 [63]。

5 DITA 在 AI 时代的新价值

DITA 的核心架构——基于主题（Topic）的写作、信息分类、语义标记与元数据——与 AI 智能体的内容消费方式高度契合；尽管该标准早于当下“智能体时代”约二十年提出 [47, 48]。

5.1 信息分类架构天然匹配智能体检索

DITA 的概念（concept）/任务（task）/故障排除（troubleshooting）/参考（reference）四类主题类型，可较自然地映射智能体在检索与回答时的“意图分类”[47]：**概念**主题提供背景、原理与约束解释，**任务**主题提供可执行的步骤与操作序列，**故障排除**主题面向异常情境，组织症状—原因—处置/绕行方案，**参考**主题则承载数据规格、参数定义与枚举值等确定性信息。

这样的**信息分类**使智能体能够在生成过程中明确当前需要的是“解释”“操作”“诊断”还是“查表”，从而减少将解释性文字误当作执行指令、把排障建议混入标准流程、或在步骤与概念之间来回跳转造成的混淆与误执行风险。与此同时，DITA 的单源发布机制使同一事实与同一规则在不同输出形态中保持一致，降低版本差异与文本矛盾带来的幻觉触发点。如 Bluestream 2026 年分析所述：“更少的冲突来源带来更可靠的 AI 生成答案”[47]。

5.2 最先进的 DITA-to-AI 管道

当前较为先进的 DITA-to-AI 管道由 Sana Remekie（Conscia.AI）在 The Content Wrangler（2025 年 8 月）中记录 [49]。该架构从 Heretto CCMS 摄取 DITA 内容，将其转换为保留主题关系的知识图谱，并在检索阶段采用关键词、向量与图检索相结合的混合 RAG；随后引入业务规则进行智能排序与过滤，最终通过 MCP 服务器对外提供结果——端到端响应时间控制在 **200 毫秒**以内 [49]。Michael Iantosca（Avalara）在 AI The Docs 2025 上展示了互补路线：**DOM Graph RAG 模型**利用 DITA 的层级结构作为本体论封装器自动构建知识图谱，从而实现更接近“确定性的神经符号推理”，而非仅依赖概率性的向量相似度检索 [50]。

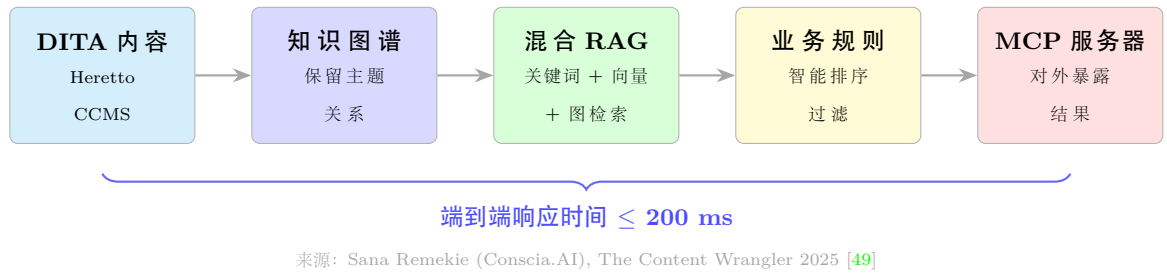

图 9: 最先进的 DITA-to-AI 管道架构——从结构化 DITA 内容到 MCP 服务器的端到端流水线

5.3 DITA-OT 社区的 LLM 优化工作

DITA-OT 仓库（GitHub 讨论 #4476）中的社区讨论正在探索生成“更适合 LLM 摄取”的 Markdown 输出 [51]。其基本思路是将 DITA 语义显式注入 Markdown：例如检测到 `<prereq>` 元素时，在相关段落前补充“以下条件是后续步骤的前提”；或在平台过滤（profiling）后的内容前加注“对于 EKS 平台”。其中一个关键难点是**条件化内容（conditional content）**：若 AI 摄取未过滤的全量 DITA（包含所有产品条件），则可能混淆不同平台/版本的功能边界。社区倾向于在 AI 摄取前使用 DITAVAL 进行预过滤，而非将条件逻辑留给智能体在运行时推断 [51]。

