When Language Models Should Reason, Not Calculate: Rethinking Event-Aware Forecasting

Astrid Atle and David Perntoft

Department of Mathematical Statistics, Industrial Engineering and Management, Lund University

A practical look at where large language models actually add value to time series prediction, and where they don't.

Every year on the Monday after Black Friday, logistics operators face the same problem. Their forecasting systems, trained on years of clean weekly cycles, suddenly need to predict what happens when four overlapping events collide: a Singles Day pulse, a Black Week build-up, the Black Friday spike itself, and a delivery after-wave into early December. A purely numerical model, even a state-of-the-art foundation model trained on billions of time points, will recover the weekly cycle perfectly and then completely miss the spike.

This is the defining limitation of nearly every quantitative forecasting system in production today. They operate on numbers alone, in what is best described as a *multimodal vacuum*. The signal that would let them anticipate Black Friday is sitting in the operations team's planning documents, in news articles, in the marketing calendar. It is just not in a form the model can consume. Large language models are the obvious candidate to bridge this gap, but the naive solution, where the LLM receives a sequence of numbers along with some text and is asked to predict the future, runs into a different problem: LLMs are unreliable processors of raw numerical sequences. They tokenize numbers in ways that destroy ordinal relationships. They hallucinate trends. They produce confident-sounding forecasts that are subtly wrong.

This master's thesis investigated a specific resolution. The premise is simple: **LLMs should never calculate. They should reason.** All numerical computation is delegated to validated statistical implementations such as SARIMA, state-space models, and STL decomposition. The LLM's contribution is limited to what it is actually good at: interpreting natural language, retrieving relevant historical analogues, and translating qualitative event descriptions into structured, quantitatively grounded adjustments to a statistical baseline.

The architecture

The system is a pipeline of specialised agents, each with a narrow responsibility (Figure 1). Statistical descriptors and domain context feed a Hypothesis Generator that proposes up to five candidate model specifications through a Tree-of-Thoughts reasoning step. Surviving hypotheses enter a Forecaster and Evaluator loop that iteratively refines parameters against held-out validation data. The best-scoring hypothesis becomes the statistical baseline. Finally, a Scenario Generator takes a future event description, matches it against historical analogues from both the target series' own history and an external knowledge bank of precedent cases, and produces three scenario specifications, namely optimistic, expected, and conservative, that adjust the baseline through deterministic multiplicative transforms.

The critical design constraint runs through every stage: the LLM never produces a number. It selects from a library of statistical models, ranks analogues, and picks the shape of a temporal envelope from a fixed list of forms. The actual magnitudes come from empirical quantiles of historical impacts, and the temporal application is deterministic. Every numerical output can be traced back to either a statistical procedure or a piece of measured evidence, never to an LLM's free-form generation.

