Automated Smart Contract Summarization via LLMs

Yingjie Mao
Hainan University
Haikou, China
maoyingjie12138@gmail.com

Xiaoqi Li*
Hainan University
Haikou, China
csxqli@gmail.com

Zongwei Li
Hainan University
Haikou, China
lizw1017@gmail.com

Wenkai Li
Hainan University
Haikou, China
liwenkai871@gmail.com

ABSTRACT

Automatic code Summarization generation technology is widely used in the development and maintenance of smart contracts. In recent years, with the advent of Large Language Models (LLMs), Gemini has received a lot of attention as the first Large Multimodal Models (LMMs) to support multimodal input. However, it is unclear how LMMs can generate contract code summarization from multimodal inputs. In this paper, we focus on evaluating Gemini on real-world smart contracts, comparing it to the MMTrans, and exploring how to combine multimodal prompts to generate a contract code summarization. We adopt several widely used metrics (BLEU, METEOR, and ROUGE-L) to measure the quality of the generated summarization. Our experiments show that Gemini-Pro-Vision achieves 21.17% and 21.05% scores for code comments generated by three-shot prompts under METEOR and ROUGE-L metrics. The above scores are better than those generated by one-shot and five-shot prompts.

KEYWORDS

Gemini, Smart Contract, Code Summarization

1 INTRODUCTION

Smart contracts[1] are automatically executed contract terms running on the Ethereum system[2]. Due to the immutability of the blockchain system[3], it is complicated to maintain and modify the vulnerabilities of smart contracts. smart contract code comments can help developers understand the contract's logic and functionality and are considered an effective method to reduce the risk of smart contract vulnerabilities. However, he et al.[4]found that misuse of uncommented code is a significant cause of 10% of vulnerabilities. Yang et al.[5] have observed that most smart contract comments are unavailable. Therefore, it is necessary to use automatic code summarization [6, 7] to automatically generate a natural language summarization for the specified contract code.

In recent years, smart contract code summarization generation technology based on natural language processing has gained wide attention. For example, Yang et al. [5] proposed an approach that uses multimodal learning techniques to take the SBT sequence[8] of the contract and the graph based on the abstract syntax tree as the input to the model. Then, MMTrans uses two encoders to extract semantic information from the two inputs and uses a joint encoding to generate code comments.

*The corresponding author

Most recently, a new LMMs called Gemini[9] based on multimodal technology has aroused great interest in natural language processing, compared with previous DL. Gemini-Pro-Vision supports more significant input tokens and multimodal input of pictures and text, which significantly improves the performance of processing various natural language tasks[10–12]. In addition to processing natural language, Gemini-Pro-Vision can also process source code. However, the current exploration of Gemini-Pro-Vision in generating code comments is still lacking.

In this paper, we evaluate the performance of the Gemini-Pro-Vision model for generating code summarization with one-shot, three-shot, and five-shot prompts, and compare the generated code summarization with the currently common SOTA. We also explore methods to build the best prompt for multimodal inputs. Furthermore, we evaluate the performance of the Gemini-Pro-Vision model for generating code summarization with one-shot, three-shot, and five-shot prompts. We use three widely used metrics (i.e., Bleu, Meter, Rouge-L), to measure the quality of the summarization generated by Gemini-Pro-Vision and to investigate Gemini-Pro-Vision's code performance summarization capabilities. We also compare the results obtained by MMTrans[5].

In summary, we make the following contributions:

- To the best of our knowledge, we are the first to study and analyze Gemini-Pro-Vision's performance in smart contract code summarization.
- We evaluated Gemini-Pro-Vision on a widely used smart contract code summarization dataset and compared it to other baselines.
- Based on the above research, we summarize several challenges and opportunities for Gemini-Pro-Vision code summarization generation.
- We open-source our experimental data and codes on https://doi.org/10.6084/m9.figshare.25144871

The rest of this paper is organized as follows. Section 2 shows the background and details of our study. Section 3 presents our study design, including research ideas, research questions, performance measurements, baselines, and datasets used for experiments. Section 4 presents the analysis of our results for the research questions. Section 5 discusses the advantages and limitations of our current research direction. Finally, Section 6 concludes our study and shows potential future directions.

