Agentic Backtest Evaluator

By Vibhu Gangina, William Seward, and Tanay Mehrish
Northeastern University Systematic Alpha
04/14/2026

Table of Contents

Motivation and Research Foundation

Backtests

Simulations of how a trading strategy would have performed using historical market data

Traditional backtesting tells you what happened (your strategy made 15% with Sharpe 1.2) but not **why or when** it's vulnerable/profitable. Traders **waste hours** manually observing data by time periods, cross-referencing macro conditions, and scouring for regime dependencies that could destroy a strategy. Furthermore, there's features that traders don't even notice after days of looking at data that can be indicative of a strategy's performance under a certain interest rate regime, or under certain market conditions.

Not only do we want to address this time-constraining issue, but we also want to look at how GenAI can be adopted to **provide useful insight** to traders, instead of serving as chatbots for infra purposes. Founder and CEO of Citadel, Ken Griffin, mentioned in an article that he believes **GenAI fails** to help hedge funds produce alpha. While there is a little bit of truth to that, we disagree with how he's approaching AI use. We believe that AI can be used as a **second set of eyes** to point traders/researchers in the right direction, offering them another opinion.

Research Question: Can we leverage GenAI to analyze a backtest to find loss periods, provide suggestions, and validate a strategy?

The Tedious Process of Analyzing Backtests

There are a myriad of reasons as to why backtests take time to analyze as expressed below

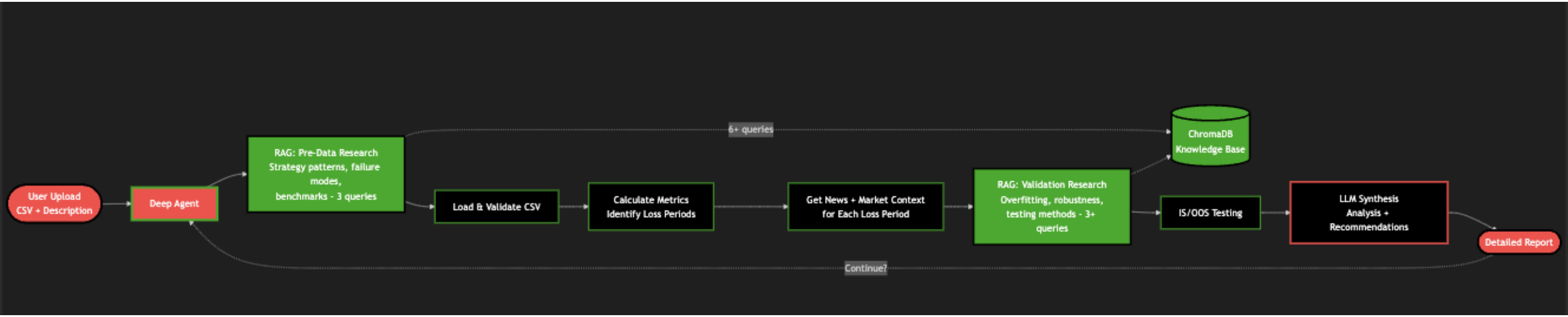
- **Data Complexity**
- **Context Requirements**
- **Statistical Validation**
- **Domain Knowledge Gaps**
- **Trade-Off Evaluation**

Table of Contents

Methodology

Evaluator Workflow

Below is a diagram showing how every piece of our system works with one another



Methodology

RAG Tool + Pipeline

Knowledge retrieval system and integration tool

The RAG tool and pipeline gives the deep agent access to a curated knowledge base about trading strategies, backtesting methodology, and market analysis, filling gaps the LLM wouldn't otherwise know.

Workflow

1) Build phase (one-time, offline)
   - Downloads 22 curated web sources (quantstart.com, investopedia.com, alphaarchitect.com, etc.)
   - Parses HTML, splits into chunks (1000 chars, 200-char overlap)
   - Generates OpenAI embeddings, stores in Chroma vector database persisted to disk (./chroma_db)
2) Startup phase
   - Loads the persisted vectorstore from disk on server startup (~2-3 seconds vs. 1-2 min rebuild)
   - Creates a retriever with k=2 (2 most relevant chunks per query)
   - Sets a single global retriever instance shared across all requests
3) Per-request retrieval
   - Agent calls retrieve(query) with a natural language question
   - Performs cosine-similarity semantic search
   - Returns top 2 chunks joined together as context

