AI Agent 开发工程师学习路线图（工程落地版）

> **目标岗位**：AI Agent 开发工程师（应用型、工程型）；
>
> **学习时长**：8 周（全职投入）；
>
> **最终产出**：2-3 个生产级、可部署的 Agent 系统 + 完整的全栈技术能力。

一、你能获得什么

> **用8周，打造从原型到生产的完整工程能力。**
>
> ✅ **8周系统课程**：从 LangChain 基础到生产级 Agent 系统架构；
>
> ✅ **每周代码实战**：手撕 RAG、Agent、多智能体，将想法变为高可用服务；
>
> ✅ **2个工业级项目**：完成从需求分析、技术选型、开发部署到监控优化的全流程；
>
> ✅ **生产级技术栈**：掌握 FastAPI, Docker, Redis, Prometheus 等后端必备技能；
>
> ✅ **顶级面试能力**：搞定系统设计、性能优化、故障排查等高频面试题。

二、开发岗核心要求

2.1、你需要具备的能力

1. 系统设计
    - 高可用、高并发架构；
    - 性能与成本优化；
    - 监控、告警与日志；
    - 故障排查与容错。
2. 工程实现
    - 熟练使用 Agent 框架；
    - 高质量、可维护的代码；
    - 快速开发与迭代能力；
    - 强大的调试与问题定位。
3. 业务理解
    - 用户需求转化为技术方案；
    - 场景适配与方案选型；
    - 数据驱动的系统优化；
    - 评估技术方案的 ROI。

2.2、开发岗简历必备

✅ **至少2个完整系统项目**：端到端可运行，有线上部署经验；

✅ **量化的业务指标提升**：如 QPS+100%、P99延迟-80%、成本-50% 等数据；

✅ **丰富的生产级技术栈**：LangChain + FastAPI + Milvus + Redis + Docker + Prometheus；

✅ **生产化经验**：有部署、监控、性能优化、异常处理的实战经历。

三、推荐学习资源与工具

1. 📚 **核心课程与书籍**
    1. **课程**: 吴恩达: Generative AI for Everyone；
    2. **课程**: 微软: Generative AI for Beginners；
    3. **课程**: HuggingFace NLP Course；
    4. **教程**: 《动手学大模型应用开发》 - Datawhale开源教程；
    5. **教程**: 《面向开发者的 LLM 入门教程》 - 吴恩达课程中文版；
    6. **教程**: 《开源大模型食用指南》 - 快速微调与部署教程；
    7. **教程**: 《AI-Guide-and-Demos》 - API到本地部署微调指南；
    8. **书籍**: 《Build a Large Language Model (From Scratch)》。
2. 🛠️ **开发框架与工具**
    1. **LLM框架**: LangChain, LlamaIndex, Dify；
    2. **Agent框架**: AutoGen, CrewAI, AgentScope；
    3. **向量数据库**: Milvus, Qdrant, Chroma；
    4. **推理引擎**: vLLM, SGLang, Ollama；
    5. **评估工具**: RAGAs, DeepEval, LangSmith。
3. 🌐 **学习社区与资源**
    1. **社区**: HuggingFace, ModelScope, 魔乐社区；
    2. **博客**: Lil'Log (OpenAI), 科学空间（苏剑林）, Chip Huyen；
    3. **资源库**: Awesome LLM Resources；
    4. **可视化**: 100+ LLM/RL 算法原理图 - 通过图解理解算法原理；
    5. **可视化**: Interactive Transformer Explainer - 交互式理解Transformer。

四、8周详细学习计划

第 1 周：大模型应用开发基础 + 手撕 Naive RAG

> **学习内容:**
>
> - **后端基础**: FastAPI 路由、异步 I/O、Pydantic 数据校验；
> - **LangChain 核心**: LLM, Prompt Templates, Output Parsers, LCEL；

- **Naive RAG 流程**: Document Loaders, Text Splitters, Embeddings, Vector Stores；
- **向量数据库**: FAISS/ChromaDB 本地化使用。

