Large Language Models for Blockchain Security and Analytics: A Survey

Collette Eguakun Okundia[1] , **Cuneyt Akcora**[2] , **Arijit Khan**[1]
[1]Bowling Green State University, Ohio, USA
[2]University of Central Florida, Florida, USA
{ceguaku, arijitk}@bgsu.edu, cuneyt.akcora@ucf.edu

Abstract

Large Language Models (LLMs) are transforming blockchain security and analytics, yet a systematic evaluation of their capabilities remains limited. This survey provides a comprehensive, AI-centric assessment of LLM-based methods across over 70 recent studies spanning 11 application domains, such as security auditing, transaction fraud detection, and cryptocurrency portfolio management. Our unified taxonomy standardizes task formulations and evaluation practices to enable a comparison of six LLM roles across domains. For each domain, we review input representations tailored to blockchain data; LLM architectures, learning and inference paradigms, *e.g.,* fine-tuning, retrieval-augmented generation, and agentic strategies. Our review analyzes the strengths, limitations, and emerging patterns of LLM roles observed in current systems. Finally, we provide practical guidance for selecting LLMs for specific roles and outline promising research directions.

1 Introduction

Large Language Models are increasingly incorporated into blockchain systems for both security and financial analytics, including smart contract auditing, transaction monitoring, fraud detection, market analysis, and decentralized finance applications [Toyoda *et al.*, 2024; Witt *et al.*, 2025; Gong *et al.*, 2026; Nguyen *et al.*, 2025]. Blockchain technology, introduced with Bitcoin, has evolved into a heterogeneous ecosystem of networks and decentralized applications that manage large volumes of digital assets [Lee *et al.*, 2020]. This evolution has expanded the analytical surface of blockchains. On the one hand, security failures such as access-control flaws continue to cause large financial losses. On the other hand, the same infrastructure facilitates trading, lending, and portfolio allocation, which creates demand for scalable analytics to support financial decision-making under uncertainty.

Blockchain analytics encompasses several interdependent goals. Security-focused analyses detect vulnerabilities, anomalous transactions, and malicious behaviors. Financial analytics, in contrast, addresses price prediction, portfolio optimization, and risk assessment using on- and off-chain data. Despite different goals, both rely on the same underlying data sources: smart contract code, execution traces, transaction histories, and protocol interactions [Khan and Akcora, 2022]. Yet the field remains challenging: blockchain data is heterogeneous and rapidly evolving, and adversarial behavior adapts quickly. Thus, existing approaches tend to be domain-specific and ad hoc rather than unified or generalizable.

LLMs have emerged as a unifying analytical tool across these domains because they operate on semantic structure rather than fixed syntactic patterns. In security analytics, LLM-based systems reason about authorization intent, state transitions, and contract interactions that are only implicitly expressed in code, complementing static and symbolic analyzers [Sun *et al.*, 2024a; Sun *et al.*, 2024b]. In financial analytics, LLMs process heterogeneous information sources such as transaction sequences, market indicators, news, and social media to extract behavioral patterns, infer sentiment, and support forecasting and portfolio decisions. Many high-impact security exploits stem from logic and compositional flaws, while market dynamics depend on narrative and coordination that numeric models miss. By jointly processing code, structured data, and natural language, LLMs address these gaps.

The deployment of LLMs in blockchain analytics is shaped by practical constraints. Labeled data are limited across both security and finance: fraud labels are scarce, ground truth for anomalies is ambiguous, and financial regimes shift faster than supervised datasets can be curated [Gai *et al.*, 2023]. Existing systems thus rely on heterogeneous strategies, *e.g.,* retrieval-augmented generation to ground inference in historical incidents and financial records [Jin *et al.*, 2025], lightweight fine-tuning on small domain datasets [Feng and Fan, 2025], prompt-based control to encode domain constraints [Chegenizadeh *et al.*, 2025], and agentic orchestration that combines reasoning with external tools. These design choices vary widely across studies, complicating comparison.

The resulting growing literature remains fragmented across application areas. Prior surveys (Table 1) focus on traditional machine learning for blockchain analytics [Palaiokrassas *et al.*, 2025; He *et al.*, 2024], general LLM applications in software engineering and cybersecurity [Zhang *et al.*, 2026], or narrow subtopics such as smart contract vulnerability de-

Survey	SC	Tx	G	LT	BS	TS	OD
Blockchain-Enhanced ML [Ural and Yoshigoe, 2023]	×	×	×	×	✓	×	×
LLMs for SE: Survey [Fan *et al.*, 2023]	×	×	×	✓	×	×	×
Blockchain meets ML [Kayikci and Khoshgoftaar, 2024]	✓	✓	×	×	✓	×	×
Smart Contract Vulnerability Detection[Biagio *et al.*, 2024]	✓	×	×	✓	✓	×	×
Blockchain Data An. in the Era of LLMs [Toyoda *et al.*, 2024]	✓	✓	✓	✓	✓	✓	✓
When LLMs meet Cybersecurity [Zhang *et al.*, 2025a]	✓	×	×	✓	×	×	×
ML for Blockchain Data Analysis [Azad *et al.*, 2025]	✓	✓	✓	×	✓	✓	✓
LLMs and Blockchain: A Living Survey [Gong *et al.*, 2026]	✓	✓	×	✓	✓	×	×
ML on Blockchain Data [Palaiokrassas *et al.*, 2025]	✓	✓	✓	×	✓	✓	×
A Survey on LLMs for SE [Zhang *et al.*, 2026]	×	×	×	✓	×	×	×
Large Language Models for Blockchain [ours]	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparison of Blockchain and LLM research surveys across key topics. SC: smart contracts; Tx: transactions; G: graph-based; LT: LLM techniques; BS: blockchain-specific; TS: time series; OD: off-chain data.

tection [Biagio *et al.*, 2024; Gong *et al.*, 2026]. Financial applications are often treated separately from security, despite sharing data, models, and methodological challenges. To date, no survey has systematically examined how LLM techniques are applied across both the security and financial dimensions of blockchain analytics, including smart contract dynamic analysis, portfolio management, dApp behavior modeling, and DeFi security validation, making ours the first to do so.

Through this survey, we provide a unified assessment of large language model methods for blockchain security and financial analytics by examining not only which blockchain task is addressed, but how the LLM functions within each system, distinguishing among models that generate artifacts, classify behaviors, explain decisions, synthesize heterogeneous signals, translate between representations, or orchestrate external tools. Our review spans smart contract development, auditing, and dynamic analysis; transaction anomaly and fraud detection; financial modeling and portfolio design and dApp and DeFi validation.

We introduce a taxonomy that standardizes task definitions, input representations, model adaptation strategies, tool interaction patterns, and evaluation practices, while explicitly characterizing LLM roles such as generation, translation, and classification. Organizing the literature around these roles reveals recurring design patterns, persistent methodological gaps, and open challenges in benchmarking, explanation faithfulness, robustness, scalability, and financial reliability. To our knowledge, this is the first survey to comprehensively cover LLM applications across these major areas of blockchain analytics, offering a structured reference for research at the intersection of blockchain systems, security, and AI-driven analytics.

