Using Silver-Standard Machine Learning Models to Determine Usefulness of C Comments

Aritra Mitra[1,*]

[1]Indian Institute of Technology, Kharagpur (IIT-KGP), West Bengal-721302, India

Abstract

Comments are very useful to the flow of code development. With the increasing use of code in commonplace life, commenting the codes becomes a hassle for rookie coders, and often they do not even think commenting as a part of the development process. This in general causes the quality of comments to degrade, and a considerable amount of useless comments are found in such codes. In these experiments, the usefulness of C comments are evaluated using LLM-generated silver standard machine learning models. The results of the experiments create a baseline for better results that can be found in the future through more research. Based on these findings, more complex and accurate machine learning models can be created that can improve the accuracy achieved in performing said task.

Keywords

Model Generation, Large Language Model, Machine Learning, Natural Language Processing,

1. Introduction

Comments provide valuable insights within codebases, thereby facilitating easier maintenance. However, when comments are verbose, irrelevant, or unhelpful, they can contribute to storage overhead without offering real benefit. With the rising prevalence of programming, many novice developers tend to overlook proper commenting practices, leading to a reduction in both the quality and quantity of comments [1]. As a result, large portions of comments often turn out to be uninformative, and sifting through them during code maintenance can be inefficient or even counterproductive.

Several deep learning–based approaches for automatic commenting aim to reduce the volume of comments [2]. Yet, research specifically addressing the quality of comments remains limited. Recent efforts have begun addressing these challenges by applying machine learning models capable of detecting and categorizing comments based on their usefulness.

The author explores a range of machine learning (ML) models to tackle this problem. This study aims to investigate key questions in the context of the Information Retrieval in Software Engineering (IRSE) shared task at the Forum for Information Retrieval Evaluation (FIRE) 2025.

- What level of complexity is necessary for an ML model to reliably distinguish between useful and non-useful comments?
- With the growing capabilities of large language models, can they be effectively applied to this task?

This work proposes that LLM-based models can serve as strong baseline approaches for the problem. More advanced models may be built upon these foundations, while remaining mindful of the risks of overfitting.

This study also investigates the important question of whether the dataset for the learning task can be augmented with data generated by large language models, such as GPT-3. This consideration is key to exploring the potential of leveraging artificial intelligence to improve comment quality evaluation. The research analyzes the impact of incorporating AI-generated data into the dataset, aiming to clarify both the benefits and challenges of integrating advanced language models into the machine learning

Forum for Information Retrieval Evaluation, December 17-20, 2025, India

*Corresponding author.

aritramitra2002@gmail.com (A. Mitra)

workflow. This aspect represents a central component of the study, emphasizing the intersection of human-authored and AI-generated content in the context of assessing comment quality.

2. Related Work

Software metadata [3] plays a crucial role in code comprehension and maintenance. Various tools have been developed to extract knowledge from software metadata, including runtime traces and structural code properties [4, 5, 6, 7, 8, 9, 10, 11, 12].

Several studies have explored the quality of code comments in the context of mining. Steidl et al. [13] employ techniques such as Levenshtein distance and comment length to measure similarity in code-comment pairs, effectively filtering out trivial or non-informative comments. Rahman et al. [14] focus on distinguishing valuable from inconsequential code review comments on review platforms, leveraging insights from attributes identified in a survey conducted with Microsoft developers [15]. Majumdar et al. [16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26] propose a framework to evaluate comments based on key concepts necessary for code understanding. Their approach constructs textual and code correlation features and utilizes a knowledge graph to semantically analyze the content of comments.

These approaches combine semantic and structural features to address the predictive challenge of identifying useful versus unhelpful comments, thereby supporting codebase cleaning. With the emergence of large language models such as GPT-3.5 and LLaMA, assessing the quality of code comments in comparison with human judgment has become increasingly important. The IRSE track at FIRE 2024 extends the methodology introduced in prior work [16]. This study examines various vector space models and feature sets for binary classification of comments, emphasizing their relevance to code comprehension. Additionally, it conducts a comparative analysis of model performance when incorporating GPT-generated labels for code and comment quality derived from open-source projects. Other studies [27, 28] explore the similar aspects of LLMs for the ongoing research.