2 BACKGROUND

2.1 Large Multimodal Models (LMMs)

As LLMs (e.g., LLAMA[13]) technology advances, LMMs have been inspired by many of its advantages, and LLMs provide a promising approach to AGI. These models combine the linguistic textual reasoning capabilities of LLMs with the image and video capabilities of models. As a result, LLMs can deal with more complex tasks that require deep understanding and expressive generation across various modalities. Open-source models such as LLAva[14] have demonstrated their ability for tasks such as image, visual, and text question answering.

However, the architecture of these models limits their ability to understand and reason about a single image. Conversely, OPENFLAMINGO[15] models can better represent real-world scenes due to their dedicated architecture, which allows them to handle multiple image features compared to their predecessors.

2.2 Multimodal Prompting Methods

In the field of LLMs, a variety of prompt language methods have been established to increase the reasoning ability of the model and ensure the accuracy of the pre-stored results. such as zero-shot[16], few-shot[17], and Chain-of-Thought (CoT)[18]. In recent years, the application of prompting technology in multimodal models has been studied to enhance the understanding and reasoning ability of LLMs multi-image data and text data. At present, multimodality in LMMs still has limitations in capturing the complex relationship between visual information and text information, especially when multiple pictures are input.

2.3 Chain-of-Thought (CoT)

The CoT methodology can significantly enhance the performance of LLMs when dealing with complex reasoning, and this principle holds in the realm of LLMs. For instance, Chancharik et al.[19] introduced a Compositional Chain of Thought (CCoT) as an approach to enhance LLMs' performance. CCoT can be divided into two steps. The first step involves generating scene graphs without relying on ground truth SG data. This is achieved using input images and task prompts, such as visual questions. The second step prompts the LLMs to generate a response using the image, task cue, and the previously generated scene graph. Incorporating scene graphs into prompts eliminates the need for fine-tuning and prevents forgetting.

3 STUDY DESIGN

3.1 Research Questions (RQs)

We aim to answer the following questions through experimental evaluation.

RQ1:What does the comment generated by Gemini-Pro-Vision look like?

RQ2:How does Gemini-Pro-Vision perform with different multimodal input prompt?

To answer RQ1, we want to look at what the comments generated by Gemini-Pro-Vision look like for a special code snippet. We use Fig. 1 as an example, use the screenshot of the code in Fig. 1 as an input image, and ask Gemini-Pro-Vision to generate code comments for the code in Fig. 1.

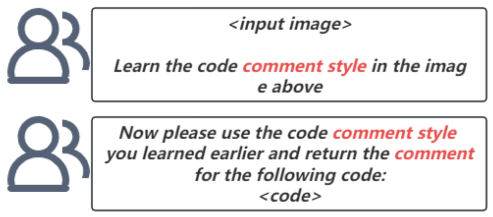
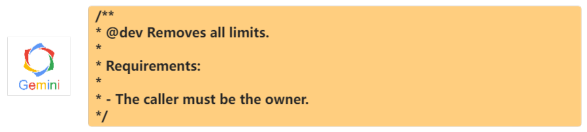

<input image>
Learn the code comment style in the imag e above
Now please use the code comment style you learned earlier and return the comment for the following code:
<code>

Figure 1: Prompt P1 for RQ1.

```
function removeLimits() external onlyOwner {
    limitsInEffect = false;
    transferDelayEnabled = false;
    emit RemovedLimits();
}
```
Figure 2: Code snippet of the detected contract



```
/**
* @dev Removes all limits.
*
* Requirements:
*
* - The caller must be the owner.
*/
```

Figure 3: Code comment generated by Gemini-Pro-Vision.

Fig. 3 shows what a comment generated by Gemini-Pro-Vision for a specific code snippet looks like. It can be seen that Gemini-Pro-Vision's reply only focuses on the result itself and does not include other redundant information.

RQ2 aims to evaluate the quality of comments generated by Gemini-Pro-Vision for specific code snippets through different Few-shot scenarios. To answer RQ2, We mainly studied the performance of Gemini-pro-vision with one-shot, three-shot, and five-shot prompts through experimental data.

3.2 Baseline

We compare our proposed method with the widely used MMTrans approach, meanwhile mainly comparing the BLEU, METEOR, and ROUGE-L scores and the length of comments. We chose the MMTrans approach because his research method and source code are all open source.

3.3 Evaluation Metrics

To evaluate the performance of Geminis pro-vision against the baselines, we decided to adopt several widely used automated evaluation metrics, such as BLEU, METEOR, and ROUGE-L. The above-automated evaluation criteria can effectively measure the familiarity between the generated self-powered contract code comments and the real-world comments vocabulary and have been widely used in smart contract code comments[5]. We show the details of these criteria as follows.