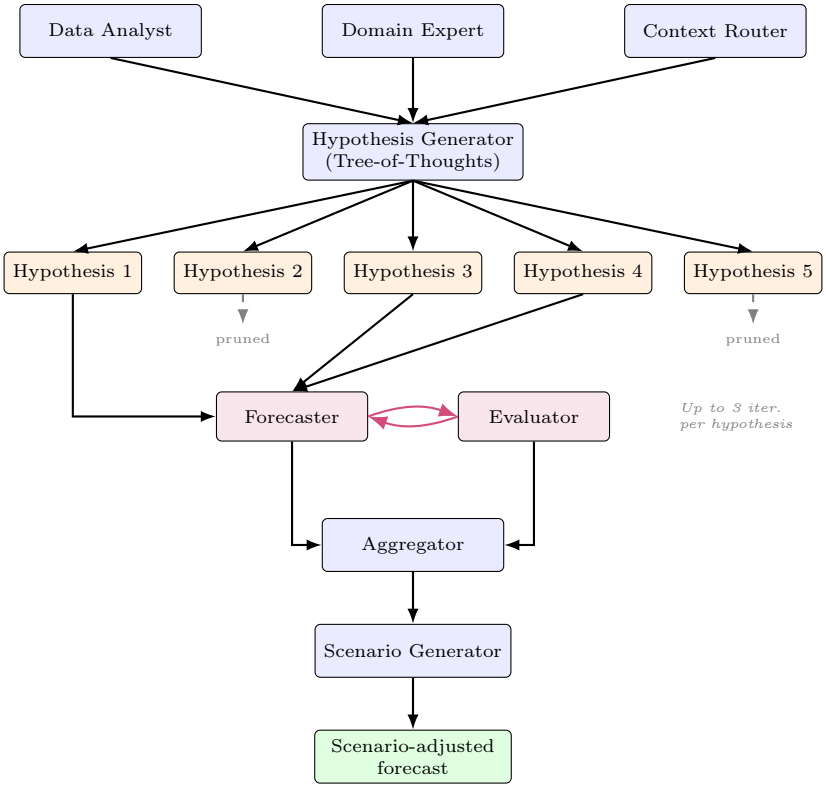

Figure 1: The agentic forecasting pipeline. Statistical descriptors and domain context feed the Hypothesis Generator, which branches into up to five candidate model specifications. Pruned hypotheses are dropped; survivors enter the Forecaster–Evaluator refinement loop. The Aggregator selects the best-performing hypothesis as the statistical baseline, which is then adjusted by the Scenario Generator using future event information.

When the system works, and when it doesn't

We evaluated the system on three simulated datasets with controlled data-generating processes: a primary care centre, a logistics hub, and a music streaming catalogue. Simulated data is a deliberate choice, since real-world series rarely come with ground truth for event effects, and we wanted to measure not just whether the forecast was accurate but whether the system reasoned about the right things. On the logistics case, the full pipeline reduced symmetric MAPE by 66 % relative to running the same statistical pipeline without any event reasoning. The health centre case showed a 59 % reduction. A purely numerical foundation model (Chronos-Bolt) produced MASE values 2.0 to 2.5 times higher on event-driven test windows, not because it is a poor model, but because it cannot anticipate level shifts driven by events it has never seen in the numerical signal.

The more informative result came from ablating the system. The music streaming case collapsed by 276 % relative to the full pipeline when both grounding sources were removed but the future event description was retained. With no historical analogue, neither own nor external, to anchor the magnitude of an album release, the language model had no way to translate "a new single will drop on June 21st" into a calibrated numerical adjustment. It produced a narrow interval that missed the actual release spike entirely. This collapse is, paradoxically, evidence that the system is behaving correctly. A worse-designed system would have invented confidence: produced a wide interval to nominally cover the actual outcome, or fabricated a magnitude from thin air. The system we built honestly signalled that it had no evidence to work with. In an operational setting, a system that knows when it does not know is far more valuable than one that hallucinates calibration it has not earned.

When this approach actually helps

LLM event reasoning is not a general-purpose improvement to forecasting accuracy. It produces measurable signal when three conditions hold simultaneously. First, events have to actually matter for the forecast horizon. On a calm Tuesday in March, the system adds no value over a well-tuned statistical baseline, and the overhead is wasted. Second, at least one grounding source must contain structurally analogous events: pure novelty defeats the system, and the LLM correctly refuses to invent a magnitude it cannot justify. Third, the qualitative description of the event must be informative enough to drive retrieval. "A marketing campaign" is too vague; "a 30 % price reduction on athletic footwear, three-day duration, social media driven" gives the retrieval system something to work with. When all three conditions hold, the gains are substantial. When any fails, the system degrades gracefully, but it does degrade.

This article summarises the master's thesis **Integrating Natural Language Events into Time Series Forecasting through Agentic LLM Orchestration** *by Astrid Atle and David Perntoft, Department of Mathematical Statistics and Industrial Engineering and Management at Lund University, conducted in collaboration with Predli. The full thesis is available on request.*