FedGPT: Large-Scale Federal Reserve Communication Scoring and Portfolio Signal Research

Technical Report for Sponsor Review

Prepared by: Nicholas Wagner
Lehigh University Masters in Financial Engineering
Sponsor: Mikhail Samonov
Report Date: May 11, 2026

Abstract

FedGPT was an applied research and engineering project to convert Federal Reserve communications into structured, machine-readable policy sentiment variables. The project combined public-source acquisition of Federal Open Market Committee materials, large-language-model batch scoring, cloud execution, time-series storage, and a SolidJS/ECharts dashboard intended to support explanatory analysis of static versus dynamic portfolio construction. My primary contribution was to build the FOMC release scraping and scoring pipeline and then to connect the resulting scored communication series to the visualization layer. The project produced a useful data product and exposed the practical difficulty of converting sparse, irregular central-bank text events into a robust forecasting signal.

Executive Summary

FedGPT addressed a question that is both financially relevant and technically difficult: can a machine-learning pipeline transform Federal Reserve communications into timely, interpretable signals that improve portfolio understanding or portfolio decisions? The sponsor asked the team to investigate whether language in Federal Reserve releases, especially FOMC materials, could be scored in a systematic way and then used in research workflows. The hoped-for outcome was not simply a dashboard. The deeper objective was to determine whether central-bank language could become a dynamic explanatory variable for portfolios exposed to rates, inflation, growth, credit, and interest-rate duration regimes.

My contribution centered on the ingestion and scoring layer. I built a pipeline that scraped FOMC releases, transformed the scraped materials into batch requests for OpenAI models, processed those requests through the OpenAI Batch API, and emitted structured score series such as short-rate sentiment, long-rate sentiment, inflation sentiment, employment sentiment, credit conditions, economic growth, surprise, and policy phase. The downstream project export contains a broad set of FOMC score series, with many categories represented by direct condition, outlook, percent, adjusted-percent, and probability variants.

The project succeeded at producing a technically coherent prototype. It demonstrated that Federal Reserve documents can be acquired, normalized, scored, stored, and visualized as time series. However, it did not mature into a reliable predictive model. The main analytical obstacle was the mismatch between the communication-event frequency and the target financial time series. FOMC releases are sparse, irregular, and strongly regime-dependent. The team also encountered serial correlation problems when trying to map language scores into predictive variables. Those issues reduced the usefulness of a naive direct-score-to-return pipeline.

The most important postmortem finding came after the project ended, when the sponsor pointed to the related FedLock project. FedLock demonstrated three lessons: first, a large Federal Reserve speech corpus already existed and could have been used instead of spending so much effort on scraping; second, pairwise tournament scoring with TrueSkill is a compelling alternative to direct numerical scoring; and third, including thousands of speeches creates a more continuous communication signal than a release-only corpus. Those lessons do not invalidate FedGPT, but they do change the recommended next step. A continuation should start with historical data, use already available communication corpora, run locally on Lehigh MFE hardware where possible, and only then consider a production cloud pipeline.

Report requirement	Where addressed in this report
Description of project essence	Sections 1 and 2
Team organization and terminology	Section 3 and Appendix A
Challenge question and sponsor task	Section 2
Steps taken and personal contribution	Sections 4, 5, 6, and 7
Technical specifications	Sections 4 through 7 and Tables 2-5
Research methods and findings	Sections 8 and 9
Charts, graphs, tables, and results	Figures 1-4 and Tables 1-6
Sponsor-facing postmortem and next steps	Sections 10 and 11

Table of Contents

1. Project Description and Essence

FedGPT was designed as a central-bank communication intelligence system. Its essence was the conversion of unstructured monetary-policy text into structured variables that could be plotted, compared, joined to market data, and discussed in a portfolio context. In the language of data engineering, the project was a document-ingestion and semantic-feature-generation system. In the language of quantitative finance, it was an attempt to create event-time factors from central-bank language. In the language of portfolio management, it was a research platform for understanding whether policy communication shifts explain portfolio performance better than static macro labels.

The project focused on Federal Reserve communications because the Federal Reserve affects short rates, long rates, inflation expectations, credit conditions, equity valuation, and risk appetite. A single FOMC statement or minutes release can change the market interpretation of the future policy path. Textual analysis is attractive because it may capture forward-looking information before it appears in lagged macroeconomic data. The technical challenge is that such communication is not a regularly sampled sensor feed. It is a sparse, institutional, highly contextual text process. The same phrase can mean different things in different inflation regimes, rate regimes, and leadership regimes.

The prototype treated the Federal Reserve document stream as a source of event-level observations. A release was scraped, parsed, and sent to an LLM prompt that produced one or more structured scores. Those scores became fields in the FOMC_scores namespace. For example, a release could be mapped into short-rate sentiment, long-rate sentiment, phase, surprise, inflation condition, employment outlook, credit-condition outlook, or economic-growth condition. The resulting series could then be rendered in the dashboard and compared against future return or portfolio-regime targets.

The project therefore had two simultaneous goals. The first was an engineering goal: build a reliable pipeline that could acquire, process, score, store, and visualize a corpus of Federal Reserve documents. The second was a research goal: determine whether the generated scores were useful for prediction or explanation. The first goal was substantially achieved as a prototype. The second goal produced negative or inconclusive evidence, which is still valuable because it revealed where the research design was underpowered.

