TxSum: User-Centered Ethereum Transaction Understanding with Micro-Level Semantic Grounding

Anonymous ACL submission

Abstract

Understanding the economic intent of Ethereum transactions is critical for user safety, yet current tools expose only raw on-chain data or surface-level intent, leading to widespread "blind signing" (approving transactions without understanding them). Through interviews with 16 Web3 users, we find that effective explanations should be structured, risk-aware, and grounded at the token-flow level. Motivated by these findings, we formulate *TxSum*, a new user-centered NLP task for Ethereum transaction understanding, and construct a dataset of 187 complex Ethereum transactions annotated with transaction-level summaries and token flow-level semantic labels. We further introduce *MATEX*, a grounded multi-agent framework for high-stakes transaction explanation. It selectively retrieves external knowledge under uncertainty and audits explanations against raw traces to improve token-flow-level factual consistency. *MATEX* achieves the strongest overall explanation quality, especially on micro-level factuality and intent quality. It improves user comprehension on complex transactions from 52.9% to 76.5% over the strongest baseline and raises malicious-transaction rejection from 36.0% to 88.0%, while maintaining a low false-rejection rate on benign transactions.

1 Introduction

Understanding the economic intent of an Ethereum transaction is crucial for both proactive risk mitigation and post-incident analysis. It enables users to preview outcomes and avoid irreversible errors before signing, and supports regulatory compliance, anti-money laundering, and forensic diagnosis after exploits—such as the TraceLLM (Wang et al., 2025) case, where on-chain analysis uncovered attack vectors in major DeFi incidents.

However, this understanding is critically lacking in practice. Modern DeFi transactions are highly compositional, leading to widespread "blind signing" (Liu et al., 2024; Kauri, 2023)—a phenomenon with severe consequences. In early 2025, attackers stole $1.5B from Bybit by tricking operators into approving a malicious transaction that appeared innocuous in their interface (BlockSec, 2025a). Even in routine interactions, users struggle: new users are confused by dual-sign patterns (approve + swap), while experienced users rely on fragile heuristics like "reject unlimited approvals," which fail in multi-contract scenarios.

These incidents reveal a fundamental gap: current tools expose raw on-chain data but fail to **translate it into human-understandable economic narratives**. While tools like MetaSuites (BlockSec, 2025b), EigenPhi (EigenPhi, 2025), and Tenderly (Tenderly, 2025) expose detailed token flows and contract interactions, they stop at the syntactic level. They do not interpret what these low-level events mean economically (Design, 2023).

As Figure 1 shows, a user's simple intent (e.g., "swap ETH for DAI") is often executed as a fragmented, multi-contract transaction. Raw traces reveal *what* happened but not *why*—leaving users unable to distinguish a routine swap from a high-risk operation. This gap between user mental models and on-chain reality motivates our research questions (RQs):

- **RQ1:** How do Web3 users understand transaction details when signing?
- **RQ2:** What do users expect from transaction explanations in terms of content and format?
- **RQ3:** How to design a framework and how the generated explanations by it affect users' comprehension and signing behavior?

To address this gap, we conducted semi-structured interviews with 16 Web3 users spanning traders, developers, and casual users. Three findings directly shaped our formulation: (1) blind signing is widespread even among experienced users;

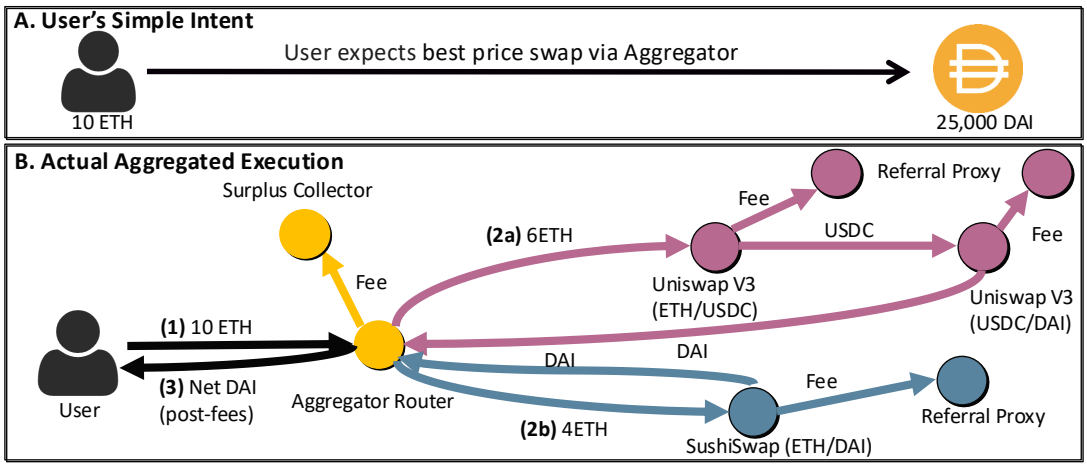

Figure 1: Mismatch between user intent and on-chain execution. A simple swap intent is realized through a complex and opaque transaction trace. The example is simplified for clarity.

(2) existing tools expose syntactic traces but not user-facing economic semantics; and (3) users naturally reason at the token-flow level while still expecting a concise transaction-level summary. Full study details are provided in Appendix Section A.

Guided by these findings, we formulate the paper around one central thesis: we introduce a new user-centered NLP task for Ethereum transaction understanding and a grounded method for improving micro-level factual explanation under high-stakes reasoning. As illustrated in Figure 2, our study proceeds from user interviews, to task and dataset construction, to grounded explanation modeling, and finally to multi-faceted evaluation. Concretely, we propose *TxSum*, a task and schema for transaction understanding at the token-flow level. *TxSum* defines five semantic attributes (see Section 2) that ground explanations in step-wise, verifiable actions aligned with user mental models. We then construct a dataset of 187 complex Ethereum transactions, each annotated with (1) a 3–4 sentence natural-language summary and (2) fine-grained flow-level labels following our predefined schema.

We further introduce *MATEX*, a grounded multi-agent framework motivated by how human experts analyze transactions. Rather than using unrestricted tool-augmented generation, *MATEX* implements a dual-process reasoning workflow (Booch et al., 2021; Hasan, 2021): fast pattern recognition (System 1) flags uncertainty and triggers targeted evidence-seeking investigation (System 2), followed by adversarial factual auditing against raw traces. The design explicitly targets the core NLP challenge in this setting: generating faithful, micro-level explanations for complex DeFi traces without hallucinating unseen protocol behavior.

We evaluate *MATEX* through automatic, expert, and user studies. Results show that *MATEX* consistently outperforms strong baselines—including monolithic prompting, VLM prompting with flow visualization, a single-agent tool user, Few-shot CoT, and a RAG pipeline—in factuality, intent quality, and risk coverage (see Section 4). In summary, we make the following contributions:

- We identify user-centered explanatory needs through semi-structured interviews with 16 diverse Web3 participants and define a structured schema for transaction understanding grounded in real usage.
- We introduce *TxSum*, a new user-centered NLP task and dataset for Ethereum transaction summarization and explanation, pairing natural-language summaries with step-wise semantic labels across 187 complex, real-world transactions.
- We propose *MATEX*, a grounded multi-agent framework that combines uncertainty-triggered retrieval, evidence synthesis, and adversarial factual auditing to generate accurate, risk-aware explanations from raw on-chain data, significantly improving user comprehension and reducing "blind signing" across automatic, expert, and user evaluations.

2 Understanding Transaction Flow

Our user interviews directly motivated *TxSum*'s schema. While a concurrent dataset (Mao et al., 2025) offers only coarse-grained intent classification, *TxSum* targets user-centered transaction understanding at two levels: (1) a concise transac-

tion summary that explains the overall intent and outcome, and (2) token-flow annotations that capture step-wise economic actions. This formulation shifts the problem from coarse transaction labeling to grounded micro-level explanation, which better matches how users reason about transaction safety.

Task Definition. *TxSum* is the task of generating a faithful explanation of a DeFi transaction from strictly on-chain observable data. The input consists of four components: (1) transaction metadata (e.g., sender, value, timestamp), (2) the full call trace, (3) a sequence of token transfers (ERC-20/721), and (4) net balance changes for all involved accounts. The output comprises two parts: (1) A transaction-level summary: a 3-4 sentence natural-language narrative that integrates the economic purpose, execution pathway, and outcomes of the transaction, including critical non-token state changes such as approvals, ownership transfers, or admin control updates. (2) Token-flow annotations: for each token transfer, five semantic attributes—action_type, intent, mechanism, precondition, and result—that ground the summary in step-wise, verifiable actions. The full specification of the five attributes is detailed in Section 2.1.

2.1 Identified Attributes

User interviews reveal that effective explanations must articulate the economic logic behind each token flow. To formalize this, we define five semantic attributes for every flow:

1. action_type, a standardized label for one of 14 financial primitives (e.g., *redeem*, *deposit*); defined in Section B Table 5.
2. intent, the user's high-level purpose (e.g., "swap out USDT for USDC");
3. mechanism, the technical pathway (which contract and function were called, e.g. "UniswapV3#exactInput");
4. precondition, the on-chain conditions required for success (e.g., USDC approved and balance > 10); and
5. result, the factual state change observed after execution (e.g., +1500 USDC balance).

Together, these attributes turn raw transaction traces into explanations that are both easy to follow and explicit about risks, and that are aligned with user mental models. Detailed definitions and examples are provided in Section B.

2.2 TxSum Dataset

Data Collection. To build a dataset that reflects real-world DeFi complexity while remaining feasible for expert annotation, we first sampled 500 Ethereum transactions from a full node over 2025. Specifically, we sampled 100 transactions from each of five expert-defined semantic categories: swaps (including multi-hop and aggregator-routed), lending/borrowing cycles, staking and delegation, liquidity pool operations, and compositional or cross-protocol interactions (e.g., flash loans, arbitrage, or custom logic not fitting the other categories). Because many real-world DeFi transactions exhibit multiple behaviors simultaneously, these categories were used only for sampling and later profiling, rather than as mutually exclusive semantic labels within the task itself. For reporting purposes, each retained transaction was assigned a primary category based on its dominant economic purpose, as determined by expert annotators.

We then filtered the 500 sampled transactions by token-flow length. We removed transactions with fewer than 5 token flows because they were usually simple transfers or otherwise too easy for step-wise semantic annotation. We also removed transactions with more than 30 token flows because they were too costly to annotate reliably. After filtering, the final *TxSum* dataset contains 187 transactions. For each transaction, we collect transaction metadata and the full call trace from an Ethereum archival node, ordered token transfers and net balance changes from EigenPhi (EigenPhi, 2025), and the token-flow graph from EigenPhi.

Data Annotation. The 187 transactions were annotated by six graduate-level annotators whose research focuses on DeFi. Annotators were organized into three fixed pairs, and each pair independently annotated a disjoint subset of 62 or 63 transactions. For every transaction, the two annotators in the assigned pair independently completed the full annotation, including token-flow labels and the transaction-level summary. Any disagreements were then resolved by a third annotator. Our annotation pipeline mirrors the cognitive process users described in interviews: (1) understand token movement, (2) infer purpose and mechanism, and (3) synthesize a coherent narrative. The process consists of three steps:

Step 1: Transaction Overview. Annotators first review all collected data to form a high-level understanding of the transaction's purpose. They write a

coarse-grained summary (not included in the final dataset) to guide subsequent labeling.