手撕系列:

- FastAPI 搭建 "Hello, World" API 服务；
- LangChain LCEL 编写第一个 LLM Chain；
- 30分钟手撕一个完整的 Naive RAG 应用。

解锁技能:

- 熟练使用 FastAPI 搭建 API；
- 掌握 LangChain 核心组件与 LCEL 表达式语言；
- 能够从零开始，快速构建一个基于文档问答的 RAG Demo。

每日学习计划

天数	学习计划	资源链接	目标
1	FastAPI 快速入门	教程: 学习 - FastAPI	掌握 FastAPI 基础，能够创建路由、处理请求
2	LangChain 核心概念	文档: LangChain Quickstart 课程: 吴恩达: LangChain for LLM Application Development 课程: Building Systems with the ChatGPT API	理解 LangChain 六大核心模块，熟练使用 LCEL
3	RAG Part 1: 加载与分割	文档: RAG 学习指南 工具: Unstructured.io, MinerU, Docling	掌握不同格式文档 (PDF, MD) 的加载和文本分块策略
4	RAG Part 2: 向量化与存储	教程: FAISS Intro 理解教程中 Basics 的概念即可 教程: Sentence Transformers	理解嵌入(Embedding) 原理，使用 FAISS/Chroma 构建本地向量索引
5-6	手撕 Naive RAG 系统	教程: RAG from Scratch 概念: LLM Powered Autonomous Agents 教程: 动手学大模型应用开发 参考: 面向开发者的LLM入门教程	整合 FastAPI + LangChain，完成一个端到端的文档问答 API
7	周度总结与项目部署		将本周的 RAG 项目用 Docker 打包，并成功运行

技术栈作用和重点分析：

1. FastAPI：是一个用于构建 API 的现代、快速（高性能）的 web 框架 —— 总结：Python后端框架，**必学**。
2. LangChian：使用 LLM 构建代理和应用程序的框架 —— 总结：调用 大模型 的框架，**必学**。

3. LlamaIndex：是一个 RAG 框架，通过构建上下文来增强 大模型 的能力；在 LangChain 中通常作为一个工具整合到工作流中使用，**必学**：
    1. 提供了上百种开箱即用数据连接器(数据库、S3/云存储、Confluence、Notion、Salesforce、API、PDF/PPT等)；
    2. 将异构数据统一转换为 `Document` 对象，存储到介质中；
    3. 智能索引：能自动构建从摘要到细节的层级索引，无缝结合向量检索（语义）和关键词检索（精确匹配）；
    4. 提供多种模式（如“累积”、“提炼”、“不合成”），灵活控制 LLM 如何基于检索到的上下文生成答案；
    5. 让 LLM 不仅能回答基于文档的问题，还能对数据进行计算、分析和推理。
4. FAISS Intro：专注于向量搜索的底层库，**了解原理**。
5. Embedding 原理：是一种将高维、离散或非结构化数据(如文本、图像、类别标签)映射到低维连续向量空间的技术，**了解原理**：
    1. 简单来说就是将数据根据语义转换成向量的技术。
6. Sentence Transformers：将文本、图片、音频等文件，转换为计算机能理解的、包含语义的"向量"，**了解原理**。
7. Chroma：一个完整的向量数据库，不仅能搜索向量，还存储向量、关联的元数据、原始文档，提供数据管理、持久化、多租户等数据库功能，**项目使用**。
8. Naive RAG：RAG四种设计模式中最简单的一种，**必学**：
    1. RAG 由**检索（Retrieval）** 与**生成（Generation）** 两大模块构成；
    2. 检索模块：从预设知识库中精准定位与用户问题相关的信息片段（如文档、段落、句子）；
    3. 生成模块：基于检索到的信息，结合大语言模型生成符合上下文、逻辑连贯的答案。

