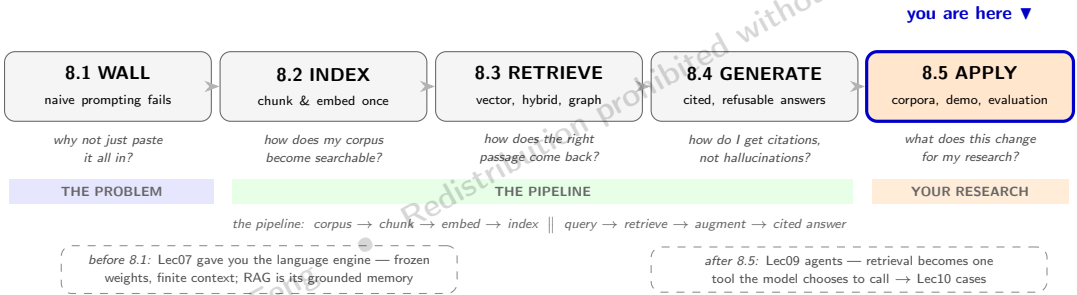
AI for Economic Research: Dynamic Models, Language, and Agents
Lecture 8.5: Apply — Research Applications, Demo, and Evaluation

Zhigang Feng

2026-07-10 19:45:42

Map: Act 8.5 — What This Changes for Your Research

The pipeline is built; this act spends it: six research corpora, a demo you run offline, the evaluation that makes it citable, and the bridge to agents.



Route this deck: six corpora (same plumbing, six research designs) → a classroom OLG demo you run offline → six failure modes + RAGAS + Cohen's κ → retrieval becomes one agent tool. Thread A (FOMC) is Use Case 1 and 6; Thread B (OLG literature) is the live demo.

Six Research Corpora

The same pipeline, six research designs

Use Case 1: FOMC Minutes & Press Conferences

- **Corpus.** The Federal Reserve publishes statements (post-meeting), minutes (3-week lag), press conference transcripts, and (5-year lag) full meeting transcripts. Available 1993–present at `federalreserve.gov`.
- **Index pipeline:**
  1. Scrape PDFs / HTML from the Fed archive.
  2. Parse into speaker turns (minutes) or Q&A pairs (press conferences).
  3. Chunk with rich metadata: `meeting_date`, `doc_type`, `speaker`, `topic_section`.
  4. Embed with `bge-large-en-v1.5`.
  5. Store in Chroma; run BM25 in parallel for hybrid retrieval.
- **Example research queries:**
  - *"How did the Committee's framing of inflation expectations evolve from 2021 to 2024?"*
  - *"Find every appearance of 'transitory' in FOMC minutes between 2020 and 2022."*
  - *"Which participants most often dissented on rate decisions, 2017–2024?"*
- **Reference:** Hansen, McMahon & Prat (2018), *QJE*, "Transparency and Deliberation Within the FOMC: A Computational Linguistics Approach."

Use Case 2: NBER Working Papers

- **Corpus.** ∼32,000 NBER working papers, 1973–present, available at `nber.org`. Each has metadata: title, abstract, authors, JEL codes, full PDF.
- **Use case: literature review automation.** Instead of typing keywords into Google Scholar and reading 50 abstracts, ask:
  > *"Find all NBER working papers from 2019 onwards that estimate the slope of the Phillips curve. Return title, authors, year, the central estimate, and the data source."*
- **Pipeline:**
  1. Index titles + abstracts as one collection (cheap, fast, good for keyword screening).
  2. Index full-paper chunks as a second collection (richer evidence for follow-up questions).
  3. Hybrid retrieval over both. Filter on `year`, `JEL_code`, `author`.
  4. Generator outputs a structured table from retrieved papers.
- **Caveat:** Always verify citations against the actual NBER record. Even with RAG, models occasionally garble author lists or year fields.
- **This is one of the highest-value RAG applications for an active researcher.**