2 Taxonomy

This survey organizes the literature using a two-axis taxonomy that separates which blockchain problem is addressed from how the LLM is operationally used within the system. This distinction is necessary because many studies target the same nominal task while embedding the LLM in different positions in the pipeline. In some systems, the LLM directly produces predictions; in others, it mediates between representations, explains another model's output, or controls external analysis tools. Treating these usages as equivalent obscures the differences in design and evaluation.

We therefore place the LLM's functional role at the center of the taxonomy. Roles are treated as first-class descriptors that cut across domains and enable consistent comparison of model behavior. Across the surveyed literature, six recurring roles emerge that capture the dominant ways LLMs interact with blockchain data, code, and infrastructure.

A generator (G) produces blockchain artifacts from intent or specifications, including smart contracts, test cases, audit reports, or trading strategies. A translator (T) maps between representations while preserving semantic meaning, such as converting natural language requirements into formal properties, front-end claims into on-chain invariants, bytecode into readable source, or high-level algorithms into deployable smart contracts. A classifier (C) assigns labels or scores to inputs, including vulnerability classes, fraud categories, attack types, or risk levels. A synthesizer (S) aggregates heterogeneous signals into a unified output, for example, by combining on-chain activity, off-chain text, and retrieved knowledge into a single assessment or portfolio. An explainer (E) generates human-facing rationales in analyzed data, clarifying why an entity or behavior is flagged and highlighting relevant evidence. A tool user (U) orchestrates external analyzers and infrastructure, deciding when to invoke static analyzers and formal verifiers and integrating their outputs into subsequent reasoning. The roles are not mutually exclusive but analytically separable. We note that these roles are specific to the blockchain domain, and new roles may appear or exist in other domains.

Alongside the role axis, we group the domain axis applications by the primary blockchain data and task context, as illustrated in Figure 1. Smart Contracts is the most mature domain and includes development, auditing, and dynamic analysis, where generators and translators support specification-to-code pipelines, classifiers and explainers support vulnerability detection and reporting, and tool users enable hybrid reasoning with program analysis and verification tools [Sun *et al.*, 2024b; Xia *et al.*, 2024; Ding *et al.*, 2025; Jin *et al.*, 2025]. Transaction data stores on-chain activity at the address, trace, and graph levels, where classifiers detect anomalies and fraud, synthesizers fuse behavioral cues, explainers produce analyst-readable evidence, and tool users support interactive exploration over large transaction databases. Markets covers financial analytics such as price prediction and portfolio construction, where synthesizers and explainers combine on-chain indicators with off-chain signals, and generators propose candidate strategies that are evaluated through backtesting. Applications and DeFi address system-level behavior in decentralized applications and protocols, where translators extract intended properties from interfaces and documentation, tool users execute multi-step workflows through wallets and APIs, and classifiers and explainers support validation under compositional interactions. Ecosystem captures broader community and infrastructure perspectives, including sentiment analysis, governance, and blockchain-AI integration, where large-scale narrative aggregation and continuous monitoring are central.

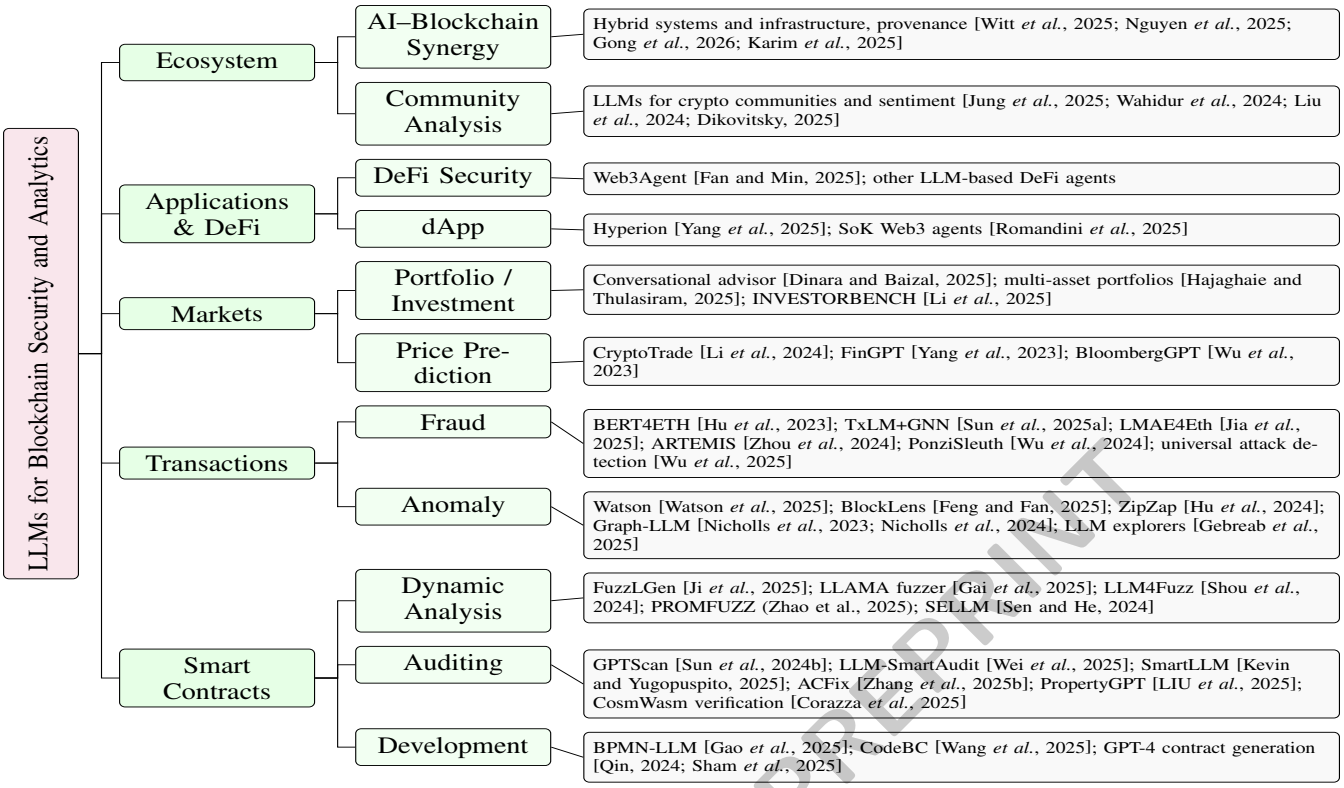

Figure 1: Taxonomy of LLM-based blockchain analytics with representative works per subdomain.

3 LLM-based Smart Contract Analysis

We analyze smart contract systems across three subfields: development, auditing, and dynamic analysis.

3.1 Smart Contract Development (G) (T)

Smart contract development translates high-level intent into immutable code in a high-level coding language, such as Solidity, that must be semantically correct, secure under adversarial interaction, and efficient under strict execution costs. Errors cannot be easily patched after deployment, and even small logic mistakes can lead to permanent financial loss. This makes the gap between business or legal specifications and correct contract code implementations a central challenge.