第 2 周：Advanced RAG 与生产级向量数据库

> **学习内容:**
>
> - **Advanced RAG 技术**: Query Transformation, Re-ranking, Hybrid Search；
> - **RAG 评估**: 使用 RAGAs, TruLens 进行自动化评估；
> - **生产级向量数据库**: Milvus/Zilliz Cloud 部署与使用；
> - **数据处理**: Unstructured.io 解析复杂文档。
>
> **手撕系列:**
>
> - 实现 BM25 + 向量的混合检索；
> - 引入 Cohere Rerank 模型提升检索精度；
> - 使用 RAGAs 评估 RAG 系统的 Faithfulness 和 Answer Relevancy；
> - Docker 部署 Milvus 并进行增删改查操作。
>
> **解锁技能:**
>
> - 掌握 10+ 种 RAG 优化策略；
> - 能够建立 RAG 系统的自动化评估流水线；
> - 熟练使用生产级的分布式向量数据库 Milvus；

- 具备处理复杂、非结构化文档的能力。

🌟 每日学习计划

天数	学习主题	资源链接	目标
8	Query Transformation	教程: LlamaIndex Query Transforms	实现 HyDE, Multi-Query 等查询改写策略
9	混合检索与重排 (Rerank)	教程: LlamaIndex Reranking 论文: Modular RAG	实现 BM25 + Embedding 混合检索，并集成 Reranker
10-11	RAG 评估体系	文档: RAGAs 评估框架 工具: FlashRAG, DeepEval, Lighteval	学习 RAG 核心评估指标，并用 RAGAs 评估优化前后的系统性能
12	生产级向量数据库 (Milvus)	文档: Milvus Quick Start 替代: Infinity, Qdrant	使用 Docker 部署 Milvus，并掌握其 Python SDK
13	高级数据处理	文档: Unstructured.io 工具: MinerU, PDF-Extract-Kit, Docling, GOT-OCR2.0	使用 Unstructured/MinerU 解析包含表格、图片的复杂 PDF
14	周度总结与系统升级		将第一周的 RAG 系统升级，集成混合检索、Reranker 和 Milvus

技术栈作用和重点分析：

1. Query Transformation：RAG 的四大主流模式中的Advanced RAG，**必学**。
2. Rerank：RAG中的关键技术，对初步检索结果进行再次排序的关键步骤，**必学**。
3. Milvus：开源的向量数据库，**必学**。
4. Unstructured.io：文档解析工具，可以从PDF和Word文档等原始源文档中提取干净的文本，**必学**。

第 3 周：Agent 开发与 Tool Calling

学习内容:

- **Agent 核心**: ReAct 框架, Planning, Tool Use, Memory；
- **Tool Calling**: OpenAI Function Calling, Tool Schema 定义；
- **工具开发**: 如何将 API, 数据库查询等封装为 Agent 可用的工具；
- **错误处理**: 工具调用失败的重试、降级策略。

手撕系列:

- 实现 3个 自定义工具 (天气查询, SQL数据库查询, API调用)；
- 基于 LangChain 构建一个可以链式调用工具的 Agent；
- 使用 OpenAI Function Calling 实现结构化数据提取。

解锁技能:

- 深刻理解 Agent 的"思考-行动"工作流；
- 能够开发、测试、维护自定义工具集；
- 掌握 Function Calling 的原理与应用；
- 具备构建能处理真实世界任务的 Agent 的能力。

🌟 每日学习计划

天数	学习主题	资源链接	目标
15	Agent 核心概念	博客: LLM Powered Autonomous Agents 文档: LangChain Agents 论文: ReAct	理解 ReAct 框架，并运行一个 LangChain 官方的 Agent 示例
16	自定义工具开发	教程: LangChain Custom Tools 参考: MCP协议, MCP教程	编写一个查询天气的自定义工具，并集成到 Agent 中
17	SQL & 数据库工具	教程: LangChain SQL Agent	构建一个能根据自然语言查询数据库的 SQL Agent
18	Function Calling 实战	文档: OpenAI Function Calling 指南: GPT Best Practices	使用 OpenAI API 实现一个能根据用户问题调用函数的 Agent
19	Agent Memory	文档: LangChain Memory 工具: Mem0, MemoryScope	为 Agent 添加对话历史记忆 (ConversationBufferMemory)
20	Agent 错误处理	教程: Error Handling in Agents	为工具调用添加重试机制 (`tenacity` 库) 和降级策略
21	周度总结与项目构建		构建一个集成 RAG 和 Web 搜索工具的 "研究助手" Agent