5.4 Oxygen AI Positron：高级 DITA+AI 集成

Oxygen AI Positron 8.1 代表目前较为先进的 DITA+AI 集成方案 [52]。其 DITA Agent 模式支持面向 DITA XML 工程的智能体对话，并提供探索 DITA map、解析引用关系等工具。值得注意的是，Positron 的提示工程反过来使用 DITA XML 进行复用与治理：人物角色与固定措辞可通过 conref 在提示间共享，profiling 属性则可据不同 AI 引擎变体调整提示 [53]，形成"用 DITA 的结构化机制管理 DITA 内容所驱动的 AI 提示"的递归闭环。

5.5 技术写作社区的共识

Write the Docs 社区积极参与这一转型。其 2025 年 11 月的 Newsletter 强调：**编写结构化且简洁的内容，配以清晰的引言段落与信息性标题，可在不额外添加 FAQ 的情况下改善 AI 排名；对人类读者有效的写作方式，对 LLM 同样适用。** [54]。Intercom 的 Beth-Ann Sher 将写作心态概括为："写作时就像在做广播采访——你不希望被断章取义地引用"[54]。AI The Docs 2025 将技术写作者定位为"AI 工作流管理者"与"真相守护者"[50]；LavaCon 2025 则提出**"AI 是新客户"**——AI 成为内容的直接消费者，而人类在更多场景中转为间接消费者 [55]。

6 实践指南、工具与常见反模式

6.1 写作原则

面向智能体可消费（machine-consumable）的文档写作，业界正在形成一组可操作的共识原则：

- **前置关键信息**——OpenAI 的 o3/o4-mini 指南显示，仅将最重要的规则移到工具描述的前部即可**提升约 6% 的准确率** [64]。
- **以规范性写法替代纯描述性写法**：不仅说明工具"做什么"，还要明确"何时用、如何用、在什么约束下用"[41]。
- **有限取值显式结构化**：对有限参数集优先使用 `enum`，避免在正文中以散文方式罗列 [64]。
- **强化结构约束(OpenAI)**:对 OpenAI 模型启用 `strict: true`,并设置 `additionalProperties: false`；必要时将关键字段标记为 `required`，以提高结构遵循与解析稳定性 [65]。
- **控制工具规模与参数复杂度**：将工具数量控制在 100 以内、每个工具的参数控制在 20 以内；超过该阈值后整体性能可能下降 [64]。
- **大目录用"按需发现"替代"全量预加载"**：Anthropic 建议对大型工具目录采用"Tool Search Tool"模式，通过动态发现减少预加载带来的 token 开销 [42]。

6.2 文档平台与工具

多个文档平台已将 AI 优化能力内置为产品功能：

- **Fern** 将机器可读文档作为核心产物：自动生成 token 优化的 llms.txt；检测到 LLM 流量时优先提供 Markdown；并通过内容标签（`<llms-only>` 与 `<llms-ignore>`）控制 AI 与人类读者的可见性 [14]。
- **GitBook** 允许将任意页面以 Markdown 形式获取（在 URL 后添加 `.md`），并为每个发布空间自动暴露 MCP 服务器 [66]。
- **ReadMe** 可从 API 文档生成 MCP 服务器，提供 AI 驱动的 linter（如检测被动语态、缺失示例），并支持"secret context"——用于提升 AI 准确性但不向人类读者展示的隐藏内容 [67]。
- **Mintlify** 自动生成 llms.txt/llms-full.txt/MCP 服务器，提供段落级向量索引，并提供监控代码库的 AI Writing Agent [43, 44]。
- llms.txt 生成器包括 Mintlify 的免费工具 [68]、llms-txt.io [69]、Firecrawl，以及 Parallel.ai 的开源 LLMTEXT 工具包。

6.3 测试面向 AI 智能体的文档

测试仍是当前最薄弱的环节。尽管尚无评估"面向 AI 消费"的文档质量的标准化基准，但可采用以下实用方法：

- **基于 RAG 的测试**：将文档接入 RAG 系统，对代表性问题集进行回归测试；
- **LLM-as-a-Judge 评估**：使用 DeepEval 等框架对回答质量与引用一致性进行评估 [62]；

- **工具轨迹测试**：验证智能体能否按正确顺序调用正确工具，并满足关键约束 [70]。

Agent CI 提供以 TOML 文件存储的评估配置，可用于 CI/CD 管道中的版本化回归测试 [71]；Google 的 Agent Development Kit 则提供 `tool_trajectory_avg_score` 与 `rubric_based_tool_use_quality_v1` 等评估指标 [72]。

