evalSmarT: An LLM-Based Framework for Evaluating Smart Contract Generated Comments

Fatou Ndiaye Mbodji[1], Mame Marieme C. Sougoufara[2], Wendkûuni A. M. Christian Ouedraogo[1], Alioune Diallo[1], Kui Liu[3], Jacques Klein[1], Tegawendé F. Bissyandé[1]
[1]SnT – University of Luxembourg, [2]Université Cheikh Anta Diop, [3]Huawei

***Abstract*—Smart contract comment generation has gained traction as a means to improve code comprehension and maintainability in blockchain systems. However, evaluating the quality of generated comments remains a challenge. Traditional metrics such as BLEU and ROUGE fail to capture domain-specific nuances, while human evaluation is costly and unscalable. In this paper, we present `evalSmarT`, a modular and extensible framework that leverages large language models (LLMs) as evaluators. The system supports over 400 evaluator configurations by combining approximately 40 LLMs with 10 prompting strategies. We demonstrate its application in benchmarking comment generation tools and selecting the most informative outputs. Our results show that prompt design significantly impacts alignment with human judgment, and that LLM-based evaluation offers a scalable and semantically rich alternative to existing methods.**

Resources

Video Demo: **https://youtu.be/HXS_Yiszoz4**
Code and Data: **https://anonymous.4open.science/r/SC_code_summarization-4653**

***Index Terms*—Smart contracts, Comment generation, Large language models (LLMs), LLM as judge**

I. Introduction

Smart contracts represent a foundational technology in decentralized systems, enabling the autonomous execution of agreements. Due to their immutable and financial nature, they are particularly sensitive to design flaws and documentation errors. Clear, accurate, and complete documentation is therefore essential not only for code maintainability, but also for security audits, regulatory compliance, and effective collaboration across teams.

Despite this need, smart contracts are often poorly documented [1]. Even when comments are present, they frequently lack relevance or alignment with the underlying code. This has motivated the development of automatic comment generation models tailored to smart contracts [2], [3]. However, evaluating the quality of these generated comments remains a significant challenge.

Traditional evaluation methods rely on surface-level metrics such as BLEU, METEOR, and ROUGE, which fail to capture the semantic and domain-specific aspects of smart contract behavior. Human evaluation, while more nuanced, is time-consuming, subjective, and difficult to scale. In contrast, the emerging paradigm of using large language models (LLMs) as evaluators referred to as *LLM-as-a-Judge* offers a promising alternative. Prior work has demonstrated the effectiveness of LLMs in evaluating requirements [4] and code summaries [5], particularly when guided by carefully designed prompts.

In this paper, we introduce `evalSmarT`, a modular and extensible framework for evaluating smart contract comment generation using LLMs. Unlike general-purpose summarization tasks, smart contract documentation requires domain-specific knowledge of Solidity and the Ethereum ecosystem. `evalSmarT` addresses this challenge by combining multiple LLMs with ten prompting strategies that incorporate domain awareness, language-specific features, and evaluation framing. Our framework enables scalable, reproducible, and semantically rich evaluation of generated comments, bridging the gap between traditional metrics and expert judgment

II. Motivation and Problem Statement

Smart contracts are critical components of decentralized systems. Their immutable and financial nature makes them particularly sensitive to design flaws and documentation errors. While automatic comment generation models have emerged to support smart contract comprehension, their evaluation remains limited. Traditional metrics such as BLEU, METEOR, and ROUGE fail to capture domain-specific concerns like security, gas optimization, or Solidity-specific constructs. Human evaluation, though more nuanced, is costly and unscalable.

Below, we present evaluation methods used in existing studies.

TABLE I: Existing code comment methods and their evaluation metrics

Method names	Evaluation metrics
SMTranslator [6]	Human judgment
STAN [7]	Human judgment
MMTrans [8]	BLEU, METEOR, ROUGE
SMARTDOC [9]	BLEU, ROUGE, Human judgment
CCGIR [2]	BLEU, METEOR, ROUGE
SolcTrans [10]	BLEU, Human judgment
SCCLLM [3]	BLEU, ROUGE
SCLA [11]	BLEU, METEOR, ROUGE
FMCF [12]	BLEU, METEOR, ROUGE
SmartBT [13]	BLEU, ROUGE, Human judgment
CCGRA [14]	BLEU, METEOR, ROUGE-L, Human judgment

2643-1572/25/$31.00 ©2025 IEEE
DOI 10.1109/ASE63991.2025.00389

Motivation: Lack of Evaluation Diversity

Most studies rely on traditional automatic metrics such as BLEU, METEOR, and ROUGE, with **0% employing LLM-based evaluation** and 55% incorporating human judgment. This reveals a significant gap in methodological diversity, particularly the absence of modern LLM-based evaluation approaches.

