Build Your Own LLM Wiki — From Raw Sources to Skills

Scaffold, ingest, enrich — 2026 DCI Summer Workshop, Day 2 Session 4

Kwok-leong Tang

2026-05-19

Table of contents

Today's session 2

Step 0 — Read Karpathy's gist (15 min) 3

The three-layer architecture 3

Step 1 — Get the corpus and scaffold the workspace 5

Step 2 — One prompt builds the scaffold 6

Step 3 — One more prompt ingests the entire corpus 7

Step 4 — Look at what you built, in Obsidian 9

Step 5 — Commit 9

Step 6 — Skills, the external-data layer 9

- Install a Skill — just ask Codex in plain English . . . 10
- Demo — enrich one wiki page with a Skill . . . 11

Optional follow-on — rename source files to article titles 11

Wrap-up — Day 2 12

Looking ahead — Day 3 12

Summary — what you should take away 13

Today's session

In Session 07 you read someone else's wiki. This session you **build your own** — starting from a real, narrowly-focused research corpus: **29 sources on Tang–Song fiscal history**, centered on the Two-Tax Law (兩稅法). Codex will scaffold the wiki, write extraction scripts, and ingest the whole corpus into a bilingual, citation-forward Obsidian vault that you take home with you.

The arc:

0. **Read Karpathy's gist** — 15 minutes with the source that named the pattern.
1. Set up a scaffold for your new wiki — folder structure, governance files, page templates.
2. Drop the **liangsui** corpus (29 PDFs + a few Markdown sources) into the `raw/` layer.
3. Send Codex one prompt — it generates `AGENTS.md`, a build script, source stubs, and Obsidian Bases views.
4. Send Codex one more prompt — it bulk-ingests the entire corpus into source / people / concept / institution / debate pages.
5. Enrich one page with a **Skill** that reaches outside the corpus (CBDB, Wikidata, calendar).

> **Note**
>
> **Teaching goal**: leave with a working personal LLM Wiki you have produced, plus a clear understanding of (a) the three-layer architecture (raw / wiki / schema), (b) the ingest operation, and (c) what Skills add on top.

> **Session materials**
>
> **Corpus** (download once, ~19 MB):
>
> - `liangsui.7z` — 29 sources on Tang–Song fiscal history (PDFs + cleaned Markdown). Extract with `7z x liangsui.7z` (macOS: `brew install p7zip`; Windows: install 7-Zip).
>
> **Companion**:
>
> - `README.md` — a one-screen overview of what's inside `liangsui.7z` and the two prompts you will paste into Codex.

Step 0 — Read Karpathy's gist (15 min)

Before we build anything, read the original statement of the pattern in the author's own voice. Open the gist and read it silently for 15 minutes:

> Andrej Karpathy, "LLM Wiki" gist (April 2026)