Large language models help bridge the gap between high-level intent and executable smart contracts by acting as **generators** that translate workflows, specifications, and algorithmic descriptions directly into code. BPMN-LLM [Gao *et al.*, 2025] converts Business Process Model and Notation workflows into Solidity through structured prompting, achieving 100% compilability and over 98% structural consistency across GPT-3.5, GPT-4, and Gemini. Security-aware generation further improves robustness: CodeBC [Wang *et al.*, 2025] fine-tunes CodeLlama-7B on curated secure and insecure Solidity samples, reducing vulnerability incidence by more than 50% and reaching a CodeBLEU score of 0.67. Despite these advances, benchmarks reveal persistent limitations; the IEEE BRAINS study [Daspe *et al.*, 2024] shows that while models such as Gemma-7B, CodeGemma-7B, Llama3-8B, Mistral-7B, and CodeLlama-7B perform well on simple contracts, their accuracy degrades in multi-contract settings where correctness depends on cross-contract interactions and shared state assumptions.

LLMs have also been used to **translate** machine learning logic into deployable smart contracts. [Sham *et al.*, 2025] convert PyTorch models into gas-optimized Solidity, while other systems improve generation by grounding the model with additional context such as domain documentation, protocol rules, and oracle feedback. For example, [Qin, 2024] fine-tune ChatGPT-3.5 on cryptocurrency and Solidity data to improve contextual accuracy, and [Xian *et al.*, 2024] integrate ChatGPT-4o, Gemini, and Llama-3 within a decentralized oracle-driven framework for multi-agent contract synthesis.

Overall, LLMs reduce the semantic burden of smart contract development, but especially generation has three limitations: i) prompt dependency, ii) incomplete security assurance, and iii) inconsistent gas efficiency.

3.2 Smart Contract Auditing (C) (E) (T) (U)

Contract auditing seeks to detect vulnerabilities before deployment and typically combines static and dynamic analyses with developer oversight. Auditing is the most active area of LLM usage in blockchain. It spans vulnerability detection, specification verification, access control repair, and bytecode decompilation. Across these tasks, LLMs augment or even replace symbolic and rule-based methods by reason-

ing about code semantics, generating human-interpretable explanations, and aligning results with domain knowledge.

From static detection to context-aware reasoning. Early research has employed general-purpose LLMs such as GPT-3.5 and GPT-4 for direct **classification** of smart contract vulnerabilities. [Chen *et al.*, 2025] conduct the first systematic benchmark, testing ChatGPT on the SmartBugs dataset across nine DASP10 categories. GPT-4 achieves high recall (88.2%) but low precision (22.6%), revealing sensitivity to prompt phrasing and comment bias. [Biagio *et al.*, 2024] extend this by fine-tuning `Llama-2-7B-chat-hf` using QLoRA on an `OWASP-SWC`-mapped dataset. Their fine-tuned model improves accuracy to 59.9% and provides line-level vulnerability annotations, outperforming traditional analyzers such as Mythril and Slither in recall.

Multi-agent and cooperative auditing systems. Multi-agent auditing frameworks distribute complementary LLM roles across agents. In LLM-SmartAudit [Wei *et al.*, 2025], GPT-based agents instantiate **classifier**, **explainer**, and **tool user** roles under a coordinating manager, achieving 98% accuracy and rediscovering 12 of 13 known exploits. Similarly, CodeSpeak [Chang *et al.*, 2025] adopts a dialogic setup in which one LLM (GPT) **explains** detected vulnerabilities and another **classifies** them, iterating toward consensus.

Retrieval-augmented and role-driven auditing. Retrieval-Augmented Generation is particularly effective in grounding model outputs to external sources, reducing hallucinations and false positives. SmartLLM by [Kevin and Yugopuspito, 2025] introduces role-based orchestration with Llama 3.1: a **classifier** identifies issues, an **explainer** explains them, and cross-checks results against retrieved Ethereum Request for Comment documentation. With QLoRA fine-tuning on 300 annotated contracts, SmartLLM reaches 100% recall and 70% accuracy, outperforming both static and zero-shot LLM baselines.

Toward generalizable LLM auditors. Across all these systems, the trend is clear: smart contract auditing is evolving from rule-based static checks toward interactive, reasoning-driven frameworks that integrate detection, explanation, and repair. However, challenges remain: Models depend heavily on limited, non-standard datasets, and no universal benchmark yet exists for logic and access control vulnerabilities. Explanation faithfulness still lacks formal guarantees. Future work must standardize evaluation datasets, integrate formal semantics into LLM embeddings, and explore cooperative agents for joint reasoning over multi-contract ecosystems.

3.3 Smart Contract Dynamic Analysis (G) (U)

Dynamic analysis executes smart contracts to observe runtime behavior and uncover vulnerabilities that static analysis may miss. Traditional "fuzzers" and symbolic execution tools systematically explore program paths but struggle with semantically rich input generation and path explosion.

Recent work integrates LLMs to mitigate these limitations and explore complex state spaces. [Sun *et al.*, 2025b] develop an adversarial method where a chain of prompts guides GPT-4 to **generate** and then refine fuzz-testing seeds. [Ji *et al.*, 2025] combine LLM-based agents with static dependency analysis to cluster functions and **generate** transaction sequences. Its BlockAnalyzer agent extracts basic blocks and control-flow information, while the PathDesigner agent uses this information to **generate** valid transaction sequence proposals. The resulting high-quality seeds boost fuzzing efficiency by 22%.

In dynamic analysis, LLMs help greatly by understanding business logic, generating semantically meaningful seeds, and prioritizing high-risk execution paths. In a sense, an LLM acts as the technical director, generating decisions for resource allocation and **using** the tools in dynamic analysis. Across methods, common themes include multi-stage architectures (seed generation, pre-fuzzing, feedback loops), integration with symbolic execution or control-flow analysis, and the use of event or transaction logs.

3.4 Future Directions

Current approaches remain dominated by prompt-engineered heuristics on pre-trained models rather than new model design with orchestration or retrieval. Systematic evaluation of how LLMs reason about program structure, handle symbolic constraints, and maintain factual consistency is still missing. The effects of multi-agent coordination, retrieval-augmented reasoning, and long-context memory on model reliability have yet to be rigorously studied. Future progress will depend on developing standardized benchmarks for reasoning, methods for interpretable intermediate representations, and architectures that unify natural language reasoning with formal, verifiable semantics. These changes would move LLMs from ad hoc assistants to accountable reasoning systems.

4 LLM-based Fraud and Anomaly Detection

Transaction analysis serves two distinct goals. Anomaly detection targets unusual on-chain behavior under weak or no labels, while fraud detection classifies known illicit activity using curated evidence. From a role perspective, the former emphasizes explanation and synthesis, while the latter emphasizes classification.

4.1 Anomaly Detection (C) (E) (S)

Transaction anomaly detection is one of the earliest and most active applications of large language models in blockchain analytics. These methods seek to identify abnormal or malicious transaction patterns that deviate from legitimate on-chain behaviors. Unlike classical graph- or rule-based detectors, LLM-driven systems can reason over opcode traces, transaction graphs, and behavioral metadata while producing natural language explanations for analysts.