Inference of the Diversity Lack

Although some recent works explore LLMs for the **generation** of smart contract comments [3], [13], to the best of our knowledge, **none** use LLMs as **evaluators**. This contrasts with broader blockchain research, where LLMs are increasingly adopted [15], and highlights the need to investigate their potential in comment evaluation.

Problem and Objective

Research Problem

Problem Statement: Despite advances in smart contract comment generation, evaluation practices remain limited to surface-level metrics or manual human judgment. This lack of semantic depth and scalability highlights a critical gap: the potential of large language models (LLMs) as evaluators remains unexplored.
Objective: To design a tool that uses large language models (LLMs) as automated evaluators for smart contract comment generation, aiming to enhance evaluation depth and scalability, while addressing smart contract-specific concerns.

III. System Overview: evalSmarT

We present `evalSmarT`, a modular framework for evaluating automatically generated comments for smart contracts. It leverages the *LLM-as-a-Judge* paradigm to assess comment quality. The system is designed to be model-agnostic and can integrate any large language model accessible via local deployment (e.g., through Ollama) or remote APIs (e.g., via OpenRouter). It supports flexible prompt engineering and evaluation workflows, enabling researchers and practitioners to experiment with diverse model–prompt configurations tailored to smart contract documentation.

`evalSmarT` is implemented using both local (Ollama) and remote (OpenRouter) LLM access, and supports evaluation of comments generated by tools such as SCCLLM, MMTrans, and CCGIR.

A. Evaluation Metrics

The evaluators assess comment quality across four dimensions: accuracy, completeness, clarity, and helpfulness. The helpfulness metric incorporates audience-specific utility assessment, acknowledging the heterogeneous stakeholder landscape in smart contract ecosystems.

Evaluation Metrics

1. Accuracy (0–100)
2. Completeness (0–100)
3. Clarity (0–100)
4. Helpfulness: Identify which audiences would find the comment useful from:
- developer_maintaining_contract
- developer_reusing_code
- developer_integrating_contract
- non_technical_user
- business_analyst

B. Prompting Strategies and Evaluation Protocol

We define an LLM evaluator as a tuple $\langle M, P \rangle$, where M is the model and P is the prompt template. Prompts are designed to incorporate:

- **Domain knowledge** (e.g., blockchain-specific logic, permission enforcement)
- **Language features** (e.g., Solidity constructs, modifiers, events)
- **Evaluation framing** (e.g., QA-based reasoning)

TABLE II: Prompting Strategy Design Matrix

Prompt	Domain	Language	QA
P1: Baseline	No	No	No
P2: Domain-aware	Yes	No	No
P3: Language-aware	No	Yes	No
P4: Baseline + QA	No	No	Yes
P5: Domain + QA	Yes	No	Yes
P6: Language + QA	No	Yes	Yes
P7: Unguided Domain	Min	No	No
P8: Unguided Language	No	Min	No
P9: Domain + Language + QA	Yes	Yes	Yes
P10: Domain + Language	Yes	Yes	No

Legend: Yes = Full integration; Min = Minimal guidance; No = Not applied

evalSmart

Type: a modular and extensible framework for evaluating automatically generated comments for smart contracts.
Components: Multiple LLM evaluators with diverse prompt strategies;
Coverage: Around 400 evaluators: 40 LLMs with 10 prompting strategies
Metrics accuracy, completeness, clarity, and helpfulness.

IV. Demonstration and Use Cases

We demonstrate `evalSmarT` through the evaluation of a smart contract comment generation tool. The system loads a set of (code,comment) pairs produced by the tool, applies multiple LLM-based evaluators, and generates structured assessments across four dimensions: accuracy, completeness, clarity, and helpfulness. This process illustrates how `evalSmarT` can be used to benchmark the performance of comment generation models in a reproducible and scalable manner.

Beyond this core demonstration, `evalSmarT` enables several practical and research-oriented use cases:

A. Benchmarking for Research

Researchers can use `evalSmarT` to compare outputs from multiple comment generation tools. The framework supports up to 400 evaluator configurations (based on approximately 40 LLMs and 10 prompt strategies), allowing for fine-grained analysis of model performance under varied evaluation conditions.

B. Best Output Selection

In practical settings, `evalSmarT` ranks multiple generated comments and selects the most appropriate one. This supports developers in choosing the most informative and accurate documentation, especially when integrating or maintaining smart contracts.

C. Prompt and Evaluator Extension

The system is designed to be extensible. Users can add new prompts or integrate additional models, enabling continuous refinement of evaluation strategies and adaptation to emerging documentation needs in blockchain development.