技术栈作用和重点分析：

1. ReAct 框架：让 AI 从“单线程”工作进化到像人类一样“思考–行动–再调整”，**必学**。
2. 自定义工具开发：MCP的应用，**必学**。
3. SQL & 数据库工具：构建一个能根据自然语言查询数据库的 SQL Agent，**必学**。
4. Agent Memory：代理记忆，**必学**。
5. Agent 错误处理，**必学**。

第 4 周：系统性能优化

学习内容:

- **缓存策略**: Redis 缓存 LLM 响应和 Embedding 结果；
- **异步处理**: `asyncio`, `aiohttp` 实现高并发；
- **批处理优化**: Embedding 和 LLM 调用的批处理；
- **推理加速**: vLLM, TensorRT-LLM 部署与使用。

手撕系列:

- 为 RAG 系统引入 Redis 缓存，对比优化前后性能；
- 将 FastAPI 的同步接口改造为异步接口；
- 部署 vLLM 并通过 API 进行推理。

解锁技能:

- 掌握 LLM 应用的核心性能优化手段；
- 能够将系统的 QPS 提升 10 倍以上；
- 熟练使用 Redis 进行缓存设计；
- 具备部署和使用高性能推理引擎的能力。

✳ 每日学习计划

天数	学习主题	资源链接	目标
22	性能瓶颈分析	工具: py-spy, Scalene	学习使用 `cProfile`, `py-spy` 等工具分析现有 Agent 系统的性能瓶颈
23	缓存优化 (Redis)	教程: FastAPI with Redis 工具: LiteLLM Caching	为 Agent 系统添加 Redis 缓存，缓存 LLM 响应
24-25	异步处理 (Async)	教程: FastAPI Async 示例: LangChain Async	将系统中 I/O 密集型操作 (如 API 调用) 改造为异步
26	批处理优化 (Batching)	教程: Batch Processing	实现 Embedding 和 Reranker 的批处理，提升吞吐量
27	高性能推理 (vLLM)	文档: vLLM Quickstart 替代: SGLang, TensorRT-LLM, LMDeploy 概览: Awesome Inference	使用 vLLM 部署一个开源模型 (如 Llama 3)，并测试其吞吐量
28	周度总结与性能压测		使用 `locust` 或 `jmeter` 对优化前后的系统进行压测，并记录 QPS, P99 等指标

技术栈作用和重点分析：

1. py-spy：在不修改代码或重启程序的情况下，实时分析 Python 程序的性能的工具，**会使用。**
2. Scalene：是一个高性能的Python分析器，专为发现CPU和内存使用效率问题而设计，**了解，py-spy与Scalene会一个即可。**
3. Redis：NoSQL数据库，主要用于缓存，分担关系型数据库的压力，**必学，越深入越好。**
4. Batching：批处理，**必学。**
5. vLLM：开源的大语言模型高速推理框架，旨在极大地提升实时场景下的语言模型服务的吞吐与内存使用效率，**必学。**
6. locust：容易上手的分布式用户负载测试工具，**会使用。**
7. jmeter：JMeter是一个纯Java编写的开源软件，主要用于进行性能测试和功能测试，**了解，locust与jmeter会一个即可。**

8. QPS：一秒钟内接收多少请求，**一个衡量服务器性能的指标**。
9. P99：100个请求中，按响应时间从小到大排序，第99的请求的响应时间，**一个衡量服务器响应耗时的指标**：
    1. P1就是响应时间最小的请求，P10就是排名第十的请求，P100就是响应时间最长的请求；
    2. P50: 即中位数值。100个请求按照响应时间从小到大排列，位置为50的值，即为P50值。

