Time series Augmented Generation for Financial Applications

Anonymous ACL submission

Abstract

Financial analysis demands bridging natural language queries with complex quantitative time series (TS) computations. While Large Language Models (LLMs) excel at language, they often falter on precise numerical reasoning and grounding in volatile financial data. We propose Time Series Augmented Generation (TSAG), an alternative to conventional Retrieval-Augmented Generation (RAG) approach. TSAG relies on tool-based infrastructure provided by LangChain framework and utilizes an LLM agent to parse natural language queries, select and invoke appropriate predefined time series analysis tools, and synthesize the tool outputs into coherent, accurate responses. We implement and evaluate TSAG, initially focusing on a proof-of-concept (POC) with cryptocurrency data and robust, predefined tools for seasonality, volatility, price, and correlation analysis. We compare multiple LLM agents (Llama 3.x, Qwen2 variants, GPT-4o variants, DeepSeek-V3 API). We provide evaluation benchmark which includes a set of typical questions with expected answers and framework to evaluate the LLM performance against them. The framework evaluates metrics such as Return Rate, Match Accuracy, LLM-accessed Accuracy, Hallucination Rate, and query latency as Seconds per Query (SPQ). Results demonstrate TSAG, particularly with capable agents like GPT-4o and Qwen2 (7B), achieves high levels of accuracy and low hallucination rates, validating the tool-based LLM integration approach for financial applications.

1 Introduction

The financial domain demands timely and accurate insights derived from complex, dynamic data (Tsay, 2005). Natural language offers an intuitive interface for accessing this information, yet effectively bridging conversational queries with the necessary underlying quantitative analysis remains a significant hurdle. Large Language Models (LLMs) have demonstrated remarkable capabilities in natural language understanding and generation (Brown et al., 2020; Wu et al., 2023). However, their application to finance is often constrained by limitations in precise numerical computation, robust temporal reasoning, and reliable grounding in volatile, high-frequency financial time series data (Fons et al., 2024). Directly applying LLMs to tasks requiring high analytical fidelity can lead to inaccurate or unsubstantiated outputs. Conversely, traditional quantitative models and libraries (Box and Jenkins, 1970), while accurate, lack accessible natural language interfaces, limiting their usability for broader audiences.

This capability gap hinders the seamless integration of sophisticated data analysis into interactive financial workflows. Standard Retrieval-Augmented Generation (RAG) systems primarily retrieve textual documents (Lewis et al., 2020) and are insufficient for queries demanding real-time computation over numerical time series. Recent RAG adaptations for time series forecasting focus on retrieving similar historical data sequences (Zaremba et al., 2024; Wu et al., 2025), a different task from executing diverse, on-demand analytical computations required for complex financial question-answering (Q&A) systems.

To address these limitations, we propose Time Series Augmented Generation (TSAG), a framework conceptualized as a **Tool-Augmented RAG** system, based on LangChain infrastructure (Kotiyal et al., 2024). TSAG employs an LLM agent not merely as a text processor, but as an orchestrator that translates natural language queries into calls to specialized, predefined financial time series analysis "grounding" functions (tools in terms of LangChain). This architecture synergistically combines the LLM's linguistic competence and reasoning capacity with the computational precision and verifiability of dedicated analytical tools operating on grounded data. By explicitly delegating quantitative tasks to reliable tools, TSAG aims to significantly enhance the accuracy, reliability, and interpretability of LLM-driven financial analysis compared to end-to-end models or less constrained agentic approaches (Data Science Dojo, 2023).

The main contributions of this work are:

- We design and implement TSAG, a novel Tool-Augmented RAG framework for financial Q&A via LLM agents and specialized tools, demonstrating a proof-of-concept within the cryptocurrency domain.

- We define and implement a library of specialized time series analysis tools targeting common financial queries related to seasonality, volatility, price dynamics, and correlation.

- We propose and apply a comprehensive, multi-dimensional evaluation benchmark and framework assessing execution success (Return Rate), accuracy based on query answer matching expected patterns (Match Accuracy), accuracy based on evaluation of answer against complete expected answer (evaluated by DeepEval (Confident AI, 2023)), Hallucination Rate(evaluated by DeepEval, and runtime efficiency as response latency measured as average Seconds per Query.

- We provide extensive empirical validation comparing multiple modern LLM agents (Llama 3.x, Qwen2 variants, GPT-4o variants, DeepSeek) within TSAG, analyzing performance trade-offs with different extreme settings for temperature - Temperature = 0.0 for the most conservative response and Temperature = 1.0 for the greatest diversity.

- We publicly release our evaluation benchmark consisting of 100 questions with expected patterns and full answers (`benchmark.tsv`) and evaluation framework (`test_eval_tsag.py.py`) to facilitate reproducibility and encourage further research into reliable, tool-grounded financial AI assistants.

This paper presents the related work (section 2), describes the TSAG framework (section 3), experimental setup (section 4) and results (section 5), and derives the conclusion (section 6).

2 Related Work

Our work intersects with Large Language Models (LLMs) in finance, time series analysis, Retrieval-Augmented Generation (RAG), and the paradigm of tool-based LLM agents.

LLMs in Finance LLMs have been applied to various financial tasks, including sentiment analysis (Loughran and McDonald, 2011), news summarization (Jin et al., 2020), report generation, and forecasting. Domain-specific models like BloombergGPT (Wu et al., 2023) show strong performance on financial NLP benchmarks. However, ensuring numerical accuracy and grounding in real-time quantitative data remains a challenge for these models when faced with complex analytical queries, which motivates our tool-based approach.

