Using SVM and Naïve Bayes Classifier to Assess the Utility of C Comments

Anamitra Mukhopadhyay[1,*]

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

Abstract

Comments have a significant positive impact on code development flow. The increasing prevalence of code in everyday life often leads inexperienced programmers to overlook commenting as an essential stage in the development process. As a result, comments are typically of lower quality, and there are a lot of useless comments in these programs. In these tests, the utility of C comments is evaluated using the Naïve Bayes Classifier and Support Vector Machine (SVM). The results set a baseline for further research that could produce better results. These findings can be utilised to create advanced machine learning models that improve the accuracy when doing the task in question.

Keywords

Machine Learning, SVM, Naïve Bayes Classifier, Natural Language Processing

1. Introduction

Comments require a lot of time and are essential to the development process in order to increase code readability. However, not all comments support the aforementioned goal. As coding becomes more common, novice programmers tend to overlook the practice of commenting, which leads to a decrease in both quantity and quality of comments. It can be frustrating and time-consuming to sift through lengthy comments only to discover that they are meaningless, and many comments turn out to be useless.

Many deep learning-based automated commenting methods can increase the number of comments. Unfortunately, there hasn't been enough focus on the issue of comment quality in the literature. Current efforts, however, are addressing these problems by developing machine learning models that are able to identify and categorise comments based on their usefulness.

The author has investigated a variety of Machine Learning (ML) models in order to solve this problem. As part of the Forum for Information Retrieval Evaluation (FIRE) 2025's Information Retrieval in Software Engineering (IRSE) joint task, this study attempts to address the following important questions:

- How complex does it have to be for a machine learning model to consistently differentiate between useful and useless comments?
- Even though they were not designed for this specific situation, how do popular general-purpose models like SVM and Naïve Bayes Classifier fare in this situation?

The aim of this research is to demonstrate that models such as SVM and Naïve Bayes can serve as effective starting points for tackling the problem. These baseline methods provide a foundation for developing more advanced models while carefully considering the risk of overfitting.

In addition to this primary objective, the study also explores whether data produced by large language models, such as GPT-3.5, can be used to enrich the training dataset. This question is especially important for examining how artificial intelligence can support the evaluation of comment quality. By analyzing the impact of supplementing the dataset with AI-generated examples, the research seeks to highlight 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.

anamitra137@gmail.com (A. Mukhopadhyay)

pipeline. This inquiry, positioned at the intersection of human-authored and AI-generated text in the context of assessing comment quality, represents a central focus of the work.

2. Related Work

Software metadata plays a vital role in code maintenance and subsequent comprehension. Several tools have been developed to support knowledge extraction from software metadata, including code structure and runtime traces [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13].

The area of mining and assessing code comments has been widely studied. For instance, Steidl et al. [14] filtered out irrelevant and uninformative comments by applying measures such as Levenshtein distance and comment length to assess word similarity in code-comment pairs. Rahman et al. [15], building on features identified through a survey of Microsoft developers, focused on distinguishing between helpful and unhelpful code review comments in review portals [16]. Majumdar et al. [17, 18, 19, 20] proposed a framework for evaluating comments based on concepts central to code comprehension. Their approach leverages a knowledge network to semantically analyze comments by extracting both textual and code correlation features. Collectively, such methods contribute to improving codebases by utilizing semantic and structural information to address the classification problem of separating useful from non-useful comments.

With the advent of large language models such as GPT-3.5 and LLaMA, evaluating the quality of code comments and comparing them with human interpretation has become increasingly important. The methodology presented in earlier work [17] was extended in the IRSE track at FIRE 2023 [21]. This track examined multiple vector space models [22] and feature sets for binary classification and evaluation of comments, particularly in relation to their role in code understanding. Furthermore, it compared the performance of the prediction model against GPT-generated labels of code and comment quality derived from open-source projects. In addition, [23, 24] provides deep insights of LLMs behaviour for these kind of tasks.

3. Task and Dataset Description

In this section, we outline the task under consideration and the dataset used. *The goal is to enhance a binary classification model for code comment quality by incorporating generated code–comment pairs that can boost its accuracy.*
The dataset was divided into two parts:

- A training set containing 8048 samples, and
- A testing set containing 1000 samples.

The training set was randomized and further divided into 70% for model training and 30% for cross-validation. The labeling scheme was defined as:

- **Useful**: Comments that aid in understanding the code
- **Not Useful**: Comments that do not aid in understanding the code

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*/

4. Augmentation

To expand the existing dataset, data generated by the advanced language model GPT-3.5-turbo was incorporated through an augmentation process. By leveraging its natural language generation abilities, additional comment data was created to both enlarge the dataset and increase its diversity. The purpose of this augmentation was to introduce comments reflecting a broader spectrum of writing styles, structures, and topics. Incorporating GPT-produced data allowed for an assessment of its effectiveness in improving the training of machine learning models for comment quality evaluation. This approach enabled the exploration of how AI-generated content can complement human-written data, resulting in a more comprehensive and robust dataset that supports stronger model performance.