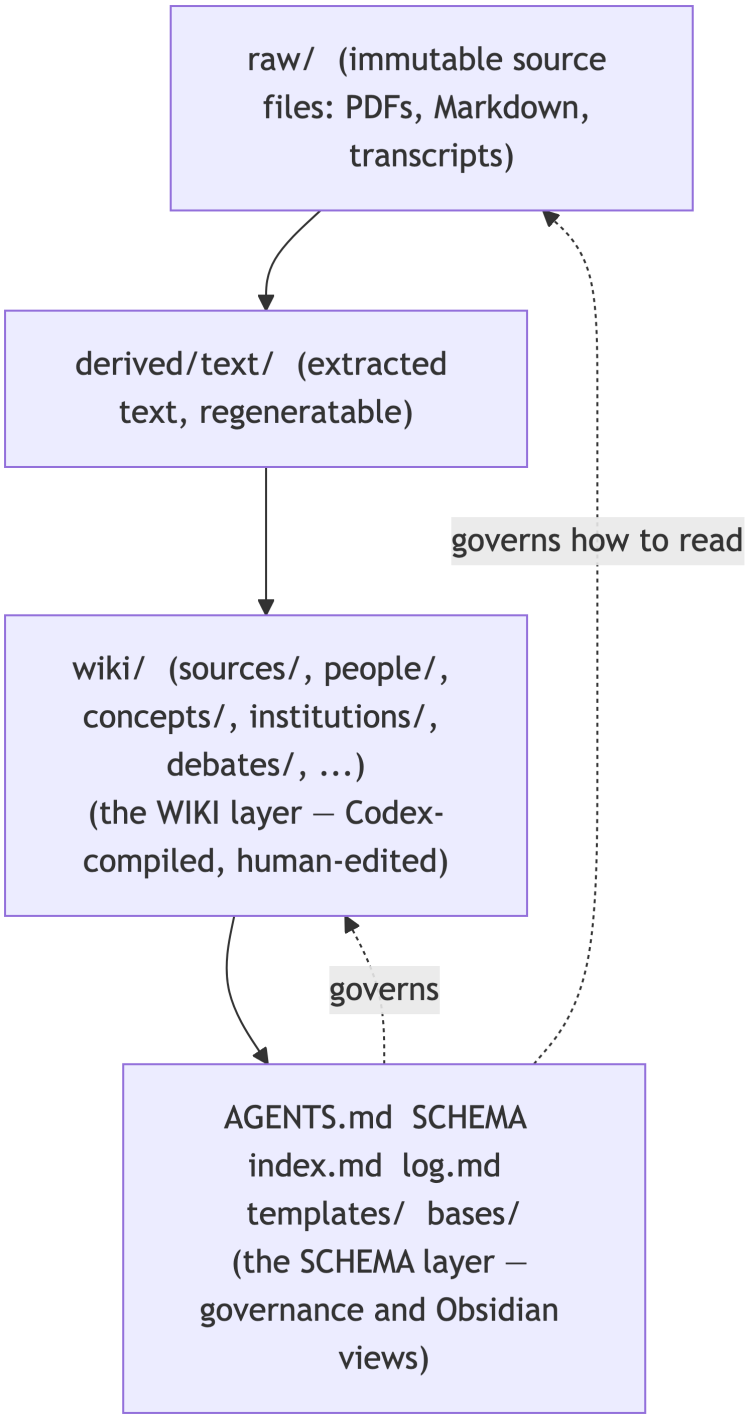
It's short — a few screens. Read it twice if you finish early.

> Read with these three questions in mind
>
> 1. **What are the three layers** Karpathy names, and what does each one *not* do?
> 2. **Why is `raw/` immutable?** What goes wrong if you edit it?
> 3. **What is `AGENTS.md` for?** In Karpathy's account, who reads it, and when?

Take notes in the margin or on paper. We will use your answers in the next section as we walk through the three-layer diagram and as the basis for the prompts in Steps 2 and 3 — both prompts encode answers to exactly these questions.

The three-layer architecture

With Karpathy in mind, here is the mental model in compressed form. Every LLM Wiki in this curriculum follows the same three-layer shape:


raw/ (immutable source files: PDFs, Markdown, transcripts)
derived/text/ (extracted text, regeneratable)
wiki/ (sources/, people/, concepts/, institutions/, debates/, ...) (the WIKI layer — Codex-compiled, human-edited)
AGENTS.md SCHEMA index.md log.md templates/ bases/ (the SCHEMA layer — governance and Obsidian views)
governs how to read
governs

Layer	What lives there	Who writes it	Mutability
Raw	Original source files — PDFs, MD, scans, transcripts	You (you collect them)	**Immutable** — never modify
Derived	Extracted/cleaned text from `raw/` — page-aware text from PDFs	A script (reproducible)	Regenerated on demand
Wiki	Short, single-topic pages compiled from the derived layer	Codex drafts, you verify	Edited continuously
Schema	Governance — `AGENTS.md`, templates, Obsidian Bases, `index.md`, `log.md`	You (you set the rules)	Edited rarely, but deliberately