第 5 周：监控、可观测性与部署

> **学习内容:**
>
> - **Agent 链路追踪**: LangSmith, OpenTelemetry；
> - **指标监控**: Prometheus 监控业务和系统指标；
> - **可视化**: Grafana 创建监控大盘；
> - **日志系统**: ELK Stack (Elasticsearch, Logstash, Kibana)；
> - **容器化部署**: Docker, Docker Compose。
>
> **手撕系列:**
>
> - 为 Agent 应用集成 LangSmith，追踪每一步的调用和延迟；
> - 使用 Prometheus 暴露自定义指标 (如 Token 消耗, 缓存命中率)；
> - 使用 Docker Compose 将 FastAPI + Milvus + Redis 整套系统一键部署。
>
> **解锁技能:**
>
> - 具备构建完整 LLM 应用可观测性体系的能力；
> - 能够快速定位和诊断线上问题；
> - 掌握基于 Docker 的容器化部署和编排；
> - 拥有完整的 DevOps for LLM Apps 经验。

每日学习计划

天数	学习主题	资源链接	目标
29	链路追踪 (LangSmith)	文档: LangSmith 替代: OpenTelemetry, LangFuse	将 LangSmith 集成到现有 Agent 应用中，分析调用链路
30	指标监控 (Prometheus)	教程: Prometheus Python Client 集成: FastAPI Instrumentator	暴露 API 的 QPS, 延迟, 错误率等核心指标
31	可视化 (Grafana)	教程: Grafana Dashboard	安装 Grafana，并创建一个简单的监控大盘来展示 Prometheus 指标
32	容器化 (Docker)	教程: Docker for FastAPI 最佳实践: Docker Best Practices	为 FastAPI 应用编写 Dockerfile 并成功构建镜像

天数	学习主题	资源链接	目标
33	服务编排 (Docker Compose)	教程: Docker Compose 示例: Full Stack FastAPI	编写 `docker-compose.yml` 文件，一键启动整个应用栈
34	日志系统	教程: Python Logging 工具: Loguru, structlog	配置应用将日志输出为 JSON 格式，为接入 ELK 做准备
35	周度总结与生产环境模拟		模拟一次线上故障，并使用本周学习的工具链进行问题定位

技术栈作用和重点分析：

1. LangSmith：是 LangChain 生态系统中的工具，用于调试、测试、评估和监控基于大语言模型（LLM）的应用程序，**会使用**。
2. Prometheus：是一个开源的监控和告警工具，专为动态云环境设计，广泛用于监控微服务和 Kubernetes 集群，**会使用**。
3. Grafana：是一款开源的数据可视化工具，主要用于大规模指标数据的可视化展现，**会使用**。
4. Docker：隔离应用程序的运行时环境，容器之间可以共享同一个操作系统，**会使用，懂流程**。
5. Docker Compose：Compose 是用于定义和运行多容器 Docker 应用程序的工具，**会使用**。
6. Loguru、structlog：Python中的日志库，**会使用**。

第 6 周：Multi-Agent 系统开发

> **学习内容:**
>
> - **Multi-Agent 框架**: AutoGen vs. CrewAI;
> - **Agent 角色定义**: 如何设计具有不同职责和能力的 Agent;
> - **通信机制与工作流**: GroupChat, Sequential/Hierarchical flow;
> - **状态管理**: 如何在多个 Agent 之间共享和传递状态。
>
> **手撕系列:**
>
> - 使用 AutoGen 构建一个“研究员-程序员-测试员”协作的软件开发团队;
> - 使用 CrewAI 构建一个“旅行规划师-本地向导-预订专员”的旅行 Agent。
>
> **解锁技能:**
>
> - 掌握至少两种主流的 Multi-Agent 开发框架;
> - 能够根据复杂业务需求，设计和实现多智能体协作系统;
> - 理解不同协作模式 (如层级式 vs. 对话式) 的优缺点。