Key Design Decisions

Balance speed and effectiveness

Decision	Rationale
Build once, persist to disk	Avoids costly embedding API calls on every analysis
k=2 chunk retrieval	~1000-2000 tokens balances richness vs. agent context budget
200-char chunk overlap	Prevents concepts from being split across chunk boundaries
Global retriever (single instance)	Thread-safe, low memory footprint across concurrent requests|Delta|∈[0.375,0.625]
Static curated sources (22 total)	Focused on timeless concepts: Sharpe, drawdown, survivorship bias, Monte Carlo

Methodology

Backtest Analyzer

Loads the CSV + standardizes data

The backtest analyzer is a universal, format-agnostic tool that accepts any backtest CSV (timeseries or summary) without pre-built assumptions. It auto-detects format, standardizes to a clean state, then runs modular analysis.

Workflow

1) Load: read raw CSV, expose information with no interpretation
2) Inspect & detect
   - Detects timeseries (date + returns) or summary (sharpe + strategy_id)
3) Standardize
   - Timeseries: [date, strategy_id, returns, equity, position]
     - Equity recalculated from returns, missing values filled with 0
   - Summary: [strategy_id, sharpe, ann_return, ann_vol, max_drawdown]
     - Auto-detects column names with fallback aliases, can override
4) Analysis
   - Timeseries
     - Total/annualized return, vol, Sharpe, max drawdown, cost fragility
     - Groups consecutive losses into distinct periods
     - Tests IS/OOS stability; walk-forward optional
   - Summary
     - Top performers, overall stats, column breakdown, threshold filters

Notable Metrics and Algorithms

Cost fragility

- Counts signal transitions in the position column
- Estimates return after adding 5 bps or 10 bps per round-trip
- Flags if strategy unprofitable

IS/OOS split logic

- Data >= 8 years -> 60/30/10 IS/OOS/holdout
- Data < 8 years -> 70/30
- High risk flags (overfitting/data snooping) widen OOS to 40%

Loss period grouping

- Consecutive rolling-window losses within 14 days merged into on period
- Prevents noise from fragmenting single regime failure

Walk-forward

- Trains on 3 years, tests on 1 year, steps by 12 months
- Mimics real deployment

Methodology

Period Context Analyzer

Provides high-level market context

The Period Context Analyzer synthesizes market intelligence to provide explanations for strategy performance during specific temporal windows. Furthermore, it integrates regime detection, market microstructure analysis, and NLP (of financial news), to analyze what truly happens **"under the hood"** when a strategy fails.

At its core, the tool includes market regime classification based on VIX spikes/thresholds, multi-source news fetching and categorization, and context synthesis after previous sub-tools are called. The main focal point of this tool is its news aggregation which we used NewsAPI and Massive for.

Workflow:

1) Agent identifies significant drawdown (ex: -18% over 3 weeks)
2) Input loss period dates + primary ticker(s) from strategy description
3) Output a JSON w/ VIX stats, top 3-5 headlines with dates/sources, detected themes (ex: "banking crisis")
4) Agent uses the output to construct a causal narrative, linking macro events to strategy failure

Context Window Optimization

A few design decisions that we made to not crowd the agent's context window

Decision	Rationale
30-90 Day Window Limits	Prevents token bloat while maintaining temporal granularity
Ticker to Company-Name Mapping	Reduces query ambiguity (ex: "KO" -> "Coca-Cola")
Multi-Provider Redundancy	Falls back to macro context if ticker-specific news unavailable
Theme Extraction over Raw Headlines	Compresses 50+ articles into actionable regime labels

Methodology

Frontend

React app that serves as the user-facing interface.

