Automated Assessment of Code Comment Quality

Hard Kapadia[1,*]

[1]Indian Institute of Technology, Goa, 403401

Abstract

The usefulness of code comments in software development can vary widely and recognizing this requires methods capable of rigorously measuring their true benefits. This work seeks to help improve the classification of code comment usefulness through a hybrid approach that combines manually-retagged datasets with synthetic data augmentation. For our augmentation, we provided GPT-3.5-turbo, a leading large language model, with prompts to create additional labelled examples of comments to aid the project. We constructed a random forests baseline classification model. Importantly, despite the synthetic examples added to the dataset, we showed no drop in the models performance, with F1 scores remaining around 0.79 before and after augmentation. The findings of this study shine a light on some of the benefits and limitations of applying synthetic data augmentation in the classification of code comments usefulness.

Keywords

Random Forest, Data Augmentation, Comment Classification, Qualitative Analysis

1. Introduction

Developers often face pressure that leads to suboptimal coding and outdated documentation, complicating **software maintenance** and **knowledge transfer**. Automated program comprehension is key to addressing this. We focus on **code comments** as a crucial source of truth for understanding program design, logic, and intent. However, comments vary in informative value, necessitating automated classification. A major hurdle is the lack of large, well-annotated datasets.

To tackle this, we propose a **hybrid data augmentation approach**. We introduce a **binary classification task** for C programs, labeling comments as **Useful** or **Not Useful**. We start with over 11,000 manually annotated samples and use **GPT-3.5-turbo** to generate an additional 200 synthetic labels for **data augmentation**. We used **Random Forests** to establish a baseline. Interestingly, the model's performance remained **stable** with an **F1 score of 0.79** for both the baseline and the augmented dataset.

This study investigates the value of combining manual and synthetic data for code comment usefulness classification, aiming to contribute a scalable solution to dataset limitations in software engineering.

The rest of the paper is organized as follows. Section 2 discusses background work. The task and dataset are described in 3. Our methodology is discussed in section 4. Results are addressed in section 5. Section 6 concludes the paper.

2. Related Work

Software metadata, such as runtime traces and structural attributes, is integral to code maintenance and comprehension, leading to the development of numerous extraction tools [1, 2, 3, 4, 5, 6, 7, 8]. In terms of mining code comments, initial quality assessment efforts relied on lexical and structural analysis, comparing word similarity (e.g., Levenshtein distance) and comment length to filter non-informative entries [9, 10, 11, 12, 13, 14]. More sophisticated approaches used feature engineering based on developer

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

*Corresponding author.

hard.kapadia.22063@iitgoa.ac.in (H. Kapadia)

0009-0005-1707-2566 (H. Kapadia)

surveys [15, 16] or semantic interpretation via knowledge graphs [17, 18] to classify comments as useful or not useful, thereby aiding codebase decluttering. The recent emergence of Large Language Models (LLMs) [19] introduces the need to evaluate if their automated quality assessments (e.g., using GPT-3.5 or LLaMA) align with human interpretations. The IRSE track at FIRE 2023 [20] addresses this by extending prior methodology [17, 21, 22, 23, 24, 25, 26, 27], specifically examining the effectiveness of various vector space models and the impact of incorporating GPT-generated labels on the performance of models designed for comment utility prediction in open-source software. Similarly, [28, 29] also explores LLMs for the tasks related to this topic.

3. Task and Dataset Description

This paper addresses a **binary classification task** aimed at categorizing source code comments as either **Useful** or **Not Useful**. The system takes a code comment along with its surrounding lines of code as input and outputs a binary label, which ultimately helps developers more effectively understand the associated code. This classification system is developed using classical machine learning algorithms, specifically **Random Forests**.

3.1. Task Definition

The two categories of source code comments are defined based on their relevance to the surrounding code:

Label	Definition
Useful	The given comment is relevant to the corresponding source code.
Not Useful	The given comment is not relevant to the corresponding source code.

3.2. Datasets

Our study utilizes two distinct datasets for training and analysis:

1. **Primary Manually Annotated Dataset:** This dataset comprises over **11,000 code-comment pairs** written in the **C programming language**. Each instance, sourced from GitHub, includes the comment text, a corresponding code snippet, and a binary label (Useful/Not Useful). The entire dataset was meticulously annotated by a team of **14 human annotators**. A sample of this data structure is presented in Table 1.
2. **GPT-Labeled Augmentation Dataset:** We created a secondary, similarly structured dataset also sourced from GitHub. In this case, the binary labels were assigned by the **GPT** large language model. This dataset is explicitly used to **augment** the primary manually annotated dataset during subsequent analyses to assess model performance with synthetic data.

4. Working Principle

We employ a **Random Forest (RF)** algorithm to implement the binary classification system. The system classifies code comments as **Useful** or **Not Useful** by taking the comment text and its surrounding code snippet as input.