System layer	Purpose	Primary output
Source acquisition	Download or scrape Federal Reserve documents and metadata	Raw HTML/PDF/text artifacts and normalized document records
Batch scoring	Convert documents into LLM-ready JSONL tasks and process asynchronously	Structured score objects keyed by document and prompt version
Time-series materialization	Transform document-level scores into named series	FOMC_scores.* JSON series and metadata
Visualization	Render scored series and user-configurable charts	Dashboard widgets, selectable data sources, date filters, and chart exports
Research analysis	Evaluate explanatory or predictive relationships	Diagnostics, findings, limitations, and model recommendations

2. Challenge Question and Sponsor Task

The sponsor-facing challenge question was: can Federal Reserve communication be converted into a robust, interpretable, and potentially predictive signal for portfolio analysis? The sponsor wanted the team to undertake a practical research workflow, not only a literature review. The team needed to acquire the data, generate signals, test whether the signals were useful, and present the results in a way that could support future research or sponsor-facing analysis.

The project initially leaned toward a predictive interpretation. If a model could score a policy release as hawkish, dovish, surprising, inflation-focused, growth-focused, or credit-tightening, then those scores might become inputs to a dynamic portfolio model. The desired downstream model would respond to communication shocks with an impulse or transfer function: a policy-text event would create a decaying effect on expected returns, volatility, duration exposure, or factor allocation. In systems terminology, the release would be an impulse input, the market

or portfolio response would be the output, and the transfer function would encode how strongly and how long the communication shock persisted.

The empirical reality was more difficult. The team found that direct use of the scores for prediction was weakened by two statistical problems. First, release data are sparse and irregular. The number of observations is much smaller than daily market data, and event gaps vary. Second, the data are serially correlated and regime-dependent. A hawkish score in a zero-rate environment, in a tightening cycle, and near the end of a hiking cycle can have different implications. A naive linear model using the raw score as a predictor can therefore confuse regime persistence with predictive content.

The sponsor then reframed the likely value of the project. Even if the scores were not immediately tradable, they could still be useful for explanations, portfolio attribution, and comparison between static and dynamic portfolios. A static portfolio keeps broad exposures fixed or changes them slowly. A dynamic portfolio changes weights as the macro and policy regime changes. FedGPT scores could help annotate why a dynamic strategy changed exposures and whether the language of the Federal Reserve supported that decision.

3. Team Organization and Definitions

The team was organized around a pipeline-plus-research structure. One workstream was responsible for acquiring and scoring Federal Reserve text. A second workstream attempted to use the resulting scored data for prediction and portfolio analysis. A third workstream focused on presentation: the dashboard, visual output, and the user interface for exploring the scores. My role overlapped the first and third workstreams. I built the scraping and scoring pipeline and then spent substantial time building a dashboard to make the score series explorable.

This organization had a practical advantage: the data product and visualization could progress while the modeling work was uncertain. It also created a management risk. I became fixated on the dashboard and spent a disproportionate amount of time improving the visualization layer while the core modeling question remained unresolved. In hindsight, the team should have frozen a minimal visualization layer earlier and devoted more time to testing whether the scores had enough statistical power to support the sponsor's intended use.

Key terms used throughout the project are defined below. These definitions matter because a central-bank NLP project can fail simply from mixing up document acquisition, semantic scoring, forecast modeling, and portfolio implementation. The FedGPT prototype proved that the first two were feasible. It did not prove that the third and fourth were reliable without additional methodological corrections.

Term or program	Definition in this project
FOMC	Federal Open Market Committee, the Federal Reserve body whose statements, minutes, transcripts, projections, and related materials communicate monetary policy.
FedGPT	The project name for the LLM-assisted Federal Reserve communication scoring and dashboard pipeline.
OpenAI Batch API	An asynchronous batch-processing API used to submit large JSONL request files for lower-cost, non-immediate scoring jobs.
Direct scoring	Prompting a model to assign explicit numeric or categorical scores to one document at a time.
Pairwise scoring	Comparing two documents at a time and using ranking or rating machinery to infer relative hawkishness or dovishness.
TrueSkill	A Bayesian rating algorithm originally developed for ranking players or teams, useful here as an analogy for ranking speeches from pairwise comparisons.
Static portfolio	A portfolio with fixed or slowly changing allocations that does not explicitly react to each communication event.
Dynamic portfolio	A portfolio that updates exposures when new signals, such as policy-communication scores, arrive.
Impulse/transfer function	A time-series model structure that maps an event shock into a lagged market or portfolio response.

4. System Architecture and Technical Specifications

The architecture can be summarized as a document-to-signal pipeline. Federal Reserve source material is acquired from official publication surfaces, normalized into a stable representation, sent through an LLM scoring process, and materialized as time-series data. The visualization layer then treats each score as a normal time series. This separation is important. Acquisition should be reproducible and auditable; model scoring should be versioned by prompt and model; storage should preserve raw artifacts; and analysis should never depend on a hidden browser state or an unreproducible manual copy/paste workflow.