Time Series Analysis and Forecasting Traditional methods like ARIMA (Box and Jenkins, 1970) and GARCH (Engle, 1982), along with deep learning models like LSTMs (Hochreiter and Schmidhuber, 1997) and Transformers (Vaswani et al., 2017; Lim and Zohren, 2021), are widely used for financial time series, typically focusing on prediction. Integrating complex reasoning or natural language interaction for diverse Q&A tasks is less common. Furthermore, understanding model uncertainty in time series predictions is critical (Fons et al., 2024), reinforcing the need for reliable, verifiable computation where possible.

Retrieval-Augmented Generation (RAG) for Time Series Standard RAG enhances LLMs by retrieving relevant textual documents (Lewis et al., 2020). Adapting RAG for time series is an active research area. Recent works focus on retrieving relevant **past time series segments** to improve forecasting. Zaremba et al. (2024) introduced Retrieval Augmented Forecasting (RAF), using embedding similarity to find relevant historical patterns for Time Series Foundation Models (TSFMs). Similarly, Wu et al. (2025) proposed FinSeer, using an LLM-enhanced retriever and specific training objectives to find historically significant sequences for financial forecasting with their StockLLM model. While highly relevant in applying RAG to financial time series, these approaches focus on augmenting forecasting tasks by retrieving similar **data sequences**. Our TSAG framework differs significantly by employing an agent to select and invoke computational **tools** based on the query's analytical requirements, generating answers grounded in the tool's output rather than retrieved historical data.

Tool-Using LLMs and Agents for Time Series A significant recent trend involves augmenting LLMs with the ability to use external tools or APIs. Models like Toolformer (Schick et al., 2023) learn implicit API calls, while frameworks like LangChain (Chase, 2022) and systems using explicit function calling like Gorilla (Patil et al., 2023) enable LLMs to act as agents orchestrating external computations. The application of LLMs to time series analysis, including the combination with tools, is recognized as a key research direction (Jin et al., 2023). Various agentic architectures are being explored for time series. For instance, Ma et al. (2024) use LLM agents to reason about and integrate **news events** (external text) for forecasting. Liu et al. (2025b) (TimeCAP) employ a two-agent system where one LLM generates textual context from the time series, aiding the second agent in event prediction. Liu et al. (2025a) (TimeXL) utilize a multi-modal encoder and three collaborating LLMs (predict, reflect, refine) for explainable prediction. Our TSAG aligns with the tool-using agent paradigm but distinguishes itself by focusing specifically on financial Q&A, utilizing a curated set of **predefined, reliable quantitative "grounding" functions** as tools, and tasking a single LLM agent (in the current architecture) with orchestrating the entire process from NL query parsing to grounded NL response synthesis. This emphasizes verifiable computation for accuracy in the financial domain, contrasting with approaches focused solely on forecasting, integrating text, or using more complex multi-agent loops.

3 Methodology: The TSAG Framework

TSAG operates as a Tool-Augmented RAG system using an LLM agent to orchestrate specialized tools (Figure 1). This approach leverages predefined, reliable computations, contrasting with open-ended code generation or embedding-based retrieval.

3.1 Architecture

The architecture consists of four layers, presented in Figure 1.

First, there is a User Layer with "front-end" components which may be represented by a Telegram bot, Jupyter Notebook for research purposes, any end-user application (e.g. in Python) or the Evaluation Agent as part of the evaluation framework

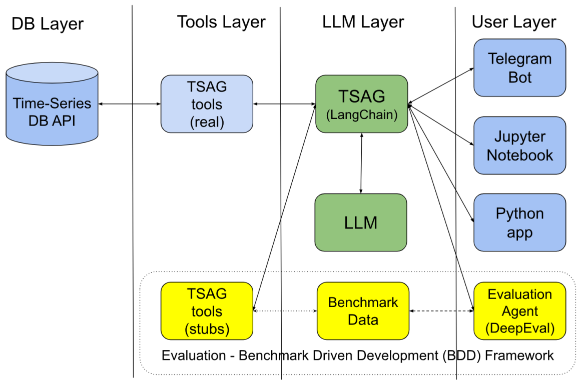


Figure 1: The TSAG Tool-Augmented RAG architecture and workflow.

based on DeepEval (Confident AI, 2023) being presented in this paper. Second, the LLM Layer consists of the TSAG kernel based on LangChain (Chase, 2022), connecting to one of the selected LLM agents - either self-hosted locally or cloud-hosted such as ChatGPT or DeepSeek. Third, the Tools Layer contains the tools plugged into the TSAG kernel as a "grounding" functions. Finally, the time series (TS) database layer (DB Layer) provides API for accessing the TS data.

For the evaluation purposes discussed in this paper, the Tools Layer contains grounding function "stubs" with expected responses hard-coded according to the Benchmark Data. The latter makes it possible to abstract from specific transient temporal data in the TS DB to perform **Benchmark-Driven Development (BDD)** of the TSAG framework on the basis of the Benchmark Data, supporting designated set of grounding functions.

3.2 Workflow

Given a natural language query (NLQ), TSAG proceeds via LLM-orchestrated steps to obtain a natural language response (NLR):

1. **Query Parsing & Tool Selection:** LLM agent parses NLQ, identifies intent, extracts parameters (using defaults from subsection 3.4) and selects tool(s) (subsection 3.3) based on prompt descriptions (subsection 3.5.

2. **Tool Execution:** System invokes selected tool(s) (e.g., `volatility(...)` according to Parkinson (1980)) via predefined Python code, querying data sources and performing verifiable TS computation.

3. **Result Synthesis:** Tool returns structured result (e.g., `{'volatility_percent': 5.0}`).

4. **Response Generation:** LLM agent synthesizes result into a grounded NLR answering the NLQ.

This structured workflow ensures calculations are handled by dedicated, verifiable logic.