To prepare the data for the model, we use a pre-trained **Universal Sentence Encoder** to generate **embeddings** for both the code snippets and their corresponding comments. These resultant vector embeddings form the input features for the RF model.

The complete dataset is partitioned into a training set and a testing set using an **80%/20% split**, respectively, for all experiments.

ID	Description	Context Snippet	Utility Class
1	/*fix issue 404 handler*/	-10. int err_code = 0; -9. Request *req = NULL; ... -1. #ifndef ISSUE404_FIX /*fix issue 404 handler*/ 1. handle_error();	Unnecessary
2	/*check for carriage return followed by null character*/	-1. if (end_of_stream) /*check for carriage return...*/ 1. c = read_char(); 2. if (c == '\n') { 3. line_count++;	Informative
3	/*Process security context message*/	-10. do_cleanup(); ... -2. int status = 0; -1. while(status == 0) { /*Process security context message*/ 1. send_msg(msg_ctx);	Useful

Table 1
Modified sample data instances from the dataset, including context.

4.1. Random Forest Model

We leverage Random Forest, an **ensemble method** based on decision trees, to enhance predictive accuracy and mitigate overfitting. The RF prediction is based on the **majority vote** of its constituent trees:

$$RF(\mathbf{x}) = \text{majority}\left(\{T_i(\mathbf{x})\}_{i=1}^{n}\right) \tag{1}$$

where $T_i(\mathbf{x})$ is the prediction of the i-th tree for input vector $\mathbf{x}$, and n is the total number of trees.

Each tree in the ensemble is constructed through **bootstrapping** (sampling the training data with replacement) and **random feature selection** at every node before splitting based on a criterion like Gini impurity.

Key advantages of using Random Forest include its ability to handle multi-dimensional feature spaces without requiring feature scaling and its robustness in dealing with missing values. The **out-of-bag (OOB)** error is used during training to provide an unbiased estimate of the generalization error for hyperparameter tuning. While a default threshold of 0.5 is used for binary classification, this can be **adjusted** to prioritize the identification of the **Useful** comment class.

5. Results

The results for the binary classification task on both datasets are summarized in Table 2.

The negligible change in performance metrics between the two experiments suggests that the synthetically generated data is practically indistinguishable from the human-annotated original data. This observation validates the utility of using a model like GPT-3.5-turbo for effective **data augmentation** in this domain.

Dataset	Acc. (%)	Prec.	Rec.	F1-Score
Initial Dataset	81.12	0.7915	0.8020	0.7955
GPT-Augmented Set	81.08	0.7922	0.8015	0.7958

Table 2
Classification performance comparison across different datasets.

6. Conclusion

This article discusses a binary classification problem in source code comment classification, specifically targeted towards the usefulness of comments embedded within source code written in C language. The primary classification method was Random Forests. In total, two experiments were completed; the first used only the original dataset while the second included both the original dataset and a synthetic dataset created by the Generative Pre-Trained Transformer (GPT). The similar results from both experiments indicated the syntehtic data closely resembled the original data, showcasing how the generation of synthetic data can improve the volume of data needed for developing models. The accuracy of the synthetic data in comparison to the original dataset is, in part, evident based upon those results. Therefore, overall, the generation of synthetic data is useful for data augmentation and can be applied in various pipelines.

Declaration on Generative AI

In the course of preparing this manuscript, the author(s) employed the generative AI tool ChatGPT. Its use was limited to performing checks for grammar and spelling. Following this, the author(s) conducted a thorough review and revision of the text and assume full responsibility for the final published content.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[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. C. B. de Souza, N. Anquetil, K. M. de Oliveira, A study of the documentation essential to software maintenance, Conference on Design of communication, ACM, 2005, pp. 68–75.

[8] S. Majumdar, P. P. Das, Smart knowledge transfer using google-like search, arXiv preprint arXiv:2308.06653 (2023).

[9] 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.

[10] 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).

[11] D. Steidl, B. Hummel, E. Juergens, Quality analysis of source code comments, International Conference on Program Comprehension (ICPC), IEEE, 2013, pp. 83–92.

[12] 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.
[13] 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: FIRE (Working Notes), 2022, pp. 1–9.
[14] J. L. Freitas, D. da Cruz, P. R. Henriques, A comment analysis approach for program comprehension, Annual Software Engineering Workshop (SEW), IEEE, 2012, pp. 11–20.
[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] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Advances in neural information processing systems 33 (2020) 1877–1901.
[20] S. Majumdar, S. Paul, D. Paul, A. Bandyopadhyay, 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, arXiv preprint arXiv:2311.03374 (2023).
[21] 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.
[22] 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).
[23] 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.
[24] N. Chatterjee, S. Majumdar, P. P. Das, A. Chakrabarti, Parallelc-assist: Productivity accelerator suite based on dynamic instrumentation, IEEE Access (2023).
[25] 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.
[26] 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.
[27] 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.
[28] 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.
[29] 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).