Step 2: Semantic Labeling. For each token transfer, annotators label five attributes derived from user interviews: action_type, intent, mechanism, precondition, and result.

Step 3: Narrative Synthesis. Annotators write a 3-4 sentence natural-language summary that strictly reflects observable evidence, without invoking external knowledge, protocol reputations, or speculative intent. Detailed annotation process and examples are provided in Section C.

Data Statistics. The final dataset contains 187 annotated transactions with an average of 12.7 token-flow steps per transaction, focusing on compositionally complex DeFi interactions rather than routine transfers or short single-step executions. The resulting summaries average 3.3 sentences (101.3 tokens). Based on a heuristic of protocol and control-flow complexity, we categorize 43 transactions as simple (single protocol), 90 as medium (2–3 protocols), and 54 as complex (≥ 4 protocols or nested logic). Full statistics are shown in Section C.2.

To assess annotation consistency at the token-flow level, we evaluate the agreement of annotators. For action_type, we compute inter-annotator agreement using Cohen's κ, yielding action_type = 0.98. For the natural language fields intent, mechanism, precondition, and result, exact-form agreement would underestimate reliability because annotators may produce semantically equivalent paraphrases. We therefore report pairwise semantic agreement using BERTScore F1, computed between the two independently written annotations for each token flow and averaged over the dataset. Specifically, for each token flow, we compute BERTScore F1 between the two independently written annotations using the official BERTScore (Zhang et al., 2020) implementation with a roberta-large backbone and baseline rescaling enabled, and then average the resulting scores over the dataset; all four fields score above 0.91 (see details in Table 6). These results indicate high consistency across attributes, supporting the reliability of the dataset.

3 Multi-Agent Framework: *MATEX*

To address **RQ3** (how to explain it), we start from how human experts analyze complex transactions. They do not rely on memorized patterns alone; instead, they follow a clear workflow: (1) quickly scan token flows for familiar patterns, (2) stop when something is unclear, (3) look up documentation or code, (4) synthesize the evidence into an explanation, and (5) check again for hidden risks before concluding (Hasan, 2021).

The key decision is when to move from System 1 to System 2. Most token flows are routine and can be interpreted from on-chain evidence alone. Only a small number are ambiguous and need extra protocol context. For this reason, *MATEX* uses Dynamic Knowledge Injection Protocol (DKIP). The Profiler computes a semantic uncertainty score u_i for each token flow t_i. If u_i is low, the system keeps a lightweight explanation path. If u_i is above a threshold τ, the Investigator retrieves protocol evidence such as ABI entries or verified documentation. The Profiler also emits a retrieval key when escalation is needed. This design avoids unnecessary retrieval and keeps the context focused on uncertain flows.

After retrieval, System 2 produces and checks the explanation. The Synthesizer combines on-chain evidence, retrieved protocol context, and flow-level hypotheses to produce a draft summary. The Safety Auditor then checks this draft against the raw trace. Here *MATEX* uses Auditor-Guided Context Pruning (AGCP). The intuition is that unsupported claims can remain in the working context and affect later drafts. Instead of only appending critique, AGCP identifies the error-causing context P_{mask} and removes it before the next refinement step. This reduces repeated hallucinations across iterations. The full pseudocode of DKIP and AGCP are provided in Appendix Section D.1.

- **Profiler** implements System 1. It scans token flows and computes a semantic uncertainty score u_i for each token flow t_i.
- **Investigator** supports System 2 with retrieving live protocol information only when u_i is above the threshold τ, following DKIP.
- **Synthesizer** is the main reasoning module in System 2, which combines on-chain evidence, retrieved protocol context, and flow-level hypotheses to produce a transaction summary.
- **Safety Auditor** checks the draft explanation against the raw trace and applies AGCP when it finds unsupported facts or missing risks.

Together, these components realize the full workflow of *MATEX*: System 1 decides whether a flow can be handled directly or should be escalated, DKIP controls what evidence enters System 2, the Synthesizer turns that evidence into a flow-

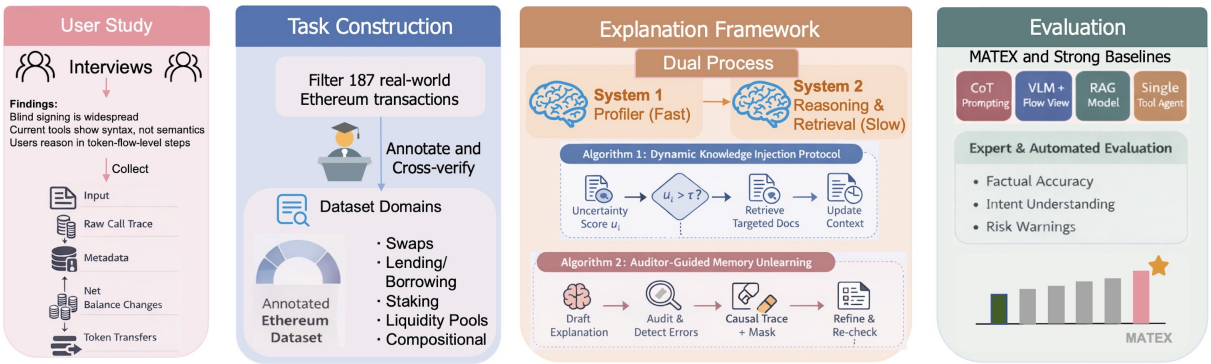

Figure 2: Overview of our study. We first conduct user interviews to identify the explanatory needs behind blind signing, which motivate *TxSum*, a new user-centered natural langauge task and dataset for Ethereum transaction understanding. We then introduce *MATEX*, a grounded multi-agent method for this task, and evaluate it against strong baselines using automatic, expert, and user-centered assessments.

grounded explanation, and AGCP removes unsupported reasoning before the next draft. This design supports our main goal: grounded micro-level factual explanation rather than only fluent transaction narration. In our implementation, all agents use Qwen3-Max as the backbone LLM and are orchestrated with LangGraph (LangChain, 2025) to support stateful and iterative interactions. The agents share structured tools for on-chain data access and protocol document retrieval. Full implementation details are provided in Section D.

4 Evaluations and Results

Our interviews reveal that existing interfaces fail to support safe DeFi interaction: users sign transactions they don't understand, often missing latent risks. To address **RQ3** (user comprehension and signing behavior), we give 3 evaluation objectives (EOs) that map directly to user-identified needs:

- **EO1: Explanation Quality.** We evaluate the accuracy, intent coverage, outcome correctness, and clarity of generated summaries using automatic metrics and expert annotation.
- **EO2: User Comprehension.** We measure users' understanding, trust, and clarity via a controlled user study.
- **EO3: Risk Awareness.** We evaluate whether users can identify real-world security threats with the help of *MATEX*.

4.1 Baselines

We compare *MATEX* against five representative baselines: **(1) Monolithic LLM.** An LLM that processes the full input (trace, metadata) in one pass. **(2) Few-shot CoT.** A prompting baseline using the same Qwen3-Max backbone with 2 in-context demonstrations and an explicit chain-of-thought reasoning template. **(3) VLM.** A VLM augmented with a token-flow graph. **(4) RAG.** A retrieval baseline that extracts contract addresses, function signatures, and protocol names from the trace, retrieves external documents with a fixed workflow, and appends the retrieved evidence directly to the model. **(5) Single-Agent.** A ReAct-style agent using Qwen3-Max as its backbone, with access to the same web search and Etherscan tools as *MATEX*.

To ensure a fair comparison, all text-only baselines (*MATEX*, Monolithic LLM, Single-Agent, Few-shot CoT, Pure RAG) use the same Qwen3-Max (Team, 2025a) backbone, while the visualization baseline uses Qwen3-VL-Plus (Report, 2025) due to its multimodal input requirement. All retrieval-enabled systems are granted the same retrieval permissions, tool budget, and temporal alignment: the same Etherscan API access, the same web-search scope, the same maximum number of external retrieval calls, and the same transaction-time protocol information window. Single-Agent can call the same web search and Etherscan tools as *MATEX*, while Pure RAG pipeline uses the same retrieval sources but without role specialization, iterative auditing, or uncertainty-triggered control. This setup isolates the contribution of *MATEX*'s selective retrieval and auditing framework rather than confounding it with information asymmetry.

All systems receive identical on-chain inputs and output the result for the whole transaction. We assess EO1 via automatic metrics and expert annotation, EO2 through a controlled user study, and EO3 using a user study on real attack transactions (see metrics in Section 4.2). Full evaluation details

are provided in Section E.

4.2 Evaluation Details and Results

4.2.1 EO1: Explanation Quality

We assess transaction summaries using 4 binary (pass/fail) criteria: **T1** (outcome correctness) and **T2** (clarity & conciseness) at the transaction level, and **F1** (faithfulness) and **F2** (intent quality) at the token-flow level. (see Appendix Table 12)

We use two evaluation methods: (1) **LLM-as-judge**: GPT-5.1 (OpenAI, 2025) scores all system outputs with binary decisions for each metric, and we explicitly validate the judge against expert labels; (2) **Expert evaluation**: 3 DeFi Ph.D. researchers rate all transactions, assessing factual correctness, intent alignment, and risk awareness. The resulting pass rates from both evaluation sources are reported in Table 1.

For binary judgments, we additionally report ROUGE-L and BLEU for transaction-level summaries, Exact Match (EM) for flow-level semantic attribute generation, and Accuracy / Macro-F1 for the 14-way action_type classification subtask. As shown in Section E.1.1, *MATEX* achieves the best performance on both summary-generation metrics (ROUGE-L, BLEU) and structured prediction metrics (EM, Accuracy, Macro-F1), consistent with the pass-rate results in Table 1.

Judge Reliability and Significance. We validate the GPT-5.1 judge against expert annotations and find strong agreement overall, with detailed accuracy, per-metric Cohen's κ, confusion-matrix analysis, and verbosity-bias checks reported in Section E.1.2. Expert inter-rater reliability is also high (Fleiss' $\kappa = 0.76$), supporting the reliability of the human evaluation protocol. To assess robustness, we compare *MATEX* against the strongest baseline using paired significance tests on the same 187 transactions: McNemar tests for binary pass/fail metrics and paired tests for ROUGE-L, EM, and action_type Macro-F1. The results confirm statistically significant gains on the key micro-level factuality and intent metrics (see Section E.1.2).

Results. Results from both LLM-as-judge and expert human evaluation are summarized in Table 1, while standard NLP metrics are reported in Section E.1.1 and table 10. *MATEX* achieves the highest scores on F1 and F2 across both evaluators, demonstrating its strength in grounded, step-wise reasoning. On T1, *MATEX* performs on par with the best baselines—scoring 76.0% in human evaluation, the highest. We manually inspected T1 failures and found that *MATEX* occasionally omits minor fees or implicit approvals when they are not explicitly traceable in the call trace, adhering strictly to its principle of only reporting verifiable on-chain facts.

The standard NLP metrics reinforce the same pattern: *MATEX* achieves the best ROUGE-L, BLEU, Exact Match, Accuracy, and Macro-F1 under matched retrieval conditions. This indicates that the gains are not confined to judge-based or human pass/fail assessment, but also transfer to standard generation and classification metrics.