3.3 Grounding Functions (Tools)

A core TSAG component is a library of specialized Python functions serving as tools. This proof-of-concept (POC) focuses on cryptocurrency trading data sourced from a Time-Series Database, as shown in Figure 1. We implement tools based on the NLQ → Function → NLR structure identified in our design. In this work, in order to provide an evaluation criterion, all real implementations of functions are replaced by corresponding "surrogate" stubs with predefined responses, depending on the input parameters. The initial tool set, prioritizing reliability for this study, covers the following:

- **Seasonality/Pattern Analysis:** Tools (`peak_traded_volume`, `lowest_traded_volume`, `round_the_clock_pattern`, `abnormal_deviations`) identifying recurring/anomalous volume patterns via statistical analysis.

- **Price and Volatility Analysis:** Tools (`price`, `volatility`) retrieving current data or calculating historical metrics (e.g., Parkinson volatility (Parkinson, 1980)) along with respective prediction tools (`predict_price`, `predict_volatility`).

- **Correlation Analysis:** Tools (`correlation_between_exchanges`, `correlation_between_tokens`) computing Pearson correlation coefficients (Pearson, 1895) across different instruments and exchanges.

- **Metadata Retrieval:** Tools (`get_base_tokens`, `get_exchanges`, `get_valid_time_units`, etc.) that provide valid parameters for the tools listed above.

Future work includes implementing more sophisticated "real" functions and expanding the scope of applications to traditional finance.

3.4 Function Parameters and Defaults

The LLM extracts parameters from the NLQ. Predefined defaults (e.g., `quote_token='USDT'`, `exchange='BINANCE'`) handle short queries with poorly defined input parameters.

3.5 LLM Agents

We evaluated several LLMs (Table 1) as agents, selected to explore trade-offs between size, cost, accuracy, hallucinations and run-time performance. Initial experiments involved use of different prompts, however at some point we have sorted out that using standard LangChain contextualization together with sufficient context size does not require specific prompts in order to maximize accuracy and minimize hallucinations. In particular, we discovered that context of 8192 is sufficient for holding the tool context for our set of tools, while smaller contexts of 4096 and default 2048 is insufficient. Specifically, insufficient context does not fit all tools to have their parameters (function arguments) identified properly.

4 Experimental Setup

We evaluated TSAG reliability, accuracy, level of hallucinations, and run-time efficiency across different LLM agents.

4.1 Tasks

We run the evaluation framework against benchmark consisting of 100 natural language questions with different levels of brevity corresponding to the tools described above in subsection 3.3: seasonality and patterns, price/volatility, correlation, and metadata retrieval.

4.2 Benchmark and Test Corpus

The test corpus used for evaluation consists of 100 items each represented by triplet derived from the original business specification, including the original natural language query (NLQ, as a "zero shot" sample), a set of expected words or numbers to be found in the text of the natural language response (NLR), and the full expected NLR text in the expected wording at the expected level of detail.

4.3 Evaluation Framework and Metrics

We evaluated across LLM agents using the evaluation framework based on DeepEval (Confident AI, 2023), having the following metrics evaluated based on the benchmark, having the DeepEval backed up with locally hosted Qwen2 7B LLM agent (Bai et al., 2024).

- **Return Rate (RR):** For each of the benchmark items - end-to-end execution success indicator, set to 1.0 if any non-empty text result is returned by the TSAG framework in response to the benchmark NLQ, and 0.0 otherwise. Across the entire benchmark - the average result, the higher the better.
- **Match Accuracy (MA):** For each of the benchmark items - Presence of all of the expected words or numbers in the benchmark indicated as 1.0 in case for the match, and 0.0 otherwise. Across the entire benchmark - the average result, the higher the better.
- **LLM-accessed Accuracy (LA):** For each of the benchmark items - measure of accuracy comparing the expected NLR from benchmark against the actual NLR, assessed by `DeepEval` (Confident AI, 2023) in the range between 0.0 and 1.0. Across the entire benchmark - the average result, the higher the better.
- **Hallucination Rate (HR):** For each of the benchmark items - degree of contextual divergence of the actual NLR text away from the expected one, assessed by `DeepEval` in the range between 0.0 and 1.0. Across the entire benchmark - the average result, the lower the better.
- **Seconds per Query (SPQ):** For each of the benchmark items - the amount of seconds to obtain NLR given NLQ. Across the entire benchmark - the average result, the lower the better.

4.4 Comparative Analysis

Our primary analysis compares the performance of different LLM agents, hosted locally or available online in the cloud, given the TSAG framework (Table 1) to identify models best suited for orchestrating financial tools, a key goal of our work. We evaluated these LLM agents against our benchmark. As expected, raw models not augmented with the tools provided ultimate Return Rate but showed zero Match Accuracy and negligible LLM-assessed Accuracy with high Hallucination Rate, confirming TSAG's necessity for grounded quantitative Q&A. These baseline results are omitted from Table 1 for clarity.

4.5 Implementation Details & Hyper-parameters

For TSAG development and evaluation we used Python 3.11, langchain 0.3.20, langchain-core 0.3.45, langchain-deepseek-official 0.1.0, langchain-ollama 0.2.2, langchain-openai 0.3.8, deepeval 2.5.4. LLM agents hosted locally were accessed via Ollama v0.1.32 for local models and official vendor APIs (OpenAI, DeepSeek) accessed in April 2025. The following models were evaluated.