6.4 常见反模式

常见反模式可概括为三类 [28, 41, 64]：

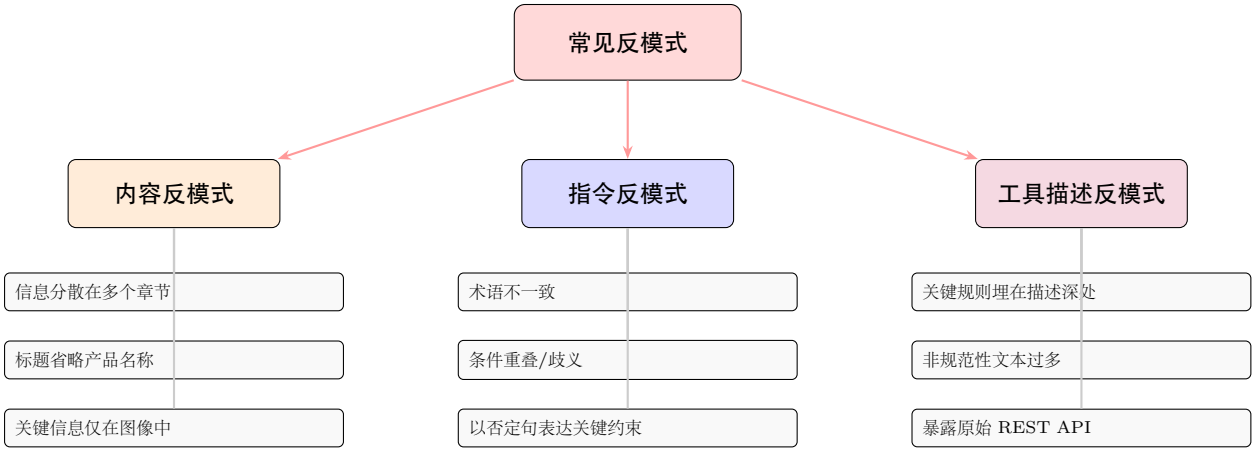

图 10: 面向智能体文档的三类常见反模式分类概览 [28, 41, 64]

6.4.1 内容反模式

- 将相关信息分散在多个章节中（对分块与检索而言是高风险做法）；
- 标题中省略产品名称（削弱语义检索与跨页对齐）；
- 仅将关键信息嵌入图像中（缺少可解析的文本等价物）；
- 假设 AI 系统具备其无法推断的前置知识。

6.4.2 指令反模式

- 术语不一致；
- 条件重叠（如“如果数据缺失”vs “如果数据不完整”）；
- 告诉智能体 HOW 而非 WHAT（引用 API 名称而非描述业务意图）；
- 用否定句表达关键约束（如“如果……则不要发送邮件”）而非肯定句（如“仅在……时发送邮件”）[28]。

6.4.3 工具描述反模式

- 将关键规则埋藏在冗长描述的深处；
- 非规范性文本过多，导致约束不清或优先级不明确；
- 缺少 few-shot 示例；
- 暴露原始 REST API，而非面向业务任务封装的高层工具 [41]。

7 面向未来的三层文档架构

文档生态正在收敛为一套**三层架构**：人类可读层（传统 HTML 与交互能力）、AI 可读层（llms.txt 及全站 Markdown 表示），以及智能体可执行层（MCP 服务器，提供可程序化访问与执行能力）[11, 73]。随着 AI 成为开发者工作流的关键中介，若企业未能落实三层架构，其产品在被检索、被推荐与被调用层面将出现“可见性衰减”。Mintlify 的最新数据（图 6）显示，文档站点**近半数流量已来自 AI 智能体** [1]。这意味着三层架构的每一层都必须将智能体视为**一等公民**：尤其是 AI 可读层与可执行层，需要通过内容协商（content negotiation）为人类与智能体分别提供最合适的内容表示与交互入口 [1]。