🌟 **每日学习计划**

天数	学习主题	资源链接	目标
36-37	AutoGen 核心概念	文档: AutoGen Tutorial 论文: AutoGen Framework	学习 `ConversableAgent`, `GroupChat` 等核心概念，并运行官方示例
38	AutoGen 实战	示例: AutoGen Examples	实现一个"研究员-程序员-测试员"的 Multi-Agent 系统
39-40	CrewAI 核心概念	文档: CrewAI Docs 教程: CrewAI Quickstart	学习 Agent, Task, Crew, Process 的概念，并运行官方示例
41	CrewAI 实战	示例: CrewAI Examples	实现一个"旅行规划师-本地向导-预订专员"的 Multi-Agent 系统
42	框架对比与总结	更多框架: agentUniverse, AgentScope, Qwen-Agent, Lagent, PraisonAI 概览: Awesome Agents	对比 AutoGen 和 CrewAI 的设计哲学、优缺点和适用场景

技术栈作用和重点分析：

1. AutoGen 框架：致力于简化多智能体系统的开发过程，使开发者能够轻松构建出智能体之间能够相互协作、交流并共同解决问题的应用程序，**（了解，会使用）**。
2. CrewAI：协调多个AI智能体的协作来完成复杂任务，**（了解，会使用）**：
    1. 这个框架模拟了现实世界中的工作团队，让不同角色的智能体能够自主地相互委派任务和交流，从而实现比单一语言模型更强大的性能表现。

第 7-8 周：工业级项目实战与面试准备

> **核心目标**：完成 1-2 个可写进简历的完整系统，并准备面试。

项目1：企业级智能客服 RAG 系统

> **业务场景**: 为某电商公司构建智能客服系统，自动回答 80% 的重复性用户问题 (订单状态、物流、退款等)。
>
> **技术要求**:
>
> - **数据源**: 对接 FAQ 文档、商品信息数据库 (PostgreSQL)。
> - **核心**: 实现一个混合检索 RAG，优先从数据库精确查询，无法命中再从文档模糊检索。
> - **性能**: 系统 QPS > 200, P99 延迟 < 500ms。
> - **监控**: 完整的 LangSmith + Prometheus + Grafana 监控体系。
> - **部署**: 使用 Docker Compose 部署。
>
> **简历亮点**: 高并发、低延迟、生产级监控、节省XX人力成本。

项目2：Agent 驱动的自动化投研系统

> **业务场景**: 为投资分析师构建自动化报告生成 Agent，输入公司名，自动完成信息搜集、分析和报告撰写。
>
> **技术要求**:
>
> - **Multi-Agent**: 使用 CrewAI 构建，包含 `信息搜集Agent` (调用搜索引擎、API)、`财报分析Agent` (解析PDF财报、计算关键指标)、`报告撰写Agent`。
> - **工具集**: 集成 Google Search, SEC API, 文件读写等至少 5 个工具。
> - **稳定性**: 强大的异常处理和重试机制，任务成功率 > 95%。
> - **工作流**: 设计一个顺序工作流，并记录每一步的中间产出。
>
> **简历亮点**: Multi-Agent 协作、复杂工作流自动化、为分析师提升XX%工作效率。

学习计划 (2周)

天数	学习主题	目标	
43-47	项目一：智能客服 RAG	完成需求分析、架构设计、核心功能开发	
48-51	项目一：优化与部署	完成性能优化、监控集成和 Docker 部署，撰写项目文档	
52-56	项目二：自动化投研 Agent	完成需求分析、Agent 设计、工具开发和工作流实现	
57-58	简历撰写与项目总结	指南: Tech Resume Guide 参考: AI面试指南	按照开发岗模板，将两个项目经历量化地写入简历
59-60	系统设计与面试 Mock	资源: OpenAI Cookbook, GPT Best Practices 题库: LLM系统设计面试题 课程: LLM Evaluation: A Complete Course	准备高频系统设计题，并进行 1v1 模拟面试