- Llama 3.1 (8B) (Meta AI, 2024b)
- Llama 3.2 (3.2B) (Meta AI, 2024a)
- Qwen2 (0.5B, 1.5B, 7B) (Bai et al., 2024)
- Qwen2.5 (0.5B, 1.5B, 3B, 7B) (Lu et al., 2024)
- GPT-4o (OpenAI, 2024b)
- GPT-4o-mini (OpenAI, 2024a)
- DeepSeek-V3 (API) (AI, 2024)

Some LLM agents that we were considering to use initially, including local Gemma 7B (Gemma Team, 2024), DeepSeek hosted online (AI, 2024), and online Qwen (Lu et al., 2024) were excluded from evaluation results due to incompatibility with the required tooling in the used version of LangChain (Kotiyal et al., 2024).

We evaluated the LLM agents with different temperatures: Temperature=0.0 was used for them most "conservative" responses (Figure 2), Temperature=1.0 was used for the greatest "diversity". In case of Temperature=1.0, we used 3 different random seeds [1, 10, 100] for 3 runs (Table 1, Figure 3).

The following hardware was used for the evaluation: MSI Raider GE77HX 12UGS notebook with 12th Gen Intel(R) Core(TM) i7-12800HX 2.00 GHz, 32.0 GB RAM, 23.9 GB GPU NVIDIA GeForce RTX 3070 Ti Laptop GPU. The computational budget in hours was taking about 2 hours for each run of the benchmark including TSAG with LLM agent execution and evaluation carried our with DeepEval. The total research time with all experimental runs and debugging was about 3 machine-months.

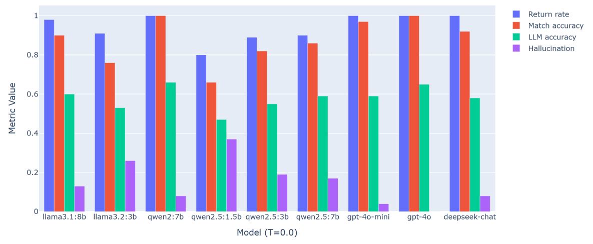


Figure 2: TSAG performance with different LLM agents (Temperature = 0.0). Metrics: return rate, match accuracy, LLM-measured accuracy and hallucination rate measured by DeepEval framework. Figure 3 shows results for Temperature = 1.0.

5 Results and Discussion

We analyzed the performance of TSAG, focusing on comparing LLM agents, using multiple runs with different random seeds with non-zero temperature to assess the reliability of our evaluations.

5.1 Quantitative Results

Table 1 and Figure 2 summarize the average performance of TSAG agents at Temperature=0.0. Figure 3 visualizes these averages at Temperature=1.0 and run variability based on three random seeds with indication of mean percentage error (MPE) and indication of respective error bars indicating reliability of our assessments.

Our quantitative evaluation reveals several key insights into TSAG performance. Firstly, the framework's viability is confirmed by high Return Rate values for most agents (Table 1), indicating successful orchestration of the NLQ-to-NLR pipeline. Secondly, Match Accuracy results highlight the variance in LLM agents' ability to correctly parse queries, invoke tools and generate output; state-of-the-art models like GPT-4o and Qwen2 7B achieve perfect scores, whereas most of smaller quantified models (Llama 3.2B, Qwen2.5 1.5B) struggle more, suggesting complex parameter extraction and response generation from tool output remains challenging for less capable agents. Some of the smallest models (Qwen2 0.5-1.5B, Qwen2.5 0.5B) delivering no responses at all with Return Rate at 0.0 were not included into resulting analysis at all.

Response quality, assessed via DeepEval, shows GPT-4o minimizes Hallucination Rate (0.02), demonstrating exceptional reliability in synthesis. Qwen2 7B leads slightly in average LLM-assessed Accuracy (0.66), indicating its responses closely mirror tool outputs, while also maintaining low Hallucination Rate (0.08). The variability across runs, indicated by MPE error bars (Figure 3), suggests good consistency for top models like GPT-4o and Qwen2 7B, whereas models with lower average performance also tend to exhibit higher variability between runs.

Comparing the deterministic "conservative" performance at Temperature=0.0 shown in Figure 2 with the "diverse" (which can be considered "exploratory") results at Temperature=1.0 shown in Figure 3a shows that the top-performing models deterministically maintain high Return Rate, Match Accuracy, LLM-assessed Accuracy, and low Hallucination Rate. Zero temperature generally leads to slight reductions in hallucination across the board, reinforcing its suitability for high-stakes applications requiring maximal factuality.

Latency analysis (Figure 3b) reveals significant differences. Local models like Qwen2 7B (2.2s) and the smaller Qwen2.5 3B (2.8s) offer fast responses. API models vary, with GPT-4o-mini (2.4-3s) being relatively quick, while DeepSeek-V3 exhibited high latency (14s) in our tests. This highlights a crucial trade-off between accessing potentially highest-performing proprietary models versus leveraging efficient open models for lower latency.

Based on the experiment with "conservative" Temperature=0.0 setting, the Qwen2 7B and GPT-4o appear to be equally good options for either local self-hosting on proprietary or leased hardware or remote access on subscription basis (Figure 2).

Based on the experiment with "diversity" Temperature=1.0 setting, both remotely accessible cloud versions of GPT-4o and GPT-4o-mini appear equal to Qwen2 7B in terms of accuracy and hallucinations, however they both become significantly less attractive due to increased response times (Figure 3).

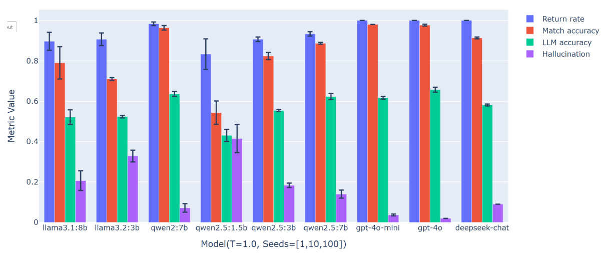
In both cases of the temperature, DeepSeek-V3 hosted online appeared little behind the top competitors in terms of accuracy and hallucinations, and losing dramatically with largest response time.

