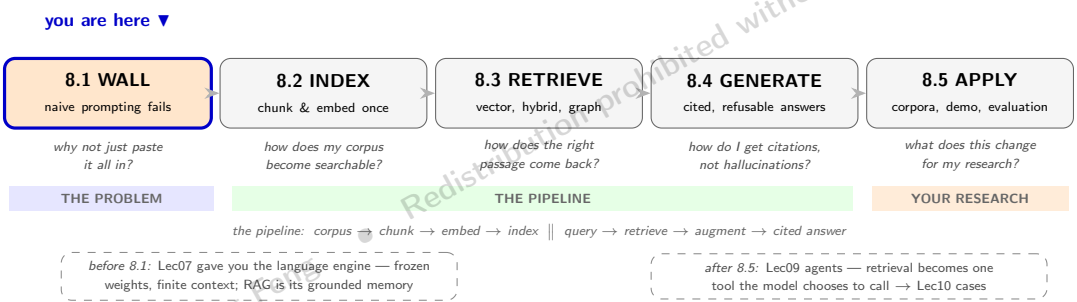
AI for Economic Research: Dynamic Models, Language, and Agents
Lecture 8.1: The Wall — Why Naive Prompting Fails

Zhigang Feng

2026-07-10 19:43:27

Lecture 8 in One Map

One lecture, five acts: hit the wall, build the machine that walks around it, then put it to research work.



Route this deck: the frozen brain → four walls → a wiki by hand. Case threads: (A) the **FOMC corpus**, *opened by the December-2025 question; (B) the* **OLG literature** *(157 papers), queried by graph in Act 8.3 and by hand in Act 8.5.*

Where We Stand

The language engine: frozen weights, finite context

The Frozen Brain: What Lecture 7 Left Us

Lecture 7 left us a powerful reader whose knowledge is frozen: everything it "knows" lives in weights fixed at the end of pretraining.

- **The machine** (Lec. 7): tokens → embeddings → attention layers → a probability distribution over the next token. Knowledge is *parametric* — there is no memory beyond the weights and the current context window.
- **Knowledge cutoff:** every model's training corpus stops months before you use it. And your *private* corpora — working papers in progress, internal memos, FOIA-released records — were never in anyone's training data at all.

Lecture 8's opening question (Lec. 7 T5a closed on it): *"What did the FOMC decide at its December 2025 meeting?"* A model trained before that meeting has *never seen* the statement. It will either refuse — or, worse, hallucinate a plausible but fictional answer.

Today's question: how do we extend the model's knowledge *at inference time*, without retraining it?

Three Pain Points for the Economist

For research use, the frozen brain fails in three specific ways.

- **1. Stale knowledge.** Anything published after the cutoff is invisible. So is anything proprietary. Macro and finance research runs on the *latest* releases.
- **2. Hallucinated citations.** Ask a vanilla LLM to "cite five NBER papers on the slope of the Phillips curve" and it will happily produce five plausible titles, plausible authors, and entirely fake DOIs. This is the most common failure mode in academic use.
- **3. No source attribution.** Even when the answer is correct, the model cannot tell you *where* the claim came from. Academic research requires footnotes.

Requirements for research-grade work: a system that can (i) read documents the model never saw, (ii) refuse to invent facts, and (iii) tell us which document supplied each claim.

Two Ways to Add Knowledge

There are exactly two places to put new knowledge: into the weights, or into the prompt.

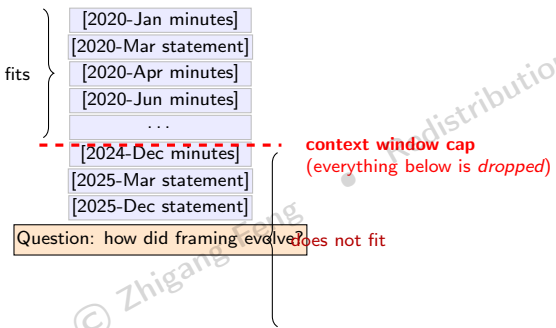
- **Option A — Fine-tuning:** re-train (some of) the model's weights on your new corpus.
  - Pros: knowledge becomes "baked in," fast at inference.
  - Cons: expensive, fragile, hard to update, no source attribution, doesn't fix hallucination.
- **Option B — Retrieval (RAG):** leave the model untouched. At query time, fetch the relevant text from an external store and paste *only that* into the prompt.
  - Pros: cheap, updatable in real time, gives source attribution, dramatically reduces hallucination.
  - Cons: more moving parts; quality depends on the retriever.
- **This lecture's promise:** Acts 8.2–8.3 build the index and the retrieval; Act 8.4 turns retrieved evidence into a cited answer — and settles A vs. B with a decision matrix; Act 8.5 applies the pipeline to six research corpora.

