

Describe, Don't Predict: LLM Agents as Point-in-Time State Builders

Oct 9, 2026 · @Zach McCormick

The opportunity

Most information that moves prices is unstructured: filings, call transcripts, credit agreements, court dockets, regulatory text, news, job postings. Systematic investing has mostly used the thin structured slice of it (prices, reported fundamentals, a handful of vendor datasets). Turning the rest into reliable features took human analysts, and analysts don't scale.

LLM agents change that. They can read, cross-reference and compute at close to analyst quality across thousands of entities, continuously and consistently. That is potentially the largest expansion of the systematic feature space since alternative data.

The core problem: lookahead bias

A strategy you cannot backtest is a strategy you cannot trust, and LLMs break backtests. A model trained on text through its cutoff knows which deals closed, which drugs failed, which firms blew up and how markets moved. Asked in a 2021 backtest whether a merger will close, it may simply remember the answer. That memory looks exactly like skill, so the backtest overstates performance, and only live trading reveals by how much.

The obvious fixes fall short:

- **Instructions** ("answer as of March 2021"). Models don't reliably forget on request, and there is no way to verify that they did.
- **Masking names and dates.** It removes useful information while leaving amounts, industries and wording that often still identify the case.
- **Point-in-time models** trained only on data before each date. That would require a frontier-quality model for every date; models small enough to retrain lack the sophistication that made LLMs worth using.
- **Testing only after the cutoff.** It is clean, but it takes years to accumulate statistically meaningful history, and the model is replaced before it does.

Lookahead bias is therefore the central obstacle: without a credible answer to it, no LLM-driven strategy can be validated before risking capital. What follows is a design for working around it.

The thesis

Use LLM agents to measure the present, never to forecast the future. An LLM's weights were trained on the future and we cannot control that; we can control what its output is allowed to depend on.

The architecture has two stages:

1. **State model (agents).** At time t , agents turn everything knowable at t into a structured, quantitative state representation S_t . The correct S_t is fully determined by evidence dated at or before t .
2. **Predictive model (trained).** A separate model maps S_t to outcomes $y_{\{t+h\}}$. It is fit walk-forward, so we control exactly which future it has seen.

This works because **in a description task, future knowledge can only show up as error; in a prediction task, it shows up as skill**. When the right answer is fixed by the as-of evidence, an agent that knows how things ended gains nothing by using it. It can only deviate from the evidence, and that deviation is auditable. All forecasting moves into the layer whose training window we control.

Example: Silicon Valley Bank, February 2023

On February 24, 2023, SVB Financial Group filed its annual report for 2022. Fourteen days later, on March 10, the bank failed after a deposit run. Take a backtest date t of February 24: what should an LLM contribute to a position in SVB?

The wrong way: ask the model. "How likely is this bank to get into trouble?" Any model trained after March 2023 knows that SVB failed. The agent might read and cite 50 documents, but its final answer inevitably blends analysis with memory, and the backtest cannot tell them apart. The strategy looks great in the backtest and then underperforms live.

The right way: ask the agent for the bank's state at t . The agent works through the 10-K's securities, deposit, liquidity and derivatives disclosures and the prior three 10-Qs, and runs the same procedure on every listed US bank:

State field	Value at February 24, 2023	What the agent had to do
Equity after marking securities to market	\$15.2B unrecognized loss on held-to-maturity securities against \$11.8B of tangible common equity; about \$0.5B left after tax	Take fair value from the securities note, tax-effect the loss at an assumed 25%, and net it against equity less preferred stock, goodwill and intangibles. This number appears nowhere in the filing.
Deposits outside FDIC coverage	\$165.4B of \$173.1B, or 96%	Combine estimated uninsured US deposits (\$151.5B) with foreign deposits (\$13.9B), which are disclosed separately
Liquidity available in a run	About \$83B, half of the uninsured deposits	Add cash (\$13.8B) to unused capacity at the FHLB (\$25.9B), discount window (\$5.3B), repo lines (\$35.0B) and fed funds lines (\$3.2B), from the liquidity section
Deposit trend	Down 12.6% since March 2022, from \$198.1B; non-interest-bearing deposits down 37%	Line up four quarterly balance sheets; management attributes outflows to client cash burn
Interest-rate hedges	Designated swap notional cut from \$10.7B to \$0.55B during 2022	Compare the derivatives note with the prior year's
Balance-sheet mix	Securities at 55% of total assets	Same calculation for every bank, to rank SVB against peers

A separate model, trained walk-forward on the same state for every bank across past cycles, maps it to outcomes such as next-quarter deposit outflows or equity drawdowns. It learns how states like this one have tended to play out; the LLM never makes that call.

Why this holds up. A grader who knows SVB failed on March 10 would score every cell exactly as one frozen on February 24, because each is arithmetic on disclosed numbers. To leak its memory, the agent would have to misstate a disclosed figure, and the source check catches it. The work is still sophisticated: the inputs sit in five different parts of a 200-page filing, and the most important number has to be constructed.