5.2 Qualitative Analysis

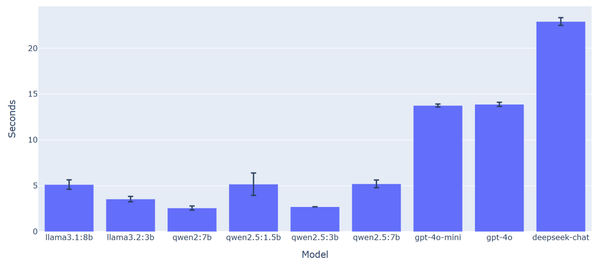
Manual review of generated responses complements the quantitative findings. Successful cases demonstrate the LLM agent's ability to parse complex natural language, correctly identify and parameterize the appropriate tool (e.g., handling multiple constraints in a query handled by function like `peak_traded_volume( base_token, quote_token, exchange, time_interval,`

LLM agent	**RR↑**	**MA↑**	**LA↑**	**HR↓**	**SPQ↓**
Llama 3.1 (8B) (Meta AI, 2024b)	0.98	0.90	0.60	0.13	4.6
Llama 3.2 (3.2B) (Meta AI, 2024a)	0.91	0.76	0.53	0.26	2.7
Qwen2 (7B) (Bai et al., 2024)	**1.00**	**1.00**	**0.66**	**0.08**	**2.2**
Qwen2.5 (1.5B) (Lu et al., 2024)	0.80	0.66	0.47	0.37	5.7
Qwen2.5 (3B) (Lu et al., 2024)	0.89	0.82	0.55	0.19	2.8
Qwen2.5 (7B) (Lu et al., 2024)	0.90	0.86	0.59	0.17	5.3
GPT-4o (API) (OpenAI, 2024b)	**1.00**	**1.00**	**0.65**	**0.02**	**2.4**
GPT-4o-mini (API) (OpenAI, 2024a)	1.00	0.97	0.59	0.04	2.9
DeepSeek-V3 (API) (AI, 2024)	1.00	0.92	0.58	0.08	14.1

Table 1: TSAG performance with different LLM agents (Temperature = 0.0). Metrics: return rate (RR), match accuracy (MA), LLM-measured accuracy (LA) measured by DeepEval framework, hallucination rate (HR) measured by DeepEval framework, seconds spent by query (SPQ). Arrows pointing up (↑) indicate the greater the better. Arrows pointing down (↓) indicate the smaller the better. Best results are highlighted as "bold".



(a) Metrics: return rate, match accuracy, LLM-measured accuracy and hallucination rate measured by DeepEval framework.



(b) Seconds spent by query (SPQ). Use of GPT and especially DeepSeek takes longer, compared to locally hosted models.

Figure 3: TSAG performance with different LLM agents as average over 3 runs (Temperature = 1.0). Error bars indicate run variability as mean percentage error (MPE). Figure 2 shows results for Temperature = 0.0.

`time_unit`, `period_unit`, `granularity_unit`, `threshold_percent`)), and synthesize the numerical or list-based output into a fluent, accurate sentence in natural language. For example, given "What was the price correlation between BTC and ETH quoted in USDT on BINANCE exchange in the past 7 days?", a high-performing agent correctly invokes `correlation_between_tokens` with extracted parameters and generates a response like "The price correlation between tokens BTC and ETH quoted in USDT on BINANCE exchange in the past 7 days trading window is 1.0." Common failure modes, particularly with less capable agents or ambiguous queries, included selecting an incorrect tool, failing to extract all necessary parameters (relying incorrectly on defaults), or minor hallucinations where the synthesized text slightly misrepresents the exact nuance of the tool's output (though major fabrications were rare with top models, reflected in low HR scores).

5.3 Discussion

The experimental results strongly validate Time-Series Augmented Generation (TSAG) approach for enabling reliable quantitative Q&A over financial time series using LLMs. By delegating computation to predefined, verifiable tools, TSAG effectively grounds LLM responses, achieving high accuracy scores (Table 1) and mitigating the inherent weaknesses of LLMs in complex numerical reasoning (Fons et al., 2024). The comparison across different agents underscores the critical role of the LLM's underlying capabilities; models like GPT-4o and Qwen2 (7B) excel at the multi-step reasoning involved in parsing, tool selection/invocation, and synthesis. The performance drop-off with smaller models suggests that robust instruction-following and parameter extraction are non-trivial requirements for successful tool orchestration in this domain.

Our analysis of run variability (MPE error bars, Figure 3) indicates good stability for the top-performing models across different random seeds with high temperature, lending confidence to their reliability. The comparison between Temperature=0.0 (Figure 2) and Temperature=1.0 average results (Figure 3a) suggests that while deterministic generation slightly reduces hallucination, performance on core accuracy metrics remains high for capable agents across these temperatures, offering flexibility depending on application needs (consistency vs. minor linguistic variation).

The TSAG approach, using predefined tools, offers significant advantages in terms of reliability and interpretability compared to alternatives considered in our design phase, such as agents generating Python code on the fly (Data Science Dojo, 2023) or end-to-end fine-tuning of a Large Language-Numeric Model (LLNM). While less flexible than code generation, the verifiable nature of the tools is paramount in the high-stakes financial domain. The challenges encountered, primarily related to natural language ambiguity and tool coverage, highlight key areas for future work. Improving the TSAG robustness through advanced prompting, agent fine-tuning, or incorporating query clarification steps could enhance performance. Expanding the tool library, potentially incorporating more sophisticated forecasting models beyond the current POC stubs, and developing mechanisms for multi-tool composition are necessary steps to broaden TSAG's analytical capabilities and address more complex financial queries. The latency differences observed also motivate further exploration of optimized local models like Qwen2 (7B).