The Wall

Four walls: the window, the middle, the meter, the memory

A Tempting First Attempt: Paste It All In

Setup. The naive, RAG-less workflow is the simplest possible baseline: paste the *whole corpus* into the prompt and ask directly. **Research question:** *"How did the FOMC's framing of inflation expectations evolve from 2020 to 2025?"*



Four walls block this approach (toured in pairs next):

1. the **window** — context cap (red line)
2. the **middle** — "lost in the middle" (Lec. 7 T5a)
3. the **meter** — pay every token, every query
4. the **memory** — no persistence across queries

In principle the model has all the data;
in practice all four walls bind.

Hard Walls: The Window and the Middle

Wall 1 is a hard cap; wall 2 degrades reading even inside the cap.

- **The window.** Frontier models read ∼1M tokens at once (mid-2026); many deployed models cap far lower. Long corpora *force* chunking, summarization, or retrieval (Lec. 7 T2a).
- **How big is the FOMC corpus?** 8 meetings/year × [statement ∼1K + minutes ∼15K tokens] ≈ 130K/year *excluding* transcripts; five years (2020–2024) ≈ 650K — it *barely* fits, and only as the whole budget.
  - Transcripts run ∼100K *each* (5-year lag): one year approaches 1M alone. Thirty years of statements+minutes ≈ 3.9M; add speeches, Beige Books, NBER papers, 10-Ks — the research corpus **never fits**.
- **The middle.** Even what fits is read unevenly: recall drops 20–30 pp. when evidence sits *mid-prompt* — the position–accuracy curve is U-shaped (Liu et al. 2024, *TACL*; Lec. 7 T5a). A ∼600K-token prompt also takes 30+ seconds to the first output token.

The December 2025 statement, concatenated after four years of minutes, lands mid-prompt — where the model reads worst. Cramming *hurts* quality, not just cost.

Economic Walls: The Meter and the Fresh Start

Even when pasting works once, it fails as a workflow: you re-pay the whole corpus on every question, and nothing accumulates.

- **Wall 3: the meter.** You pay for *every input token, every query* (per-token rates: Lec. 7 T4, "Tokens, Cost, and Reproducibility"). Stuff ∼600K tokens of FOMC text into each query and one thousand exploratory queries re-read — and re-bill — the archive one thousand times. Prohibitive at the scale of a literature review.
- **Wall 4: the memory.** Naive prompting treats every query as a **fresh start**: no reuse of reading work across queries, no incremental update when next month's minutes arrive, no way to grow past the window — ever.
- **Compare a real research workflow:** collect documents *once*; index them *once*; ask many questions over time, paying only for what you actually retrieve; index new arrivals incrementally.

The pattern we need: separate the cost of *ingesting documents* from the cost of *asking questions*. (The cost crossover gets its own picture in Act 8.4.)

The Wall

Line up what naive prompting can do against what research actually needs, and the gap is structural — not a matter of better prompts.

Naive prompt engineering can do	Research workflows actually need
QA on a single short document	QA over millions of tokens of corpus
Tens of queries before cost bites	Thousands of queries over weeks
Whatever fits in the context window	Decades of FOMC + NBER + 10-K filings
Today's prompt only	Incremental updates as new docs arrive
"Trust me" answers	Cited, source-attributed answers

Conclusion: prompt engineering hits a hard wall on freshness, scale, latency, cost, and attribution. We need an architecture that **separates indexing from querying**.

This is what RAG does.

The Human-Scale Fix: A Personal Wiki

Retrieval by hand — the pre-formal RAG

Before the Formalism: A Personal Wiki

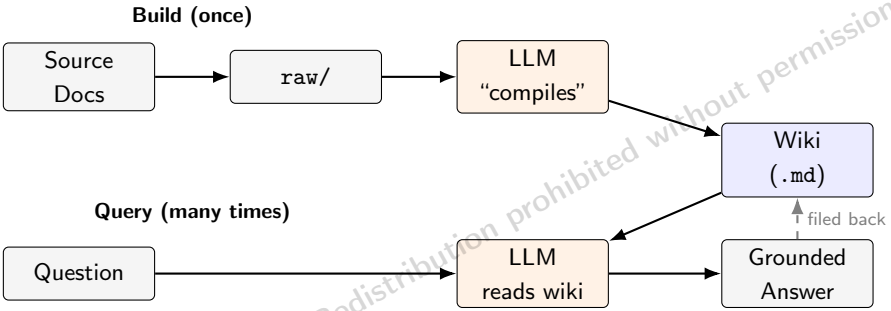
Before the full machinery, meet the same idea at human scale — a knowledge base you can set up today.