- **BLEU**. BLEU[20] (Bilingual Evaluation Understudy) is a metric to evaluate machine translation, which measures the matching degree of the corresponding n-gram content in the standard translation by the overlap calculation of n-grams. In this experiment, we use BLEU-4 to measure the quality of smart contracts generated intelligently.
- **METER**. METER[21](Metric for Evaluation of Translation with Explicit ORdering) is also a metric to evaluate machine translation. However, METER first creates a sentence alignment between them and then calculates the similarity between the sentences.
- **ROUGE-L**. ROUGE-L[22] A ROUGE variant calculates the similarity of two sentences by the length of the longest common subsequence (LCS).

3.4 DATASET

We used the so-comment dataset from the SCCSD-Solidity. After we filtered it, we got 22,000 valuable pieces of data and conducted experiments.

4 RESULTS

Table 1: Compare the word count of Gemini-Pro-Vision generated comments with the word count of real-world comments.

Method	SCCSD-Solidity			
	Median	Average	Maximum	Minimum
Gemini-Pro-Vision (one-shot)	9	12	173	1
Gemini-Pro-Vision (three-shot)	10	14	240	1
Gemini-Pro-Vision (five-shot)	10	14	276	1
Real-world	8	9	188	2

Table 2: Comparison results between Gemini-Pro-Vision and baselines in terms of three performance measures.

Method	SCCSD-Solidity		
	BLEU	METEOR	ROUGE-L
Gemini-Pro-Vision (one-shot)	15.38 %	20.41 %	19.99 %
Gemini-Pro-Vision (three-shot)	15.31 %	21.17 %	21.05 %
Gemini-Pro-Vision (five-shot)	14.29 %	19.48 %	20.00 %
MMTrans	34.14 %	48.56 %	43.24 %

In this section, to answer RQ2, we will compare Gemini-Pro-Vision's comments generated by one-shot, three-shot, and five-shot prompts with MMTrans's comments generated by 22000 pairs of code-comment in the SCCSD-Solidity dataset. We consistently use the prompt in Fig. 1 to guide Gemini-Pro-Vision to generate code comments for the 22000 code fragments in the data test collection.

Table. 1 shows the difference between the comment length generated by Gemini-Pro-Vision with one-shot, three-shot, and five-shot prompts and the actual review length in the real world. It can be observed that the median and average length of comments generated by one-shot prompts are lower than those generated by three-shot and five-shot prompts, which are only 9 words and 12 words respectively. The maximum length of the generated annotation is only 173 words, which is far less than three-shot and five-shot prompts. Although the comment length generated by Gemini-Pro-Vision with a one-shot prompt is lower than that generated by a three-shot and five-shot, there is still a certain gap between Gemini-Pro-Vision and real-world on average, but the gap is small in the median

Table. 2 shows the overall performance of Gemini-Pro-Vision in one-shot, three-shot, and five-shot compared with MMTrans. It can be observed from the table that Gemini-Pro-Vision has the highest scores for METEOR and ROUGE-L with three-shot and five-shot prompts, while the BLEU score is slightly worse than that of one-shot. In comparison with MMTrans, it is evident that MMTrans significantly outperforms Gemini in three metrics: METEOR, BLEU, and ROUGE-L.

Fig. 4 shows the score breakdown of the comments generated by Gemini-Pro-Vision on the SCCSD-Solidity dataset with a one-shot, three-shot, and five-shot prompts. According to Fig. 4, we can see that the median METEOR, BLEU, and ROUGE-L scores obtained by Gemini-Pro-Vision with one-shot, three-shot, and five-shot prompts have little difference. The score of three-shot is higher than that of one-shot and five-shot under METEOR and ROUGE-L.

5 DISCUSSION

In this section, we summarize the current advantages and limitations of Gemini-Pro-Vision for generating code comments, as well as future expectations.

5.1 Benefit

- **More concise**. The code comments produced by Gemini-Pro-Vision exhibit a more robotic style than GPT-4. They contain fewer redundant words or phrases in the generated code summary, allowing for immediate use without trimming. As illustrated in Fig. 3, we prompted both Gemini-Pro-Vision and GPT-4 to generate code comments for the provided code example. The results, shown in Fig. 5, depict the distinct styles of the generated code comments. Gemini-Pro-Vision's responses are notably concise, and its ability to incorporate image input facilitates learning the code comment style from boilerplate images.
- **More stronger reasoning ability**. Gemini-Pro-Vision can integrate multiple pieces of image information for reasoning, enhancing its ability to learn and understand real code comment styles through visual cues. This feature contributes to improving the accuracy and readability of the generated code comments, bringing them closer to the authentic style observed in real-world comments.