LLMs for explainable anomaly detection. [Watson *et al.*, 2025] augments graph-based anomaly detectors with an LLM layer that converts node-level feature importance into natural language **explanations**, combining local statistics such as degree, betweenness, and transaction value with feature weights to produce concise, audit-friendly rationales. At the transactional level, LLMs increasingly serve as **classifiers**: [Feng and Fan, 2025] fine-tune Llama 3.2-1B with LoRA to analyze Ethereum execution traces–including opcodes, gas us-

age, and control flow–by chunking transactions and generating both maliciousness scores and localized explanations, achieving over 90% recall on real DeFi exploits. Because full-scale LLM inference over long, heterogeneous traces is costly, [Hu *et al.*, 2024] compress a transformer backbone via quantization and adapter fusion to 7.5% of its original size, reducing training overhead by nearly 60% while maintaining competitive performance (F1-score $\approx$ 70%). Complementing these approaches, [Lei *et al.*, 2025] summarize Bitcoin transaction graphs into human-readable representations that GPT, DeepSeek, and Llama can use for address classification.

LLMs for multimodal investigation. [Nicholls *et al.*, 2024] propose an LLM-based **explainable** AI pipeline for Bitcoin, where GPT-3.5 generates contextual narratives for each transaction, and GPT-4 **synthesizes** clusters of similar illicit activities into reports. Complementary to this, the multimodal system in [Nicholls *et al.*, 2023] combines textual and structural data to **explain** illicit flows across DeFi protocols.

LLM-based blockchain exploration and retrieval. [Gebreab *et al.*, 2025] introduce a blockchain explorer integrating a real-time LLM agent with a retrieval-augmented SQL engine for historical data. Using GPT-4o and Claude 3.5 as reasoning cores, the system supports both live transaction tracing and retrospective anomaly analytics, achieving accuracy improvements of 6%-24%.

4.2 Fraud Detection (C) (S)

Pre-trained transformers on transaction sequences. Several studies model blockchain activity as sequences of transactions and apply pre-trained Transformer models to detect fraud. [Hu *et al.*, 2023] train a Transformer encoder with a masked prediction objective that is structurally aligned with BERT-style pretraining, but its tokens are Ethereum addresses and transaction features, not language units. The model achieves strong results in phishing detection and deanonymization of addresses. [Yu *et al.*, 2025] further introduce a modular tokenizer for encoding transaction contents and use improved pre-training to improve detection. The model employs RoPE embedding and FlashAttention, and notably detects novel Solana attacks missed by all baselines.

Graph-integration and multimodal methods. To incorporate structural context, several models fuse LLMs with graph learning. [Sun *et al.*, 2025a] embeds transaction sentences with a Transformer and fuses them with similarity and interaction graphs via multi-head attention and graph neural networks. This yields $\sim$ 10% F1-score improvements on multiple Ethereum fraud datasets. [Jia *et al.*, 2025] combine contrastive language modeling with masked graph autoencoders. A cross-attention module unifies both views, achieving top performance and strong generalization to unseen scams. Beyond Ethereum, [Zhou *et al.*, 2024] use transaction graphs with textual and visual NFT metadata, processed via LLMs and CNNs, to detect Sybil users in NFT airdrops.

LLMs for code and execution analysis. [Wu *et al.*, 2024] introduce a zero-shot detection method for Ponzi schemes that uses taint analysis and a two-stage chain-of-thought prompt to guide LLMs (GPT-3.5, Llama3, and Mistral) in reasoning about fund flow logic. With no fine-tuning, GPT-3.5 achieves up to 96% balanced accuracy and **classifies** previously unknown scams. [Feng and Fan, 2025] process Ethereum execution traces through a LoRA-fine-tuned Llama model. The authors tokenize opcode traces and use overlapping chunks to capture deep call patterns. The approach **classifies** malicious transactions and also identifies trace segments contributing to the decision.

Unsupervised and few-shot approaches. [Wu *et al.*, 2025] introduce a self-supervised method for universal attack detection. The authors train a Transformer to reconstruct transaction traces and use smart contract function documentation as auxiliary supervision. Transactions with high reconstruction loss are **classified** as anomalies. This approach generalizes in low-data settings; it identifies previously unseen 1,692 real-world poisoning scams, with no dedicated labels.

LLMs in domain-specific blockchain systems. LLMs are also used in sector-specific blockchain platforms. [Islayem *et al.*, 2025] deploy retrieval-augmented LLMs (GPT-4o, Gemini 1.5 and Claude 3.5 Sonnet) to **classify** fraud in health insurance claims stored on Ethereum. The model accesses historical records, interprets unstructured medical notes, and flags suspicious claims such as inflated bills. It achieves over 99% accuracy in controlled experiments and demonstrates how LLM-based fraud detection generalizes beyond DeFi to broader smart contract ecosystems.

4.3 Future Directions

A key limitation of LLM-based fraud and anomaly analysis is the cost. Running large models directly on raw transaction streams, long traces, or full graphs does not scale. As a result, most systems rely on summarization, trace compression, or text-aligned graph representations before invoking the LLM.

Integration with graph models is often ad hoc, with limited routing or filtering before LLM use. Future progress depends on pipelines that are representation-aware, selectively invoke LLMs, and evaluate performance under explicit cost and latency constraints rather than accuracy alone.

Another interesting direction is reliably detecting bridge transactions, which is essential for tracing cross-chain fund flows, identifying laundering routes, and assessing systemic risks as assets move across heterogeneous blockchain environments. LLM-based agents are well-suited to this task because they can combine multi-hop reasoning with tool-driven log inspection and graph traversal to reconstruct complex cross-chain paths in an accurate and explainable manner.

5 LLMs for Financial Analysis

Financial analytics systems use LLMs to model market dynamics and construct investment decisions, spanning price prediction and portfolio design.

5.1 Price Prediction (C) (S)

LLMs have emerged as powerful sequence learners capable of adapting to diverse tasks through few- or zero-shot transfer learning. Their strong pattern recognition abilities–demonstrated by models such as Llama and GPT-4 across

multiple modalities–make them a natural candidate for time-series forecasting, including cryptocurrency price prediction. Yet their performance on complex, non-stationary domains like cryptocurrency markets, where clear seasonality and trends are often absent, remains a significant challenge.

Language models as time series forecasters. Time-LLM [Jin *et al.*, 2024] **encodes** multivariate time series for a frozen LLM by splitting each series into patches, projecting those patches onto learned text-like vectors via cross-attention, and prepending a short prompt to define the task. [Chaffard *et al.*, 2025] adapt Time-LLM for Bitcoin price prediction by replacing patching with an inverted sequence embedder to enable true multivariate attention, replacing learned prototype compression with offline PCA for efficient vocabulary alignment, applying fractional differencing to stabilize non-stationarity while preserving long-range dependencies, and augmenting inputs with prompt-encoded technical indicators to better leverage the LLM's financial knowledge. BreakGPT [Simonyan, 2024] presents a hybrid framework that combines a time-series encoder with an LLM to detect sharp upward price moves. Its main contributions are a surge-focused architecture, a task-aware training objective, and empirical evidence that LLM-based predictors can complement transformer encoders.