6 Conclusion

In this paper, we presented and evaluated a Time-Series Augmented Generation (TSAG) approach, implemented as a Tool-Augmented RAG framework designed to empower LLM agents with reliable quantitative analysis capabilities for financial time series Q&A. By orchestrating calls to a library of specialized, predefined computational tools, TSAG effectively grounds language model outputs in verifiable data analysis, addressing critical limitations of LLMs in numerical reasoning and factuality within the demanding financial domain.

Our comprehensive experiments, comparing multiple state-of-the-art LLM agents (Llama 3.x, Qwen2 variants, GPT-4o variants, DeepSeek-V3), demonstrate the viability and effectiveness of this approach. We show that capable agents like GPT-4o and Qwen2 7B, when integrated into TSAG, achieve high accuracy and low hallucination in executing the tool-based pipeline, accurately matching tool outputs, generating faithful responses with minimal hallucination (validated by DeepEval). We analyzed performance trade-offs across model size, accuracy (RR, MA, LA) and hallucination (HR) metrics, latency (SPQ), and consistency (MPE), providing insights into agent selection. Furthermore, analysis of deterministic (Temperature=0.0) versus stochastic (Temperature=1.0) generation highlights the framework's stability and potential for tuning based on application requirements for factuality versus linguistic variation.

We provide a benchmark that can be used to evaluate other LLM agents to improve the described evaluation metrics, and can also be extended to obtain richer and more functional sets of tools that extend the applicability of TSAG.

This work validates tool-augmentation as a powerful and pragmatic strategy for building more reliable, accurate, and interpretable artificial intelligence (AI) applications for quantitative financial analysis. Key future directions include expanding the sophistication and coverage of the tool library (including robust prediction models and tools for traditional finance), enhancing the agent's ability to handle complex multi-step reasoning and natural language ambiguity, and incorporating rigorous uncertainty quantification, while continuing to prioritize ethical considerations and responsible AI development.

7 Limitations

Our work has limitations:

- **Tool Coverage & Brittleness:** Scope limited by predefined tools; cannot answer out-of-scope queries. Performance of the actual non-stub tools depends on actual implementation of them and content of the actual time-series database. Tools may be brittle to API/data format changes.

- **Tool Correctness:** Accuracy hinges on meticulous tool implementation and valid data sources. Further work on uncertainty quantification (Fons et al., 2024) is also needed.

- **Natural Language Robustness:** Parsing ambiguous or complex natural language queries remains challenging; robustness to query permutations needs further testing, as identified in our design.

- **Static Tools:** No dynamic code generation, limiting flexibility compared to some agent approaches. Reliance on LangChain-compatible tooling excluded some models (e.g., online Qwen).

- **Evaluation Scope:** Focused on crypto-finance Q&A domain; generalization to traditional finance requires significant tool adaptation. Run variability analysis based on 3 seeds provides initial insights but more runs would be beneficial.

- **Compositionality:** Limited exploration of multi-tool reasoning chains required for more complex analyses.

- **Sensitivity to Agent/Parameters:** Performance varies significantly with LLM choice and temperature, necessitating careful tuning and selection.

8 Ethical Considerations

Deploying and using TSAG, especially in production environment, requires addressing ethical concerns, as follows. (ACL Rolling Review Chairs, 2023):

- **Potential Risks** Improper use of any financial instrument cay cause financial damage to a user. From this perspective, grounding an LLM agent performance with manually predefined function with control over hallucination level, as we do in our work, can mitigate the risk. Anyway, using our TSAG framework in production, as using any financial instrument on the market requires personal responsibility and awareness.

- **Personally Identifying Info Or Offensive Content:** The benchmark we created do not include any personally identifying info or offensive content.

- **Transparency vs. Opacity:** Tool use enhances computational transparency. LLM reasoning remains partially opaque; we provide code details with tool stubs, TSAG framework and evaluation framework.

- **Accuracy and Reliability:** Crucial in finance. Tool accuracy dependence requires rigorous testing. Low hallucination rates are confirmed in our study for top performing models. Users must understand outputs are tool-based information, not infallible financial advice.

- **Data Bias:** In production use, reliance on historical time-series data may reflect historical market biases (e.g., asset popularity, exchange-specific patterns). Auditing for bias propagation is needed.

- **Potential Misuse:** Generating convincing analyses requires safeguards against misinformation/manipulation. Not a substitute for professional advice.

- **Data Privacy:** Uses public data; adaptation for private data requires robust security/privacy protocols.

- **Computational Resources:** LLM inference incurs costs. Experiments used remote access over Web API to GPT and Qwen LLMs hosted in the cloud and local LLM hosting on MSI Raider GE77HX 12UGS notebook with 12th Gen Intel(R) Core(TM) i7-12800HX 2.00 GHz, 32.0 GB RAM, 23.9 GB GPU NVIDIA GeForce RTX 3070 Ti Laptop GPU. We note the efficiency benefits of capable local models like Qwen2 (7B).

- **Reproducibility:** Enhanced by releasing code for function stubs with tool descriptions, entire TSAG framework and evaluation framework. API access and specific model versions may limit full replication.

- **Fairness & Equity:** When using the TSAG platform for production purposes, potential biases may arise if the data or actual implementation of real tools favors certain assets or exchanges. Expanding tool coverage requires attention to equitable representation.

Adherence to responsible AI principles is paramount.

References

ACL Rolling Review Chairs. 2023. Responsible NLP Research Checklist.