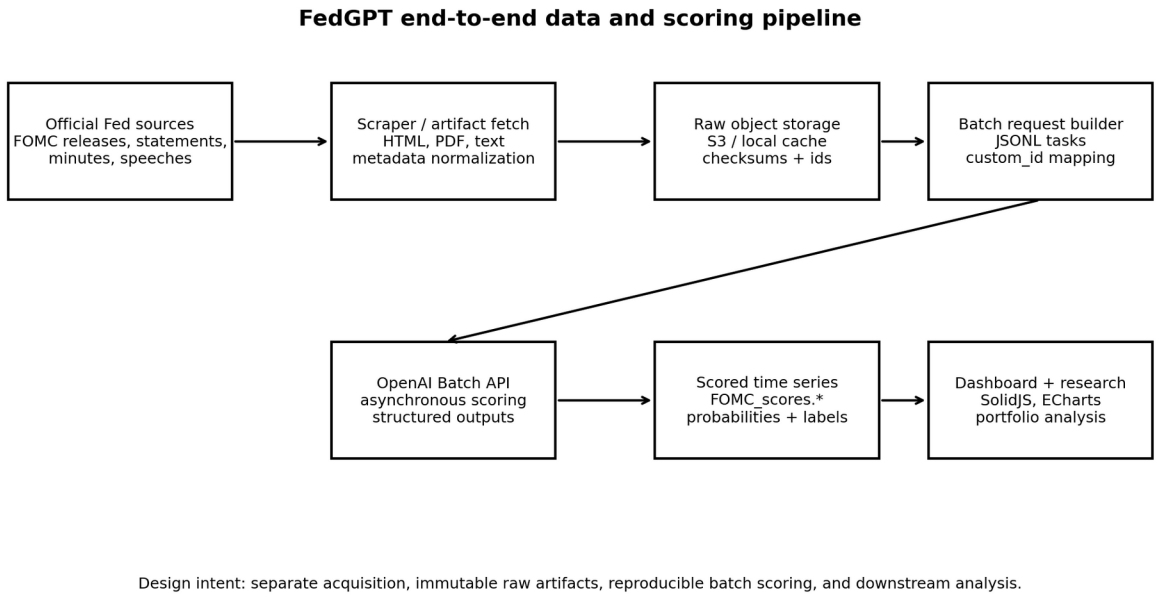

Figure 1. FedGPT pipeline architecture: acquisition, raw storage, batch scoring, time-series materialization, and dashboard research loop.

The initial cloud-oriented design used AWS because it offered a straightforward way to separate storage, compute, and long-running batch jobs. Amazon S3 was suitable for raw artifacts and structured outputs because it is object storage; AWS Lambda was suitable for event-driven glue code and smaller orchestration tasks; and AWS Batch was conceptually appropriate for larger jobs that process many documents or many scoring requests (Amazon Web Services, 2026a, 2026b, 2026c). This design was technically reasonable, but cost control became a constraint because repeated scoring and cloud execution accumulated expenses before the modeling value was clear.

The dashboard implementation used a modern SolidStart/SolidJS stack. The project export shows dependencies such as SolidJS, SolidStart, Vinxi, Muuri, ECharts, echarts-solid, marked, unstorage, and related UI/runtime packages. SolidJS supplied fine-grained reactivity; Muuri handled draggable grid layouts; ECharts handled high-density time-series visualization; unstorage provided a development-mode abstraction over local file-backed storage; Vinxi/SolidStart enabled server-side route and RPC-style execution. This stack was strong for an exploratory dashboard, but it also increased the surface area of the project.

Component	Technical specification	Project rationale
Frontend framework	SolidStart / SolidJS	Fine-grained reactivity and SSR-aware application structure.
Charting	ECharts 5.x via echarts-solid	Interactive line/scatter charts for event and score series.
Layout	Muuri grid	Draggable/resizable dashboard widgets.
Local development storage	unstorage with file-system drivers	Local-first development without requiring production database access.
Production storage concept	S3-style object storage plus API-backed retrieval	Durable raw artifacts, processed outputs, and score files.
Batch processing	OpenAI Batch API JSONL input	Asynchronous large-scale scoring with lower marginal cost than synchronous calls.
Cloud compute concept	AWS Lambda and AWS Batch	Event-driven glue and scalable batch execution.

The later py-fed-m direction provides a cleaner architecture for the same domain. It explicitly separates Board/FOMC documents, statistical release snapshots, and officially linked time series, aligning with the Federal Reserve's own separation between FOMC historical materials, statistical releases, and data-download surfaces (Federal Reserve Board, 2022, 2026). It also defines normalized runtime authorities such as documents.parquet, document_artifacts.parquet, dataset_snapshots.parquet, release_artifacts.parquet, and series_points.parquet. This is closer to what FedGPT needed from the beginning: a local-first, reproducible acquisition engine before semantic analysis.

5. My Pipeline: Scraping, Normalization, and Batch Scoring

My main contribution was the acquisition and scoring pipeline. I treated each Federal Reserve release as an input object with a source URL, publication date, document type, raw content, parsed text, and score outputs. The acquisition step had to be careful because Federal Reserve materials span decades, formats, and page structures. A robust pipeline must handle HTML pages, linked PDFs, changing archive structures, missing text, duplicate links, and publication dates that do not always align perfectly with market-observation dates.

