AI KNOWLEDGE SYSTEM

karpathy's LLM Wiki

OBSIDIAN PRACTICAL GUIDE

新手快速理解 karpathy's LLM Wiki

這套方法的重點不是「把資料全部塞進去」，而是把原始資料、AI 生成 Wiki、規則文件和日常提問串成一個能持續運作的知識系統。

三種文件　三個操作　三個提效工具

> 適合對象：剛開始搭建個人或團隊 AI 知識庫、想用 Obsidian 落地的人。

@Akiryou0206
iiaiuii

FRAMEWORK

先記住總框架：3 類文件 + 3 個動作 + 3 個提效工具

把它看成一個知識工廠：原料進來，整理成 Wiki，再被查詢、修正和持續維護。

FILES

3

文件類型

先分清楚要存什麼

Raw Resource
Wiki
Schema

WORKFLOW

3

日常操作

知識庫怎麼運轉

Ingest
Query
Link / 審查

SPEED UP

3

提效工具

讓 AI 更快找到東西

Index
Log
RAG

最重要的理解：這不是「多記筆記」，而是讓 AI 按規則把資料加工成可重用知識。

AIU

@Akiryou0206

iiaiuii

FILES

三種文件：先分清原料、知識頁和規則

新手最容易把三種文件混在一起，結果越記越亂。

RAW RESOURCE

原始資源

原料倉

存原始資料，例如 PDF、文章、影片、逐字稿。
它不是總結頁，而是你的原材料倉庫。

THE WIKI

維基文件

知識頁

由 AI 讀取原始資源後生成頁面。
每個實體、概念、專案單獨成頁，並建立引用關係。

THE SCHEMA

規則文件

說明書

定義加工流程、命名規範、頁面結構、連結規則。
它是你和 AI 的協作協議。

實務上：Raw 保存原文，Wiki 保存「已經理解後的結構化知識」，Schema 保存「以後都按什麼標準理解」。

WORKFLOW

三個日常操作：攝取、問答、審查

真正讓知識庫「活起來」的，不是收集，而是這三個動作形成循環。

1

Ingest

攝取

把新資料餵給 AI。

AI 按 Schema 讀取、抽取實體和概念、更新 Wiki 與索引。

2

Query

問答

基於 Wiki 提問。

AI 用已經整理好的知識回答，並回指原始來源，不必每次重讀原文。

3

Link

審查

讓 AI 定期掃描 Wiki。

找矛盾、舊結論、孤立頁面、缺失連結，並給出合併或拆分建議。

建議固定節奏：有新資料就 Ingest，有問題先 Query，每週或每月做一次 Link 審查。

TOOLS

三個提效工具：Index、Log、RAG

它們不是給人看的花活，而是讓 AI 更快找到東西、更少重複勞動。

INDEX

目錄頁

幫 AI 先知道有哪些頁

AI 自動維護所有 Wiki 頁面清單，並給每頁一句摘要。
作用：讓 AI 快速知道「庫裡有哪些內容可查」。

LOG

操作日誌

幫 AI 記得最近做過什麼

按時間記錄操作：讀了什麼、建立哪些頁面、做了什麼連結。
作用：幫助 AI 追蹤最近研究脈絡。

RAG

大庫搜尋

Wiki 變大後再啟用

當 Wiki 超過 1000 頁後，用本地搜尋工具做 BM25 + 向量混合檢索。
作用：避免大庫查詢變慢、變漏。

一句話理解：Index 幫 AI 找「頁」，Log 幫 AI 找「過程」，RAG 幫 AI 在大庫裡找答案。

OBSIDIAN

Obsidian 怎麼搭：資料夾、頁面類型、工作流

新手不要先追求花俏外掛，先把結構定清楚。

推薦目錄結構

```
Raw
Wiki / Concepts
Wiki / Entities
Wiki / Compare
Wiki / Source Notes
Index.md
Log.md
Schema.md
```

Raw 放原文，Wiki 放 AI 生成頁，Schema 單獨放。

這套結構怎麼運轉

- **Schema 先寫什麼** 頁面結構、命名規則、連結規則、哪些內容由 AI 生成。
- **Ingest 時做什麼** 讀取 Raw，抽實體與概念，更新 Wiki / Index / Log。
- **Query 時做什麼** 優先基於 Wiki 回答，必要時回看 Raw，並保留來源。
- **Link 時做什麼** 掃孤立頁、矛盾頁、過時表述，給出修正建議。

WIKI OUTPUT

AI 生成出來的 Wiki，通常長這樣

重點不是頁面多，而是「每一頁都有明確角色，並且彼此可連結」。

CONCEPT

概念頁

例如：**AI agent**、**skill** 系統

會寫定義、摘要、相關概念與來源。
概念之間要能互相跳轉。

ENTITY

實體頁

例如：人物、組織、專案

會寫核心理念、專案歷程、與哪些概念相關。
實體要能掛回具體材料。

COMPARE

對比頁

例如：**PAI vs OpenCola**

把多個系統放在同一張表裡橫向比較。
對比頁通常最適合複習和復盤。

在 Obsidian 裡，這些頁面真正有價值的地方，是能透過雙向連結形成知識關係圖。

BLIND SPOTS

三個最容易被忽略的盲點

很多人失敗，不是因為不會做 Wiki，而是忽略了這些使用層面的現實。

01

原始資料本身也要學

入門階段，教程 PDF 和影片的章節結構，比 AI 總結頁更適合系統學習。

Wiki 更適合回顧與複習。

02

AI 生成內容必須驗收

不要無腦囤積。

要檢查連結是否有上下文、頁面是否真的可讀、內容有沒有誤導。

03

內容也要為 AI 可讀

Index、Log、type 屬性這些設計，很多時候是為了讓 AI 日後更好檢索、更好維護。

一句話：別把知識庫當「倉庫」，要把它當「持續被 AI 和人共同維護的系統」。

AIU

@Akiryou0206

iiaiuii

START SMALL

新手可以怎麼開始

先做一個能跑起來的小系統，再慢慢補規則和自動化。

1 **先建資料夾**
Raw、Wiki、Index、Log、Schema 先分開。

2 **先寫 Schema**
先約定命名、頁面模板和連結規則。

3 **先跑一次 Ingest**
拿 1 份 PDF 或 1 篇文章做樣板。

4 **開始 Query**
讓知識庫先服務你自己的提問。

最後記住這 4 句

1. Raw 是原料，不是結論。
2. Wiki 是整理後的知識頁。
3. Schema 是你和 AI 的協作規則。
4. 知識庫的價值，在於可查詢、可修正、可持續維護。

先小規模跑通，再慢慢擴大。

AIU
@Akiryou0206
iiaiuii