DeepSeek AI. 2024. DeepSeek LLM: Scaling Open-Source Language Models with Long Training. *Preprint*, arXiv:2405.04434.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yunxiang He, Jianxin Li, Lei Li, Yankai Lin, Jian Long, Peng Lu, Jiaxu Mao, Chengqiang Lu, Jianmin Wang, Wei Wang, Rui Yan, and 9 others. 2024. Qwen2: The New Generation of Qwen Large Language Models. *Preprint*, arXiv:2406.04728.

George EP Box and Gwilym M Jenkins. 1970. *Time series analysis: Forecasting and control*. Holden-Day.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, pages 1877–1901. Curran Associates, Inc.

Harrison Chase. 2022. Langchain. `https://github.com/langchain-ai/langchain`.

Confident AI. 2023. Deepeval: The evaluation framework for llms. `https://github.com/confident-ai/deepeval`.

Data Science Dojo. 2023. Langchain Agents for Time Series Analysis. `https://datasciencedojo.com/blog/langchain-agents-for-time-series-analysis/`.

Robert F Engle. 1982. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, 50(4):987–1007.

Joan Fons, Javier Conde, Adrián Martín, Paula Gordaliza, and Jordi Vitrià. 2024. MEASURING AND MODELING THE IMPACT OF MODEL UNCERTAINTY ON FINANCIAL TIME SERIES PREDICTION. *Preprint*, arXiv:2408.14484.

Gemma Team. 2024. Gemma: Open Models Based on Gemini Research and Technology. *Preprint*, arXiv:2403.08295.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Di Jin, Yixin Genc, Eugene Bart, Vick Singh, and Mirella Lapata. 2020. Hooks in the Headlines: Learning to Generate Headlines with Controlled Hooks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8716–8732.

Ming Jin, Yifan Chen, Haoming Chen, Qinkai Zheng, Chenjuan Guo, Lu Chen, Ruidi Chen, Hassan Foroosh, Reza Zandifar, Ramtin Zand, Huajie Shao, Shiliang Sun, Bolin Ding, and Liang Wang. 2023. Large Language Models for Time Series: A Survey. *Preprint*, arXiv:2310.10195.

Arnav Kotiyal, Praveen Gujjar J, Guru Prasad M S, Raghavendra M Devadas, Vani Hiremani, and Pratham Tangade. 2024. Chat with pdf using langchain model. In *2024 Second International Conference on Advances in Information Technology (ICAIT)*, volume 1, pages 1–4.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Preprint*, arXiv:2005.11401.

Bryan Lim and Stefan Zohren. 2021. Time series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209.

Chen Liu, Qiao Jin, Zheyuan Wang, Yuxuan Liang, and B. Aditya Prakash. 2025a. Explainable Multi-modal Time Series Prediction with LLM-in-the-Loop. *Preprint*, arXiv:2503.01013. Version 1.

Chen Liu, Qiao Jin, Yu Zhang, Yifan Zhao, Ajitesh Srivastava, and B. Aditya Prakash. 2025b. Time-CAP: Learning to Contextualize, Augment, and Predict Time Series Events with Large Language Model Agents. *Preprint*, arXiv:2502.11418. Version 2.

Tim Loughran and Bill McDonald. 2011. When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks. *The Journal of finance*, 66(1):35–65.

Chengqiang Lu, Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Yewen Dong, Yang Fan, Wenbin Ge, Yunxiang He, Yue Huang, Luo Ji, Zhengran Jin, Peng Lu, Jiaxu Mao, Run Ruan, Jiaqi Wang, Jianmin Wang, Wei Wang, and 5 others. 2024. Qwen2.5: Advancing Large Language Models in Reasoning, Comprehension, and Safety. *Preprint*, arXiv:2407.00678.

Yunjie Ma, Bingsheng Yao, Quan Tu, Ya Su, Yong Li, and Dongsheng Li. 2024. From News to Forecast: Iterative Event Reasoning in LLM-Based Time Series Forecasting. *Preprint*, arXiv:2409.17515. Version 1.

Meta AI. 2024a. Introducing Meta Llama 3.2.

Meta AI. 2024b. Llama 3.1 (8B) Model - Ollama Library. `https://ollama.com/library/llama3.1`. Accessed: 2025-04-22.

OpenAI. 2024a. GPT-4o mini [Model Information].

OpenAI. 2024b. Introducing GPT-4o.

Michael Parkinson. 1980. The Extreme Value Method for Estimating the Variance of the Rate of Return. *The Journal of Business*, 53(1):61–65.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large Language Model Connected with Massive APIs. *Preprint*, arXiv:2305.15334.

Karl Pearson. 1895. Note on Regression and Inheritance in the Case of Two Parents. *Proceedings of the Royal Society of London*, 58:240–242.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. *Preprint*, arXiv:2302.04761.

Ruey S Tsay. 2005. *Analysis of financial time series*, volume 543. John Wiley & Sons.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 5998–6008.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. BloombergGPT: A Large Language Model for Finance. *arXiv preprint arXiv:2303.17564*.

Zihao Wu, Cheng Yang, Fan Zhou, Jiaqi Wang, Qiang Zhang, Keyu Duan, Yi Yang, and Chenjuan Guo. 2025. Enhancing Financial Time-Series Forecasting with Retrieval-Augmented Large Language Models. *Preprint*, arXiv:2502.05878. Version 2.

Adam Zaremba, Kacper Chwiałkowski, Tomasz Zięba, Piotr Rybinski, and Marcin Andrychowicz. 2024. Retrieval-Augmented Financial Forecasting. *Preprint*, arXiv:2411.08249.