One caution: we think to compute uninsured-deposit coverage partly because SVB happened. That is researcher lookahead, not model lookahead. The defense is a schema derived from standard banking theory (run risk, asset-liability mismatch), fixed in advance and applied to every bank.

Sources: [SVB Financial Group 10-K for 2022](#), with balance-sheet values cross-checked against its [SEC XBRL data](#). Tax rate and run-liquidity sum are our calculations.

What makes a task work

Define a task as the process and output schema for a single agent run. The best tasks demand a lot of intelligence while giving hindsight nowhere useful to go. The key distinction: **sophistication is not what leaks; dated knowledge is**. Timeless knowledge (what a springing covenant does, how ASC 606 works, what a Fed conditional means) is the point of using an LLM. Knowledge of what happened to *this* firm, deal or drug is the only thing that leaks.

A task belongs in the state model when it passes these tests:

- **Grader invariance.** A grader who knows what happened next would score the answer the same as one frozen at t .
- **High timeless, low dated demand.** The work needs legal, accounting or economic expertise, not recall of the entity's later history.
- **Mechanical verifiability.** Every value traces to a source dated at or before t , and every derived number can be recomputed.
- **Ingredients, not judgments.** Extract product lists, not "competitors"; stated conditionals, not "hawkishness"; guidance and its track record, not "credibility." The predictive model does the judging.
- **Universe-level procedures.** The agent's intelligence decides *how to measure*, applied identically to every entity, never *where to dig* for a particular one.

Where agents add the most

Single-document extraction is just the beginning of what is possible here. Agents pay off most when the state exists in no single document and must be assembled, checked or computed.

Archetype	What the agent does	Example state feature
Reconstruction	Builds an object no document contains	Full capital structure and recovery waterfall from every indenture and amendment
Implied expectations	Solves for what prices and consensus at t assume	Unit growth implied by the price vs. the firm's and peers' history
Triangulation	Checks one source against independent others	Management claims scored as corroborated, unverifiable or contradicted
Cross-entity consistency	Reconciles narratives across a value chain	Distributor says channel inventory is high; supplier says demand is robust
Absence detection	Compares disclosures to history and peers	A KPI that quietly stopped being reported
Event-time screening	Applies fixed exposure criteria across the universe when a shock hits	Firm-level exposure to a new tariff, with citations
Situation assembly	Reports where a process stands	Open approvals and statutory clocks for a pending deal; projected vote arithmetic

Each row describes a state. The predictive model learns how states turn into outcomes: when implied-vs-history gaps close, whose narrative is usually wrong, how exposed firms react.

What to avoid

The main danger is a task that is point-in-time in form but predictive in substance. Sophistication and predictiveness rise together, so this slide is easy to miss.

- **Outcome-defined judgments.** "Relevant," "material," "key risk," "surprising" and 1–10 scores are partly defined by what turned out to matter. Break them into fixed-schema sub-answers.
- **Environment leakage.** Live search, current Wikipedia pages, restated data and today's ticker lists all carry the future. Agents need a point-in-time sandbox: an as-of corpus, archived snapshots, point-in-time databases and a frozen clock.
- **Navigation leakage.** An agent with a hunch to dig into a firm that later blew up leaks hindsight without stating a single false fact. Selection by an agent is the same problem as selection by a model, only larger.
- **Analog selection.** Pre- t base rates are legitimate, but choosing the analogs is where hindsight hides. The agent writes similarity criteria; retrieval runs mechanically.
- **Outcome-correlated error.** Extraction that is more careful for firms that later became famous is a signal the predictive model will learn.
- **Researcher lookahead.** Designing the schema around what we know mattered in hindsight. This leak is in us, not the model.

How we know it holds

We cannot prove leakage is zero; realistic contamination effects need more events than most corpora contain. The case is structural, backed by audits:

1. **Design.** Tasks give hindsight no useful outlet, so leakage that does occur shows up as error against the evidence.
2. **Provenance.** Every output value traces to a source with an as-of timestamp at or before t , checked mechanically.
3. **Invariance.** Masked-vs-named runs, counterfactual document edits and supplied-future-note probes should change neither the answers nor the agent's path through the sources. Trajectories are logged and replayable, with a fresh context per entity and date.
4. **Post-cutoff holdout.** Documents published after the model's training cutoff are the only assumption-free test. State quality and the state-to-outcome relationship should hold there.

Conclusion

We cannot make an LLM forget the future, but we can avoid asking it questions the future would answer. That constraint shapes what we ask agents to build, not how capable they are allowed to be.

The payoff goes beyond a cleaner backtest. A state representation is reusable: the same deal state can feed a closing-probability model, a spread model and a risk model. It is inspectable: when a forecast misses, we can tell whether the state was wrong or the mapping was. And it survives model upgrades. A newer, more capable agent re-run on history produces a more accurate description of the same past, which is still checkable against dated evidence, rather than a better memory of what came next.