The point of the separation is **reproducibility**. If your wiki ever gets corrupted, looks wrong, or grows in a way you regret, you can re-ingest from the raw layer with a better `AGENTS.md` and produce a cleaner wiki. **Your sources are safe; your compiled knowledge is regeneratable.**

> **i** The wiki shape is not fixed
>
> Session 06 introduced a generic LLM-Wiki shape — `concepts/, patterns/, tools/, examples/, readings/`. That shape fits workshops about AI tooling. For a *Tang–Song fiscal-history* corpus, the natural shape is different — `sources/, people/, concepts/, institutions/, debates/, chronology.md, bibliography.md`. The wiki shape is whatever your `AGENTS.md` says it is; the architecture (three layers, plain-text, version-controlled) is what stays constant.

Step 1 — Get the corpus and scaffold the workspace

Create the workspace and download the corpus into `raw/`:

```
mkdir -p ~/genai-workshop/liangsui-wiki/raw
cd ~/genai-workshop/liangsui-wiki
# Download liangsui.7z from the session's materials page, then:
7z x ~/Downloads/liangsui.7z -oraw
```

After extraction you should see ~29 files under `raw/` — Chinese-language PDFs and a few `.md` conversions, all on the Two-Tax Law and adjacent Tang–Song fiscal topics.

> ⚠ **Do not edit raw/**
>
> The whole point of the raw layer is that it is **immutable**. Anything you change inside `raw/` gets re-ingested next time and confuses the wiki. If you want to annotate a source, write a wiki page about it — do not annotate the source PDF itself.

Now open the folder as a Codex project (and as an Obsidian vault — click the vault switcher in Obsidian's bottom-left corner, then *Open another vault* → *Open folder as vault*).

Step 2 — One prompt builds the scaffold

Send Codex this prompt. It is the **scaffold prompt** — it does not ingest content; it builds the wiki's *operating system* and writes per-source stubs.

```
I want to use the material in raw/ to create an LLM-Wiki following
Karpathy's pattern:
https://gist.github.com/karpathy/442a6bf555914893e9891c11519de94f

The corpus is ~29 sources (PDFs + Markdown) on Tang–Song fiscal
history, centered on the Two-Tax Law (兩稅法).

My preferences:

1. Bilingual — Chinese as the primary scholarly language, with English
   summaries after major sections.
2. Page-level citations everywhere claims matter
   (e.g., `Source: [[sources/陈明光-量出制入-1986]], p. 2–4`).
3. Both a research companion for reading AND a queryable reference wiki.
4. Optimize for Obsidian and Obsidian Bases (.base files):
   https://obsidian.md/help/bases

Please set up — without ingesting content yet:

- AGENTS.md with the wiki rules (language convention, citation
  convention, page-type definitions, ingest/query/lint workflow)
- derived/text/ for page-aware extracted PDF text
  (use pdftotext where available; fall back to OCR only if needed)
- wiki/ with: sources/, concepts/, people/, institutions/, debates/,
  plus index.md, log.md, chronology.md, bibliography.md
```

```
- scripts/build_source_scaffold.py — a reproducible script that
  extracts text from raw/ into derived/text/, generates
  derived/source_manifest.csv, and writes a stub page per source
  into wiki/sources/
- wiki/templates/ — source, concept, debate page templates
- wiki/bases/ — Obsidian Bases for sources, concepts, and a research
  workbench, using clean YAML page properties (page_type, status,
  topics, citation fields)
- A handful of seed pages for the most central concepts (兩稅法,
  量出制入, 兩稅三分, 錢荒, 法外加徵), people (楊炎, 劉晏, 陸贄,
  唐德宗), and institutions (度支, 鹽鐵, 戶部, 藩鎮), plus one debate
  page for the historiographical controversy around 量出制入

Then run the script, validate that all YAML/frontmatter and .base
files parse cleanly, and report what was produced.
```

Watch the Review pane on the right. Codex will:

1. Inspect `raw/`, check that `pdftotext` is on the system, plan the structure.
2. Write `AGENTS.md` (the schema layer — this is the most consequential file in the wiki).
3. Write `scripts/build_source_scaffold.py`.
4. Run the script — extract every PDF, generate 29 source stubs, build a manifest CSV.
5. Add the seed concept/people/institution/debate pages.
6. Validate.