5.2 Limitation

- **Lack of high-quality benchmark dataset**. The data in the SCCSD-Solidity dataset is outdated, spanning back 21 years. Given the rapid evolution of Solidity in recent years, the comments within SCCSD-Solidity lack coverage of the language's new features. Additionally, the majority of code comments are brief and fail to offer a comprehensive overview of the

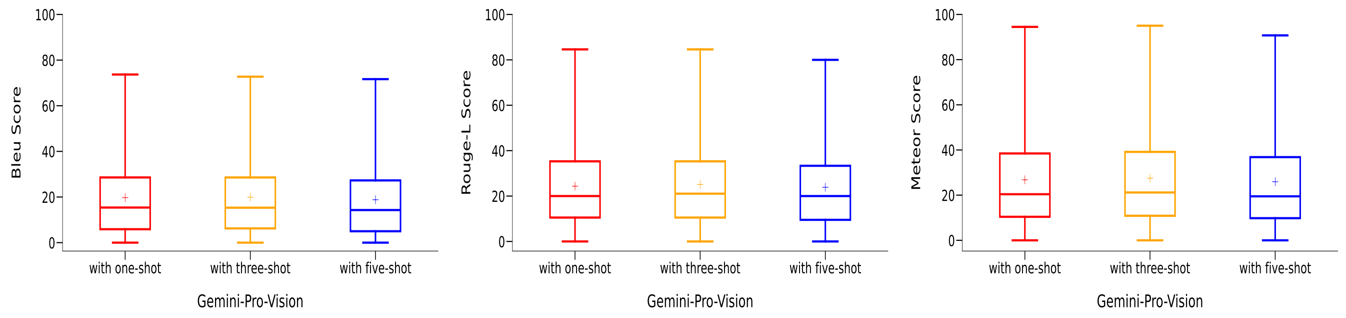
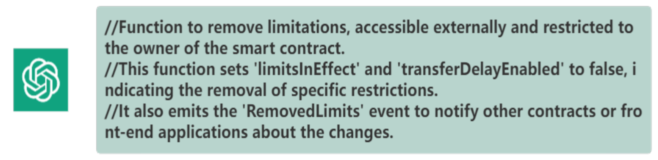
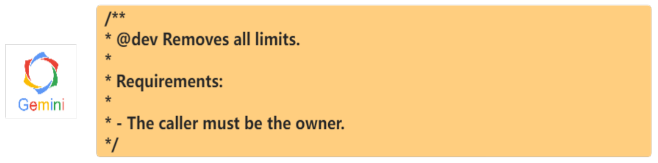


Figure 4: Performance metric scores for Comments generated by Gemini.



//Function to remove limitations, accessible externally and restricted to the owner of the smart contract.
//This function sets 'limitsInEffect' and 'transferDelayEnabled' to false, i ndicating the removal of specific restrictions.
//It also emits the 'RemovedLimits' event to notify other contracts or fro nt-end applications about the changes.

(a) Code comments generated by GPT4



```
/**
* @dev Removes all limits.
*
* Requirements:
*
* - The caller must be the owner.
*/
```

(b) Code comments generated by GEMINI

Figure 5: Comment generated by Gemini-Pro-Vision and GPT-4.

code's functionalities. Hence, there is a significant need to explore further and invest in constructing a high-quality test dataset for Gemini-Pro-Vision, specifically focusing on code comment generation.

- **Lack of metrics**. Currently, there is no suitable metric for evaluating comments generated by LLMs such as Gemini-Pro-Vision. Traditional metrics such as METEOR, BLEU, and ROUGE-L, which are generally used in machine translation, are not suitable for evaluating the quality of comments generated by Gemini-Pro-Vision.

6 CONCLUSION

In this paper, we formulate heuristic questions to investigate a suitable multimodal approach for assessing the quality of Gemini-Pro-Vision-generated code comments. We compare the performance of Gemini-Pro-Vision and MMTrans, a dedicated code summarization model, on the SCCSD-Solidity dataset for code summarization generation. Our findings reveal that, in terms of METEOR, BLEU, and ROUGE-L metrics, Gemini-Pro-Vision is inferior to MMTrans on the SCCSD-Solidity dataset. However, Gemini-Pro-Vision generates slightly higher-quality code comments with three-shot prompts compared to one-shot and five-shot prompts. Based on our findings, we outline two opportunities and adjustments for utilizing Gemini-Pro-Vision to generate code comments. We hope that our results and observations contribute to future work in the field of generating code comments with Gemini-Pro-Vision.