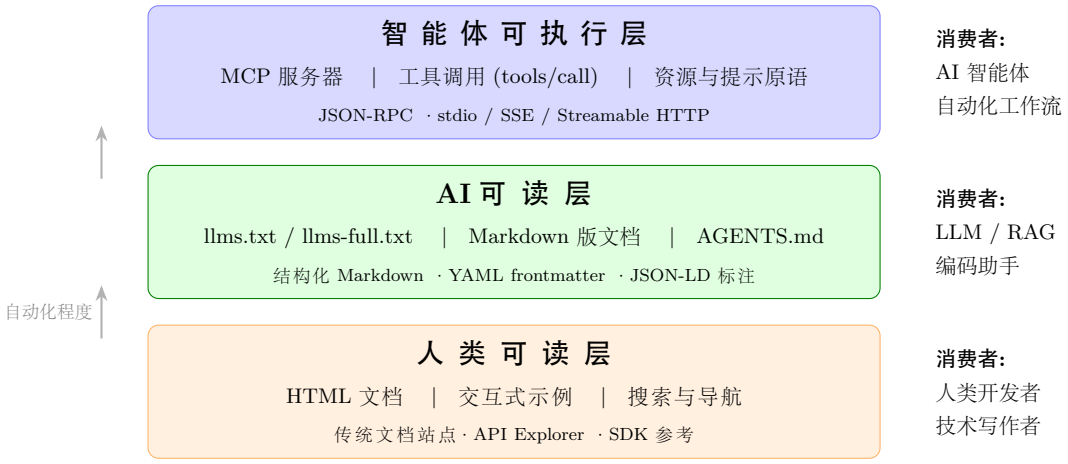

图 11: 面向未来的三层文档架构

展望 2026 年及以后，至少有三股力量将持续塑造该领域：其一，**MCP 的扩散**正在加速——Document360 预测到 2026 年，**75% 的开发者**将采用 MCP [73]。其二，**生成引擎优化（GEO）**作为新兴方法体系出现，目标是面向 AI 驱动搜索（而非传统 SEO）优化内容 [74]。其三，技术写作者的角色正在从“内容生产者”转向“信息架构与治理的协调者”：Cherryleaf（2025）调查显示，55% 的技术传播从业者已定期使用 AI [75]；到 2026 年，AI 更可能默认生成初稿，而人类负责审查、纠错与结构化组织 [75]。

文档质量与 AI 能力之间的关系正在成为核心设计约束。正如对 Anthropic 工具写作指南的分析所概括：**“瓶颈将不再是模型的智能，而是我们为其构建的工具质量”** [76]。学术研究进一步支持这一判断：EASYTOOL 表明，良好文档化的工具可使较小模型超越依赖劣质文档的较大模型 [3]；DRAFT 证明文档改进具备跨模型族系的泛化能力 [56]。文档不再只是软件旁的参考资料，它正在演变为 AI 智能体理解、导航并操作数字世界的关键接口。

8 结论

本研究展示了一个快速演进但主线清晰的新趋势：文档正在从”供人阅读的参考资料”转向”供模型检索与执行的交互接口”。在这一转型中，标准、平台与研究工作彼此牵引，共同推动实践收敛。

有五点结论值得关注：

1. **Markdown 暂时胜出**，并正逐步成为智能体可读文档的通用载体：llms.txt [4]、AGENTS.md [8]、MCP 工具描述 [15] 以及主流文档平台 [43, 66] 都在收敛到干净、结构稳定的 Markdown 表示。

2. **文档质量直接约束智能体能力上限**：学术证据显示，结构良好的文档可在一定程度上弥补模型局限，而低质量文档会使前沿模型也出现系统性失效 [2, 3, 57]。

3. **传统结构化写作（DITA）与 AI 需求高度对齐**：其基于主题的架构、信息分类以及元数据框架，天然映射到 RAG 分块、意图分类与检索过滤 [47, 49]，但仍依赖有效的转换与摄取管道 [51]。

4. **智能体可读文档的评估体系仍然缺位**：测试”面向 AI 智能体的文档有效性”仍以临时方法为主，这是当前实践中最显著的空白之一 [71, 72]。

5. **文档经济学正在逆转**：当近半数流量来自 AI 智能体 [1]，歧义成本会在每次智能体交互中复合累积；且智能体失败往往不产生可观测信号 [1]，使文档质量从”锦上添花”转变为系统可靠性的直接输入 [46, 11]。

参考文献

[1] Langlois, P. (2026). "Almost Half Your Docs Traffic is AI." *Mintlify Blog*. https://www.mintlify.com/blog/ai-traffic

[2] Hsieh, C.-Y., et al. (2023). "Tool Documentation Enables Zero-Shot Tool-Usage with Large Language Models." arXiv:2308.00675. https://arxiv.org/abs/2308.00675