Approve each file. By the end you should see ~29 source stubs in `wiki/sources/`, ~5 concept pages, ~4 people, ~4 institutions, 1 debate, and three `.base` files in `wiki/bases/`. Refresh Obsidian — the new vault is visible in the sidebar.

> **!** Important
>
> **`AGENTS.md` is the most consequential file in the wiki.** Open it. Read it end to end. It tells Codex (and any future agent) how to write into your wiki — language, citations, page shape, what to do when sources disagree. Anything you want enforced across every future ingest goes there.

Step 3 — One more prompt ingests the entire corpus

The scaffold has 29 *stub* source pages — they have frontmatter but no real content. Now we tell Codex to actually read each source and fill in the source pages, expand the concept/people graph, and write the bibliography. **No ingest script this time** — Codex itself reads each source and writes each page. The agent *is* the ingester.

```
Now do a full first-pass ingest of every source in raw/. Do this
directly — read each source's page-aware extracted text in
derived/text/ and write the corresponding wiki/sources/*.md page
yourself. Do not write a script for the ingest; you are the ingester.

For each source:
  - Read the extracted text in derived/text/
  - Identify the abstract (usually the first paragraph after the
    title, often labeled 提要 / 内容提要 / 摘要) and place it at the
    top of the source page
  - Write one page-level note section per page of the source
  - The first time a recurring concept term appears in a source page
    (e.g., 兩稅法, 量出制入, 兩稅三分, 錢荒, 攤逃, 色役, 唐宋轉折),
    render it as a [[wikilink]]
  - Set `ingested: true` in the source page's frontmatter
  - If you encounter a concept, person, institution, or debate that
    deserves its own page and one doesn't yet exist, create the stub
    and link to it
  - Skip any source page whose frontmatter shows
    `manual_refined: true` so hand-refined work is not overwritten

When all sources are written:

1. Rebuild wiki/index.md, wiki/log.md, and wiki/bibliography.md from
   the source manifest.
2. Validate: YAML/frontmatter parses cleanly; all .base files parse
   cleanly; every source page shows `ingested: true`.
3. Report a summary — number of source pages, number of new
   concept/people/institution/debate pages added, number of
   page-level note sections, anything that needs follow-up.
```

This is the moment the wiki goes from a skeleton to something with intellectual content. The pass produces ~250 page-level note sections across the 29 sources, expands concepts from the seed five to closer to 15–20, and gives every source a fillable bilingual record.

> **!** Important
>
> **You are not done when the ingest finishes — you are at *first pass*.** Auto-generated page notes are excellent for navigation but quotations and bibliographic metadata should be checked before formal citation. The wiki's *intellectual spine* — the debate pages and the connections between them — is the part that pays off when you hand-refine after class.

Step 4 — Look at what you built, in Obsidian

Switch to Obsidian. Three things to look at:

- **`wiki/bases/research-workbench.base`** — open it. Bases turn YAML properties into live tables. You can filter sources by topic, status, or whether they take a position in a debate.
- **The graph view** (☐ G) — the corpus is now a network. Click 量出制入 in the graph; see which sources connect to it.
- **One source page** — open `wiki/sources/` and pick any source. Read the bilingual summary. Click through the `[[wikilinks]]` to the people and concepts it cites. This is the "research companion for reading" your scaffold prompt asked for.

If the wiki is going to live with you for months, an obvious next step is to hand-refine the **debate page** (`wiki/debates/debate-liangchu-zhiru-principle.md`). That is the wiki's intellectual spine — the thing you would actually publish from. We will not do it in class; it is the kind of slow work the wiki exists to support.

Step 5 — Commit

Same git workflow as Session 05. In the terminal pane (☐ J):

```
cd ~/genai-workshop/liangsui-wiki
git init
git add -A
git commit -m "Initial scaffold + bulk ingest of liangsui corpus"
```

You can push it to GitHub now (`gh repo create liangsui-wiki --private --source=. --push`) or wait.