3. Description of Task and Dataset

This section provides an overview of the task and its corresponding dataset. The IRSE assignment at FIRE 2025 was defined as follows:
Enhancing a binary code comment quality classification model by incorporating generated code-comment pairs to improve its accuracy.

The dataset associated with this task was organized into two components:

Training dataset: Contains 8,048 samples.

Testing dataset: Contains 1,000 samples.

The training dataset was shuffled and split into 70% for model training and 30% for cross-validation. The data was labeled as follows:

- **Useful**: Comments that contribute to understanding the code
- **Not Useful**: Comments that do not aid code comprehension

4. Augmentation

The dataset augentation was performed using GPT-4o-mini and GPT-3.5-Turbo. The augmented dataset was subsequently input into CodeT5, GPT-4o-mini, GPT-3.5-Turbo, and Code-LLaMA to generate labels for the data. Three distinct prompting strategies were employed for this task, described as follows:

Table 1
Description of the Dataset for the Task

Label	Example
Useful	/*not interested in the downloaded bytes, return the size*/
Useful	/*Fill in the file upload part*/
Not Useful	/*The following works both in 1.5.4 and earlier versions:*/
Not Useful	/*lock_time*/

I/O Prompting: In this approach, the LLM performs both labeling and data generation using a roleplay-based method, providing only the requested label or generated data without any additional content in its response.

Chain of Thoughts (CoT) Prompting: [29] Here, the LLM produces the desired output along with an explanation of the reasoning process used to arrive at the answer. This encourages the generation of more coherent and meaningful data, though it necessitates additional processing of the response.

Tree of Thoughts (ToT) Prompting: [30, 31] In this method, the LLM generates multiple candidate responses for a task. Subsequent questions are then asked based on these responses, forming a tree of thoughts. The optimal output can be selected from the leaves of this tree, improving accuracy but requiring substantial post-processing.

5. System Description

5.1. Text Preprocessing

All hyperlinks, punctuation marks, numerals, and stop words were removed. Next, words with POS tags other than Noun, Verb, Adverb, or Adjective were discarded. Lemmatization was applied to merge different forms of a word into a single canonical term, using NLTK WordNet [32]. The same preprocessing steps were applied to both the training and testing datasets.

5.2. Feature Extraction

The TfidfVectorizer [33] was used to convert text into numerical features. Additionally, the Keras Tokenizer was employed alongside the TfidfVectorizer from the SciKit-Learn library.

5.3. Machine Learning Models

Three models were applied for this task, described as follows:

SilverCodeBERT: Utilizes a **CodeBERT-base** model to generate embeddings for both the Natural Language (NL) comment and the Programming Language (PL) code context. Predictions are made using a Multi-Layer Perceptron (MLP) applied to these embeddings.

SilverDoubleBERT: Employs a **BERT-base-uncased** model for the NL comment embeddings and a **CodeBERT-base** model for the PL code context embeddings. Inference is performed via a Multi-Layer Perceptron (MLP) using the combined embeddings.

SilverLSTM: Uses a **GRU** model to generate embeddings for both the NL comment and PL code context. Predictions are made using a Support Vector Machine (SVM) classifier on the resulting embeddings.

NOTE: Although these models were initially generated by LLMs, minor modifications were required to ensure syntactic correctness. The models were then fine-tuned on both the original and augmented datasets.

6. Findings

6.1. Without Augmentation

Table 2
Results of Classifier Runs

Run	Macro F1 Score	Macro Precision	Macro Recall	Accuracy%
SilverCodeBERT	0.807	0.823	0.768	78.12
SilverDoubleBERT	0.799	0.807	0.791	79.08
SilverLSTM	0.779	0.793	0.765	77.22

6.2. With Augmentation

Table 3
Result of Classifier Runs

Run	Macro F1 Score	Macro Precision	Macro Recall	Accuracy%
SilverCodeBERT	0.807	0.826	0.788	79.31
SilverDoubleBERT	0.824	0.834	0.814	81.35
SilverLSTM	0.790	0.771	0.811	78.49

7. Conclusion

The tasks were carried out using machine learning models. Results from the SilverDoubleBERT classifier indicate room for improvement, enabling the development of more sophisticated models that better align with the problem requirements and yield improved performance. Srijoni Majumdar et al. [34] have achieved notable results using ELMo and BERT-based models, and it is anticipated that these outcomes can be further enhanced in future work.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT in order to: Grammar and spelling check. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

Acknowledgments

Thanks to the creators of IRSE FIRE for giving this wonderful opportunity to work on such a project, and their constant technical support throughout the timespan.