REFERENCES

[1] Shenhui Zhang, Wenkai Li, Xiaoqi Li, and Boyi Liu. Authros: Secure data sharing among robot operating systems based on ethereum. *IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS)*, pages 147–156, 2022.
[2] Xiaoqi Li, Ting Chen, Xiapu Luo, and Jiangshan Yu. Characterizing erasable accounts in ethereum. In *Information Security: 23rd International Conference, ISC, Bali, Indonesia, December 16–18, Proceedings 23*, pages 352–371. Springer, 2020.
[3] Xiaoqi Li, Ting Chen, Xiapu Luo, and Chenxu Wang. Clue: towards discovering locked cryptocurrencies in ethereum. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pages 1584–1587, 2021.
[4] Ningyu He, Lei Wu, Haoyu Wang, Yao Guo, and Xuxian Jiang. Characterizing code clones in the ethereum smart contract ecosystem. In *Financial Cryptography and Data Security*, pages 654–675, 2020.
[5] Zhen Yang, Jacky Keung, Xiao Yu, Xiaodong Gu, Zhengyuan Wei, Xiaoxue Ma, and Miao Zhang. A multi-modal transformer-based code summarization approach for smart contracts. In *IEEE/ACM 29th International Conference on Program Comprehension (ICPC)*, pages 1–12. IEEE, 2021.
[6] Tong Ye, Lingfei Wu, Tengfei Ma, Xuhong Zhang, Yangkai Du, Peiyu Liu, Shouling Ji, and Wenhai Wang. CP-BCS: Binary code summarization guided by control flow graph and pseudo code. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 14740–14752, 2023.
[7] Aakash Bansal, Siyuan Jiang, Sakib Haque, and Collin McMillan. Statement-based memory for neural source code summarization. *arXiv preprint arXiv:2307.11709*, 2023.
[8] Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. Deep code comment generation. In *Proceedings of the 26th conference on program comprehension*, pages 200–210, 2018.
[9] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
[10] Zongwei Li, Dechao Kong, Yuanzheng Niu, Hongli Peng, Xiaoqi Li, and Wenkai Li. An overview of ai and blockchain integration for privacy-preserving. *arXiv preprint arXiv:2305.03928*, 2023.
[11] Daoan Zhang, Junming Yang, Hanjia Lyu, Zijian Jin, Yuan Yao, Mingkai Chen, and Jiebo Luo. Cocot: Contrastive chain-of-thought prompting for large multimodal models with multiple image inputs. *arXiv preprint arXiv:2401.02582*, 2024.
[12] Yuqing Wang and Yun Zhao. Gemini in reasoning: Unveiling commonsense in multimodal large language models. *arXiv preprint arXiv:2312.17661*, 2023.
[13] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
[14] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.
[15] Anas Awadalla, Irena Gao, Josh Gardner, Jack Hessel, Yusuf Hanafy, Wanrong Zhu, Kalyani Marathe, Yonatan Bitton, Samir Gadre, Shiori Sagawa, et al. Open-flamingo: An open-source framework for training large autoregressive vision-language models. *arXiv preprint arXiv:2308.01390*, 2023.
[16] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2022. *URL https://arxiv.*

org/abs/2205.11916.

[17] Xin Liu, Daniel J. McDuff, G. Kovács, Isaac R. Galatzer-Levy, Jacob Sunshine, Jiening Zhan, Ming-Zher Poh, Shun Liao, Paolo Di Achille, and Shwetak N. Patel. Large language models are few-shot health learners. *ArXiv*, abs/2305.15525, 2023.

[18] Ge Zheng, Bin Yang, Jiajin Tang, Hong-Yu Zhou, and Sibei Yang. Ddcot: Duty-distinct chain-of-thought prompting for multimodal reasoning in language models. *arXiv preprint arXiv:2310.16436*, 2023.

[19] Chancharik Mitra, Brandon Huang, Trevor Darrell, and Roei Herzig. Compositional chain-of-thought prompting for large multimodal models. *arXiv preprint arXiv:2311.17076*, 2023.

[20] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[21] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.

[22] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.