5. System Description

5.1. Text Preprocessing

Initially, stop words, punctuation marks, digits, and hyperlinks are removed. Subsequently, words whose POS tags are not nouns, verbs, adverbs, or adjectives are filtered out. Lemmatization is then applied to normalize different word forms into a single representation, using the NLTK WordNet module. These preprocessing steps are applied consistently to both the training and testing datasets.

5.2. Feature Extraction

To transform text into numerical features, `TfidfVectorizer` is employed. Additionally, the Keras `Tokenizer` is used in conjunction with the `TfidfVectorizer` from the SciKit-Learn library.

5.3. Machine Learning Models

For this task, two models were employed: a Support Vector Machine (**SVM**) and a **Naïve Bayes classifier**. Both models were implemented using the SciKit-Learn library. The SVM model was configured with the following parameters:

- **C**: `(regularization parameter)` = 1
- **kernel**: `(kernel type)` = 'linear'

6. Findings

6.1. Without Augmentations

With the specified parameters for the SVM model, the validation set achieved an accuracy of 77.32% and an F1 score of 0.768.
In comparison, the Naïve Bayes classifier yielded an accuracy of 60.93% on the validation set, with an F1 score of 0.688.

Table 2
Results of Classifier Runs

Run	Macro F1 Score	Macro Precision	Macro Recall	Accuracy%
SVM	0.768	0.791	0.747	77.32
Naïve Bayes	0.688	0.732	0.649	60.93

6.2. With Augmentation

Using these parameters, the SVM model obtained a validation accuracy of 77.62% with an F1 score of 0.782.
For the Naïve Bayes classifier, the validation accuracy was 64.10%, accompanied by an F1 score of 0.695.

Table 3
Results of Classifier Runs

Run	Macro F1 Score	Macro Precision	Macro Recall	Accuracy%
SVM	0.782	0.793	0.772	77.62
Naïve Bayes	0.695	0.739	0.656	64.10

7. Conclusion

Basic machine learning models, namely SVM and the Naïve Bayes classifier, were applied to address the task. The results from the SVM classifier indicate room for enhancement, suggesting the possibility of developing more sophisticated models that align more closely with the problem statement and yield improved performance. Notably, Srijoni Majumdar et al. [25] have previously employed neural networks to achieve strong results, and it is anticipated that such approaches will continue to deliver even better outcomes in the future.

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] 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.
[2] 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.
[3] 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.
[4] 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.
[5] 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.

[6] 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.
[7] 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).
[8] 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.
[9] N. Chatterjee, S. Majumdar, P. P. Das, A. Chakrabarti, Parallelc-assist: Productivity accelerator suite based on dynamic instrumentation, IEEE Access 11 (2023) 73599–73612.
[10] 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.
[11] 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.
[12] 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.
[13] S. Majumdar, P. P. Das, Smart knowledge transfer using google-like search, arXiv preprint arXiv:2308.06653 (2023).
[14] D. Steidl, B. Hummel, E. Juergens, Quality analysis of source code comments, International Conference on Program Comprehension (ICPC), IEEE, 2013, pp. 83–92.
[15] M. M. Rahman, C. K. Roy, R. G. Kula, Predicting usefulness of code review comments using textual features and developer experience, International Conference on Mining Software Repositories (MSR), IEEE, 2017, pp. 215–226.
[16] A. Bosu, M. Greiler, C. Bird, Characteristics of useful code reviews: An empirical study at microsoft, Working Conference on Mining Software Repositories, IEEE, 2015, pp. 146–156.
[17] 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.
[18] 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.
[19] 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, in: Forum for Information Retrieval Evaluation, ACM, 2022.
[20] 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.
[21] S. Majumdar, S. Paul, D. Paul, A. Bandyopadhyay, B. Dave, S. Chattopadhyay, P. P. Das, P. D. Clough, P. Majumder, Generative ai for software metadata: Overview of the information retrieval in software engineering track at fire 2023, in: Forum for Information Retrieval Evaluation, ACM, 2023.
[22] S. Majumdar, A. Varshney, P. P. Das, P. D. Clough, S. Chattopadhyay, An effective low-dimensional software code representation using bert and elmo, in: 2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS), IEEE, 2022, pp. 763–774.
[23] 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.
[24] 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).
[25] 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.