References

[1] J. Raskin, Comments are more important than code, ACM Queue 3 (2005) 64–. doi:10.1145/1053331.1053354.

[2] E. Wong, J. Yang, L. Tan, Autocomment: Mining question and answer sites for automatic comment generation, in: 2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2013, pp. 562–567. doi:10.1109/ASE.2013.6693113.
[3] S. C. B. de Souza, N. Anquetil, K. M. de Oliveira, A study of the documentation essential to software maintenance, 2005.
[4] L. Tan, D. Yuan, Y. Zhou, Hotcomments: how to make program comments more useful?, in: Conference on Programming language design and implementation (SIGPLAN), ACM, 2007, pp. 20–27.
[5] S. Majumdar, S. Papdeja, P. P. Das, S. K. Ghosh, Smartkt: a search framework to assist program comprehension using smart knowledge transfer, in: 2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS), IEEE, 2019, pp. 97–108.
[6] N. Chatterjee, S. Majumdar, S. R. Sahoo, P. P. Das, Debugging multi-threaded applications using pin-augmented gdb (pgdb), in: International conference on software engineering research and practice (SERP). Springer, 2015, pp. 109–115.
[7] S. Majumdar, N. Chatterjee, S. R. Sahoo, P. P. Das, D-cube: tool for dynamic design discovery from multi-threaded applications using pin, in: 2016 IEEE International Conference on Software Quality, Reliability and Security (QRS), IEEE, 2016, pp. 25–32.
[8] S. Majumdar, N. Chatterjee, P. P. Das, A. Chakrabarti, A mathematical framework for design discovery from multi-threaded applications using neural sequence solvers, Innovations in Systems and Software Engineering 17 (2021) 289–307.
[9] S. Majumdar, N. Chatterjee, P. Pratim Das, A. Chakrabarti, Dcube_ NN D cube NN: Tool for Dynamic Design Discovery from Multi-threaded Applications Using Neural Sequence Models, Advanced Computing and Systems for Security: Volume 14 (2021) 75–92.
[10] J. Siegmund, N. Peitek, C. Parnin, S. Apel, J. Hofmeister, C. Kästner, A. Begel, A. Bethmann, A. Brechmann, Measuring neural efficiency of program comprehension, in: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, 2017, pp. 140–150.
[11] Y. Wang, H. Le, A. D. Gotmare, N. D. Bui, J. Li, S. C. Hoi, Codet5+: Open code large language models for code understanding and generation, arXiv preprint arXiv:2305.07922 (2023).
[12] J. L. Freitas, D. da Cruz, P. R. Henriques, A comment analysis approach for program comprehension, 2012.
[13] D. Steidl, B. Hummel, E. Juergens, Quality analysis of source code comments, 2013.
[14] M. M. Rahman, C. K. Roy, R. G. Kula, Predicting usefulness of code review comments using textual features and developer experience, 2017.
[15] A. Bosu, M. Greiler, C. Bird, Characteristics of useful code reviews: An empirical study at microsoft, 2015.
[16] S. Majumdar, A. Bansal, P. P. Das, P. D. Clough, K. Datta, S. K. Ghosh, Automated evaluation of comments to aid software maintenance, Journal of Software: Evolution and Process 34 (2022) e2463.
[17] S. Majumdar, S. Papdeja, P. P. Das, S. K. Ghosh, Comment-mine—a semantic search approach to program comprehension from code comments, in: Advanced Computing and Systems for Security, Springer, 2020, pp. 29–42.
[18] S. Majumdar, A. Bandyopadhyay, S. Chattopadhyay, P. P. Das, P. D. Clough, P. Majumder, Overview of the irse track at fire 2022: Information retrieval in software engineering, 2022.
[19] S. Majumdar, A. Bandyopadhyay, P. P. Das, P. Clough, S. Chattopadhyay, P. Majumder, Can we predict useful comments in source codes?-analysis of findings from information retrieval in software engineering track@ fire 2022, in: Proceedings of the 14th Annual Meeting of the Forum for Information Retrieval Evaluation, 2022, pp. 15–17.
[20] S. Majumdar, A. Deshpande, P. P. Das, P. P. Chakrabarti, Comprehending c codes with llms: Effective comment generation through retrieval and reasoning, Pattern Recognition Letters (2025).
[21] S. Paul, S. Majumdar, R. Shah, S. Das, M. Ghosh, D. Ganguly, G. Calikli, D. Sanyal, P. P. Das, P. D. Clough, et al., Overview of the "information retrieval in software engineering"(irse) track at forum for information retrieval 2024, in: Proceedings of the 16th Annual Meeting of the Forum for