- **Andrej Karpathy** (June 2025):
  "I'm finding very useful recently: using LLMs to build personal knowledge bases for various topics of research interest. [...] *A large fraction of my recent token throughput is going less into manipulating code, and more into* **manipulating knowledge** *(stored as markdown and images)."*

- **The workflow is disarmingly simple:**
  1. Collect source documents (papers, articles, data) into a `raw/` directory.
  2. Ask an LLM to *compile* them into a **wiki** — a directory of `.md` files with summaries, concept articles, and cross-links.
  3. Query the wiki by asking the LLM questions; it reads the relevant articles and answers from them.

- **Why this matters for us:** this is the *same core insight* behind RAG — separating data ingestion from querying — but without any of the formal machinery.

The Wiki Pipeline



- **Build:** papers, articles, and data land in `raw/`; the LLM compiles them into wiki articles — summaries, concept pages, back-links. *You rarely edit the wiki manually.*
- **Query:** ask a question; the LLM reads the wiki's index and relevant articles, then answers with grounded citations.
- **Feedback loop (dashed):** answers get filed *back* into the wiki, so your research "adds up."

Hands-On: A Mini Wiki for This Course

A working example ships in this lecture's folder — browse it, extend it, query it.

```
wiki_demo/
  index.md                    -- master table of contents
  concepts/
    embeddings.md             -- text-to-vector mappings (Lec 7-8)
    rag.md                    -- retrieval augmented generation
    transformers.md           -- the architecture behind LLMs
  lectures/
    lec07_llm.md              -- key ideas from Lecture 7
    lec08_rag.md              -- key ideas from Lecture 8
  papers/
    vaswani_2017.md           -- Attention Is All You Need
    lewis_2020_rag.md         -- RAG original paper (NeurIPS)
  connections/
    cross_references.md       -- links across lectures
```


- Ask: *"How do the embeddings from Lecture 7 connect to vector stores in Lecture 8?"* — the LLM reads the relevant pages and gives a **grounded, cross-referenced** answer.

Wiki = Manual RAG: Now Automate It

The wiki is retrieval, LLM-curated — and its ceiling tells us exactly what to build next.

	Personal Wiki	**Full RAG Pipeline**
Ingestion	LLM reads raw docs, writes summaries	Chunk → embed → store in vector DB
"Retrieval"	LLM reads index, picks articles	Embedding similarity search (top-K)
Scale limit	∼400K words (index fits in context)	Millions of documents
Update	LLM rewrites/extends articles	Add new chunks incrementally

- **The natural progression:** naive prompting (dies at the window) → personal wiki (dies at ∼400K words, when the LLM can no longer hold its own index) → **RAG** (embeddings + similarity search; scales to millions of documents).
- Karpathy again: *"the LLM has been pretty good about auto-maintaining index files* [...] *at this ∼small scale."* Thirty years of FOMC text is **not** small scale.

Bridge: From Hand-Curated to Machine-Indexed

Act 1 hit the wall: a frozen reader behind a metered window, a corpus that will never fit, and a hand-built wiki that fixes it only at human scale.

- RAG is the wiki idea *formalized and automated*: instead of an LLM maintaining an index by hand, embeddings and similarity search do retrieval at arbitrary scale. (Karpathy: *"there is room here for an incredible new product instead of a hacky collection of scripts."*)
- **Next — Act 8.2 (INDEX):** the offline half of the pipeline. One picture of the whole machine, then its first two decisions: *chunking* (what is a retrieval unit?) and *embeddings* (text becomes geometry).
- Keep the two threads in sight: the FOMC minutes get chunked and embedded next; the OLG literature waits for Acts 8.3 and 8.5.

Arc: **Act 1 WALL** → *Act 2 INDEX* → *Act 3 RETRIEVE* → *Act 4 GENERATE* → *Act 5 APPLY.*