The performance patterns align with the architectural limitations of each baseline. Both the Monolithic LLM and the VLM (augmented with a token-flow graph) process the input in a single pass without explicit decomposition or verification, making them prone to hallucination or missed details—especially on minor but critical flows such as small fees. Few-shot CoT improves local reasoning structure but still lacks grounded verification. The Pure RAG benefits from external evidence, yet its fixed retrieval-and-append workflow cannot selectively investigate ambiguous flows or audit generated claims against the raw trace. The Single-Agent integrates decomposition, retrieval, synthesis, and narration into one cognitive loop, lacking role specialization or adversarial validation, which leads to error propagation in complex transactions.

4.2.2 EO2: User Trust & Comprehension

We conduct a separate controlled user study with a new cohort of 20 active DeFi users, distinct from the 16 interview participants (full details are provided in Section E.2). Participants were recruited via Web3 communities and professional networks, and all reported at least three on-chain DeFi transactions in the past six months.

Each participant views 9 transactions in total. For each complexity levels (simple, medium, and complex), they see one transaction under each of the following conditions using the EigenPhi explorer interface: (a) Raw: the default explorer view only; (b) Single-Agent: explorer view augmented with annotations from the strongest baseline in EO1; (c) *MATEX*: explorer view augmented with *MATEX* explanations. Transaction-condition assignment is balanced across participants using a Latin Square design (Section E.2).

Results. Table 2 summarizes the EO2 results. *MATEX* consistently improves both objective com-

Method	Pass Rate (%, LLM-as-judge)				Pass Rate (%, Human)			
	T1	T2	F1	F2	T1	T2	F1	F2
LLM	79.1	80.7	51.2	64.3	71.2	82.1	53.4	65.7
Few-Shot CoT	74.9	86.1	62.5	63.9	73.9	85.2	62.1	64.8
VLM	77.0	82.9	60.8	61.2	75.8	86.0	61.4	62.8
RAG	78.1	81.8	52.0	64.6	72.1	82.0	53.9	65.8
Single-Agent	66.8	89.8	67.6	67.5	70.1	88.1	69.0	69.2
MATEX	**73.3**	**93.0**	**73.2**	**76.6**	**76.0**	**94.0**	**75.1**	**78.3**

Table 1: EO1 pass rates (%): LLM-as-judge and expert human evaluation on baselines and *MATEX*.

prehension and subjective trust, with the largest gains on complex transactions. In particular, comprehension accuracy on complex transactions rises from 35.3% in the Raw condition and 52.9% in the Single-Agent condition to 76.5% with *MATEX*. Perceived trust and clarity show a similar trend, increasing from 1.8–2.6 in the comparison conditions to 3.7–4.5 with *MATEX*.

the improvements suggest *MATEX* not only improves transaction understanding, but also supports safer signing decisions. The advantage of *MATEX* is especially pronounced on medium and high complexity transactions, where users benefit most from grounded, step-wise explanations. Detailed statistical comparisons are reported in Section E.2.

Cond.	Accuracy (%)			Trust and Clarity		
	S	M	C	S	M	C
Raw	88.2	64.7	35.3	3.2	2.5	1.8
Agent	94.1	76.5	52.9	3.8	3.2	2.6
MATEX	**98.0**	**88.2**	**76.5**	**4.5**	**4.1**	**3.7**

Table 2: EO2 results ($N = 20$): Comprehension accuracy (%) and perceived trust/clarity (1–5) across transaction complexity (S=Simple, M=Medium, C=Complex).

4.2.3 EO3: Risk Awareness

We evaluate whether *MATEX* helps users recognize security threats in real transactions. We select 5 malicious transactions from real-world DeFi exploits (detailed in Section E.3) and 5 benign transactions of comparable complexity as controls.

We conduct a between-subjects study using the same 20-participant evaluation cohort as EO2, split into two groups of 10. The first group views all 10 transactions in a standard block explorer (Raw). The second group views the same transactions with *MATEX* explanations on the same interface.

For each transaction, participants report their signing intention (Yes/No) and describe any perceived risks in an open-ended response. We measure the rejection rate on malicious transactions, the proportion of users who correctly identify the core threat, and the false-rejection rate on benign transactions. Risk recognition is evaluated by two independent researchers using manual semantic coding guided by core threat concepts.

Results. *MATEX* substantially improves users' ability to identify security risks without increasing false alarms. On malicious transactions, the rejection rate increases from 36% (Raw) to 88% (*MATEX*), and the rate of correctly identifying the underlying threat rises from 28% to 72%. On benign transactions, the false-rejection rate remains low and stable: 12% in the Raw condition versus 8% in the *MATEX* condition.

Qualitative analysis of open-ended responses confirms this shift: users with *MATEX* provide specific, actionable risk descriptions (e.g., "unlimited approval granted to 0x...789"), while Raw users often offer vague judgments (e.g., "looks sketchy") or miss threats entirely. This demonstrates that *MATEX* enables non-experts to make more informed, security-aware signing decisions. Using Fisher's exact tests, the increase in malicious-transaction rejection is statistically significant ($p =< 0.001$), and the increase in correct threat identification is also significant ($p = 0.002$). In contrast, the difference in false-rejection rates on benign transactions is not significant ($p = 0.741$).

4.3 Ablation Study

We then explore which components are responsible for these gains. We perform ablations to assess the contribution of each major *MATEX* component by disabling one component at a time while keeping the rest of the pipeline unchanged, using the same inputs and tool access as the full system. Detailed results are reported in Appendix E.3.1.

The ablation results show that all four components contribute meaningfully, but in different ways

(full results in Appendix E.3.1). Removing the Profiler causes a modest drop in flow-level factuality (F1: 73.2→70.5), suggesting that uncertainty control mainly helps suppress local hallucinations on ambiguous flows. Removing the Investigator leads to a much larger degradation (F1: 73.2→62.8; action_type Macro-F1: 68.5→60.4), confirming that targeted protocol retrieval is essential for resolving unseen contracts and protocol-specific mechanisms. Without the Synthesizer, both explanation quality and summary coherence collapse (T2: 93.0→64.2; ROUGE-L: 52.6→41.8), showing that evidence fusion is necessary to turn step-wise observations into coherent user-facing narratives. Finally, removing the Safety Auditor leaves macro-level outcome correctness relatively stable (T1: 73.3→72.2) but sharply reduces micro-level faithfulness (F1: 73.2→62.5), indicating that adversarial trace-based checking is especially important for preserving intermediate factual completeness rather than only the bottom-line outcome.

5 Related Work

LLM-Driven Transaction Analysis and Auditing. Recent benchmarks such as XFinBench (Zhang et al., 2025) and surveys (Chung et al., 2025) highlight growing interest in applying LLMs to complex financial reasoning (e.g., forecasting, scenario planning). Early Ethereum analysis relied on static analysis and symbolic execution (Torres et al., 2018; Tsankov et al., 2018). These methods offer precision but cannot infer high-level user intent.

Recent work leverages Large Language Models or agents for smart contract auditing (Li et al., 2025b; Ma et al., 2025), bug detection (Li et al., 2025a; Wei et al., 2025), and gas optimization (Zheng et al., 2025). However, these approaches focus on code-level properties, which are distinct from explaining the runtime economic behavior of a specific transaction.

Human-Centric Web3 Security and Interfaces. Web3 security is fundamentally a human–computer interaction problem: users often sign transactions "blindly" because current interfaces expose only opaque calldata and low-level traces (Academy, 2025; Team, 2025b). Tools such as PrettiSmart and PonziLens+ improve trace visibility, but they rarely provide user-facing economic semantics such as hidden fees, slippage, or cross-protocol risk (Wen et al., 2025, 2024).

Recent LLM-based approaches attempt to bridge this gap by classifying user intent from traces (Mothukuri et al., 2024; Mao et al., 2025). In particular, the concurrent work of Mao et al. (2025) is closely related to ours, but it addresses a different problem formulation. Their Transaction Intent Mining (TIM) framework models DeFi understanding primarily as a macro-level intent mining task over a predefined intent taxonomy, with the goal of assigning one or more coarse transaction-level intent labels. By contrast, *TxSum* is a user-centered transaction-understanding task based on our findings in the interviews: it requires not only a transaction-level natural-language explanation, but also micro-level token-flow semantic parsing over five attributes. As a result, the two settings differ substantially in both granularity and output space.

This difference is important in complex DeFi settings: a system may infer a coarse label such as *swap* or *arbitrage* while still failing to explain intermediate fees, approvals, routing, or other risk-bearing state changes relevant to blind-signing decisions. Consistent with this task formulation, *MATEX* is designed for grounded explanation rather than only intent prediction: it selectively retrieves protocol context when on-chain evidence is insufficient and validates generated claims against raw traces, producing step-wise, risk-aware explanations for complex DeFi transactions.

6 Conclusion

Through formative interviews, we find that safe transaction understanding in DeFi requires explanations that are structured, risk-aware, and grounded at the token-flow level, rather than raw traces or coarse intent labels alone. Motivated by this, we formulate *TxSum*, a new user-centered NLP task and dataset for transaction understanding that combines transaction-level natural-language explanation with micro-level token-flow semantic parsing.

We then introduce *MATEX*, a grounded multi-agent framework that combines uncertainty-triggered retrieval, evidence synthesis, and adversarial factual auditing to generate faithful explanations of complex DeFi transactions. Across automatic evaluation, expert assessment, and user studies, *MATEX* consistently outperforms baselines, with especially strong gains on complex multi-protocol transactions and real-world risk recognition. We hope this work provides a useful foundation for future research on user-centered, grounded language for high-stakes transaction understanding.

Limitations

At the same time, several limitations remain. The *TxSum* dataset covers 187 Ethereum transactions from 2025 and focuses on single-chain Ethereum mainnet. Our expert annotation protocol operates at the token-flow level and is therefore costly, which limits dataset scale. While our user study includes participants with diverse backgrounds, it captures signing behavior in a controlled setting rather than real-world high-pressure scenarios. We hope *MATEX* encourages future work on faithful, user-centered language interfaces.

Ethical Considerations

This study was reviewed and approved by the Institutional Review Board (IRB) of our institution. All interview participants provided informed consent prior to participation and were informed of their right to withdraw at any time. To protect privacy, all collected data—including interview transcripts, demographic information, and transaction logs—were anonymized and stored on encrypted, access-controlled servers. No personally identifiable information is included in the published dataset or supplementary materials. The *TxSum* dataset contains only public on-chain transactions; no private user data or wallet secrets were collected or exposed. We also note a dual-use concern. A system that helps users or analysts understand suspicious transactions more clearly could, in principle, also help attackers test whether malicious behavior is less obvious under current interfaces. However, our primary goal is defensive: improving user awareness, supporting security analysis, and reducing blind signing in high-risk DeFi environments. We therefore present *MATEX* as an assistive explanation tool rather than an automated execution or attack-planning system.

Open Science

Our code of *MATEX* is available at an anonymized repository.[1] To strengthen reproducibility, the repository includes the dataset, full multi-agent orchestration logic, system prompts, tool invocation pipeline, and evaluation scripts used in our experiments. We also release the annotation schema, detailed guidelines, and example cases described in the appendix to facilitate re-implementation and further benchmarking.