The scoring pipeline transformed documents into JSONL tasks. This matched the OpenAI Batch API operating pattern, in which each request line can carry a custom identifier and the batch job completes asynchronously rather than as an immediate interactive response (OpenAI, 2026). Each row carried a custom identifier that could be joined back to the source document. A typical request contained the model name, prompt version, document content or extracted excerpt, scoring schema, and desired structured response. After the batch completed, the output file was parsed and joined back to the original document manifest. That join step is critical. Without a stable custom_id, model outputs become impossible to audit at scale.

```
Example scoring task shape (simplified)
{
  "custom_id": "fomc_2024-12-18_statement_v3",
  "method": "POST",
  "url": "/v1/responses",
  "body": {
    "model": "<batch_scoring_model>",
    "input": [
      {"role": "system", "content": "Score Federal Reserve communication text using the
schema."},
      {"role": "user", "content": "<normalized FOMC release text>"}
    ],
    "text": {"format": {"type": "json_schema", "name": "fed_policy_scores"}}
  }
}
```

The major engineering principle was provenance. Every model score should be traceable to a source document, prompt version, model version, timestamp, and parser version. This was especially important because a direct score is not a physical measurement. It is an interpretation generated by a probabilistic model. If the prompt changes, the score may change. If the model changes, the score may change. Therefore the score must be treated as a versioned derived artifact, not as immutable ground truth.

Data field	Type	Purpose
document_id	string	Stable identifier for the source release or communication object.
source_url	string	Original URL used for provenance and re-fetching.
publication_date	date	Event-time anchor for joining to markets and macro data.
document_type	enum/string	Statement, minutes, speech, testimony, press release, transcript, or other category.
raw_artifact_path	string	Path to the raw fetched object.
parsed_text_path	string nullable	Path to extracted text; missing text is explicit state.
prompt_version	string	Version of scoring instructions and schema.
model_version	string	Model used for scoring.
score_name	string	Feature name such as short_rate_sentiment or inflation_outlook.
score_value	float/string	Model-emitted value or categorical label.
score_probability	float nullable	Optional confidence/probability companion field.

The resulting data product included many named score series. The exported project structure contains multiple FOMC_scores JSON files, including consumer demand, credit condition, economic growth, employment, fed action, inflation, long-rate sentiment, short-rate sentiment, phase, surprise, and standard-deviation style outputs. Several families were represented by condition, outlook, percent, adjusted-percent, and probability variants, allowing the dashboard to show both the model's central score and its uncertainty-like companion values.

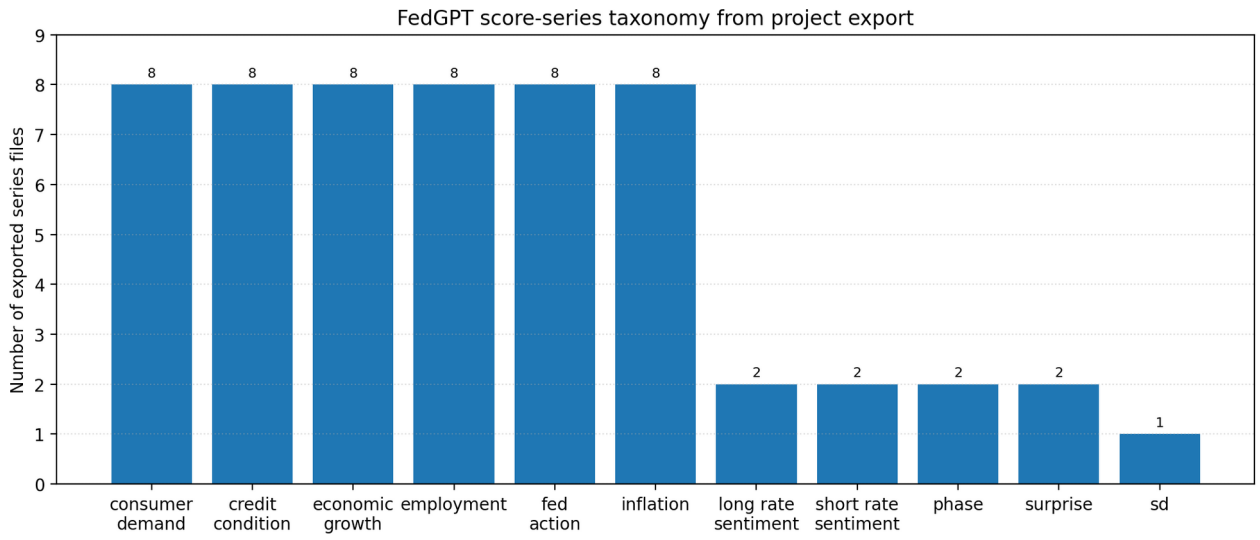

Figure 2. Count of exported score series by family, based on the project export.

6. LLM Scoring Methodology and Schema Design

The original FedGPT scoring approach was direct scoring. A prompt asked the model to inspect a release and assign values to several economically meaningful dimensions. Direct scoring is attractive because it is simple, cheap relative to human annotation, and easy to explain. A document receives a score; the score can be plotted; and the score can be joined to future returns or portfolio outcomes.

The direct-scoring approach has a calibration problem. A model may interpret the 1-to-5 scale differently across time, across document types, or across prompts. A score of 4.0 in one model version may not be equivalent to a score of 4.0 in another model version. Even within one model, the distribution can become discretized around common anchor values. This became important when reviewing FedLock because FedLock explicitly argued that direct scoring can collapse into coarse camps while pairwise ranking produces a smoother distribution.