Sentiment analysis with off-chain data. Transformer-based models show strong promise in crypto market forecasting. [Dikovitsky, 2025] evaluates headline-driven ultra-short-term prediction using a dataset of 23,423 timestamped Bitcoin news items and a two-stage cascade **classifier** that filters market-moving headlines before predicting price direction, with GPT-based variants achieving up to 79% accuracy within a one-hour horizon. Complementary advances in sentiment modeling strengthen such pipelines: [Wahidur *et al.*, 2024] demonstrate that domain-adapted models and prompt engineering substantially improve zero-shot sentiment extraction by reducing hallucinations and enhancing label consistency. Extending this line of work, [Jung *et al.*, 2025] introduce a hybrid forecasting framework that fuses lexicon- and LLM-derived sentiment with technical indicators, on-chain metrics, and cross-asset price data, showing that multi-source feature fusion yields robust predictions over a 2,700-day evaluation. [Liu *et al.*, 2024] situate these analyses within a broader synthesis of LLM-based financial sentiment analysis, surveying model families, datasets, and applications that collectively define an emerging research agenda for LLM-driven market prediction.

5.2 Portfolio Design G E S U

LLMs are transforming portfolio management by processing massive datasets, **generating** trading strategies, and providing natural language **explanations**. BloombergGPT integrates structured market data with unstructured sources like news and filings [Wu *et al.*, 2023]. FolioLLM demonstrates how LLMs can support diverse tasks, including stock, exchange-traded fund (ETF), and multi-asset allocation [Popov and Roshka, 2024]. FinGPT highlights the role of prompt engineering, fine-tuning, and retrieval-augmented generation in enhancing personalization, risk balance, and transparency [Yang *et al.*, 2023]. Cryptocurrency portfolio management presents unique challenges due to extreme volatility, sentiment-driven price movements, and limited historical data. Effective systems must combine on-chain transactions with off-chain signals, adapt rapidly to market shifts, and deliver clear rationales to maintain investor trust.

LLM agent. CryptoTrade [Li *et al.*, 2024] presents a reflective, multi-agent LLM trading framework that uses GPT-3.5-turbo, GPT-4, and GPT-4o for zero-shot cryptocurrency trading. It **combines** on-chain data, including market trends and transaction records, with off-chain signals such as financial news to address market volatility. Reflection allows agents to iteratively critique and refine reasoning, improving decision quality. The system extracts structured features, encodes them into optimized prompt templates, and generates daily trading strategies with natural language rationales. It supports Bitcoin, Ethereum, and Solana, aiming to maximize returns.

Prompt engineering and fine-tuning. [Dinara and Baizal, 2025] introduce a GPT-4o-based conversational agent that personalizes cryptocurrency portfolio strategies by **integrating** market data with investor preferences expressed through dialogue. The system uses prompt engineering and instruction fine-tuning to align recommendations with long-term objectives while providing natural language **explanations** to enhance transparency. It supports diversified portfolio construction across multiple cryptocurrencies and evaluates performance using *Expected Popularity Complement*, *Average Recommendation Popularity*, and *User Satisfaction*, showing that conversational personalization improves decision quality.

Multi-asset portfolio with RAG. [Hajaghaie and Thulasiram, 2025] explore LLM-based multi-asset portfolio construction using Meta's Llama 3.1-8B, employing prompt engineering with few-shot and chain-of-thought methods and enhancing outputs through a classic RAG framework. By incorporating 90,000 rows of up-to-date financial data across major asset classes, the model produced diversified portfolios that included cryptocurrencies alongside equities, ETFs, commodities, and bonds, with a winning portfolio identified through frequency and weight analysis. Evaluation using Total Return, Annualized Return, Volatility, Beta, Alpha, Sharpe ratio, Maximum Drawdown, and Value-at-Risk demonstrated that RAG-enhanced portfolios delivered more balanced and risk-aware crypto allocations than the vanilla model.

Benchmarking. InvestorBench [Li *et al.*, 2025] is an open-source benchmark and agentic framework to evaluate LLM-based financial decision-making across three single-asset tasks: stock trading, cryptocurrency trading, and exchange-traded fund investing. It standardizes environments, **tools**, and metrics to compare thirteen large language models across proprietary, open-source, and finance-specific categories. The study shows that cryptocurrency trading tasks are highly volatile and strongly influenced by sentiment-driven news. Larger open-source models and domain-specific financial models achieved stronger returns and better risk-adjusted performance than smaller or general-purpose models. These results highlight that effective cryptocurrency portfolio management requires models that integrate diverse signals and

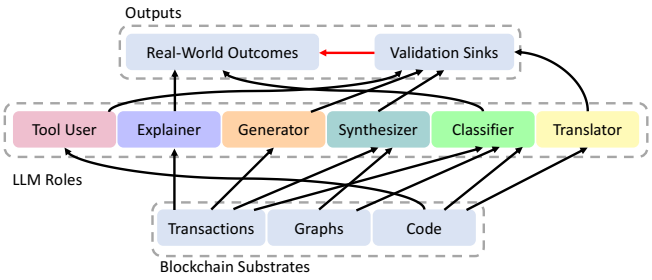

Figure 2: **Role-centric abstraction**. Blockchain substrates supply raw inputs that are processed by distinct LLM roles. Tool use is triggered when external computation, retrieval, simulation, or verification is needed. Role outputs flow through validation mechanisms such as static analysis, formal verification, backtesting, or human review before affecting real-world outcomes.

adapt to rapid market changes.

5.3 Future Directions

Many studies inadvertently leak future information through prompt design, window selection, or sentiment alignment, weakening claims of predictive power and masking the actual value of LLM-based reasoning. Rigorous evaluation demands time-aligned protocols that enforce strict information boundaries and quantify performance degradation under temporal drift. Progress is further limited by the absence of standardized benchmarks: current datasets focus on single-asset trading or simplified portfolios and fail to reflect DeFi-specific risks, *e.g.,* oracle delays.

6 Agentic AI for dApps and DeFi (E) (T) (U)

LLMs are increasingly deployed as tool-using agents that validate, execute, and reason over multi-step on-chain interactions rather than merely analyzing static artifacts.

A dApp typically integrates a user-facing front-end (web interfaces or documentation) with back-end smart contracts executing the actual logic. Misalignments between what the interface promises and what the on-chain code does can mislead users and erode trust in the system [Yang *et al.*, 2025]. Large language models have emerged as powerful tools to bridge these gaps by parsing human-facing descriptions and code in parallel to improve trustworthiness in dApps and DeFi [Romandini *et al.*, 2025; Fan and Min, 2025].

Interface consistency and transparency. [Yang *et al.*, 2025] has shown that dApp front-end descriptions often diverge from on-chain implementations in subtle but critical ways. For example, a DeFi platform's website might advertise fixed rewards that its smart contracts do not enforce. Hybrid AI analysis frameworks combine an LLM's natural language understanding with traditional program analysis to detect such inconsistencies. Hyperion by [Yang *et al.*, 2025] exemplifies this approach, using an instruction-tuned Llama2 model to interpret dApp UI text and documentation, alongside dataflow-guided symbolic execution of the corresponding smart contract. Hyperion extracts promised properties from the front-end (*e.g.,* fee policies) and cross-verifies them against the actual contract logic. The authors identify seven inconsistency types (*e.g.,* hidden fees), and report detecting such issues with high precision across hundreds of real-world dApps. This work shows how LLMs act as an **explainer** of the application's intended behavior (in natural language) and a checker against the trustless code that connects the interface to the implementation.