[1] https://anonymous.4open.science/r/matex-4373/

References

OSL Academy. 2025. Are defi protocols difficult to use?

BlockSec. 2025a. Bybit $1.5b hack: In-depth analysis of the malicious safe wallet upgrade attack.

BlockSec. 2025b. Metasuites, the swiss army knife for builders.

Grady Booch, Francesco Fabiano, Lior Horesh, Kiran Kate, Jonathan Lenchner, Nick Linck, Andreas Loreggia, Keerthiram Murgesan, Nicholas Mattei, Francesca Rossi, and 1 others. 2021. Thinking fast and slow in ai. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 15042–15046.

Richard E Boyatzis. 1998. *Transforming qualitative information: Thematic analysis and code development*. Sage.

Yoonseo Chung, Jeonghyun Kim, MiYeon Kim, Minsuh Joo, and Hyunsoo Cho. 2025. *Foundations of LLMs and Financial Applications*. Springer Nature Singapore, Singapore.

Design. 2023. Why poor ux is blocking mass adoption of web3.

EigenPhi. 2025. Eigenphi.

Ahmed Rizvan Hasan. 2021. Artificial intelligence (ai) in accounting & auditing: A literature review. *Open Journal of Business and Management*, 10(1):440–465.

Isaac Kauri. 2023. Blind signing in web3: Challenges and user strategies. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*.

LangChain. 2025. Langgraph.

Xiangyu Li, Yawen Zeng, Xiaofen Xing, Jin Xu, and Xiangmin Xu. 2025a. Hedgeagents: A balanced-aware multi-agent financial trading system. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 296–305.

Zongwei Li, Xiaoqi Li, Wenkai Li, and Xin Wang. 2025b. Scalm: Detecting bad practices in smart contracts through llms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 470–477.

Mingyi Liu, Jun Ho Huh, HyungSeok Han, Jaehyuk Lee, Jihae Ahn, Frank Li, Hyoungshick Kim, and Taesoo Kim. 2024. I experienced more than 10 DeFi scams: On DeFi users' perception of security breaches and countermeasures. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 6039–6055, Philadelphia, PA. USENIX Association.

Wei Ma, Daoyuan Wu, Yuqiang Sun, Tianwen Wang, Shangqing Liu, Jian Zhang, Yue Xue, and Yang Liu. 2025. Combining fine-tuning and llm-based agents for intuitive smart contract auditing with justifications. In *47th IEEE/ACM International Conference on Software Engineering, ICSE 2025, Ottawa, ON, Canada, April 26 - May 6, 2025*, pages 1742–1754. IEEE.

Qian'ang Mao, Yuxuan Zhang, Jiaman Chen, Wenjun Zhou, and Jiaqi Yan. 2025. Know your intent: An autonomous multi-perspective llm agent framework for defi user transaction intent mining. *arXiv preprint arXiv:2511.15456*.

Viraaji Mothukuri, Reza M Parizi, James L Massa, and Abbas Yazdinejad. 2024. An ai multi-model approach to defi project trust scoring and security. In *2024 IEEE International Conference on Blockchain (Blockchain)*, pages 19–28. IEEE.

OpenAI. 2025. Gpt-5.1: A smarter, more conversational chatgpt.

Bob Price. 2002. Laddered questions and qualitative data research interviews. *Journal of Advanced Nursing*, 37(3):273–281.

Qwen3-VL Technical Report. 2025. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

Qwen Team. 2025a. Qwen3 technical report. *arXiv*.

Token Metrics Team. 2025b. Why web3 ux is poor compared to web2 and what that means for mass adoption.

Tenderly. 2025. Tenderly.

Christof Ferreira Torres, Julian Schütte, and Radu State. 2018. Osiris: Hunting for integer bugs in ethereum smart contracts. In *Proceedings of the 34th Annual Computer Security Applications Conference*, ACSAC '18, page 664–676, New York, NY, USA. Association for Computing Machinery.

Petar Tsankov, Andrei Dan, Dana Drachsler-Cohen, Arthur Gervais, Florian Buenzli, and Martin Vechev. 2018. Securify: Practical security analysis of smart contracts. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 67–82.

Shuzheng Wang, Yue Huang, Zhuoer Xu, Yuming Huang, and Jing Tang. 2025. Tracellm: Security diagnosis through traces and smart contracts in ethereum. *Preprint*, arXiv:2509.03037.

Zhiyuan Wei, Jing Sun, Yuqiang Sun, Ye Liu, Daoyuan Wu, Zijian Zhang, Xianhao Zhang, Meng Li, Yang Liu, Chunmiao Li, and 1 others. 2025. Advanced smart contract vulnerability detection via llm-powered multi-agent systems. *IEEE Transactions on Software Engineering*.

Xiaolin Wen, Tai D Nguyen, Shaolun Ruan, Qiaomu Shen, Jun Sun, Feida Zhu, and Yong Wang. 2024. Ponzilens+: Visualizing bytecode actions for smart ponzi scheme identification. *IEEE Transactions on Visualization and Computer Graphics*.

Xiaolin Wen, Tai D Nguyen, Lun Zhang, Jun Sun, and Yong Wang. 2025. Prettismart: Visual interpretation of smart contracts via simulation. *IEEE Transactions on Visualization and Computer Graphics*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *The International Conference on Learning Representations (ICLR)*. OpenReview.net.

Zhihan Zhang, Yixin Cao, and Lizi Liao. 2025. XFinBench: Benchmarking LLMs in complex financial problem solving and reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8715–8758, Vienna, Austria. Association for Computational Linguistics.

Jingyi Zheng, Zifan Peng, Yule Liu, Junfeng Wang, Yifan Liao, Wenhan Dong, and Xinlei He. 2025. Gasagent: A multi-agent framework for automated gas optimization in smart contracts. *arXiv preprint arXiv:2507.15761*.

APPENDIX

A Interview Protocol Details

A.1 Interview Structure

Each 45–60 minute interview followed a semi-structured protocol with three phases, conducted via Zoom or in person and audio-recorded with consent. All sessions were transcribed verbatim for analysis.

Phase 1: Background and Experience (5–10 mins). We began by establishing the participant's DeFi experience to contextualize their responses. Standardized prompts included:

1. **Usage History:** "How long have you used DeFi? Which protocols do you interact with most frequently (e.g., Uniswap, Aave, Lido)?"
2. **Recent Transaction:** "Describe your most recent on-chain transaction. What were you trying to achieve, and how did you decide whether to sign it?"
3. **Tooling:** "What tools do you use to inspect transactions before signing (e.g., Etherscan, Rabby, MetaMask)? What do you find helpful or frustrating?"

This phase helps us tailor subsequent questions and identify domain-specific heuristics.

Phase 2: Think-Aloud Task (20–25 mins). Participants analyzed four real Ethereum transactions (see Table 3), presented via a simplified Etherscan-like interface showing: sender, token flows, decoded function calls, and net balance changes (no raw bytecode).

For each transaction, participants were instructed: "Please think aloud as you try to understand this transaction. Explain in your own words: (1) what it does, (2) what risks you see, and (3) whether you would sign it—and why." Interviewers used non-leading prompts to sustain reflection:

- "Can you walk me through how you reached that conclusion?"
- "What part of the trace supports that interpretation?"
- "Is there anything you're unsure about?"

Phase 3: Laddered Questioning (15–20 mins). Building on Phase 2 responses, we employed laddering (Price, 2002) to surface mental models. For each key statement (e.g., "I wouldn't sign TX4"), we asked:

1. **Concrete Attribute:** "What specifically about TX4 made you uncomfortable?"
2. **Functional Consequence:** "What could go wrong if you did sign it?"
3. **Personal Value:** "Why is that outcome unacceptable to you? What does it mean for your sense of control or security?"

We repeated this chain until reaching a fundamental value (e.g., "autonomy," "transparency," "irreversibility"). Additionally, we probed explanatory expectations:

- "What information would have made TX2 easier to understand?"
- "Should the explanation focus on individual steps or the overall goal?"
- "Would a clear explanation change your signing decision? In what cases?"

These responses directly informed the design of the *TxSum* schema and *MATEX* evaluation (RQ3).

A.2 Interview Participant Demographics

A.3 Interview Analysis and Key Findings

All interviews were transcribed verbatim and analyzed using inductive thematic analysis (Boyatzis, 1998). Two researchers independently coded the first 20% of transcripts to develop an initial codebook, then iteratively refined it through discussion while coding the remaining data. Inter-coder agreement on the shared subset reached Cohen's $\kappa = 0.82$, and disagreements were resolved by consensus. Thematic saturation was reached by the 14th interview, with no new major themes emerging in the final two interviews.

Our formative interviews revealed three findings that directly motivated the formulation of *TxSum*.

Finding 1: Blind signing is widespread. Even experienced users often approve transactions they cannot fully interpret. Rather than understanding the complete transaction, they frequently rely on fragile heuristics such as "reject unlimited approvals" or "avoid unknown contracts," which break down in compositional DeFi transactions spanning multiple contracts and hidden intermediate steps.

ID	Type	Description
TX1	Multi-hop Swap	User swaps 10 ETH for DAI via 1inch aggregator, which routes through Uniswap V3 (ETH→USDC) and SushiSwap (USDC→DAI), with fees collected by intermediary contracts.
TX2	Flash Loan Exploit	Attacker takes Aave flash loan to artificially inflate liquidity on a mock pool, enabling a large swap that drains user funds.
TX3	Revolving Leverage	User borrows USDC from Aave, swaps to ETH on CowSwap, stakes ETH on Lido, then uses stETH as collateral to borrow more USDC—creating cyclic leverage.
TX4	Stake + Approval	User stakes ETH on Lido and grants an unlimited approval to Lido's stETH contract—standard but often misunderstood pattern.

Table 3: Transaction examples used in think-aloud exercises.

ID	Role	Gender	Country	Experience
1	Trader	M	China (HK)	5
2	Analyst	F	Singapore	4
3	Analyst	M	China	6
4	Trader	M	USA	4
5	Developer	M	USA	5
6	Developer	M	China (HK)	4
7	Developer	M	UK	3
8	Unemployed	M	China	3
9	Designer	F	China	2
10	Student	M	USA	2
11	Student	M	China (HK)	4
12	Student	F	Australia	3
13	Accountant	M	Canada	1
14	Officer	M	China	2
15	Engineer	F	Singapore	3
16	Student	M	South Korea	2

Table 4: Demographic summary of interview participants. Role, Experience (years with Web3), and Country (primary residence during Web3 use) are self-reported.

Finding 2: Current tools suffer from semantic poverty. Participants repeatedly reported that existing interfaces expose low-level artifacts such as token transfers, calldata, and decoded function names, but fail to explain what those events mean economically. As a result, users often cannot infer the actual purpose, risk, or preconditions of a transaction from syntactic traces alone.

Finding 3: Users reason at the token-flow level but still want concise summaries. Most participants consistently treated token flows as the natural unit of understanding, because each flow corresponds to a small economic action such as swapping, depositing, repaying, or fee deduction. At the same time, they also wanted a concise transaction-level summary that explains the overall intent and highlights critical risk-bearing effects such as approvals, ownership changes, or hidden fees.