Demonstration Summary

The demonstration showcases how `evalSmarT` evaluates the output of a smart contract comment generation tool. A set of code–comment pairs is loaded, and multiple LLM-based evaluators are applied to assess the quality of the generated comments. The system outputs structured scores and justifications across four evaluation dimensions. This demonstration highlights the tool's ability to benchmark models, select the most informative comment, and support reproducible, scalable evaluation workflows.

V. Illustrative Evaluation and Insights

To support the demonstration of `evalSmarT`, we conducted a focused experiment showcasing its evaluation capabilities on real-world smart contract summaries. Rather than presenting an exhaustive benchmark, we aim here to illustrate how the tool operates in practice and to highlight the rationale behind its internal evaluator configuration.

A. Selecting the Default Evaluator

We experimented with 40 evaluator configurations (10 prompts × 4 LLMs). Among these, the combination of GPT-4 and prompt P6 (language-aware + QA framing) achieved the best alignment with human expert annotations across accuracy, completeness, clarity, and helpfulness dimensions. This configuration serves as the default evaluator in our demonstration.

B. Example: Comparing SCCLLM and CCGIR

To illustrate the tool in use, we applied `evalSmarT` to evaluate comments generated by two state-of-the-art smart contract summarization systems: SCCLLM [3] and CCGIR [2]. We collected real-world smart contract functions from Etherscan, processed them through both tools, and used our selected evaluator (GPT-4 with prompt P6: language-aware + QA) to assess the quality of the generated comments.

The example showcases how `evalSmarT` facilitates structured comparison between models across the four evaluation dimensions, while also providing audience-specific helpfulness annotations.

`evalSmarT` Configuration for Model Comparison

LLM Component: GPT-4
Prompt: P6–Language-aware-QA-framing
Smart Contract Source: Real-world contracts collected from Etherscan
Generated Comments: Outputs from SCCLLM and CCGIR
Expected Output: Structured evaluation across accuracy, completeness, clarity, and helpfulness, including identification of relevant audiences

C. Findings

The evaluation revealed clear differences in the performance of the two systems. SCCLLM produced more accurate and complete comments, with better alignment to contract semantics and audience needs. In contrast, CCGIR showed notable weaknesses, particularly when evaluated on smart contracts that differed significantly from its training distribution.

TABLE III: Evaluation scores for SCCLLM using the GPT-4 + P6 (Language-aware + QA) evaluator.

Acc.	Comp.	Clar.	Overall	Mnt.	Reuse	Integr.	NonTech	Analyst
88.53	73.90	96.22	86.22	0.97	0.99	0.76	0.02	0.06

The results in Table III highlight the strong performance of SCCLLM when evaluated using the GPT-4 + P6 (Language-aware + QA) configuration. The generated comments exhibit high scores in clarity (96.22) and accuracy (88.53), with a slightly lower but still solid score in completeness (73.90). These scores contribute to a robust overall evaluation average of 86.22.

From an audience-specific perspective, the comments are especially helpful for developers maintaining (0.97) or reusing (0.99) the contract, and to a lesser extent for those integrating (0.76) it. However, the helpfulness drops significantly for non-technical users (0.02) and business analysts (0.06). This indicates that while SCCLLM produces highly accurate and readable comments, its utility remains concentrated among technically proficient users. Broader accessibility would likely require additional language simplification or audience-specific tailoring.

TABLE IV: Evaluation scores for CCGIR using the GPT-4 + P6 (Language-aware + QA) evaluator.

Acc.	Comp.	Clar.	Overall	Mnt.	Reuse	Integr.	NonTech	Analyst
10.00	6.00	57.00	24.33	0.40	0.40	0.40	0.00	0.00

The evaluation scores for CCGIR (Table IV) reveal significantly lower performance compared to SCCLLM. Accuracy (10.00) and completeness (6.00) are particularly low, indicating that the generated comments often fail to correctly and fully describe the smart contract functions. While clarity (57.00) is somewhat better, it remains moderate, suggesting the comments are not sufficiently clear or informative.

Regarding audience-specific helpfulness, CCGIR's comments show limited utility for developers maintaining, reusing, or integrating the contracts, with only 40

These findings highlight CCGIR's limitations in generating high-quality and audience-tailored comments for smart contracts, especially when compared to the stronger performance of SCCLLM under the same evaluation conditions.

Summary of Findings

SCCLLM [3]:
Generated comments were generally accurate, complete, and clear. Helpfulness tags showed relevance for technical audiences, especially developers reusing or maintaining contracts.

CCGIR [2]:
Comments lacked precision and completeness, often missing key logic. Performance degraded significantly when the input contracts diverged from the types seen during training.

Conclusion:
`evalSmarT` revealed that SCCLLM generalizes better to unseen contracts, while CCGIR struggles with out-of-distribution functions. This highlights the importance of evaluation tools that account for generalization, semantic accuracy, and audience relevance.