Autonomous LLM agents. Beyond static inconsistency checks, LLM agents can **use** decentralized applications like a Web3 user. [Fan and Min, 2025] propose a Web3 assistant combining an LLM with wallet tooling and blockchain APIs so that natural language text, *e.g.,* "*swap 2 ETH for USDC on Ethereum*" are decomposed into structured workflows of balance queries, quote retrieval, approval handling, and transaction submission across multiple networks. The agent runs an end-to-end loop to understand intent, retrieve domain-specific operation and API specifications via a modular RAG setup, predict the next on-chain action, and calibrate that action against recent API feedback before execution, with a dedicated executor module that dispatches validated calls and returns normalized status signals for next steps.

Future directions. A key research direction is understanding how personalized agents interact, since many DeFi exploits emerge from contracts that are individually correct but collectively unsafe. Agentic systems must reason about coupled state transitions and shared liquidity, and not rely on isolated contract assumptions. They must also support wallet-operated safety under partial observability and delegated authority through constrained action spaces, state-dependent permissions, and rollback-aware execution policies to prevent catastrophic losses. Validation is equally challenging: current systems depend on forked or sandboxed environments that approximate mainnet conditions, and the field lacks standardized methods for assessing whether an agent's reasoning is robust to adversarial interfaces, manipulated quotes, or MEV-driven dynamics. Meaningful progress will require benchmarks that tie execution traces to financial outcomes to evaluate stability and reliability under realistic conditions. Having covered all the taxonomy fields, Figure 2 illustrates this role-centric abstraction and how role outputs flow through validation stages.

7 Conclusion

We have presented a role-centric analysis of large language models for blockchain security and analytics, showing that treating generator, translator, classifier, synthesizer, explainer, and tool-user functions as first-class roles exposes design patterns obscured by task-based taxonomies. Across these settings, LLMs serve as semantic intermediaries that connect code, structured data, and natural language, relying on external tools for grounding and validation. Our findings suggest that LLMs are most effective as components of hybrid systems, and meaningful progress requires benchmarks evaluating reasoning faithfulness, execution-grounded validation, and robustness under realistic conditions. Until such foundations mature, LLM-based blockchain systems should be deployed as assistive analytical tools rather than autonomous agents with implicit guarantees.

References

[Azad *et al.*, 2025] Poupak Azad, Cuneyt Gurcan Akcora, and Arijit Khan. Machine learning for blockchain data analysis: Progress and opportunities. *Distributed Ledger Technologies*, 5(1), 2025.

[Biagio *et al.*, 2024] Boi Biagio, Christian Esposito, and Sokjoon Lee. Smart contract vulnerability detection: The role of large language model (LLM). *ACM SIGAPP ACR*, 24(2):19–29, 2024.

[Chaffard *et al.*, 2025] Owen Chaffard, Pablo Mollá, Marc Cavazza, and Helmut Prendinger. Enhancing large language models for Bitcoin time series forecasting. *Knowledge-Based Systems*, 330:114449, 2025.

[Chang *et al.*, 2025] Shuyu Chang, Chen Geng, Haiping Huang, Rui Wang, Qi Li, and Yang Zhang. Codespeak: Improving smart contract vulnerability detection via llm-assisted code analysis. *JSS*, page 112635, 2025.

[Chegenizadeh *et al.*, 2025] Mostafa Chegenizadeh, Zixian Pang, Junyong Cao, and Lundrim Azemi. Scalable blockchain analytics: an llm-powered approach. In *ICBC*, 2025.

[Chen *et al.*, 2025] Chong Chen, Jianzhong Su, Jiachi Chen, Yanlin Wang, Tingting Bi, Jianxing Yu, Yanli Wang, Xingwei Lin, Ting Chen, and Zibin Zheng. When ChatGPT meets smart contract vulnerability detection: How far are we? *ACM TOSEM*, 34(4):1–30, 2025.

[Corazza *et al.*, 2025] Jan Corazza, Ivan Gavran, Gabriela Moreira, and Daniel Neider. Accessible smart contracts verification: Synthesizing formal models with tamed llms. In *ICST*, 2025.

[Daspe *et al.*, 2024] Etienne Daspe, Mathis Durand, Julien Hatin, and Salma Bradai. Benchmarking large language models for ethereum smart contract development. In *BRAINS*, 2024.

[Dikovitsky, 2025] Vladimir Dikovitsky. Short-term cryptocurrency price forecasting based on news headline analysis. *Frontiers Blockchain*, 8, 2025.

[Dinara and Baizal, 2025] Alfara Nafi Dinara and ZKA Baizal. Large language model-based conversational recommender system for personalizing long-term cryptocurrency portfolio recommendation. In *ICoICT*, 2025.

[Ding *et al.*, 2025] Hao Ding, Yizhou Liu, Xuefeng Piao, Huihui Song, and Zhenzhou Ji. Smartguard: An llm-enhanced framework for smart contract vulnerability detection. *Expert Syst. Appl.*, 269:126479, 2025.

[Fan and Min, 2025] Sizheng Fan and Tian Min. Web3agent: Automating on-chain operations via natural language interfaces. *ACM TWEB*, 2025.

[Fan *et al.*, 2023] Angela Fan, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, Shubho Sengupta, Shin Yoo, and Jie M Zhang. Large language models for software engineering: Survey and open problems. In *ICSE-FoSE*, 2023.

[Feng and Fan, 2025] Chi Feng and Lei Fan. Blocklens: Detecting malicious transactions in ethereum using llm techniques. In *ISC*, 2025.

[Gai *et al.*, 2023] Yu Gai, Liyi Zhou, Kaihua Qin, Dawn Song, and Arthur Gervais. Blockchain large language models. *arXiv:2304.12749*, 2023.

[Gai *et al.*, 2025] Keke Gai, Haochen Liang, Jing Yu, Liehuang Zhu, and Dusit Niyato. Llama: Multi-feedback smart contract fuzzing framework with llm-guided seed generation. *arXiv:2507.12084*, 2025.

[Gao *et al.*, 2025] Sheng Gao, Wenting Liu, Jianming Zhu, Xuewen Dong, and Jin Dong. Bpmn-llm: Transforming bpmn models into smart contracts using large language models. *IEEE Software*, 2025.

[Gebreab *et al.*, 2025] Senay A. Gebreab, Ahmad Musamih, Khaled Salah, and Raja Jayaraman. Llm-based exploration and analysis of real-time and historical blockchain data. *Expert Syst. Appl.*, page 129851, 2025.

[Gong *et al.*, 2026] Jianghao Gong, Peiqi Yan, Yue Zhang, Hongli An, and Logan Liu. Blockchain meets llms: A living survey on bidirectional integration. *Expert Syst. Appl.*, 298:129851, 2026.

[Hajaghaie and Thulasiram, 2025] Ahmadreza Hajaghaie and Ruppa K Thulasiram. Leveraging large language models and retrieval-augmented generation for enhanced multi-asset portfolio construction. In *CiFer*, 2025.