B Identified Attributes

We define five semantic attributes for each token flow, grounded in user mental models and derived from our interview study.

- action_type: A protocol-agnostic label denoting the financial primitive of a token transfer (e.g., `swap_in`, `deposit`, `fee`). Classification is determined solely by on-chain signals: zero-address interactions, event logs (e.g., `Transfer`, `Mint`), and decoded function names. The taxonomy comprises 14 mutually exclusive types, designed to cover core DeFi operations while enabling cross-protocol consistency. Full definitions are provided in Table 5.
- intent: A user-centric statement of the economic purpose behind the step (e.g., "Redeem yield-bearing tokens to unlock liquidity"). This field bridges raw token movements and human-understandable goals, reflecting what the user is trying to achieve—not how the protocol implements it. Intent is inferred from the transaction context, protocol documentation, and the action's role in the broader sequence.
- mechanism: A precise description of the on-chain execution pathway, formatted as "Call `<Contract>.<function>()`" or "`<Contract>#<function>()`". It specifies which contract and function were invoked (e.g., "Call `PendleRouter.redeem()` to burn SY and withdraw sUSDe"), grounding the explanation in decoded calldata and event logs. This enables users to verify behavior against official documentation.
- precondition: The minimal set of on-chain conditions required for the action to succeed (e.g., "User holds sufficient SY balance and has approved the router"). Preconditions are derived from protocol documentation, function `require()` statements, and observable state (e.g.,

prior approvals, pool liquidity). Numerical thresholds are included only when explicitly visible in the trace or documentation; otherwise, the field uses qualitative descriptions.

- result: A factual account of the direct state changes caused by the action (e.g., "User receives 53.5 sUSDe; SY balance is burned in the PendleSUSDESY contract"). This field strictly reports what is observable in token transfers, event logs, or state diffs—never inferred profits, market impact, or psychological outcomes. It serves as the ground-truth anchor for the entire explanation.

All attributes are derived using only on-chain data and official protocol documentation. No external knowledge, speculative intent, or subjective interpretation is permitted. Detailed examples are provided in Section C.

C Annotation Guidelines, Consistency Rules, and Edge Cases

This section details our annotation protocol, including step-by-step guidelines, attribute definitions, and concrete examples. All annotators were required to have prior experience with DeFi protocols (e.g., Uniswap, Aave, Pendle) and familiarity with blockchain explorers (Etherscan, Tenderly, Phalcon). Annotation proceeds in three steps:

Step 1: Transaction Overview. Annotators first examine the transaction's high-level effects using block explorers—focusing on net token flows, called contracts, and state changes—to form a one-sentence coarse-grained summary (e.g., "Extract yield from a Pendle YT position"). This summary is not included in the final dataset but guides subsequent labeling.

Step 2: Token Flow Segmentation. Each row in the token transfer log is treated as a single *micro action* representing an atomic economic step.

Step 3: Attribute Labeling. For each micro action, annotators fill five fields using only on-chain evidence and official protocol documentation.

Annotation Consistency Rules. To ensure uniformity, annotators follow these conventions:

- Use "the user" for the transaction initiator, "the protocol" for contract addresses, and "burn address" for zero addresses.
- Report token amounts in native units (e.g., "53.5 sUSDe"), not USD equivalents.
- Avoid speculative intent (e.g., "to short the market"); focus only on directly inferable purposes.
- Every intent and result must align with the corresponding token transfer or event log.

Edge Cases and Disambiguation Guidelines. Because real DeFi transactions are highly compositional, the main annotation difficulty is not identifying the final outcome, but disambiguating intermediate steps that are easy to overlook yet critical for faithful explanation and user safety. We therefore define explicit decision rules for common but consequential edge cases:

- **Protocol-internal routing vs. user-facing economic action.** Transfers among routers, adapters, settlers, vault helpers, or executor contracts are labeled as `route` when they merely relay assets for execution. They are labeled with a semantic action (e.g., `swap_out`, `deposit`, `repay`) only if the transfer itself realizes an economically meaningful state transition for the user. This distinction is crucial in aggregator-routed and cross-protocol transactions, where many intermediate hops are operationally necessary but not independently meaningful.
- **Material intermediate effects vs. ignorable plumbing.** Annotators do not collapse away intermediate transfers when they carry economic or security significance. In particular, protocol fees, relayer fees, approval side effects, collateral movements, liquidation repayments, or transfers into newly introduced contracts must remain explicit in the annotation even if the final net outcome appears simple. This rule is designed to support blind-signing scenarios, where omitted intermediate effects can hide the true risk of a transaction.
- **Approval and permission changes.** Pure approval events do not create token-flow labels unless accompanied by an actual token transfer, but they must be reflected in the transaction-level summary whenever they are economically or security relevant. Unlimited approvals, newly granted permissions to unfamiliar contracts, and approval-reset patterns may also be encoded in precondition when they are necessary for interpreting subsequent token flows.
- **Wrapped, unwrapped, and derivative assets.** Conversions such as ETH $\leftrightarrow$ WETH or underlying asset $\leftrightarrow$ receipt/derivative token are annotated by their economically salient primitive rather than surface transfer form. For example, receiving stETH after supplying ETH is not treated

Action Type	Definition
`mint`	Creation of new tokens issued from the zero address to a recipient (i.e., newly minted assets).
`burn`	Permanent destruction of tokens by transferring them to the zero address.
`redeem`	Conversion of derivative or wrapped assets into their underlying assets (e.g., SY → sUSDe, LP → underlying legs, WETH → ETH), where the derivative token is returned to a contract and the underlying asset is released.
`issue`	Issuance of derivative or receipt assets from underlying assets (e.g., sUSDe → SY or single-asset deposits minting LP tokens).
`swap_in`	Input leg of a token exchange, representing assets sent from the user to a DEX, router, or liquidity pool as part of a swap.
`swap_out`	Output leg of a token exchange, representing assets sent from a DEX, router, or liquidity pool to the user as the result of a swap.
`deposit`	Transfer of assets into lending protocols, staking contracts, or vaults to obtain interest-bearing positions or receipt tokens, excluding swaps.
`withdraw`	Retrieval of assets from lending protocols, staking contracts, or vaults, potentially accompanied by redemption of receipt tokens.
`borrow`	Borrowing of assets from a lending protocol, resulting in an increase in the user's debt position.
`repay`	Repayment of borrowed assets to a lending protocol, including repayments made by liquidators on behalf of borrowers.
`fee`	Fees charged by protocols, routers, bridges, or flash-loan mechanisms within a transaction.
`reward`	Distribution of protocol incentives or yield (e.g., staking rewards, vault emissions, or airdrops) without a direct input counterpart.
`route`	Transfer used solely for execution routing or delegation between protocol-internal contracts (e.g., router, settler, adapter, executor), without constituting an independent economic action.
direct transfer	Direct token transfer between two externally owned accounts or non-protocol-related addresses, without triggering swaps, minting, burning, or other specific economic actions.

Table 5: Action taxonomy used for structured DeFi token-flow annotation and analysis.

as a generic transfer but as issuance of a derivative position; conversely, burning a receipt token to recover the underlying asset is labeled as `redeem`. This prevents semantically shallow descriptions that miss the user's actual financial position change.

- **Multi-hop aggregation and nested execution.** In aggregator- or meta-router-mediated swaps, each token transfer step is annotated separately at the flow level, including intermediate hops and fee deductions. However, the transaction-level summary may compress contiguous hops into a higher-level operation only when doing so does not obscure economically material fees, hidden approvals, routing side effects, or risk-bearing contract transitions. This preserves readability at the summary level while maintaining full micro-level accountability.
- **Borrow–swap–stake–reborrow loops.** In revolving leverage or recursive collateralization patterns, annotators preserve each borrow, swap, stake, mint, collateralization, and re-borrow step independently rather than collapsing the trace into a single coarse label such as "leveraged position." This is important because user-facing risk often lies in the repeated dependency structure itself rather than in the final macro intent.
- **Flash-loan-mediated actions.** Flash-loan transactions are annotated by separating the temporary borrowed leg, intermediate execution steps, fee repayment, and final settlement. Even when the flash loan is fully repaid within the same transaction, the temporary capital injection is retained explicitly because it often explains otherwise unintuitive trade size, liquidation behavior, or exploit mechanics.
- **Control-transfer and upgrade-related transactions.** Transactions that change ownership, implementation, proxy mastercopy, operator privilege, or administrative authority are treated as high-priority cases even when token movement is minimal. In such cases, the summary must explicitly surface the control-transfer risk rather than only describing low-level call mechanics, because these transactions are especially relevant to phishing, wallet compromise, and governance attacks.

- **Direct transfer vs. protocol interaction.** A token movement is labeled as direct transfer only when it does not participate in a broader protocol-mediated economic action. If the transfer is part of a swap, staking flow, repayment path, settlement sequence, or exploit chain, it is labeled according to that semantic role rather than treated as an isolated transfer.
- **Unknown or partially verified contracts.** When verified source code or documentation is unavailable, annotators are instructed to remain conservative: they label only what is directly supported by token movements, calldata, event logs, and observable state changes. This rule prevents speculative intent attribution and aligns the dataset with the paper's broader emphasis on factual grounding under uncertainty.

Example Annotation. We provide an annotated token flow from a cross-DEX swap:

```
{
  "summary": "tx-level summary",
  "token_flow": [
   {
    "action_id": 1,
    "action_type": "swap_out",
    "intent": "Swap out ...",
    "mechanism": "ContractA#fallback()",
    "precondition": "ETH balance ...",
    "result": "... receives 0.1 ETH."
   },
   {...}, {...}
  ]
}
```

The final natural-language summary synthesizes all flows into a coherent narrative:

> *The user executed a cross-DEX swap of 0.1 ETH for 37.1 USDC. To enable the trade, they first granted a one-time approval of 0.1 ETH to the RangoDiamond router. The router then executed the swap via a multi-step aggregation path on underlying AMMs, deducting a small relayer fee and a protocol fee from the output. Unused native ETH was refunded to the user.*

Concrete examples are provided in Section F.

C.1 Semantic Consistency of Annotations

For the natural language token-flow attributes—intent, mechanism, precondition, and result—we do not report κ as the primary consistency measure, because semantically equivalent annotations may differ lexically. Instead, we perform a supplementary semantic consistency analysis on the independently written annotation pairs for each token flow. Concretely, for each field, we compute BERTScore F1 between the two independently written annotations, and then average scores across all annotated token flows. We use the official BERTScore (Zhang et al., 2020) implementation with a `roberta-large` backbone and baseline rescaling enabled. The resulting field-level semantic consistency scores are all above 0.91, as shown in Table Table 6. These results indicate that, although annotators often use different surface forms, their free-form annotations remain highly consistent at the semantic level.

Field	BERTScore F1
intent	0.962
mechanism	0.913
precondition	0.982
result	0.948

Table 6: Semantic consistency of free-form token-flow annotations, computed as average pairwise BERTScore F1 between independently written annotation pairs.