> Tip
>
> **Private vs public** — for a personal wiki that contains your half-formed thoughts and unfinished pages, `--private` is the safer default. You can flip it to public later. Public wikis like `kltng/digital-china-wiki` work because the author deliberately curates for a public audience.

Step 6 — Skills, the external-data layer

Your wiki has a real limit: **it only knows what the corpus already contained.** If you want a wiki page on the historical figure 陆贽 (Lu Zhi, 754–805) to include his ancestral region,

his official titles, his examination cohort — none of that lives in the Tang–Song fiscal-history corpus. Those facts live in CBDB.

That is where **Skills** come in. A Skill is a packaged capability the agent can invoke — usually a wrapper around an external database, a search API, or a deterministic computation. The Skill itself is a small directory of files (instructions + helper scripts) that Codex reads at the start of a session; once installed, the agent uses the Skill the way you would use a colleague's expertise.

The Skills we use in this workshop were built by **Kwok-leong Tang** for humanities researchers, and they live in a public repository: `kltng/humanities-skills`. Skills relevant to this corpus include:

Skill	What it does
`cbdb-api`	Looks up historical Chinese figures in the **CBDB** (China Biographical Database). Perfect for the Tang figures in this corpus.
`chgis-tgaz`	Looks up historical Chinese placenames with coordinates and dates — useful for tracing 藩镇 and provincial postings.
`cjk-calendar`	Converts Chinese reign-year dates (e.g., 建中元年) to Gregorian.
`wikidata-search`	Queries Wikidata for entities by name; structured facts in many languages.
`harvard-library-catalog`	Searches HOLLIS for bibliographic records — extends the bibliography beyond the 29-source corpus.

A Skill turns a question Codex *cannot* answer from training data alone into a question it *can* answer by calling the right tool — and then writing the answer into the wiki.

Install a Skill — just ask Codex in plain English

You do not need to clone the repo, copy folders, or edit any config file. Codex understands natural-language installation requests. In your Codex Desktop session, simply say:

```
Please install the cbdb-api Skill from
https://github.com/kltng/humanities-skills into this project.
```

Codex will fetch the skill directory, place it where it needs to go, and confirm. If you want several skills, list them in one prompt — `"install cbdb-api, chgis-tgaz, and cjk-calendar`

`from https://github.com/kltng/humanities-skills"` works just as well. Approve each step in the Review pane.

> **Tip**
>
> **The repo is public** — anyone can read, install, or contribute. If you build your own Skill later (Day 3), the same pattern applies: put it in a public Git repo, point Codex at the URL, and it installs.

Demo — enrich one wiki page with a Skill

With `cbdb-api` installed (see above), ask Codex:

```
Open wiki/people/陆贽.md. Use my installed cbdb-api Skill to look up
陆贽 (Lu Zhi, 754–805) by name and dates. Add a one-paragraph
bilingual biographical note (Chinese first, English second) next to
the page's opening summary — birth year, ancestral region, examination
cohort, principal offices — and cite the Skill's response in the
Sources section as `Source: CBDB <person_id>`. Do not invent any
field the Skill does not return; if a field is empty, omit it.
```

Approve. Codex calls the Skill, gets back structured data, incorporates it into the page in your bilingual house style, and updates the source attribution. Refresh Obsidian — the new content is there, indistinguishable from your hand-written prose except that it cites CBDB.

> **What just happened**
>
> You used four layers at once: **raw** (the Tang–Song corpus), **wiki** (the page you ingested), **schema** (`AGENTS.md` instructed the agent to write bilingually with page-level citations), and **Skill** (CBDB fetched at runtime). This is the full LLM Wiki architecture in action. For the rest of your research career, every new database you connect via a Skill becomes a *capability* of your wiki. The wiki grows in two directions: **deeper** through better ingestion of more raw sources, and **wider** through more Skills connecting to more data.