Use Case 3: Congressional Record & Legislative Text

- **Corpus.** The Congressional Record contains every speech delivered on the House and Senate floors since 1873. Daily editions are available at `congress.gov`.
- **Use case: tracking how policy debates frame economic issues over time.**
  - "How has the rhetoric around 'inflation' shifted in House Banking Committee testimony from 2008 to 2024?"
  - "Which Senators most frequently cite Federal Reserve research in floor speeches?"
- **The structural challenge.** Congressional Record has a deep hierarchy — Congress / session / chamber / date / speaker / bill / section. **The Act 8.2 chunking gotcha applies in full force.** A naive chunker that ignores this hierarchy will produce essentially un-citable retrieval results.
- **Practical fix:** Use a Congress-specific parser (e.g. `congress-api-python`) to extract clean speaker turns with all metadata, then feed those into the chunker.
- **Reference:** Gentzkow, Shapiro & Taddy (2019), *Econometrica*, "Measuring Group Differences in High-Dimensional Choices: Method and Application to Congressional Speech."

Use Case 4: SEC 10-K Filings (Item 1A Risk Factors)

- **Corpus.** EDGAR hosts ∼5,000,000 SEC filings, including the annual 10-K reports of every public US firm. Each 10-K has a standardized "Item 1A: Risk Factors" section disclosing material risks.
- **Use case: cross-firm comparison of disclosed risks.**
  - *"Compare how the four largest US banks discussed interest-rate risk in their 2022 vs. 2023 10-K filings."*
  - *"Find all 10-K filings from energy firms in 2024 that mention transition risk from climate policy."*
- **Pipeline:**
  1. Use `sec-edgar-downloader` to pull 10-Ks.
  2. Run a section-aware parser to extract Item 1A only.
  3. Chunk *within* Item 1A, preserving `cik`, `ticker`, `filing_date`, `fiscal_year`.
  4. Store in Chroma. Hybrid retrieval with metadata filter on `ticker` for cross-firm queries.
- **References:** Loughran & McDonald (2011), *JF*, "When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks." Lopez-Lira (2023), *working paper*, "Risk Factors That Matter."

Use Case 5: Cross-Country Central Bank Speeches

- **Corpus.** The Bank for International Settlements maintains a unified archive of central bank speeches from ∼60 institutions, 1997–present, at `bis.org/cbspeeches`. Other primary sources: ECB, BoE, BoJ, RBA, BoC, RBI, PBoC.
- **Use case: comparative monetary policy analysis.**
  - *"How did major central banks frame the post-COVID inflation surge in 2022 speeches? Compare the Fed, ECB, BoE, and BoJ."*
  - *"Which central banks most often cite financial-stability concerns alongside inflation, 2010–2024?"*
- **Pipeline notes:**
  - Add `country`, `institution`, `speaker_role`, `language` as metadata.
  - For non-English speeches, either use a multilingual embedder (`multilingual-e5-large`) or translate first.
  - Hybrid retrieval is critical — many speeches use country-specific institutional jargon a generic embedder won't recognize.
- **This use case is essentially impossible without RAG — no human can read ∼10,000 speeches across ∼60 institutions per year.**

Use Case 6: Combining Structured + Unstructured

- RAG isn't only for unstructured text. The most powerful pattern combines a **structured database** (e.g. FRED time series, BEA NIPA tables) with an **unstructured text store** (Fed speeches, FOMC minutes).

- **Example query:**

  "In months when the unemployment rate fell below 4%, what did the Fed publicly say about labor markets?"

- **Pipeline:**

  1. *Structured side:* SQL on FRED data. `SELECT date FROM unrate WHERE value < 4.0;`
  2. *Unstructured side:* retrieve speeches and FOMC minutes whose `date` falls within those months.
  3. Pass the joined evidence to the LLM with a prompt that says "summarize how Fed officials characterized the labor market in these specific months."

- **This pattern** — structured SQL filter → semantic retrieval → generation — is uniquely useful for empirical economists. It bridges what economists already know how to do (data) with what they couldn't do before (text at scale).