C.2 Dataset Profile

This section summarizes the coverage profile of the 187 annotated transactions retained after filtering. The retained transactions are intentionally skewed toward semantically rich DeFi interactions rather than routine wallet activity. In particular, the dataset contains multi-step swaps, lending loops, staking operations, liquidity management, and cross-protocol compositions that require reasoning across multiple token flows and contracts. Table 7 reports the primary-category distribution, and Table 8 summarizes token-flow complexity, protocol diversity, and contract verification status. For example, a transaction that borrows assets, swaps them through an aggregator, and then deposits the resulting tokens into a staking protocol may be assigned the primary category lending/borrowing cycle if the dominant economic purpose is leverage creation, even though it also contains swap and staking substeps. Although each transaction is assigned a single primary category for summary statistics, many examples contain multiple semantic behaviors; in particular, 57% of retained

transactions involve more than one protocol-level action pattern.

Category	Count
Swaps / Aggregator-routed swaps	47
Lending / Borrowing cycles	37
Staking / Delegation	42
Liquidity Pool operations	34
Cross-protocol interactions	29

Table 7: Primary-category distribution of the 187 annotated transactions in *TxSum* after filtering. Each transaction is assigned one expert-defined dominant category for profiling purposes.

Coverage Statistic	Value
Total annotated Txs	187
Average token-flow steps / Tx	12.7
Average distinct protocols / transaction	2.8
Verified focal contracts	94
Partially verified focal contracts	56
Unverified / closed-source focal contracts	37

Table 8: Coverage statistics of *TxSum*.

D Agent Design Details

MATEX consists of four specialized agents that collaborate through a shared structured workspace (the "Evidence Board") and interact through the two core mechanisms described in Section D.1. First, we provide the pseudocode for the two core mechanisms used in *MATEX*: Dynamic Knowledge Injection Protocol (DKIP, Algorithm 1) and Auditor-Guided Context Pruning (AGCP, Algorithm 2). Then, we describe each agent's role, inputs, outputs, and operational behavior.

D.1 Algorithmic Details of *MATEX*

Profiler. The Profiler implements the fast, pattern-based stage of *MATEX* (System 1). It ingests the transaction metadata, call trace, token transfers, and balance changes, and produces two kinds of outputs for each token flow: (1) a lightweight semantic hypothesis when the flow matches a known pattern, and (2) an uncertainty assessment used by DKIP. Operationally, the Profiler assigns each flow a semantic uncertainty score u_i and, when escalation is needed, emits a retrieval key q_i that specifies what external protocol evidence should be queried (e.g., a contract-function pair, router name, or unknown address role).

Algorithm 1 Dynamic Knowledge Injection

Require: Token flows $T = \{t_1, t_2, \ldots, t_n\}$, uncertainty threshold τ, evidence budget k
Ensure: Synthesized context C_{syn}

```
C_syn ← ∅, B ← ∅                    ▷ B: evidence cache
for all t_i ∈ T do
    (u_i, q_i) ← Profiler(t_i)      ▷ q_i: retrieval key
    if u_i ≤ τ then
        C_syn ← C_syn ∪ ToKnownPattern(t_i)
    else
        if q_i ∉ B then
            R_i ← Investigator(q_i)
            B[q_i] ← SelectTopK(R_i, k)
        end if
        C_syn ← C_syn ∪ Format(t_i, B[q_i])
    end if
end for
return C_syn
```

Critically, the Profiler is designed to remain conservative under uncertainty. When it encounters unfamiliar contracts, custom execution paths, or anomalous state changes, it does not force a semantic interpretation; instead, it raises the uncertainty score and forwards the relevant retrieval key to the Investigator. This design ensures that ambiguous flows trigger evidence-seeking rather than unsupported extrapolation.

Profiler Prompt

You are an expert Ethereum Transaction Profiler. Your goal is to create a structured investigation plan grounded in [TOKEN TRANSFERS].

CRITICAL: Anchor your analysis to the input [TOKEN TRANSFERS].
- Prefer using TT# ranges (e.g., "TT#1–TT#6") over generic "Step 1-5".
- Each segment should cover a contiguous range of TT# entries.
- For each uncertain flow or segment, assign an uncertainty score and provide a retrieval key for downstream investigation.

Output a JSON object with exactly these four keys:
1. "segments": logical phases of the transaction;
2. "initial_hypothesis": a high-level intuition about the transaction type;

Agent	Input	Output	Role in the pipeline
Profiler	Token transfers, call trace, metadata, balance changes	Flow-level hypotheses, uncertainty score u_i, retrieval key q_i when needed	Performs fast pattern-based screening and decides whether each flow should remain on the lightweight path or be escalated for retrieval.
Investigator	Retrieval key q_i and supporting trace context	Verified protocol evidence (e.g., ABI entries, documentation snippets, contract-role resolution)	Retrieves protocol context only for uncertain flows under DKIP.
Synthesizer	On-chain evidence, profiler hypotheses, retrieved evidence	Draft transaction explanation and flow-grounded semantic outputs	Fuses evidence into a coherent transaction-level explanation and structured flow-level interpretation.
Safety Auditor	Draft explanation, raw trace, token transfers	APPROVED or critique with error spans / unsupported claims	Checks factual grounding against raw traces and triggers AGCP to prune unsupported revision context before re-synthesis.

Table 9: Inputs and outputs of the four agents in *MATEX*.

3. "uncertain_flows": a list of uncertain TT# items with uncertainty scores and retrieval keys;
4. "investigation_tickets": concrete verification questions for the Investigator.

JSON FORMAT:
{
"segments": ["phase_id": 1, "range": "TT#1–TT#5", "summary_label": "..."],
"initial_hypothesis": "...",
"uncertain_flows": ["transfer_ref": "TT#3", "uncertainty_score": 0.82, "retrieval_key": "..."],
"investigation_tickets": ["target_step": "...", "contract": "...", "reason": "...", "question": "..."]
}

Algorithm 2 Auditor-Guided Context Pruning

Require: Fixed evidence context C_{syn}, raw trace Tr, max iterations M
Ensure: Verified explanation E_{final}

$D \leftarrow \emptyset$ ▷ Mutable revision context
for $iter = 1$ to M **do**
  $E_{\text{draft}} \leftarrow \text{Synthesizer}(C_{\text{syn}} \cup D)$
  $(v, F_{\text{err}}) \leftarrow \text{Auditor}(E_{\text{draft}}, Tr)$
  if $v = \text{APPROVED}$ **then**
    return E_{draft}
  end if
  $P_{\text{mask}} \leftarrow \text{LocateErrorContext}(D, E_{\text{draft}}, F_{\text{err}})$
  $D \leftarrow (D \setminus P_{\text{mask}}) \cup \text{Critique}(F_{\text{err}})$
end for
return E_{draft}

Protocol Investigator. The Investigator is only activated for flows whose uncertainty score exceeds the DKIP threshold. Given a retrieval key q_i from the Profiler, it retrieves protocol evidence from the same external sources used in evaluation: verified contract ABIs, protocol documentation, and temporally aligned web evidence. Its role is not to broadly summarize the protocol, but to return only the evidence necessary to resolve the ambiguous flow under investigation. This selective retrieval design limits context noise and keeps downstream synthesis focused on uncertain parts of the trace.

Investigator Prompt

You are a Researcher. I will provide a question about a contract and raw search results. Answer the question based STRICTLY on the search results.

Verification rule:
- Only output "Verified: ..." if the search results explicitly mention the contract/function AND describe the behavior relevant to the question.
- Otherwise, output: "No verifiable info found."

Keep it concise.

Synthesizer. The Synthesizer serves as the main reasoning module in System 2. It fuses three inputs: (1) the raw on-chain trace, (2) the Profiler's flow-level hypotheses, and (3) the Investigator's retrieved evidence patches. Its goal is to map low-level operations to user-facing economic actions and produce a coherent transaction-level explanation together with flow-grounded semantic interpretations.

To preserve factual grounding, the Synthesizer operates over evidence that remains internally attributed to its source (e.g., raw trace facts, ABI evidence, or retrieved documentation snippets). The final user-facing explanation does not need to expose these provenance tags verbatim, but it is generated from this evidence-grounded internal state. Its output is then passed to the Safety Auditor for adversarial validation. The prompt for Synthesizer is too long; we have put it in the open science repository in Section 6.

Safety Auditor. The Safety Auditor acts as an adversarial checker over the Synthesizer's draft. It compares the generated explanation against the raw trace, token transfers, and retrieved evidence, focusing on unsupported claims, omitted intermediate effects, action-type errors, and missing risk-bearing side effects.

If contradictions are found, the Auditor does not merely append a critique for the next round. Instead, *MATEX* applies Auditor-Guided Context Pruning (AGCP): the system identifies the unsupported or error-causing portion of the mutable revision context and removes it before re-synthesis. This design reduces repeated hallucinations across iterations and is especially important in long, compositional DeFi transactions where a single unsupported inference can pollute subsequent drafts.

Safety Auditor Prompt

You are a Senior DeFi Security Auditor.
Critique the Draft JSON against the Raw Trace.

Hard checks (must fail if violated):
1) Hallucinations: tokens/amounts/addresses/protocols/functions not supported by trace.
2) Wrong 'action_type' classification.
3) Missing fees or side effects IF they appear in [TOKEN TRANSFERS].
4) Format errors (JSON schema, intent/mechanism/precondition/result structure).
5) [TOKEN TRANSFERS] 1:1 coverage:
- Every TT# entry must appear EXACTLY ONCE as token_flows[i].transfer_ref.
- token_flows length must equal the number of transfers in [TOKEN TRANSFERS].
- Each token_flows[i].result must EXACTLY match its TT# facts (from/to/token/amount).

If perfect, output exactly: "APPROVED".
If errors exist, output a concise list of fixes.

E Evaluation Details

E.1 EO1: Explanation Quality

We evaluate explanation quality using both automated and expert-based methods on our full *TxSum* test set (187 complex transactions). The detailed justification is in Table 11.

LLM-as-a-Judge. We employ GPT-5.1 as an automated evaluator, with separate judging protocols for token-flow and transaction levels. For **transaction-level evaluation (T1, T2)**, the judge receives: (1) the full on-chain input (transaction metadata, decoded call trace, complete token transfer list, and net balance changes), and (2) the generated sentence's natural language summary. It then outputs binary judgments for outcome correctness and clarity. For **token-flow evaluation (F1, F2)**, the judge receives: (1) the raw on-chain sub-trace corresponding to a single token flow (e.g., a transfer event and its surrounding context), and (2) the predicted semantic attributes for that flow (e.g., intent, mechanism). It then outputs binary (YES/NO) judgments for faithfulness and intent quality.

Both protocols use structured prompts that require the model to ground every decision in the provided evidence. Prompts are validated against expert annotations; the final validation numbers are reported in the main text and Section E.1.2.

Transaction Level Judge Prompt

You are a Financial Auditor. Evaluate the "Model Summary" against the "Expert Summary" for the entire transaction.

Task
Evaluate two specific dimensions based ONLY on the summary text.
1. T1: Outcome Correctness (Financial Precision)
- **Definition**: Does the summary accurately state the **Final Net Outcome** (Profit/Loss/Balance Changes)?
- **PASS (YES)**: Directions (+/-) and amounts match the Expert Summary.
- **FAIL (NO)**: Wrong direction (gain vs loss), wrong final numbers, or missing significant fees mentioned by Expert.