[3] Yuan, Y., et al. (2025). "EASYTOOL: Enhancing LLM-based Agents with Concise Tool Instruction." In *Proceedings of NAACL 2025*. https://aclanthology.org/2025.naacl-long.44/

[4] Howard, J. (2024). "The /llms.txt file." llms-txt. https://llmstxt.org/

[5] Publii. (2025). "The Complete Guide to llms.txt: Should You Care About This AI Standard?" https://getpublii.com/blog/llms-txt-complete-guide.html

[6] Anthropic. (2024). "Introducing the Model Context Protocol." https://www.anthropic.com/news/model-context-protocol

[7] Wikipedia. (2025). "Model Context Protocol." https://en.wikipedia.org/wiki/Model_Context_Protocol

[8] AGENTS.md. (2025). "AGENTS.md: A Standard for AI Agent Instructions." https://agents.md/

[9] AIMultiple. (2025). "Agents.md: A Machine-Readable Alternative to README." https://research.aimultiple.com/agents-md/

[10] Array Analytics. (2025). "From robots.txt to AI Regulation: How Web Standards and Governance Are Evolving for Machine Consumers." https://www.arrayanalytics.io/post/from-robots-txt-to-ai-regulation-how-web-standards-and-governance-are-evolving-for-machine-consumer

[11] Mintlify. (2025). "AI Documentation Trends: What's Changing in 2025." https://www.mintlify.com/blog/ai-documentation-trends-whats-changing-in-2025

[12] Mintlify. (2025). "Simplifying docs for AI with /llms.txt." https://www.mintlify.com/blog/simplifying-docs-with-llms-txt

[13] Mintlify. (2025). "Real llms.txt examples from leading tech companies (and what they got right)." https://www.mintlify.com/blog/real-llms-txt-examples

[14] Fern. (2026). "API Docs for AI Agents: llms.txt Guide Feb 2026." https://buildwithfern.com/post/optimizing-api-docs-ai-agents-llms-txt-guide

[15] Model Context Protocol. (2025). "Tools – Model Context Protocol (MCP)." https://modelcontextprotocol.info/docs/concepts/tools/

[16] OpenAI. (2025). "Model context protocol (MCP) - OpenAI Agents SDK." https://openai.github.io/openai-agents-python/mcp/

[17] Xano. (2026). "OpenAPI Specification Guide (2026): AI Agents, MCP, & API Design." https://www.xano.com/blog/openapi-specification-the-definitive-guide/

[18] Anthropic. (2025). "Using CLAUDE.MD files: Customizing Claude Code for your codebase." https://claude.com/blog/using-claude-md-files

[19] Cursor. (2025). "Cursor – Rules." https://docs.cursor.com/context/rules

[20] DigitalAPI. (2025). "How to make your APIs ready for AI agents?" https://www.digitalapi.ai/blogs/how-to-make-your-apis-ready-for-ai-agents

[21] David John (Chukwuemeka). (2025). "Introducing OpenAPI Slimmer – Slim Down Your API Specs for AI Agents." *Medium*. https://medium.com/@mcsavvy/introducing-openapi-slimmer-slim-down-your-api-specs-for-ai-agents-b0f199ea37f2

[22] Ricardo Garcês. (2025). "Using AI Agents to abstract Complex APIs." *Medium*. https://medium.com/@ricardomsgarces/using-ai-agents-to-abstract-complex-apis-fd50f0b6d7c6

[23] GitHub. (2025). "upstash/context7-legacy: Instant LLM Context for Agents and Developers." https://github.com/upstash/context7-legacy

[24] GitHub. (2025). "therealtimex/context7-llmdocs: Context7 MCP Server." https://github.com/therealtimex/context7-llmdocs

[25] EDRLab. (2025). "TDM Reservation Protocol." https://www.edrlab.org/open-standards/tdmrep/

[26] Common Crawl. (2025). "A Further Look Into the Prevalence of Various ML Opt-Out Protocols." https://commoncrawl.org/blog/a-further-look-into-the-prevalence-of-various-ml-opt-out-protocols

[27] Biel.ai. (2025). "Practical tips to optimize technical documentation for LLMs, AI agents, and chatbots." https://biel.ai/blog/optimizing-docs-for-ai-agents-complete-guide

[28] Elements.cloud. (2025). "Agent Instruction Patterns and Antipatterns: How to Build Smarter Agents." https://elements.cloud/blog/agent-instruction-patterns-and-antipatterns-how-to-build-smarter-agents/