A technically stronger design would use direct scoring for interpretability and pairwise scoring for calibration. Direct scores can answer, “What did the model think this document was about?” Pairwise scores can answer, “Was this document more hawkish or dovish than another document from a nearby era?” For central-bank communication, pairwise scoring is especially attractive because it reduces the need for the model to maintain a universal numeric scale across decades. Instead, the model only compares two texts at a time, and an external rating algorithm infers the global ordering.

Dimension	Direct FedGPT scoring	Pairwise / TrueSkill-style scoring
Unit of judgment	One document at a time	Two documents compared at a time
Output	Absolute score or label	Relative win/loss/preference result
Calibration risk	High: model must maintain a global numeric scale	Lower: model only makes local comparisons
Interpretability	Very direct; score fields map to economic concepts	Requires converting pairwise ratings back to interpretable levels
Computational cost	Approximately linear in number of documents times score dimensions	More expensive because many comparisons are required
Best use in continuation	Feature labels, explanations, and dashboard annotations	Era-adjusted hawkishness/dovishness index and smoother communication factor

A hybrid scoring schema would preserve the most useful parts of FedGPT while learning from FedLock. The direct scoring schema should emit structured fields such as inflation_condition, inflation_outlook, employment_condition, employment_outlook, fed_action_sentiment, short_rate_sentiment, long_rate_sentiment, credit_condition, economic_growth, surprise, and phase. The pairwise schema should evaluate relative hawkishness or dovishness between two anonymized documents and then update a rating distribution. The resulting time series could include both level and uncertainty: a rating mean and a rating standard deviation.

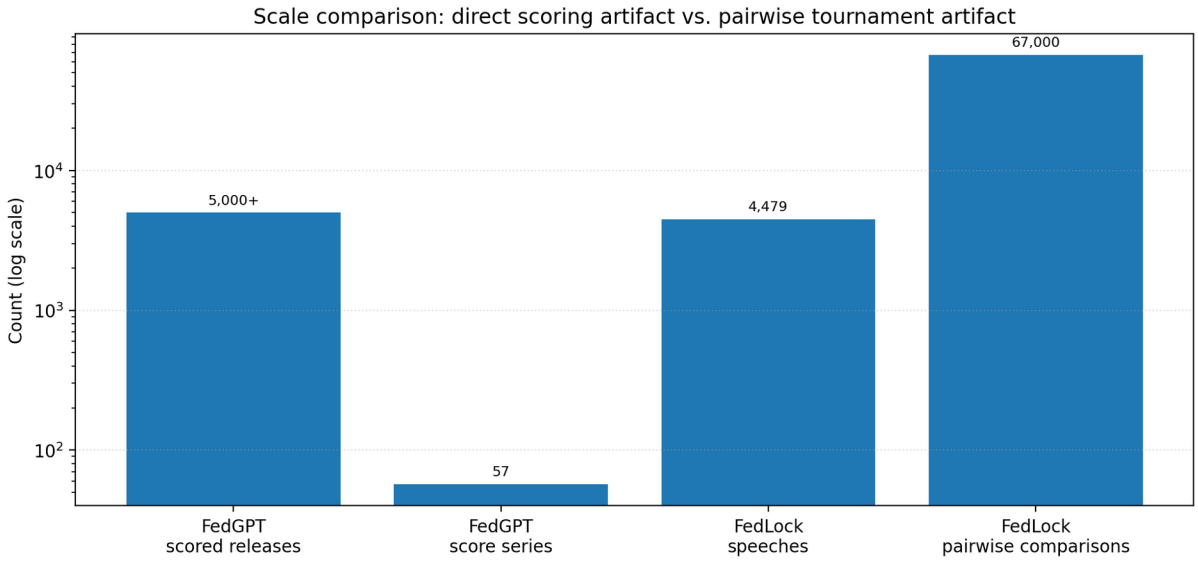

Figure 3. Scale comparison between the FedGPT direct-score artifact and the FedLock pairwise tournament artifact. Counts are shown on a log scale because comparisons are much larger than document counts.

7. Dashboard and Visualization Layer

The dashboard was intended to make the score series usable. A score pipeline without visualization is difficult to review because model outputs are high-dimensional, event-time based, and often unintuitive. The SolidJS dashboard exposed score series through configurable chart widgets. Users could select data sources, choose date ranges, plot line or scatter series, inspect metadata, change chart options, and organize widgets in a grid.

The dashboard code evolved into a relatively sophisticated widget system. A right sidebar staged the selected widget in a temporary clone, allowed edits, and then pushed changes back into the dashboard model. Data cards exposed available tables and columns. Series cards allowed a user to select a source, change start and end dates, choose chart type, and set visual options. The design separated available data sources from the current widget series, which was necessary because a research dashboard needs to add, remove, and rewire series without corrupting the underlying layout.

This dashboard work was technically valuable but strategically risky. I spent a large amount of time debugging layout behavior, chart rendering, series selection, date filtering, metadata tooltips, and right-sidebar interactions. Those tasks improved the prototype's polish, but they did not directly answer the statistical question of whether the FedGPT score series improved prediction. In hindsight, the correct project management decision would have been to lock the dashboard to a minimal design earlier and define a formal modeling test matrix before continuing UI work.