2. T2: Clarity (Readability)
- **Definition**: Is the summary concise and easy to understand for a non-expert?

Method	ROUGE-L	BLEU	EM	Acc.	Macro-F1
LLM	41.8	19.6	21.4	58.7	54.3
Few-shot CoT	43.6	21.3	24.1	61.5	57.1
VLM	42.9	20.7	22.9	60.6	55.8
Pure RAG	42.1	19.8	22.0	59.1	54.9
Single-Agent	47.9	24.3	28.6	66.8	62.7
MATEX	52.6	27.8	33.9	72.4	68.5

Table 10: Standard NLP metrics for EO1. ROUGE-L and BLEU are computed on transaction-level summaries; EM is computed on flow-level semantic attribute generation; Accuracy and Macro-F1 are computed on the 14-way action_type classification task.

Metric	PASS (YES)	FAIL (NO)
Token Flow Level		
F1: Faithfulness	All claims are directly verifiable from the on-chain trace. Correct token, amount, direction, and action type (e.g., "swaps 50 USDC for 0.03 ETH via Uniswap V3").	Any hallucination or factual error: wrong token/amount, fabricated action (e.g., claims "stakes 1 ETH" when trace only shows WETH wrap).
F2: Intent Quality	Explains economic purpose using trace evidence (e.g., "deposits USDC into Aave to borrow DAI"). Purpose must be inferable from on-chain steps.	Only describes mechanics without purpose (e.g., "transfers USDC to Aave") or speculates unsupported intent (e.g., "to invest in crypto").
Transaction Level		
T1: Outcome Correctness	Net balance changes, fees, and critical non-token state changes exactly match on-chain data. Includes all fees >= 0.1% of principal (e.g., "Net: +240 USDC, -0.05 ETH, fee: 1.2 USDT").	Any error in the final state or omission of material fees (e.g., claims "gained 240 USDC" when net is -10 USDC after fees).
T2: Clarity & Conciseness	Natural-language narrative in 3–4 sentences. No raw addresses, hex, or internal names. Groups related actions (e.g., "multi-hop swap via 1inch, then staked in Curve").	Uses EOA/contract addresses (e.g., "to 0x7a25..."), list format ("Step 1: ... Step 2: ..."), or exceeds 3 sentences.

Table 11: Detailed binary scoring rubric for *TxSum*. All judgments must be grounded in the on-chain input.

- **PASS (YES)**: Natural language, logical flow, concise (approx 2-3 sentences).
- **FAIL (NO)**: Verbose, dumps raw JSON/Hex data, robotic phrasing, or overly long.

Output Format (Strict JSON)
{
"reasoning": "Brief analysis...",
"scores": {
"T1_Outcome": "YES",
"T2_Clarity": "YES"
} }

Token Flow Level Judge Prompt

You are a DeFi Semantic Expert. Compare the "Model Prediction Flow" against the "Expert Ground Truth Flow" for a specific step in a transaction.

Task
Evaluate the prediction on two specific dimensions.

1. F1: Faithfulness (Semantic Equivalence)
- **Definition**: Does the model describe the *same* factual event? (Amounts, Tokens, Direction).
- **PASS (YES)**: "User sends 1 ETH" == "Transfer 1 ETH from User". Numbers match exactly.
- **FAIL (NO)**: Wrong amounts, wrong tokens, wrong direction, or factual contradictions.

2. F2: Intent Quality (Economic Purpose)
- **Definition**: Does the 'intent' capture the **economic purpose** ("Why"), not just the mechanical action ("What")?
- **PASS (YES)**: "User deposits ETH **to collateralize**." / "User sends funds **to initiate swap**."
- **FAIL (NO)**: "User transfers ETH." (Mechanical repetition).

Output Format (Strict JSON)
{
"reasoning": "Brief analysis...",
"scores": {
"F1_Faithfulness": "YES",
"F2_Intent_Quality": "YES"
}}

Expert Evaluation. Three DeFi researchers with $\geq$3 years of blockchain security or protocol development experience perform blind evaluations of all 187 transactions (balanced across systems). Experts assess each output on: (1) factual correctness (aligned with F1 and T1), (2) intent alignment (F2), and (3) risk awareness (whether critical risks are obscured or omitted). Each criterion is scored as `Pass`/`Fail`. Inter-annotator agreement is measured via Fleiss' $\kappa = 0.76$.

E.1.1 Additional Metrics for EO1

To complement the binary pass/fail evaluation in the main text, we also report standard NLP metrics for EO1, including ROUGE-L and BLEU for transaction-level summaries, Exact Match (EM) for flow-level semantic attribute generation, and Accuracy / Macro-F1 for the 14-way action_type classification task.

E.1.2 Judge Validation and Statistical Testing

This section provides additional details on the reliability of the GPT-5.1 judge used in EO1, including its agreement with expert labels, judge error patterns, verbosity-bias analysis, and the statistical significance tests used to compare *MATEX* with the strongest non-*MATEX* baseline.

Judge Validation Setup. For EO1, the GPT-5.1 judge evaluates each system output on the four binary criteria defined in Table 12: **T1** (outcome correctness), **T2** (clarity & conciseness), **F1** (faithfulness), and **F2** (intent quality). To assess the reliability of this automatic evaluator, we compare its judgments against expert annotations produced by three DeFi Ph.D. researchers on the same transaction set. We report overall judge accuracy with respect to the expert majority label, as well as per-metric Cohen's κ. The overall agreement is 85.6%, with Cohen's κ values of 0.71 for T1, 0.78 for T2, 0.69 for F1, and 0.73 for F2. Across the three human experts, inter-rater reliability is also high, with Fleiss' $\kappa = 0.76$.

Judge Error Patterns. To better understand judge failure modes, we inspect its confusion patterns relative to expert labels. Table 14 reports the aggregated confusion matrix over all EO1 judgments. Most disagreements arise from borderline cases involving partial factual omissions (e.g., small fees or implicit approvals) rather than purely stylistic differences. This is consistent with the judge prompt, which emphasizes grounding in on-chain evidence and semantic adequacy rather than surface token flow alone.

Level	Metric	Definition
Transaction (Macro)	T1	**Outcome Correctness**: The summary accurately describes the final net balance changes, explicit fees, and critical non-token state changes (e.g., approvals, ownership transfers).
	T2	**Clarity & Conciseness**: The summary is concise (3-4 sentences), avoids technical jargon, and forms a coherent narrative.
Token Flow (Micro)	F1	**Faithfulness**: The predicted flow is semantically equivalent to the ground truth (i.e., factually correct actions and amounts).
	F2	**Intent Quality**: The explanation captures the *economic purpose* (the "why") rather than just describing the mechanical transfer (the "what").

Table 12: Evaluation for *TxSum*, divided into macro-level summarization and micro-level flow assessment.

Metric	Judge Accuracy (%)	Cohen's κ
T1	84.0	0.71
T2	88.2	0.78
F1	83.4	0.69
F2	86.8	0.73
Overall	85.6	—

Table 13: Agreement between the GPT-5.1 judge and expert majority labels for EO1.

Expert Label	Judge Prediction	
	PASS	FAIL
PASS	496	61
FAIL	47	144

Table 14: Aggregated confusion matrix of GPT-5.1 judgments against expert majority labels across T1–F2.

Verbosity Bias Analysis. A common concern in LLM-as-a-judge evaluation is that longer outputs may receive artificially favorable scores. To test for this effect, we compute the Pearson correlation between output length and binary pass rate across the evaluated summaries. We obtain $r = 0.07$ and $p = 0.31$, indicating no statistically significant verbosity bias. This suggests that the judge is not simply rewarding longer or more verbose explanations.

Statistical Significance Testing. We compare *MATEX* against the strongest non-*MATEX* baseline on the same 187 transactions using paired statistical tests. For the binary pass/fail metrics (T1–F2), we use paired McNemar tests, since all systems are evaluated on the same instances. For continuous automatic metrics, we use paired significance tests appropriate for matched predictions: specifically, we report paired tests for ROUGE-L, Exact Match (EM), and action_type Macro-F1.[2] The resulting p-values are summarized in Table 15.

Metric	Test	p-value
T1	McNemar	0.118
F1	McNemar	< 0.001
F2	McNemar	0.003
ROUGE-L	Paired bootstrap	0.006
action_type Macro-F1	Paired bootstrap	0.004

Table 15: Key paired significance tests comparing *MATEX* against the strongest baseline on EO1.

Takeaway. These validation and significance results support two conclusions. First, the GPT-5.1 judge is sufficiently aligned with expert judgment to serve as a scalable evaluator for EO1, while expert inter-rater agreement further confirms the reliability of the human protocol. Second, the improvements achieved by *MATEX* on the key factuality- and intent-related metrics are statistically robust rather than artifacts of random variation on a limited test set.

E.2 EO2: User Trust & Comprehension

Participant Cohort and Recruitment. EO2 uses a **new cohort of 20 active DeFi users**, which is **separate from the 16 participants** in our formative interview study. The earlier 16 interviewees were used only to identify user needs and inform task design, whereas the EO2 cohort was recruited later for controlled evaluation. No participant took part in both studies.

[2] We use paired bootstrap resampling for the continuous metrics.

We recruited the EO2 participants via Web3 communities and professional networks across 6 countries/regions. Inclusion criteria were: (1) at least 3 on-chain DeFi transactions in the past 6 months; (2) familiarity with at least two major protocols (e.g., Uniswap, Aave, or Lido); and (3) age ≥ 18. Participants include practitioners, researchers/developers, students, and casual users. Their demographic distribution is summarized in Table 16.

Role	Count	Avg. Exp. (years)
Practitioners	4	4.2
R&D	5	3.8
Students	7	1.8
Casual Users	4	2.2

Table 16: Demographics of EO2 user study participants ($N = 20$). Practitioners include traders and analysts; R&D includes researchers and developers.

Study Design. EO2 is a **within-subject controlled study** using the EigenPhi block explorer interface under three conditions: (a) **Raw**: EigenPhi's default view (call trace + token transfers); (b) **Single-Agent**: Raw explorer view augmented with annotations from the strongest non-*MATEX* baseline in EO1; (c) ***MATEX***: Raw explorer view augmented with *MATEX* explanations.

Each participant viewed 9 transactions in total—3 simple, 3 medium, and 3 complex. Within each complexity level, one transaction was shown under each condition. Condition assignment and presentation order were balanced using a Latin Square design to mitigate order, transaction-identity, and complexity effects. Participants were not placed under a strict time limit, and most completed the study in approximately 40 minutes. For each transaction, participants completed the following questionnaire:

User Study Questionnaire

Comprehension (choose one for each question):

1. What was the main purpose of this transaction?
A) To swap USDC for ETH and stake the ETH in Lido
B) To withdraw liquidity from a Uniswap pool
C) To repay a loan on Aave
D) Not sure

2. Which of the following actions occurred during this transaction?
A) Approved unlimited spending for a proxy contract
B) Swapped USDC for DAI via a multi-hop route
C) Deposited ETH into a lending protocol
D) All of the above

3. What is the primary financial risk in this transaction?
A) High slippage due to low liquidity
B) Unlimited token approval granted to an unknown contract
C) No significant risk
D) Not sure

Perception (rate 1–5, where 1 = Strongly Disagree, 5 = Strongly Agree):

4. I understand what this transaction does.
5. I trust this explanation.

Scoring Procedure. The questionnaire contains two components. **(1) Comprehension Accuracy.** Participants answered three multiple-choice questions covering: (i) the main purpose of the transaction, (ii) a major action or mechanism involved in execution, and (iii) the most salient risk or side effect. Each correct answer received one point, yielding a per-transaction comprehension score from 0 to 3. We report mean accuracy as the percentage of correctly answered questions aggregated across participants. **(2) Trust and Clarity.** Participants rated the following statements on a 5-point Likert scale: (i) "I understand what this transaction does," and (ii) "I trust this explanation." We report the average of the two Likert items as the trust/clarity score for each condition and complexity level.