Tool Summary Snapshot

Users: Researchers and practitioners working on smart contract comprehension and comment generation.
Challenge: Lack of semantic, scalable, and domain-aware evaluation of smart contract comments.
Method: LLM-based evaluators using customizable prompt-model configurations for structured evaluation.
Validation: Successfully benchmarked SCCLLM and CCGIR; findings align with human judgment and reveal evaluator sensitivity to prompt design.

VI. Conclusion

We presented `evalSmarT`, a modular framework leveraging the *LLM-as-a-Judge* paradigm for evaluating smart contract generated comment. The tool enables structured, scalable assessment across multiple quality dimensions by integrating a wide range of LLMs and prompt strategies. Through a use case with SCCLLM and CCGIR, we demonstrated `evalSmarT` capabilities for benchmarking, best-output selection, and flexible evaluator configuration. The tool is open-source and readily usable by both researchers and practitioners. Future work will include a more comprehensive evaluation campaign and integration of fine-tuned domain-specific evaluators.

VII. Acknowledgments

This work is supported by the Luxembourg Ministry of Foreign and European Affairs through their Digital4Development (D4D) portfolio under the project LuxWAyS (Luxembourg/West-Africa Lab for Higher Education Capacity Building in CyberSecurity and Emerging Topics in ICT4Dev.)

References

[1] A. Pinna, S. Ibba, G. Baralla, R. Tonelli, and M. Marchesi, "A massive analysis of ethereum smart contracts empirical study and code metrics," *Ieee Access*, vol. 7, pp. 78 194–78 213, 2019.

[2] G. Yang, K. Liu, X. Chen, Y. Zhou, C. Yu, and H. Lin, "Ccgir: Information retrieval-based code comment generation method for smart contracts," *Knowledge-Based Systems*, vol. 237, p. 107858, 2022.

[3] J. Zhao, X. Chen, G. Yang, and Y. Shen, "Automatic smart contract comment generation via large language models and in-context learning," *Information and Software Technology*, vol. 168, p. 107405, 2024.

[4] S. Lubos, A. Felfernig, T. N. T. Tran, D. Garber, M. El Mansi, S. P. Erdeniz, and V.-M. Le, "Leveraging llms for the quality assurance of software requirements," in *2024 IEEE 32nd International Requirements Engineering Conference (RE)*, 2024, pp. 389–397.

[5] Y. Wu, Y. Wan, Z. Chu, W. Zhao, Y. Liu, H. Zhang, X. Shi, and P. S. Yu, "Can large language models serve as evaluators for code summarization?" *arXiv preprint arXiv:2412.01333*, 2024.

[6] M. Li, J. Weng, A. Yang, J. Weng, and Y. Zhang, "Towards interpreting smart contract against contract fraud: A practical and automatic realization," *Cryptology ePrint Archive*, 2020.

[7] X. Li, T. Chen, X. Luo, T. Zhang, L. Yu, and Z. Xu, "Stan: Towards describing bytecodes of smart contract," in *2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2020, pp. 273–284.

[8] Z. Yang, J. Keung, X. Yu, X. Gu, Z. Wei, X. Ma, and M. Zhang, "A multi-modal transformer-based code summarization approach for smart contracts," in *2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC)*. IEEE, 2021, pp. 1–12.

[9] X. Hu, Z. Gao, X. Xia, D. Lo, and X. Yang, "Automating user notice generation for smart contract functions," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2021, pp. 5–17.

[10] C. Shi, Y. Xiang, J. Yu, K. Sood, and L. Gao, "Machine translation-based fine-grained comments generation for solidity smart contracts," *Information and Software Technology*, vol. 153, p. 107065, 2023.

[11] Y. Mao, X. Li, W. Li, X. Wang, and L. Xie, "Scla: Automated smart contract summarization via llms and semantic augmentation," *arXiv preprint arXiv:2402.04863*, 2024.

[12] G. Lei, D. Zhang, J. Xiao, G. Fan, Y. Cao, and Z. Feng, "Fmcf: A fusing multiple code features approach based on transformer for solidity smart contracts source code summarization," *Applied Soft Computing*, vol. 166, p. 112238, 2024.

[13] J. Xiang, Z. Gao, L. Bao, X. Hu, J. Chen, and X. Xia, "Automating comment generation for smart contract from bytecode," *ACM Transactions on Software Engineering and Methodology*, 2024.

[14] Z. Zhang, S. Chen, G. Fan, G. Yang, and Z. Feng, "Ccgra: Smart contract code comment generation with retrieval-enhanced approach." in *SEKE*, 2023, pp. 212–217.

[15] Z. He, Z. Li, S. Yang, H. Ye, A. Qiao, X. Zhang, X. Luo, and T. Chen, "Large language models for blockchain security: A systematic literature review," *arXiv preprint arXiv:2403.14280*, 2024.