[He *et al.*, 2024] Zheyuan He, Zihao Li, Sen Yang, He Ye, Ao Qiao, Xiaosong Zhang, Xiapu Luo, and Ting Chen. Large language models for blockchain security: A systematic literature review. *arXiv:2403.14280*, 2024.

[Hu *et al.*, 2023] Sihao Hu, Zhen Zhang, Bingqiao Luo, Shengliang Lu, Bingsheng He, and Ling Liu. Bert4eth: A pre-trained transformer for ethereum fraud detection. In *WWW*, 2023.

[Hu *et al.*, 2024] Sihao Hu, Tiansheng Huang, Ka-Ho Chow, Wenqi Wei, Yanzhao Wu, and Ling Liu. Zipzap: Efficient training of language models for large-scale fraud detection on blockchain. In *WWW*, 2024.

[Islayem *et al.*, 2025] Ruba Islayem, Senay Gebreab, Walaa AlKhader, Ahmad Musamih, Khaled Salah, Raja Jayaraman, and Muhammad Khurram Khan. Using large language models for enhanced fraud analysis and detection in blockchain based health insurance claims. *Sci. Rep.*, 15(1):29763, 2025.

[Ji *et al.*, 2025] Songyan Ji, Mingyue Xu, Jin Wu, and Jian Dong. Fuzzlgen: Logical seed generation for smart contract fuzzing via llm-based agents and program analysis. In *NDSS Symposium*, 2025.

[Jia *et al.*, 2025] Yifan Jia, Yanbin Wang, Jianguo Sun, Ye Tian, and Peng Qian. Lmae4eth: Generalizable and robust ethereum fraud detection by exploring transaction semantics and masked graph embedding. *TIFS*, 2025.

[Jin *et al.*, 2024] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-llm: Time series forecasting by reprogramming large language models. In *ICLR*, 2024.

[Jin *et al.*, 2025] Yanli Jin, Chunpei Li, Peng Fan, Peng Liu, Xianxian Li, Chen Liu, and Wangjie Qiu. LLM-BSCVM: an llm-based blockchain smart contract vulnerability management framework. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 44–55. Springer, 2025.

[Jung *et al.*, 2025] Hae Sun Jung, Haein Lee, and Jang Hyun Kim. Detecting Bitcoin sentiment: Leveraging language model applications in sentiment analysis for Bitcoin price prediction. *Neural Process. Lett.*, 57(5):77, 2025.

[Karim *et al.*, 2025] Md Monjurul Karim, Dong Hoang Van, Sangeen Khan, Qiang Qu, and Yaroslav Kholodov. Ai agents meet blockchain: A survey on secure and scalable collaboration for multi-agents. *Future Internet*, 17(2), 2025.

[Kayikci and Khoshgoftaar, 2024] Safak Kayikci and Taghi M Khoshgoftaar. Blockchain meets machine learning: A survey. *J. Big Data*, 11(9), 2024.

[Kevin and Yugopuspito, 2025] Jun Kevin and Pujianto Yugopuspito. Smartllm: Smart contract auditing using custom generative ai. In *ICoCSETI*, 2025.

[Khan and Akcora, 2022] Arijit Khan and Cuneyt G. Akcora. Graph-based management and mining of blockchain data. In *CIKM*, 2022.

[Lee *et al.*, 2020] Xi Tong Lee, Arijit Khan, Sourav Sen Gupta, Yu Hann Ong, and Xuan Liu. Measurements, analyses, and insights on the entire ethereum blockchain network. In *WWW*, 2020.

[Lei *et al.*, 2025] Yuchen Lei, Yuexin Xiang, Qin Wang, Rafael Dowsley, Tsz Hon Yuen, Kim-Kwang Raymond Choo, and Jiangshan Yu. Large language models for cryptocurrency transaction analysis: A Bitcoin case study. *arXiv:2501.18158*, 2025.

[Li *et al.*, 2024] Yuan Li, Bingqiao Luo, Qian Wang, Nuo Chen, Xu Liu, and Bingsheng He. Cryptotrade: A reflective llm-based agent to guide zero-shot cryptocurrency trading. In *EMNLP*, 2024.

[Li *et al.*, 2025] Haohang Li, Yupeng Cao, Yangyang Yu, Shashidhar Reddy Javaji, Zhiyang Deng, Yueru He, Yuechen Jiang, Zining Zhu, Kp Subbalakshmi, Jimin Huang, et al. Investorbench: A benchmark for financial decision-making tasks with llm-based agents. In *ACL*, 2025.

[Liu *et al.*, 2024] Chenghao Liu, Arunkumar Arulappan, Ranesh Naha, Aniket Mahanti, Joarder Kamruzzaman, and In-Ho Ra. Large language models and sentiment analysis in financial markets: A review, datasets, and case study. *IEEE Access*, 12:134041–134061, 2024.

[LIU *et al.*, 2025] Ye LIU, Yue XUE, Daoyuan WU, Yuqiang SUN, Yi LI, Miaolei SHI, and Yang LIU. Propertygpt: Llm-driven formal verification of smart contracts through retrieval-augmented property generation. In *Network and Distributed System Security (NDSS) Symposium 2025*. Internet Society, 2025.

[Nguyen *et al.*, 2025] Cong T Nguyen, Yinqiu Liu, Hongyang Du, Dinh Thai Hoang, Dusit Niyato, Diep N Nguyen, and Shiwen Mao. Generative ai-enabled blockchain networks: Fundamentals, applications, and case study. *IEEE Netw.*, 39(2):232–241, 2025.

[Nicholls *et al.*, 2023] Jack Nicholls, Aditya Kuppa, and Nhien-An Le-Khac. Enhancing illicit activity detection using xai: A multimodal graph-llm framework. *arXiv:2310.13787*, 2023.

[Nicholls *et al.*, 2024] Jack Nicholls, Aditya Kuppa, and Nhien-An Le-Khac. Large language model xai approach for illicit activity investigation in bitcoin. *Neural Comput. Appl.*, pages 1–13, 2024.

[Palaiokrassas *et al.*, 2025] Georgios Palaiokrassas, Sarah Bouraga, and Leandros Tassiulas. Machine learning on blockchain data: A systematic mapping study. an annual review of information science and technology (arist) paper. *JASIST*, pages 1–48, 2025.

[Popov and Roshka, 2024] Andrey Popov and Oleg Roshka. FolioLLM: Constructing Portfolio of ETFs using Large Language Models. Technical report, Stanford University, Department of Computer Science, 2024. https://web.stanford.edu/class/cs224n/final-reports/256938687.pdf.

[Qin, 2024] Hao Qin. Revolutionizing cryptocurrency operations: The role of domain-specific large language models (LLMs). *IJCTT*, 72(6):101–113, 2024.

[Romandini *et al.*, 2025] Nicolò Romandini, Carlo Mazzocca, Kai Otsuki, and Rebecca Montanari. Sok: Security and privacy of ai agents for blockchain. In *2025 7th International Conference on Blockchain Computing and Applications (BCCA)*, pages 708–720. IEEE, 2025.