Statistical Analysis. Because each participant completed all three conditions, we used repeated-measures tests to compare comprehension accuracy and Likert-scale trust/clarity ratings across conditions. These analyses support the same overall pattern shown in the main text: *MATEX* yields the strongest gains in comprehension accuracy, with the largest advantage on medium- and high-complexity transactions, and also achieves the highest trust/clarity ratings overall. The overall condition effect on comprehension accuracy is assessed with Cochran's Q test, followed by post-hoc paired comparisons between *MATEX* and the two comparison conditions. For trust and clarity ratings, we similarly test for an overall condition effect and then examine pairwise differences. The detailed test statistics and pairwise comparison results are

reported below in Table 17.

The statistical results are consistent with the main-text trends: *MATEX* shows a significant overall advantage in both comprehension accuracy and trust/clarity. The largest differences are observed relative to the Raw condition, while the gains over the Single-Agent baseline remain significant but smaller in magnitude. This pattern suggests that grounded, step-wise explanations are especially beneficial when users would otherwise rely on incomplete or weakly structured transaction views.

Interpretation. This design allows us to evaluate whether *MATEX* improves users' objective understanding of transaction purpose and risk, as well as their subjective trust in the explanation, under matched interface conditions. Because the EO2 cohort is separate from the formative interview participants, the evaluation is not conflated with prior exposure to our study design or task framing.

E.3 EO3: Risk Awareness

Risk Transaction Selection. We select 5 real-world malicious transactions from major DeFi/NFT exploits, each representing a distinct and non-obvious risk pattern:

- **BadgerDAO** (Dec 2021): Infinite token approval via compromised frontend.[3]
- **WazirX** (Jul 2024): Gnosis Safe proxy upgrade via misleading UI.[4]
- **OpenSea** (Feb 2022): Malicious NFT order signature via phishing email.[5]
- **Bybit** (Feb 2025): Gnosis Safe mastercopy upgrade via compromised frontend.[6]
- **BXH Exchange** (Oct 2021): Direct fund theft due to missing multi-sig.[7]

We also include 5 benign transactions of comparable complexity as controls (e.g., standard multi-hop swaps, LP deposits).

User Evaluation. We conduct a dedicated between-subjects study with 20 participants (the same 20 participants in EO2), split into two groups of 10. (1) The **Raw group** views all 10 transactions (5 malicious, 5 benign) in a standard block explorer interface (EigenPhi). (2) The ***MATEX* group** views the same transactions with *MATEX* explanations overlaid on the same interface.

For every transaction, we simulate the pre-signing confirmation screen as it appeared in the original attack (e.g., in MetaMask or EigenPhi), showing the identical calldata and token flows that victims encountered. Participants see either (a) the raw transaction view or (b) the same view enhanced with *MATEX* annotations, with condition assignment balanced across users. Participants provide (1) their signing intention (Yes/No) and (2) an open-ended description of any risks they notice.

Two researchers independently code responses for risk detection using manual semantic coding guided by core threat concepts, rather than rigid keyword matching. For example, responses such as "this contract can keep spending my tokens forever" are credited as recognizing an unlimited approval even if the exact phrase is absent. The coders achieve Cohen's $\kappa = 0.81$.

Scoring Procedure and Real Cases. For malicious transactions, we compute (i) rejection rate and (ii) correct threat-identification rate. For benign transactions, we compute the false-rejection rate. Threat identification is counted when the participant correctly recognizes the core exploit mechanism (e.g., unlimited approval, malicious proxy upgrade, hidden ownership change, phishing signature abuse), regardless of exact wording. Section F provides real examples showing how unconventional phrasing is still credited under this rubric.

E.3.1 Ablation Results

To better understand which architectural components drive the gains of *MATEX*, we perform ablations in which we disable one major component at a time while keeping the rest of the pipeline unchanged. We report EO1 judge-based metrics together with representative standard NLP metrics. The results are shown in Table 18.

F Case Studies

We present three representative cases to illustrate why *TxSum* is practically important and technically challenging, where coarse intent labels or final-balance summaries are insufficient for safe user-facing explanation.

Case 1: Aggregator-Routed Swap with Hidden

[3] https://www.trmlabs.com/resources/blog/trm-investigates-badgerdao-defi-protocol-hacked
[4] https://www.cloudsek.com/blog/wazirx-incident-explained
[5] https://www.certik.com/resources/blog/opensea-phishing-incident-analysis
[6] https://www.certik.com/resources/blog/bybit-incident-technical-analysis
[7] https://www.certik.com/resources/blog/boy-x-highspeed-incident-analysis

Comparison	Test	Result
Overall comprehension accuracy	Cochran's Q	$p < 0.001$
MATEX vs. Raw (accuracy)	Paired McNemar	$p < 0.001$
MATEX vs. Single-Agent (accuracy)	Paired McNemar	$p = 0.012$
Overall trust/clarity	Friedman test	$p < 0.001$
MATEX vs. Raw (trust/clarity)	Wilcoxon signed-rank	$p < 0.001$
MATEX vs. Single-Agent (trust/clarity)	Wilcoxon signed-rank	$p = 0.018$

Table 17: Statistical comparisons for EO2. The pattern is consistent with the descriptive results in the main text: *MATEX* yields the strongest overall performance, with the largest gains over the Raw condition and smaller but still significant improvements over the Single-Agent baseline.

Variant	T1	T2	F1	F2	ROUGE-L	Action Macro-F1
w/o Profiler	$68.4_{\downarrow 4.9}$	$88.2_{\downarrow 4.8}$	$70.5_{\downarrow 2.7}$	$74.2_{\downarrow 2.4}$	$50.2_{\downarrow 2.4}$	$66.0_{\downarrow 2.5}$
w/o Investigator	$57.2_{\downarrow 16.1}$	$74.9_{\downarrow 18.1}$	$62.8_{\downarrow 10.4}$	$70.3_{\downarrow 6.3}$	$46.3_{\downarrow 6.3}$	$60.4_{\downarrow 8.1}$
w/o Synthesizer	$47.1_{\downarrow 26.2}$	$64.2_{\downarrow 28.8}$	$57.9_{\downarrow 15.3}$	$68.5_{\downarrow 8.1}$	$41.8_{\downarrow 10.8}$	$56.5_{\downarrow 12.0}$
w/o Auditor	$72.2_{\downarrow 1.1}$	$91.4_{\downarrow 1.6}$	$62.5_{\downarrow 10.7}$	$74.9_{\downarrow 1.7}$	$49.9_{\downarrow 2.7}$	$62.0_{\downarrow 6.5}$
Full *MATEX*	**73.3**	**93.0**	**73.2**	**76.6**	**52.6**	**68.5**

Table 18: Ablation results on EO1 and representative standard NLP metrics. Each variant disables one component while keeping the remaining pipeline unchanged. T1–F2 are LLM-as-judge pass rates (%), while ROUGE-L and action_type Macro-F1 evaluate transaction-level summary quality and structured semantic parsing, respectively. Arrows indicate absolute drops relative to full *MATEX*.

Intermediate Fees. A user swaps ETH for DAI through 1inch. On-chain, the transaction is executed through multiple intermediate contracts: the user sends ETH to the 1inch router, the router performs an ETH→USDC swap on Uniswap V3, forwards the USDC through an executor contract, and finally completes a USDC→DAI swap on SushiSwap. The trace also contains a relayer fee, a routing-related fee deduction, and a small ETH refund.

Why it matters. This case shows why user-facing transaction understanding cannot stop at a coarse label such as "swap." For signing safety, the user also needs to know whether the transaction introduces unfamiliar intermediary contracts, applies hidden deductions, or contains additional state changes beyond the intended exchange.

Why it is hard. The overall intent is relatively easy to identify, but a faithful explanation is more demanding. A model can correctly infer that the transaction is a swap while still omitting the intermediate fee transfer, misclassifying executor hops, or failing to explain why multiple protocols appear in what seems to be a single operation. This gap between macro intent and micro-level factual completeness is central to our task.

Annotation and evaluation implication. Intermediate executor hops may be labeled as `route`, but fee-bearing transfers must remain explicit. A summary that only states "the user swapped ETH for DAI" is incomplete if it omits economically material fees or intermediate deductions.

Case 2: Recursive Leverage via Aave, CowSwap, and Lido. A user borrows USDC from Aave, swaps the USDC into ETH through CowSwap, stakes the ETH into stETH through Lido, supplies the stETH back to Aave as collateral, and then borrows again. The resulting position is economically a leveraged staking strategy, but the on-chain trace is distributed across borrowing, swapping, staking, derivative issuance, collateral supply, and repeated debt expansion.

Why it matters. This case reflects a realistic DeFi workflow in which the main user risk does not lie in a single transfer, but in the dependency structure created across multiple protocols. For transaction explanation, it is not sufficient to say that the user "opened a leveraged position"; the explanation should reveal how the leverage is constructed and where the exposure comes from.

Why it is hard. A coarse transaction label can be correct while still being inadequate. The trace

mixes economically distinct actions that are easy to compress into one high-level description, yet such compression hides the repeated collateralization pattern that determines liquidation sensitivity and protocol dependence. The difficulty therefore lies in preserving enough step-wise structure to remain faithful without losing readability.

Annotation and evaluation implication. Each borrow, swap, staking, collateral deposit, and re-borrow step is annotated independently. This case motivates why *TxSum* requires micro-level semantic parsing rather than only top-level intent prediction.

Case 3: Safe Wallet Mastercopy Upgrade with Minimal Token Movement. A transaction presented in the wallet interface appears to be a routine administrative confirmation, but in fact changes the mastercopy or implementation address of a Safe-style wallet. At signing time, there may be little or no visible token movement, even though control over future wallet behavior is effectively altered.

Why it matters. This is representative of the transactions that are most dangerous for blind-signing: the immediate balance outcome looks harmless, but the security consequence is severe. A user-facing explanation must therefore surface non-token state changes when they alter ownership, upgrade authority, or execution control.

Why it is hard. Purely token-centric reasoning is insufficient here because the critical event is not a transfer but a control-plane change. A system that focuses only on token flows or final balances may produce an explanation that is mechanically accurate yet fails to communicate the actual security implication. This makes the case especially important for high-stakes explanation settings.

Annotation and evaluation implication. These cases motivate why *TxSum* includes transaction-level summaries that must mention critical non-token state changes. An explanation that only paraphrases the low-level call mechanics, but does not surface the control-transfer risk, is considered inadequate for user-facing safety.