核心学习资源推荐

> **精选业界最优质的学习资源，助你快速提升工程能力**

智能体开发

Hello-Agents - Datawhale

- **推荐指数**: ★★★★★
- **内容**: Agent 开发完整教程，从基础到进阶
- **适合**: 快速上手 Agent 开发，掌握框架使用
- **亮点**: 中文友好、实战导向、案例丰富

📊 RAG 系统搭建

All-in-RAG - Datawhale

- ⭐ **推荐指数**: ★★★★★
- 📖 **内容**: RAG 系统完整实现，涵盖文档解析、检索、生成
- 🎯 **适合**: 构建企业级 RAG 系统、性能优化
- 💡 **亮点**: 完整代码、最佳实践、生产级方案

🔧 模型微调（可选）

Unsloth - 高效微调框架

- ⭐ **推荐指数**: ★★★★☆
- 📖 **内容**: 快速微调工具，降低资源消耗
- 🎯 **适合**: 需要快速微调、资源有限的场景
- 💡 **亮点**: 速度快、易上手、成本低

LLaMA-Factory - 一站式微调平台

- ⭐ **推荐指数**: ★★★★★
- 📖 **内容**: Web UI 微调平台，支持 SFT、LoRA、DPO
- 🎯 **适合**: Function Call 微调、模型定制化
- 💡 **亮点**: 可视化界面、功能全面、易于使用

🧰 数据处理

Easy-Dataset - 数据处理工具集

- ⭐ **推荐指数**: ★★★★☆
- 📖 **内容**: 数据清洗、格式转换、质量评估
- 🎯 **适合**: RAG 数据准备、知识库构建
- 💡 **亮点**: 自动化工具、提升数据质量

🧠 理解大模型原理（加分项）

nanoGPT - Karpathy

- ⭐ **推荐指数**: ★★★★★
- 📖 **内容**: 从零实现 GPT，理解模型原理
- 🎯 **适合**: 深入理解 LLM 工作机制、面试加分
- 💡 **亮点**: 代码简洁、注释详细、理解本质

nanochat - Karpathy

- ⭐ **推荐指数**: ★★★★☆
- 📖 **内容**: 从零构建对话模型

- 🎯 **适合**: 理解对话系统、端到端实现
- 💡 **亮点**: 完整流程、实战导向

🎯 完整学习路径

[AgentGuide - AI Agent 完整学习路线]

- ⭐ **推荐指数**: ★★★★★
- 📖 **内容**: Agent 开发、RAG 系统、上下文工程、面试指南
- 🎯 **适合**: 系统化学习、求职准备、技术路线规划
- 💡 **亮点**: 开发岗/算法岗双路线、实战项目、简历模板

💡 学习建议

入门阶段（第1-2周）

1. 先学习 **Hello-Agents** 建立 Agent 开发基础
2. 浏览 **nanoGPT** 了解模型原理（可选）

进阶阶段（第3-6周）

1. 深入 **All-in-RAG** 学习 RAG 系统搭建
2. 使用 **LLaMA-Factory** 进行 Function Call 微调（可选）
3. 用 **Easy-Dataset** 处理数据

实战阶段（第7-8周）

1. 参考 **AgentGuide** 完成项目
2. 构建完整的生产级系统
3. 准备面试和简历

🛠️ 推荐技术栈组合

RAG 系统项目

```
后端: FastAPI + LangChain + Milvus/Chroma
数据处理: Easy-Dataset
监控: LangSmith + Prometheus + Grafana
部署: Docker + Docker Compose
```

Multi-Agent 项目

```
框架: CrewAI / AutoGen / LangGraph
工具: LangChain Tools + Custom Tools
工作流: State Machine + Task Queue
监控: LangSmith + 自定义日志
```