Information Retrieval Evaluation, 2024, pp. 18–21.
[22] N. Chatterjee, S. Majumdar, P. P. Das, A. Chakrabarti, Parallelc-assist: Productivity accelerator suite based on dynamic instrumentation, IEEE Access 11 (2023) 73599–73612.
[23] P. Chakraborty, S. Dutta, D. K. Sanyal, S. Majumdar, P. P. Das, Bringing order to chaos: Conceptualizing a personal research knowledge graph for scientists., IEEE Data Eng. Bull. 46 (2023) 43–56.
[24] S. Paul, S. Majumdar, A. Bandyopadhyay, B. Dave, S. Chattopadhyay, P. Das, P. D. Clough, P. Majumder, Efficiency of large language models to scale up ground truth: Overview of the irse track at forum for information retrieval 2023, in: Proceedings of the 15th Annual Meeting of the Forum for Information Retrieval Evaluation, 2023, pp. 16–18.
[25] N. Chatterjee, S. Majumdar, P. P. Das, A. Chakrabarti, Tool assisted agile approach for legacy application migration, International Journal of System Assurance Engineering and Management (2025) 1–16.
[26] S. Majumdar, P. P. Das, Smart knowledge transfer using google-like search, arXiv preprint arXiv:2308.06653 (2023).
[27] A. Deshpande, A. Maji, D. Mondol, P. P. Das, P. D. Clough, S. Majumdar, The code–llm handshake: Smarter maintenance through ai, in: Proceedings of the 17th annual meeting of the Forum for Information Retrieval Evaluation, 2025, pp. 9–12.
[28] A. Mitra, S. Majumdar, A. Mukhopadhyay, P. P. Das, P. D. Clough, P. P. Chakrabarti, Operationalizing large language models with design-aware contexts for code comment generation, arXiv preprint arXiv:2510.22338 (2025).
[29] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, D. Zhou, Chain-of-thought prompting elicits reasoning in large language models, 2023. URL: https://arxiv.org/abs/2201.11903. arXiv:2201.11903.
[30] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, K. Narasimhan, Tree of Thoughts: Deliberate Problem Solving with Large Language Models, 2023. URL: https://arxiv.org/abs/2305.10601. arXiv:2305.10601.
[31] J. Long, Large Language Model Guided Tree-of-Thought, 2023. URL: https://arxiv.org/abs/2305.08291. arXiv:2305.08291.
[32] E. Loper, S. Bird, Nltk: The natural language toolkit, 2002. URL: https://arxiv.org/abs/cs/0205028. doi:10.48550/ARXIV.CS/0205028.
[33] V. Kumar, B. Subba, A tfidfvectorizer and svm based sentiment analysis framework for text data corpus, in: 2020 National Conference on Communications (NCC), 2020, pp. 1–6. doi:10.1109/NCC48643.2020.9056085.
[34] S. Majumdar, A. Bansal, P. P. Das, P. D. Clough, K. Datta, S. K. Ghosh, Automated evaluation of comments to aid software maintenance, Journal of Software: Evolution and Process 34 (2022) e2463. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.2463. doi:https://doi.org/10.1002/smr.2463. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/smr.2463.