A Reproducibility Information

A.1 Code

The code for reproducibility of our results is available upon request from authors and submitted with the paper. The code consists of the following files, residing in respective folders, that should be deployed under common folder called `graphRAG` (due to specific historical reasons).

1. `requirements.txt` - list of 3rd party components with identified version numbers required to run the following TSAG framework and evaluation framework code

2. `tsag/tools.py` - implementation of the tools as grounding functions

3. `tsag/llm.py` - code of the TSAG framework, based on LangChain

4. `DeepEval/EvalTsag.py` - core code of the evaluation framework based on DeepEval

5. `DeepEval/deepeval_custom_llm.py` - custom code of the evaluation framework for specific LLM agents

6. `tests/test_eval_tsag.py` — root executable evaluation framework script used to evaluate specific LLM agent with the following command: `python tests/test_eval_tsag.py ./tests/benchmark.tsv qwen2.5:7b 1.0 3 1,10,100` (see the source file for the description of parameters)

A.2 Data

The data file represents the benchmark consisting of 100 items, described in subsection 4.2 in tab-separated file `tests/benchmark.tsv` which is expected to reside together evaluation framework script referenced above. The benchmark, which is a scientific artifact resulting from our work, is made publicly available under the terms of the MIT License, specified in the file `tests/LICENSE`.

The intended use of the benchmark is development and testing existing and future LLM agents against set of tools described in our work. Also, it is intended to have the benchmark extended with new typical questions and answers based on existing set of tools and parameters of respective function or extending the set of tools together with the benchmark. The benchmark data belongs to financial domain, specific to crypto-finance subject area, in English language.

B Implementation Details

B.1 Grounding Function (Tool) Summary

The grounding function used in the TSAG POC are implemented in Python. These tools expectedly encapsulate specific analytical logic corresponding to common financial queries accordingly to business specification in Table 2. The stubs of the functions delivered as part of the benchmark presented in this work are containing specific expected responses hard-coded in `tsag/tools.py` source code accordingly to the benchmark in `tests/benchmark.tsv`.

B.2 Parameters and Hyper-parameters

The following LLM parameters and hyper-parameters were used, overriding the defaults for respective LLM agents.

- LLM temperatures (`temperature`): [0.0, 1.0]
- Random seeds (`seed`): [1, 10, 100]
- Length of LLM context window in tokens (`num_ctx`): 8192
- The maximum number of tokens LLM is allowed to generate (`num_predict`): 512
- The maximum number of retries for GPT and DeepSeek LLM agents (`max_retries`): 2

Function Name	Description	Key Parameters (Defaults)	Returns
peak_traded_volume	Finds times (e.g., day of week) with highest volume within periods, exceeding relative threshold.	base_token: str quote_token: str (default='USDT') exchange: str (default='BINANCE') time_interval: int (default=1) time_unit: str (default='year') period_unit: str (default='week') granularity_unit: str (default='day') threshold_percent: float (default=5.0)	list[str]
lowest_traded_volume	Finds times with lowest volume within periods, below relative threshold.	base_token: str quote_token: str (default='USDT') exchange: str (default='BINANCE') time_interval: int (default=1) time_unit: str (default='year') period_unit: str (default='week') granularity_unit: str (default='day') threshold_percent: float (default=5.0)	list[str]
round_the_clock_pattern	Combines peak and lowest volume times for pattern summary.	base_token: str quote_token: str (default='USDT') exchange: str (default='BINANCE') time_interval: int (default=1) time_unit: str (default='year') period_unit: str (default='week') granularity_unit: str (default='day') threshold_percent: float (default=5.0)	tuple
abnormal_deviations	Finds recent time points with volume deviations exceeding historical norms by threshold.	base_token: str quote_token: str (default='USDT') exchange: str (default='BINANCE') time_interval: int (default=1) time_unit: str (default='year') period_unit: str (default='week') granularity_unit: str (default='day') threshold_percent: float (default=5.0)	tuple
price	Gets the latest price within the lookback window.	base_token: str quote_token: str (default='USDT') exchange: str (default='BINANCE') time_interval: int (default=1) time_unit: str (default='day')	float
volatility	Calculates historical price volatility (Parkinson method) over window.	base_token: str quote_token: str (default='USDT') exchange: str (default='BINANCE') time_interval: int (default=1) time_unit: str (default='day')	float
predict_price*	Predicts price for the next window (POC: simple extrapolation).	base_token: str quote_token: str (default='USDT') exchange: str (default='BINANCE') time_interval: int (default=1) time_unit: str (default='day')	float
predict_volatility*	Predicts volatility for next window (POC: simple extrapolation).	base_token: str quote_token: str (default='USDT') exchange: str (default='BINANCE') time_interval: int (default=1) time_unit: str (default='day')	float
correlation_between_exchanges	Computes Pearson price correlation across two exchanges.	base_token: str quote_token: str (default='USDT') exchange_a: str exchange_b: str time_interval: int (default=7) time_unit: str (default='day')	float
correlation_between_tokens	Computes Pearson price correlation between two tokens on one exchange.	base_token_a: str base_token_b: str quote_token: str (default='USDT') exchange: str (default='BINANCE') time_interval: int (default=7) time_unit: str (default='day')	float

Table 2: Summary of key grounding functions (tools) in TSAG POC. Parameters are listed with type and default value. *Prediction tools use simple extrapolation.

- The maximum number of retries for all LLM agents, in case if no text is generated (`retries`): 5

The search of the parameters and hyper-parameters was performed incrementally along with development course. Lower numbers or defaults of hyper-parameters were chosen at the beginning of the study and then we were incrementally increasing them in the course of debugging and benchmarking till the return rate and accuracy metrics reached the plateau and stopped improving further.