Three Roles for RAG Evidence: Signal, Outcome, Instrument

Use case	Signal	Outcome	Instrument
FOMC minutes	FOMC stance over time	asset prices, expectations	communication shocks
NBER papers	methodological frontier	citation counts	field-maturity proxy
Congressional Record	policy rhetoric	bill passage, vote share	constituent preferences
SEC 10-K	firm-disclosed risk	stock returns, default	risk-disclosure rule shock
Central-bank speeches	cross-country stance	exchange rates	comparative monetary shock

Reading the table. RAG returns *evidence*; the role you assign it — **signal** (text-as-variable), **outcome** (text-as-target), or **instrument** (text-as-IV) — decides how you use it. Lec. 7 T1 introduced these three roles of text; they transfer one-to-one to RAG outputs (companion: Lec. 7 T5b's Application Zoo).

Same plumbing, different research design. Pick the role before you write the prompt.

Synthesis: Three Workflow Patterns

1. **Literature review.** Index a paper repository (NBER, SSRN, RePEc). Query: "find work on X, summarize, return citations." Saves days per project.

2. **Policy-text monitoring over time.** Index an institutional corpus (FOMC, Congress, central banks). Query: "how did the framing of Y evolve from year A to year B?" Turns institutional text into a longitudinal research instrument.

3. **Cross-sectional comparative analysis.** Index a corpus where the unit of analysis is a firm or country (10-Ks, central bank speeches). Query: "compare how entity-1, entity-2, ... discussed Z in year Y." Enables cross-firm and cross-country comparisons that were previously infeasible at scale.

Methodological punchline. RAG turns institutional text from *background reading* into a *queryable research instrument*. That is the new capability. Everything else — chunking, embeddings, vector stores — is plumbing.

Classroom Demo: OLG RAG

A pipeline you can run offline today

Demo: One Question Through the Whole Pipeline

Classroom question: *What did Diamond (1965) add to the OLG approach?* — answerable from the source paper *iff* retrieval finds the right paragraph. **Offline by default** (no API key, no network, no vector-DB service); source `./demos/M4_rag_olg_demo`.


```
cd demos/M4_rag_olg_demo
/usr/bin/python3 run_demo.py
/usr/bin/python3 run.py ask "What did Diamond 1965 add to the OLG approach?"
PYTHONPATH=src /usr/bin/python3 -m unittest discover -s tests
```

Artifact	What to inspect
chunk file	boundaries and line references
retrieval report	top-K evidence, scores, and anchor checks
extractive answer	whether every claim is supported by retrieved text
unsupported query	whether the system refuses rather than improvises

Files land under the demo's `build/` folder, including `chunks.json` and a retrieval-report CSV.

Demo: Audit the Corpus Before You Index

- Before chunking, *look at the corpus*. The OLG demo's `run_demo.py` prints a Step-1 audit:
  - character count, line count, heading count
  - the first ∼5 headings (sections of Spear–Young's paper)
- **Why this matters.** If the audit shows zero headings, your chunker will be flying blind: no section boundaries, no metadata to filter on later. The corpus audit is the cheapest sanity check in the entire RAG pipeline.
- **Practical rule.** If you cannot describe your corpus in two minutes from an audit print-out, you are not ready to index it.
- **Output to inspect:** `build/chunks.json` after Step 2 of `run_demo.py`. Each chunk carries `chunk_id`, `citation`, and `section` — the metadata that makes retrieval auditable.

Demo: Read the Retrieval Report First

What the demo prints (per question)

- top-K chunks with `score`
- `citation` (file + line range)
- `section` heading
- *anchor hits* — did retrieval surface the gold-standard phrases the instructor expected?
- `PASS/FAIL` verdict per question

Why anchor checks matter

- Score thresholds alone can lie — a low-quality chunk can still rank highest.
- Anchors are *specific phrases* that must appear in a correct retrieval.
- Anchor hit rate is the simplest cousin of context-recall (the RAGAS metric two acts down).
- Watching the PASS/FAIL trail tells you which question types your index handles — and which it doesn't.

Rule restated: read the retrieval report before the generated answer. The diagnostic is the publication-grade artifact.