[Sen and He, 2024] Yang Sen and Jiahao He. Sellm: An integrated tool leveraging symbolic execution and llms for smart contract vulnerability detection. In *ICCWAMTIP*, 2024.

[Sham *et al.*, 2025] Nikumbh Sarthak Sham, Sandip Chakraborty, and Shamik Sural. Generation of optimized solidity code for machine learning models using LLMs. In *ICBC*, 2025.

[Shou *et al.*, 2024] Chaofan Shou, Jing Liu, Doudou Lu, and Koushik Sen. Llm4fuzz: Guided fuzzing of smart contracts with large language models. *arXiv:2401.11108*, 2024.

[Simonyan, 2024] Aleksandr Simonyan. Breakgpt: Leveraging large language models for predicting asset price surges. In *Workshop on LLMs and Generative AI for Finance@ICAIF*, 2024.

[Sun *et al.*, 2024a] Yuqiang Sun, Daoyuan Wu, Yue Xue, Han Liu, Wei Ma, Lyuye Zhang, Yang Liu, and Yingjiu Li. Llm4vuln: A unified evaluation framework for decoupling and enhancing llms' vulnerability reasoning. *arXiv:2401.16185*, 2024.

[Sun *et al.*, 2024b] Yuqiang Sun, Daoyuan Wu, Yue Xue, Han Liu, Haijun Wang, Zhengzi Xu, Xiaofei Xie, and

Yang Liu. Gptscan: Detecting logic vulnerabilities in smart contracts by combining GPT with program analysis. In *ICSE*, 2024.

[Sun *et al.*, 2025a] Jianguo Sun, Yifan Jia, Yanbin Wang, Ye Tian, and Sheng Zhang. Ethereum fraud detection via joint transaction language model and graph representation learning. *IF*, 120:103074, 2025.

[Sun *et al.*, 2025b] Jiaze Sun, Zhiqiang Yin, Hengshan Zhang, Xiang Chen, and Wei Zheng. Adversarial generation method for smart contract fuzz testing seeds guided by chain-based llm. *ASE*, 32(1):12, 2025.

[Toyoda *et al.*, 2024] Kentaroh Toyoda, Xiao Wang, Mingzhe Li, Bo Gao, Yuan Wang, and Qingsong Wei. Blockchain data analysis in the era of large-language models. *arXiv:2412.09640*, 2024.

[Ural and Yoshigoe, 2023] Ozgur Ural and Kenji Yoshigoe. Survey on blockchain-enhanced machine learning. *IEEE Access*, 11:145331–145362, 2023.

[Wahidur *et al.*, 2024] Rahman SM Wahidur, Ishmam Tashdeed, Manjit Kaur, and Heung-No Lee. Enhancing zero-shot crypto sentiment with fine-tuned language model and prompt engineering. *IEEE Access*, 12:10146–10159, 2024.

[Wang *et al.*, 2025] Lingxiang Wang, Hainan Zhang, Qinnan Zhang, Ziwei Wang, Hongwei Zheng, Jin Dong, and Zhiming Zheng. Codebc: A more secure large language model for smart contract code generation in blockchain. *arXiv:2504.21043*, 2025.

[Watson *et al.*, 2025] Adriana Watson, Grant Richards, and Daniel Schiff. Explain first, trust later: Llm-augmented explanations for graph-based crypto anomaly detection. *arXiv:2506.14933*, 2025.

[Wei *et al.*, 2025] Zhiyuan Wei, Jing Sun, Yuqiang Sun, Ye Liu, Daoyuan Wu, Zijian Zhang, Xianhao Zhang, Meng Li, Yang Liu, Chunmiao Li, et al. Advanced smart contract vulnerability detection via llm-powered multi-agent systems. *IEEE Trans. Softw. Eng.*, 2025.

[Witt *et al.*, 2025] Leon Witt, Armando Teles Fortes, Kentaroh Toyoda, Wojciech Samek, and Dan Li. Blockchain and artificial intelligence: Synergies and conflicts. In *Advances in Artificial Intelligence*, pages 89–108. Springer, 2025.

[Wu *et al.*, 2023] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance. *arXiv:2303.17564*, 2023.

[Wu *et al.*, 2024] Cong Wu, Jing Chen, Ziwei Wang, Ruichao Liang, and Ruiying Du. Semantic sleuth: Identifying ponzi contracts via large language models. In *ASE*, 2024.

[Wu *et al.*, 2025] Zhiying Wu, Jiajing Wu, Hui Zhang, Zibin Zheng, and Weiqiang Wang. Hunting in the dark forest: A pre-trained model for on-chain attack transaction detection in web3. In *WWW*, 2025.

[Xia *et al.*, 2024] Shihao Xia, Shuai Shao, Mengting He, Tingting Yu, Linhai Song, and Yiying Zhang. Auditgpt: Auditing smart contracts with ChatGPT. *arXiv:2404.04306*, 2024.

[Xian *et al.*, 2024] Youquan Xian, Xueying Zeng, Duancheng Xuan, Danping Yang, Chunpei Li, Peng Fan, and Peng Liu. Connecting large language models with blockchain: Advancing the evolution of smart contracts from automation to intelligence. *arXiv:2412.02263*, 2024.

[Yang *et al.*, 2023] Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. Fingpt: Open-source financial large language models, 2023.

[Yang *et al.*, 2025] Shuo Yang, Xingwei Lin, Jiachi Chen, Qingyuan Zhong, Lei Xiao, Renke Huang, Yanlin Wang, and Zibin Zheng. Hyperion: Unveiling dapp inconsistencies using llm and dataflow-guided symbolic execution. In *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering*, pages 2125–2137, 2025.

[Yu *et al.*, 2025] Jiahao Yu, Xian Wu, Hao Liu, Wenbo Guo, and Xinyu Xing. Blockscan: Detecting anomalies in blockchain transactions. In *NeurIPS*, 2025.

[Zhang *et al.*, 2025a] Jie Zhang, Haoyu Bu, Hui Wen, Yongji Liu, Haiqiang Fei, Rongrong Xi, Lun Li, Yun Yang, Hongsong Zhu, and Dan Meng. When llms meet cybersecurity: a systematic literature review. *Cybersecur.*, 8(1):55, 2025.

[Zhang *et al.*, 2025b] Lyuye Zhang, Kaixuan Li, Kairan Sun, Daoyuan Wu, Ye Liu, Haoye Tian, and Yang Liu. Acf ix: Guiding llms with mined common rbac practices for context-aware repair of access control vulnerabilities in smart contracts. *IEEE Trans. Softw. Eng.*, 2025.

[Zhang *et al.*, 2026] Quanjun Zhang, Chunrong Fang, Yang Xie, Yaxin Zhang, Shengcheng Yu, Weisong Sun, Yun Yang, and Zhenyu Chen. A survey on large language models for software engineering. *Science China Information Sciences*, 69(4):141102, 2026.

[Zhou *et al.*, 2024] Chenyu Zhou, Hongzhou Chen, Hao Wu, Junyu Zhang, and Wei Cai. Artemis: Detecting airdrop hunters in nft markets with a graph learning system. In *WWW*, pages 1824–1834, 2024.