[29] ARON HACK. (2025). "LLM-Friendly Documentation: Creating Content That AI Can Understand and Process Effectively." https://aronhack.com/llm-friendly-documentation-creating-content-that-ai-can-understand-and-process-effectively/

[30] kapa.ai. (2025). "Writing documentation for AI: best practices." https://docs.kapa.ai/improving/writing-best-practices

[31] Google. (2025). "Intro to How Structured Data Markup Works." *Google Search Central*. https://developers.google.com/search/docs/appearance/structured-data/intro-structured-data

[32] Ragwalla. (2025). "Cohere Embed v4.0: 128K Context Windows Transform Agentic RAG at Scale." https://ragwalla.com/blog/cohere-embed-v4-0-128k-context-windows-transform-rag-at-scale

[33] SitePoint. (2025). "Long Context vs RAG: When 1M Token Windows Replace RAG." https://www.sitepoint.com/long-context-vs-rag-1m-token-windows/

[34] Weaviate. (2025). "Chunking Strategies to Improve LLM RAG Pipeline Performance." https://weaviate.io/blog/chunking-strategies-for-rag

[35] Adnan Masood, PhD. (2025). "Chunking Strategies for Retrieval-Augmented Generation (RAG): A Comprehensive Guide." *Medium*. https://medium.com/@adnanmasood/chunking-strategies-for-retrieval-augmented-generation-rag-a-comprehensive-guide-5522c4ea2a90

[36] Databricks. (2025). "Mastering Chunking Strategies for RAG: Best Practices & Code Examples." https://community.databricks.com/t5/technical-blog/the-ultimate-guide-to-chunking-strategies-for-rag-applications/ba-p/113089

[37] Elastic. (2025). "Advanced RAG techniques: Data processing & ingestion." *Elasticsearch Labs*. https://www.elastic.co/search-labs/blog/advanced-rag-techniques-part-1

[38] GitHub. (2025). "stripe/ai: One-stop shop for building AI-powered products and businesses with Stripe." https://github.com/stripe/ai

[39] Stripe. (2025). "Build agentic AI SaaS Billing workflows." *Stripe Documentation*. https://docs.stripe.com/agents-billing-workflows

[40] Stripe. (2024). "A blueprint for AI acceleration." *Stripe Sessions*. https://stripe.com/sessions/2024/a-blueprint-for-ai-acceleration

[41] Anthropic. (2025). "Writing effective tools for AI agents—using the Model Context Protocol." https://www.anthropic.com/engineering/writing-tools-for-agents

[42] Anthropic. (2025). "Introducing advanced tool use on the Claude Developer Platform." https://www.anthropic.com/engineering/advanced-tool-use

[43] Mintlify. (2025). "AI-native documentation." https://www.mintlify.com/docs/ai-native

[44] Fern. (2026). "Best llms.txt Platforms January 2026." https://buildwithfern.com/post/best-llms-txt-implementation-platforms-ai-discoverable-apis

[45] Mintlify. (2025). "Introducing AI Assistant: Turning docs into your product expert." https://www.mintlify.com/blog/introducing-ai-assistant-2025

[46] Petrenko, E. (2026). "How 16 AI Agents Fixed Our Documentation Problem." https://jonnyzzz.com/blog/2026/01/24/16-ai-agents-documentation-refactor/

[47] Bluestream. (2026). "Top 10 Ways Structured Content Prepares Documentation for AI." https://bluestream.com/blog/top-10-ways-structured-content-prepares-documentation-for-ai/

[48] Alation. (2025). "How to Write AI-Ready Documentation: Structure, Metadata & Best Practices." https://www.alation.com/blog/how-to-write-ai-ready-documentation/

[49] Remekie, S. (2025). "Making AI Agents Smarter: How to Turn Your Technical Documentation into an AI-Powered Knowledge Assistant." *The Content Wrangler*. https://www.thecontentwrangler.com/p/making-ai-agents-smarter-how-to-turn

[50] API the Docs. (2025). "AI The Docs Online 2025." https://apithedocs.org/ai-docs-online-2025

[51] DITA-OT Community. (2025). "Generating 'smarter' Markdown for LLM ingestion." *GitHub Discussion #4476*. https://github.com/orgs/dita-ot/discussions/4476