Demo: Extractive Answer vs. Refusal

Supported query (Q3 in demo)

- Top retrieved chunks discuss Diamond's production-side extension of Samuelson OLG.
- Extractive answer quotes those chunks verbatim, with line citations.
- Anchor hits ≥ 1; `PASS`.
- This is what a publication-grade RAG should look like.

Unsupported query (refusal test)

- E.g. "What is the demo author's email address?"
- Top retrieved chunks have low score and no anchor coverage.
- Extractive answer returns: *"I do not have enough retrieved evidence in the provided source to answer."*
- This is the *feature*, not the bug.

The two behaviors are inseparable. A system that can answer "Diamond 1965" correctly *and* refuses to invent an email address is the minimum bar for using RAG in published research.

The grounded prompt behind this is exactly Act 8.4's template, with line-range citations `[file:La-b]`. Artifact: `build/terminal_retrieval_report.csv` — archive it next to your paper draft.

Failure Modes and Evaluation

When it breaks — and how to prove it works

When RAG Fails: Six Failure Modes

1. **Retriever miss.** Relevant document exists but the embedder ranks it outside top-K. Often caused by terminology mismatch.

2. **Re-ranker promotes the wrong context.** Cross-encoder mis-scores; the LLM gets a confident-but-irrelevant chunk.

3. **Generator ignores the retrieved context.** The LLM falls back on its parametric memory and answers from training data — often confidently and incorrectly.

4. **Chunk boundary cuts critical information.** The fact spans the boundary between two chunks; neither chunk alone contains the answer.

5. **Outdated index.** The corpus has been updated, but the index hasn't been re-run. The retriever returns stale chunks.

6. **Ambiguous query.** "Inflation" could mean CPI, PCE, expectations, or asset prices. The retriever picks one interpretation; the user wanted another.

Mitigations — One per Failure Mode

- **Retriever miss** ⇒ *query expansion* and *HyDE* (hypothetical document embeddings: ask the LLM to draft a fake answer, embed *that*, retrieve nearest documents). Also: hybrid sparse + dense retrieval.
- **Re-ranker mis-scores** ⇒ try a stronger re-ranker (`bge-reranker-v2`); train a domain-specific re-ranker on labeled query-document pairs.
- **Generator ignores context** ⇒ stricter system prompt ("Answer ONLY from the context"); lower temperature; faithfulness check in post-processing.
- **Chunk boundary cuts information** ⇒ semantic chunking; larger chunk overlap; or use a parent-document retriever (retrieve a chunk, return its parent paragraph).
- **Outdated index** ⇒ incremental ingestion pipeline triggered when source documents change; nightly re-embed for high-churn corpora.
- **Ambiguous query** ⇒ query rewriting (ask the LLM to clarify or expand the query before retrieval); multi-query retrieval (generate N paraphrases, retrieve for each, fuse).

Evaluation at scale stays expensive — hence the next step is a small, curated gold set, not exhaustive automatic scoring.

Evaluation: The RAGAS Framework

- Es, James, Espinosa-Anke & Schockaert (2023), *EACL*, "RAGAS: Automated Evaluation of Retrieval Augmented Generation." Now the standard.
- **Four metrics, two for retrieval and two for generation:**
  - **Context precision.** Of the retrieved chunks, how many are actually relevant to the query? *(Retriever quality.)*
  - **Context recall.** Of the chunks that should have been retrieved, how many did we get? *(Retriever coverage.)*
  - **Faithfulness.** Are all the claims in the answer supported by the retrieved context? *(Generator anti-hallucination.)*
  - **Answer relevance.** Does the answer actually address the query? *(Generator on-topic.)*
- RAGAS computes all four automatically using an LLM-as-judge. Cost: a few cents per query.
- **The right way to use it:** build a small (∼50-question) eval set for your specific corpus, run RAGAS each time you change a pipeline component, and only ship a change when the metrics improve.

Validate Like an Economist: Cohen's κ and Disclosure

RAGAS uses an LLM as the judge — fast for iteration, but it inherits the drift, sycophancy, and version-instability documented in Lec. 7 T5b (Yin, Vu & Persico, NBER 35110, 2026: $\kappa = 0.36$ across frontier LLMs on the same rubric).