Dashboard feature	Implementation detail	Research value
Selectable data sources	DataList and DataCard maintain table.column selections	Allows researchers to decide which score families are active.
Series configuration	SeriesList and SeriesCard map selected sources into ECharts series	Enables quick comparison across sentiment dimensions.
Date filtering	Series cards maintain start/end dates per source	Supports regime-specific analysis.
Metadata tooltips	DataCard loads series metadata on hover	Helps interpret axes, dates, descriptions, and debug fields.
Widget staging	RightSidebar clones selected widget before applying changes	Prevents accidental layout/data corruption during editing.
Chart export concept	ECharts option and screenshot utilities in project direction	Supports sponsor review, reproducibility, and presentation artifacts.

8. Research Methods

The team's research methods combined data engineering, LLM-based text annotation, exploratory time-series analysis, and portfolio interpretation. The first research method was corpus construction: identify relevant Federal Reserve communication materials, acquire them, extract usable text, and preserve metadata. The second method was prompt-driven semantic annotation: define scoring dimensions and use a language model to transform text into structured variables. The third method was visual and statistical inspection of the resulting series. The fourth method was downstream modeling: attempt to connect scores to future market or portfolio outcomes.

For a rigorous continuation, the modeling workflow should be formalized before new scoring begins. The correct design is not to score all available text and then search for relationships. The correct design is to predefine target variables, horizons, alignment rules, transformations, and evaluation metrics. For example, one could define a target as the one-day, five-day, and twenty-day change in Treasury yields after each communication event; or the relative return between a static balanced portfolio and a dynamic duration-adjusted portfolio. The event-time score must then be mapped to the target without look-ahead bias.

Several transformations are necessary. Irregular event series need event-time alignment and potentially interpolation or state-space smoothing. Scores should be expressed as changes, surprises, or deviations from an era-adjusted baseline rather than raw levels. Regime variables should be included because monetary-policy language has different market meaning in tightening, easing, zero-lower-bound, and inflation-shock regimes. Serial correlation should be modeled explicitly using lagged score terms or state-space models rather than treated as independent observations.

Research question	Candidate method	Risk or correction
Does policy language predict rate moves?	Event study around FOMC dates and Treasury yield changes	Must handle overlapping windows and confounding macro releases.
Does language explain portfolio attribution?	Static-versus-dynamic portfolio return decomposition	Must distinguish explanation from causal prediction.
Are scores stable across prompt/model versions?	Prompt/version robustness testing	Direct scoring can drift; needs calibration checks.
Can sparse release scores become a continuous factor?	State-space smoothing or Gaussian kernel over event dates	Smoothing may introduce artificial persistence.
Do speeches improve signal density?	Add speech corpus and compare event frequency	Speaker identity and era effects must be controlled.
Can pairwise scoring improve calibration?	Anonymized pairwise comparisons plus TrueSkill	Higher compute cost and comparison sampling design required.

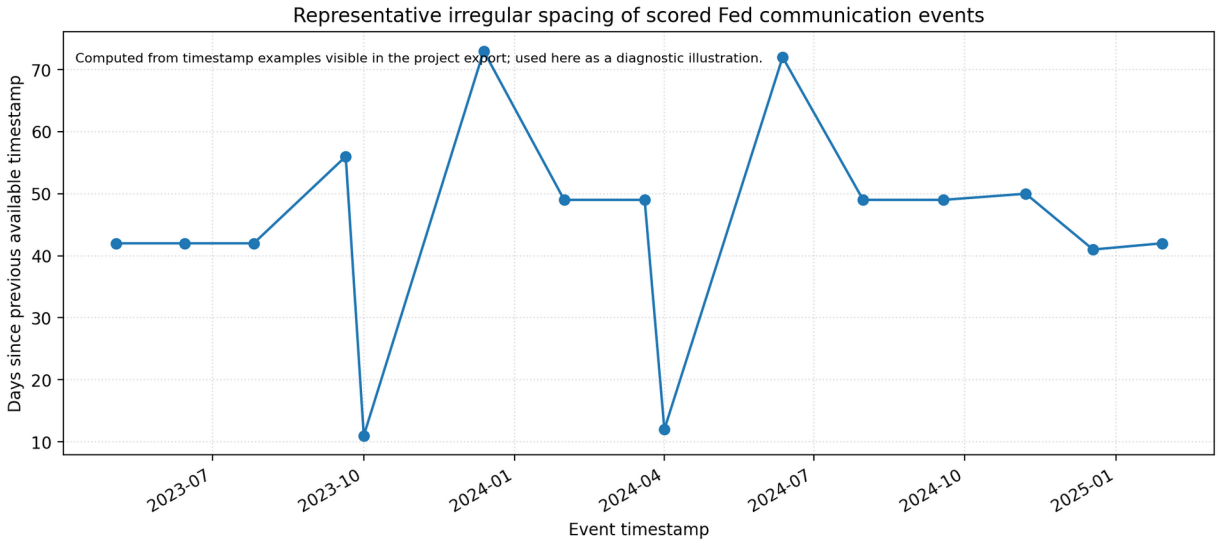

Figure 4. Representative event-spacing diagnostic from project-export timestamps, illustrating why direct release-only scoring creates irregular samples.

9. Results and High-Level Findings

The project produced several concrete outputs. First, it demonstrated an end-to-end scrape-score-store-visualize workflow. Second, it generated a large scored communication dataset from FOMC releases. Third, it created a score namespace that could be treated as time-series data in the dashboard. Fourth, it identified serious limitations in direct predictive use. These are meaningful results even though the modeling result was not a decisive success.