Optional follow-on — rename source files to article titles

The scaffold script names source pages `src-001.md`, `src-002.md`, etc. — fine for the manifest, ugly to read in Obsidian. If you want article-title filenames instead:

```
Change source filenames from src-### to article titles. Update
scripts/build_source_scaffold.py (and any other script that
references src-### names) so future regeneration uses article-title
note names. Rename existing
wiki/sources/src-*.md and derived/text/src-*.md files to their
article-title equivalents. Rewrite every Obsidian link in the wiki
to use the new filenames. If two records are duplicates of the same
article in different formats (e.g., a Markdown conversion and a PDF
extraction), distinguish them with a `-pdf` suffix. Log the rename
in wiki/log.md.
```

This is a homework-friendly task — the script-update + bulk rename + link-rewrite is the kind of refactor the wiki pattern handles cleanly because everything is plain text.

Wrap-up — Day 2

In one morning + one afternoon you went from "what is an LLM" to "I have my own LLM Wiki under version control, ingested from real research material and enriched by an external Skill." That arc is the spine of the curriculum.

Specifically you have produced today (Day 2):

- A working **HTML batch tool** for any text task an LLM can handle (Session 05).
- That tool **deployed to the public web** under your name (Session 05).
- Literacy in reading **someone else's LLM Wiki** at about a thousand pages (Session 07).
- A working **personal LLM Wiki** of your own — built from a real historical corpus (Session 08).

Looking ahead — Day 3

Day 3 deepens both directions:

- **More raw sources** — your own research material, not the liangsui corpus. We will look at OCR pipelines, file-system organization, and what makes a raw layer easy to ingest from.
- **More Skills** — you will install more of them, write your own simple ones, and learn the difference between a *Skill* and a *full MCP server*.
- **Classical DH methods** — TEI, NLP, historical GIS, network analysis. How the agent fronts the classical pipeline rather than replacing it.

Summary — what you should take away

1. **An LLM Wiki has three layers: raw / wiki / schema** — plus a `derived/` cache when sources need extraction. Raw is immutable; the wiki is regeneratable; the schema is the rules.
2. **`AGENTS.md` is the most consequential file in the wiki.** It defines the language, the citation style, the page shape, and how to handle disagreement. Everything downstream follows from it.
3. **Two prompts build the wiki.** Prompt 1 scaffolds — `AGENTS.md`, scripts, templates, stubs. Prompt 2 ingests — bulk-writes the source pages and grows the concept graph. Anything beyond is hand-refinement.
4. **The wiki shape fits the corpus.** A fiscal-history wiki has `sources/`, `people/`, `concepts/`, `institutions/`, `debates/`. A workshop wiki has `concepts/`, `patterns/`, `tools/`, `examples/`, `readings/`. Architecture stays constant; folders adapt.
5. **Two prompts, two roles.** The scaffold prompt produces a reproducible script (`build_source_scaffold.py`) because the work is mechanical — extract text, write stubs, generate a manifest. The ingest prompt has Codex *read and write directly* because the work is interpretive — judging which passage is the abstract, which terms deserve a wikilink, which page deserves its own stub. Mechanical work goes in scripts; interpretive work stays with the agent.
6. **Skills connect the wiki to the outside world** — CBDB, Wikidata, CHGIS, calendar. The difference between a wiki that knows what was in your sources and a wiki that can answer questions that need fresh data.
7. **The wiki survives.** Codex, Obsidian, MCP, Skills — every tool you use today is replaceable. The markdown files are what you keep, for as long as you keep researching.

References

- Andrej Karpathy, "LLM Wiki" gist (April 2026) — the canonical statement of the pattern.
- Obsidian Bases — official documentation for the `.base` file format.
- `kltng/digital-china-wiki` — the worked example from Session 07.
- Codex documentation hub — Skills, `AGENTS.md`, MCP.
- Wiki patterns: `patterns/llm-wiki-pattern.md`, `patterns/three-layer-architecture.md`, `patterns/ingest-query-lint.md`.