For publication-grade research, layer a human-anchored audit on top:

1. Build a 100–200-item **gold standard**: queries paired with ideal retrievals *and* answers, curated by a domain expert.
2. Have a *second annotator* score retrieval relevance independently; compute **Cohen's** κ (target ≥ 0.75).
3. Treat RAGAS as the fast inner loop, the gold-standard κ audit as the slow outer loop — publish only the latter.

Disclose (AEA, Oct 2024, extends to RAG): embedding + vector-DB + retrieval-K + re-ranker; prompt template + generator + temperature; evaluator model + version; gold-set size + second-annotator κ; archive the retrieval report with the draft.

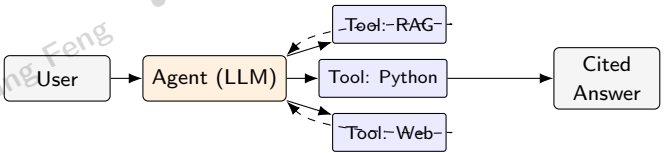
RAGAS tells you whether the pipeline got *better*; Cohen's κ tells you whether it is *good enough to cite*.

From RAG to Agents

Retrieval becomes one tool

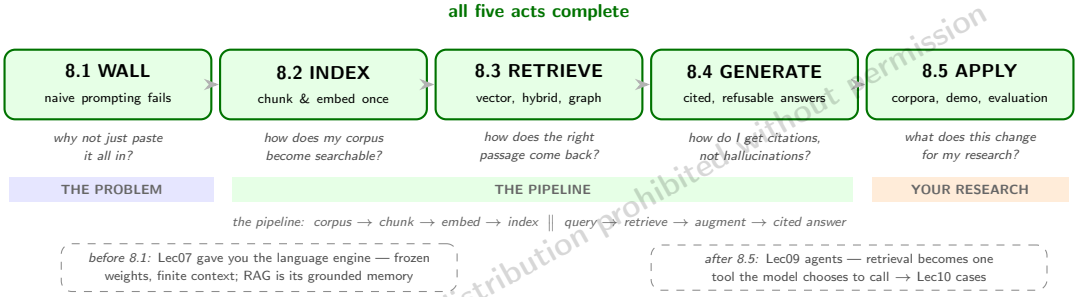
From RAG to Agentic RAG: Retrieval Becomes One Tool

- **Single-shot RAG** (today's pipeline): fixed plumbing — query → retrieve → augment → generate. One round; the developer wires the steps.
- **Agentic RAG:** the LLM *decides* whether, when, and what to retrieve, via the *JSON tool calls* introduced in Lec. 7 T4 ("Structured Output"). The loop continues until the model signals `done`.
- **Patterns this unlocks:** multi-hop retrieval (one query begets the next); *query rewriting* (the agent reformulates an ambiguous user question before retrieving); and *cross-tool composition* — RAG *plus* code execution *plus* web fetch *plus* structured-data queries.
- **Inherited failure modes (Lec. 7 T4/T5a):** the agent's context still drifts; sycophancy still bites. Worse: an agent can retrieve confirming evidence *and ignore disconfirming evidence* unless the loop is designed to penalize that.



Lec. 9 details the harness, the safety boundary, and a working terminal-agent walkthrough.

Close: One Map, Complete — Pick the Right Tool



Pick...	**when...**
RAG (today)	facts from a corpus the LLM never saw, with citations the reader can verify.
Fine-tuning	*style*, *format*, or rare *tokens* — not facts. (Pair with RAG; don't substitute.)
Agentic RAG (Lec. 9)	multi-hop retrieval, query rewriting, or cross-tool composition (RAG + code + web).
None of the above	the corpus fits one context window *and* the question is single-shot — long-context suffices.

Reasoner (LLM) + memory (index) + librarian (retriever) — RAG turns institutional text into a queryable research instrument. **Next: Lecture 9 (Session 6)** — a terminal agent that uses RAG as one tool among many. **Questions?**