The most visible positive result was the creation of many score series. The project export shows a broad FOMC_scores namespace. Each score family was designed to represent a different macro-policy dimension. This is useful because it allows researchers to ask more granular questions than "was the Fed hawkish?" For example, inflation language and employment language can diverge; rate sentiment and credit-condition sentiment can diverge; phase and surprise can reveal whether the release was routine or regime-changing.

The most important negative result was that the data were difficult to use for prediction without further corrections. The team encountered sparsity, irregularity, and serial correlation. In event-driven finance, these are not minor issues. A model trained on a few hundred or a few thousand irregularly spaced policy events has far fewer independent observations than a daily return model. If the event series is also autocorrelated, the effective sample size is smaller still. The danger is that a model appears to fit historical policy regimes but fails when the regime changes.

Result category	Observed result	Sponsor implication
Engineering feasibility	Pipeline could scrape, score, store, and visualize Federal Reserve communications	The concept is technically feasible.
Data product	Multiple FOMC score families were materialized as time-series-style JSON outputs	The project can support explanatory dashboards and further research.
Prediction quality	Downstream users struggled to make scores useful for forecasting	Predictive claims should be avoided until corrected modeling is performed.
Cost and operations	Cloud execution and repeated processing became costly	Historical-first local processing is preferable before production cloud deployment.
Dashboard value	The UI helped inspect the data but consumed excessive development time	Future scope should cap UI effort until research validity is established.
Postmortem insight	FedLock showed that broader corpora and pairwise ranking may solve key weaknesses	Continuation should adopt existing datasets and consider TrueSkill-style ranking.

The correct sponsor interpretation is therefore cautious but not pessimistic. FedGPT did not prove that LLM-scored Fed releases are a standalone trading signal. It did prove that the data product is buildable and that central-bank communication can be transformed into a structured research object. Its best near-term use is explanation, regime annotation, and portfolio attribution. Its best medium-term research path is to broaden the corpus, improve calibration, and test dynamic allocation rules on pre-specified historical hypotheses.

10. Postmortem and FedLock Comparison

After the project ended, Mikhail Samonov brought FedLock to my attention. FedLock is a similar project that scores Federal Reserve speeches and presents an era-adjusted hawkishness/dovishness index (Nathan, 2026). Reviewing it clarified several failures in our project design. The most important failure was that we overinvested in scraping when relevant historical Federal Reserve communication corpora were already available. That does not mean scraping was useless; building the scraper taught us the structure of the source domain. But for a time-limited research project, using existing datasets would have created more time for modeling and validation.

The second failure was methodological. FedGPT used direct scoring, while FedLock used a pairwise tournament approach with Microsoft TrueSkill, a Bayesian rating framework originally developed for ranking competitors while tracking uncertainty (Herbrich, Minka, & Graepel, 2007). Pairwise tournament scoring is compelling because the model does not need to define an absolute hawkishness scale. It only needs to decide which of two anonymized speeches is more hawkish or dovish. TrueSkill then turns many comparisons into ratings with uncertainty. This is a better match for historical central-bank language because the meanings of terms such as "firming," "restrictive," "patient," or "data dependent" shift across regimes.

The third failure was corpus density. FedGPT emphasized FOMC releases, which are important but sparse. FedLock reported a corpus of 4,479 speeches and approximately 67,000 pairwise comparisons, producing many more sample points and allowing a smoother communication index (Nathan, 2026). Existing historical datasets, including Federal Reserve communication data assembled for research use, also show that the project could have begun from a richer corpus rather than from live scraping alone (Acosta, n.d.). A release-only corpus creates a staircase-like event series with large gaps. A speech corpus still has selection bias and speaker effects, but it gives the model a denser measurement of policy communication tone.

Design choice	FedGPT implementation	FedLock lesson	Recommended correction
Corpus	FOMC releases and related materials scraped by project pipeline	Large speech corpus already available and broader than release-only data	Start with existing historical corpora, then add official scraping for incremental freshness.
Scoring	Direct numeric/category scoring per document	Pairwise tournament plus TrueSkill produces smoother rating distribution	Use hybrid direct + pairwise scoring.
Continuity	Sparse, irregular event series	More speeches create denser sample path	Include speeches, testimony, press conferences, and minutes paragraphs where appropriate.
Computation	Cloud-oriented AWS pipeline and repeated batch operations	Low-cost historical batch is possible with controlled design	Run historical experiments locally first; use cloud only when validated.
Research emphasis	Dashboard polish consumed too much effort	Methodological design was central to FedLock value	Freeze UI early and prioritize validation design.

This comparison changes the continuation plan. The next version should not begin by scraping the live web. It should begin by reproducing a historical corpus, defining a calibration method, and testing whether the communication index explains known policy cycles. Only after the historical backtest is convincing should the project add live updates and production dashboard features.

11. Recommended Continuation Plan

A continuation should be staged in three phases. Phase 1 should be historical and local. It should use already available Federal Reserve communication datasets, reproduce the FedGPT scoring schema, and build a pairwise TrueSkill-style calibration layer. This phase should run on local or Lehigh MFE hardware rather than AWS. The purpose is to prove methodological value before paying for operational scale.