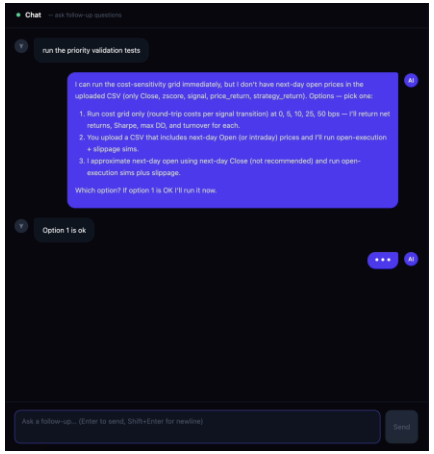
Workflow

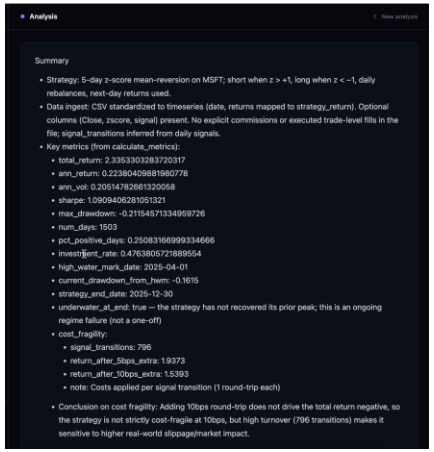
1) Upload a backtest CSV + describe your strategy
2) Agent runs full regime, risk, and factor analysis, returns markdown report with tables and metrics
3) Ask follow-up questions in the live chat panel.

Session ID persists full conversation context across questions.

UI Layout:

- Left panel: rendered markdown report (headers, tables, etc.)
- Right panel: live chat interface with message history





Routing

Connection to backend

- POST /api/invoke — accepts CSV + strategy description, triggers agent, returns analysis + session ID
- POST /api/continue — accepts session ID + follow-up question, agent recalls prior context via LangGraph Memory Saver, returns next AI response
    - Prompt shifts from "analyze this backtest" to "you are a helpful assistant continuing a conversation" — optimized for dialogue rather than deep analysis
- GET /api/health — verifies if backend is live and vectorstore is loaded into memory; drives the green/amber/red status indicator in the UI

Table of Contents

Demo

Results and Findings

Testing Pipeline & Results

Results of comparing our output vs ChatGPT vs Claude

The benchmark system evaluates agent performance across **10 diverse backtest scenarios** by comparing outputs against human-expert ground truth and baseline LLM responses, using GPT-4o as an independent judge to score each output on the metrics below.

Metric	Our Agent	ChatGPT	Claude
Avg LLM Score	8.5/10	8.5/10	**8.7/10**
Avg Tools Used	**11.2**	4.7	4.5
Temporal Periods Analyzed	**25.5**	19.3	17.8
Specific Numbers Cited	**195**	109	161

Analysis

Analysis of testing results

Looking at the chart, we can see that all three agents performed relatively similar over the 10 test cases. However, our agent used more tools on average, analyzed more temporal periods, and cited more resources. This indicates thorough analysis but potential inefficiency since our agent should have outperformed ChatGPT and Claude.

When inspecting all three outputs for a random sub-sample of test cases, research members came to the conclusion that the agent is able to **provide a wider range of suggestions**, covering news to find potential regimes. Futhermore, as it follows a **consistent structure and replicable approach**, it is easier interpreted than ChatGPT and Claude which had different formats for every single test case.

With further refinement and improvement, our agent will hopefully start to outperform the industry-standard baseline.

Table of Contents

Limitations, Risk Considerations, and Next Steps

Domain-Specific Limitations

Current limitations with output and performance

From our rigorous testing we found that there are several NewsAPI coverage gaps with noticeable **coverage degradation** pre-2018 despite our usage plan covering data from 2014 onwards. Moreover, we found that non-US equities received limited English-language news, leading to either empty or **irrelevant headlines.**

We also found that our regime detection logic was quite simple, only looking at VIX-based thresholds (stress > 30, calm < 15). What we are missing is credit spread analysis, market breadth, and yield curve dynamics. As a result, the output **misses regime transitions** (ex: 2022 rate hiking cycle without VIX spike).

Operational Risks

Risk considerations based on testing observations

It was observed that the agent occasionally **cites non-existent headlines** when news APIs fail, therefore hallucinating on edge cases. To mitigate this, we could require a URL field for all headlines and add a fact-checking step that verifies that the headline exists via web scraping.

Next Steps

Future additions and improvements

An idea that we had was to integrate academic/research papers into our vectorstore to **reduce parametric hallucination** and ground recommendations. While the vectorstore already has 20+ curated documents, we found that the retrieval tool didn't quite reach the depth we wanted it to and often scratched the surface with baseline information. With more specific research papers as context, the agent will be able to conduct further analysis of the strategy and provide insightful suggestions.

Another improvement that our team has thought of is **human-in-the-loop refinement** to improve agent analysis. It would work like a feedback system where the users can rate the analysis and members of our team can subsequently update the agent's system prompt. The feedback would not solely be limited to reviewing the analysis, but it would also include the user's preference on how they would like the analysis to be formatted, allowing a more tailored approach for each user.