[52] Oxygen XML. (2025). "Oxygen AI Positron What's New." https://www.oxygenxml.com/ai_positron_web_author/whats_new.html

[53] CIDM. (2025). "AI Prompt Development with Reuse and Profiling." https://infomanagementcenter.com/ai-prompt-development-with-reuse-and-profiling/

[54] Write the Docs. (2025). "Write the Docs Newsletter – November 2025." https://www.writethedocs.org/blog/newsletter-november-2025/

[55] Content Strategy Knowledge Base. (2025). "LavaCon Conference: What's on the Content Strategist's Agenda in 2025–2026." https://www.contentstrategy.at/event-reports/lavacon-conference-whats-on-the-content-strategists-agenda-in-2025-2026

[56] Qu, C., et al. (2025). "From Exploration to Mastery: Enabling LLMs to Master Tools via Self-Driven Interactions." In *Proceedings of ICLR 2025*. arXiv:2410.08197. https://arxiv.org/abs/2410.08197

[57] Patil, S.G., et al. (2024). "Gorilla: Large Language Model Connected with Massive APIs." In *Proceedings of NeurIPS 2024*. https://openreview.net/forum?id=tBRNC6YemY

[58] Qin, Y., et al. (2024). "ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs." In *Proceedings of ICLR 2024 (Spotlight)*. https://openreview.net/forum?id=dHng2O0Jjr

[59] Dang, H., et al. (2025). "Improving Large Language Models Function Calling and Interpretability via Guided-Structured Templates." arXiv:2509.18076. https://arxiv.org/abs/2509.18076

[60] He, S. (2024). "Achieving Tool Calling Functionality in LLMs Using Only Prompt Engineering Without Fine-Tuning." arXiv:2407.04997. https://arxiv.org/abs/2407.04997

[61] Multi-Field Tool Retrieval. (2025). arXiv:2602.05366. https://arxiv.org/abs/2602.05366

[62] Clarivate. (2025). "How to Evaluate Generative AI Output Effectively." https://clarivate.com/academia-government/blog/evaluating-the-quality-of-generative-ai-output-methods-metrics-and-best-practices/

[63] Evidently AI. (2025). "30 LLM evaluation benchmarks and how they work." https://www.evidentlyai.com/llm-guide/llm-benchmarks

[64] OpenAI. (2025). “o3/o4-mini Function Calling Guide.” https://developers.openai.com/cookbook/examples/o-series/o3o4-mini_prompting_guide/

[65] OpenAI. (2025). “Function calling.” *OpenAI API Documentation*. https://developers.openai.com/api/docs/guides/function-calling/

[66] GitBook. (2025). “LLM-ready docs.” *GitBook Documentation*. https://gitbook.com/docs/publishing-documentation/llm-ready-docs

[67] ReadMe. (2025). “ReadMe AI.” https://readme.com/ai

[68] Mintlify. (2025). “Free llms.txt generator: create AI-optimized documentation files.” https://www.mintlify.com/blog/free-llms-txt

[69] LLMs.txt Generator. (2025). “Free llms.txt Generator.” https://llms-txt.io/

[70] Stainless. (2025). “MCP API Documentation: The Complete Guide.” https://www.stainless.com/mcp/mcp-api-documentation-the-complete-guide

[71] Agent CI. (2025). “Evaluations (Evals) - Agent CI Documentation.” https://agent-ci.com/docs/core-concepts/evaluations/

[72] Google. (2025). “Why Evaluate Agents.” *Agent Development Kit (ADK)*. https://google.github.io/adk-docs/evaluate/

[73] Document360. (2026). “Major AI Documentation Trends for 2026.” https://document360.com/blog/ai-documentation-trends/

[74] LeadGen Economy. (2025). “llms.txt, robots.txt for AI, and the Technical Foundation of AI Discoverability.” https://www.leadgen-economy.com/blog/llms-txt-ai-crawler-optimization-guide/

[75] Fluid Topics. (2026). “6 Technical Documentation Trends to Watch in 2026.” https://www.fluidtopics.com/blog/industry-insights/technical-documentation-trends-2026/

[76] LaxmiKumar Reddy Sammeta. (2025). “Writing Effective Tools for AI Agents: Lessons from Anthropic.” *Medium*. https://laxmikumars.medium.com/writing-effective-tools-for-ai-agents-lessons-from-anthropic-25b85bf74f5d