Phase 2 should connect the historical score index to research targets. Candidate targets include Treasury yield changes, curve changes, equity factor returns, bond-equity correlation regimes, inflation-sensitive assets, and

static-versus-dynamic portfolio attribution. The key is to evaluate fixed hypotheses, not search the dashboard for visually appealing relationships. The report output should include out-of-sample windows, robustness checks across model versions, and sensitivity to event-window assumptions.

Phase 3 should introduce production freshness. Only after the historical system works should the project scrape official sites on a schedule, process new speeches or releases, and publish updated dashboard values. At that point, AWS or another cloud provider may be appropriate, but the pipeline should be cost-governed. Each run should know how many documents are new, how many scores are recomputed, what the expected token and compute cost is, and whether the sponsor actually needs the update frequency.

1. Rebuild the corpus from existing historical sources before scraping new documents.
2. Separate raw acquisition, text extraction, direct scoring, pairwise scoring, and portfolio modeling into versioned modules.
3. Use anonymized pairwise comparisons and a TrueSkill-style rating layer for a primary hawkishness index.
4. Retain direct FedGPT scores as interpretable subdimensions for explanation and dashboard annotation.
5. Define event-time joins and target horizons before running model selection.
6. Treat portfolio work first as attribution and explanation, then as forecasting only if validation supports it.
7. Move production freshness to Lehigh-owned or sponsor-approved hardware before returning to AWS scale.
8. Explore a future AI-agent architecture in which separate agents handle data acquisition, scoring, validation, portfolio interpretation, and sponsor-report generation.

The agentic continuation is especially promising. One agent could monitor data quality and source changes. Another could run scoring jobs. Another could test score stability across prompts and models. Another could perform portfolio attribution. A final “reflection” agent could write a postmortem after each experiment, recording what worked, what failed, and what should be tried next. This would preserve institutional memory and prevent the team from repeating failed paths.

12. Conclusion

FedGPT was a technically ambitious project that produced a functioning prototype and a useful postmortem. My primary contribution was the pipeline for scraping FOMC releases and scoring them through OpenAI batch processing. I also invested heavily in the dashboard, which improved the system’s presentation but consumed too much time relative to the unresolved modeling problem. The project demonstrated that Federal Reserve communication can be converted into structured score series and visualized as a research product. It did not demonstrate that those scores are immediately reliable as predictive trading signals.

The project’s main lesson is that central-bank communication analysis should be treated as a careful data-science problem before it is treated as a dashboard or trading system. The next version should begin with historical data, use existing corpora, add pairwise calibration, model sparsity and serial correlation explicitly, and evaluate pre-defined portfolio questions. If those results are promising, the project can return to live updating and cloud deployment. In that staged form, FedGPT remains a strong foundation for sponsor-facing research on monetary-policy language, dynamic portfolio explanation, and AI-assisted financial analysis.

Appendix A. Glossary of Technical Terms

Term	Definition
Artifact	A stored file produced or consumed by the pipeline, such as raw HTML, raw PDF, parsed text, JSONL batch input, or model output.
Batch JSONL	A newline-delimited JSON file in which each row represents one model request for asynchronous processing.
Calibration	The process of making scores comparable across time, prompts, models, and document types.
Event time	Analysis time measured relative to events, such as one day after a release, rather than regular calendar intervals.
Look-ahead bias	A modeling error where future information is accidentally used to make a historical decision.
Model drift	A change in model behavior caused by model version, prompt version, context, or distribution shift.
Provenance	Metadata that records where a datum came from and how it was produced.
Serial correlation	Dependence between observations over time; a major issue when treating policy language scores as independent events.
State-space model	A model that estimates a latent evolving state from noisy observations, useful for irregular policy-sentiment signals.
Transfer function	A system model that maps input shocks into output responses across time lags.

References

Acosta, M. (n.d.). FOMC Communications Data. Federal Reserve Bank of Philadelphia. https://www.philadelphiafed.org/surveys-and-data/real-time-data-research/acosta-fomc-communications

Amazon Web Services. (2026). Amazon S3 User Guide. https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html

Amazon Web Services. (2026). AWS Batch. https://docs.aws.amazon.com/batch/latest/userguide/what-is-batch.html

Amazon Web Services. (2026). AWS Lambda Developer Guide. https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

Federal Reserve Board. (2022). Transcripts and other historical materials. https://www.federalreserve.gov/monetarypolicy/fomc_historical.htm

Federal Reserve Board. (2026). FOMC materials and related resources. https://www.federalreserve.gov/monetarypolicy/fomccalendars.htm

Hansen, S., McMahon, M., & Prat, A. (2018). Transparency and deliberation within the FOMC: A computational linguistics approach. Quarterly Journal of Economics, 133(2), 801-870. https://doi.org/10.1093/qje/qjx045

Herbrich, R., Minka, T., & Graepel, T. (2007). TrueSkill: A Bayesian skill rating system. Advances in Neural Information Processing Systems. Microsoft Research. https://www.microsoft.com/en-us/research/publication/trueskilltm-a-bayesian-skill-rating-system/

Liu, Y., et al. (2024). Aligning with human judgement: The role of pairwise preference in large language model evaluators. OpenReview. https://openreview.net/forum?id=9gdZI7c6yr

Nathan, J. (2026). FedLock: Ranking FOMC communication hawkishness with LLM pairwise comparison and TrueSkill. https://jnathan9.github.io/fedlock/

OpenAI. (2026). Batch API guide. https://platform.openai.com/docs/guides/batch