Assessing the Utility of C Comments with SVM and Naïve Bayes Classifier

Anamitra Mukhopadhyay[1,*]

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

Abstract

The flow of code development greatly benefits from comments. As code becomes more prevalent in daily life, novice programmers frequently ignore commenting to be a necessary step in the development process. This generally lowers the quality of comments, and such programs contain a significant number of pointless comments. In these experiments, the Naïve Bayes Classifier and Support Vector Machine (SVM) are used to assess the usefulness of C comments. The outcomes establish a baseline for future study that may yield superior findings. These results can be used to develop more sophisticated and complex machine learning models that increase the accuracy attained when completing the task at hand.

Keywords

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

1. Introduction

In order to improve code readability, comments are crucial to the development process and take up a lot of time. Not every comment, though, assists the above objective. As coding becomes more widespread, inexperienced programmers often neglect the art of commenting, which results in a decline in the quantity and quality of comments. A lot of comments end up being useless, and it can be annoying and time-consuming to sort through long comments only to find that they are pointless.

The quantity of comments can be boosted by a variety of deep learning-based automatic commenting models. Unfortunately, the problem of comment quality has not received enough attention in the literature. However, current initiatives are tackling these issues by creating machine learning models that can recognize and classify comments according to their utility.

The author has explored a range of Machine Learning (ML) models in an attempt to find solutions to this problem. This paper aims to answer critical questions as part of the Information Retrieval in Software Engineering (IRSE) shared task at the Forum for Information Retrieval Evaluation (FIRE) 2024 :

- What level of complexity is necessary for a Machine Learning model to reliably distinguish useful comments from useless ones?
- How do well-known general-purpose models like SVM and Naïve Bayes Classifier perform in this context, even if they are not built for this particular scenario?

The purpose of this research is to show that models like SVM or Naïve Bayes Classifier can be used as good starting points to address the problem. These foundations can be used to build more intricate models while taking overfitting risk into account.

Apart from the above described goals, this work additionally addresses the topic of whether data generated by large language models, like GPT-3.5, can be used to augment the dataset for the learning task. This aspect is crucial when investigating the possibilities of using artificial intelligence to enhance the evaluation of comment quality. The study intends to clarify the advantages and difficulties of incorporating sophisticated language models into the machine learning pipeline by assessing the effects of augmenting the dataset with AI-generated data. This question, which highlights the intersection of

Forum for Information Retrieval Evaluation, December 12-15, 2024, India

*Corresponding author.

anamitra137@gmail.com (A. Mukhopadhyay)

https://cse.iitkgp.ac.in/~anamitra.mukhopadhyay/ (A. Mukhopadhyay)

human-written and AI-generated content in the context of comment quality evaluation, is a substantial portion of the research.

2. Related Work

Software metadata is essential to code maintenance and subsequent comprehension. Many tools have been created to help with knowledge extraction from software metadata, such as code structure and runtime traces [1, 2, 3, 4, 5, 6].

Numerous authors have studied the topic of mining code comments and evaluating their quality. By using methods like Levenshtein distance and comment length to measure word similarity in code-comment pairs, Steidl et al. [7] successfully weed out irrelevant and uninformative comments. By using characteristics found in a survey of Microsoft developers, Rahman et al. [8] concentrate on differentiating between code review comments that are helpful and those that are not in review portals [9]. Using concepts essential to code comprehension, Majumdar et al. [10, 11, 12, 13] have presented a framework for assessing comments. Their method uses a knowledge network to semantically evaluate the data in comments by developing textual and code correlation characteristics. In the end, these methods aid in the process of cleaning codebases by utilising both semantic and structural information to solve the prediction problem of differentiating between helpful and useless comments.

The development of huge language models, like GPT-3.5 or llama, makes it essential to evaluate code comments' quality and contrast them with human interpretation. The methodology introduced in a previous work [10] is expanded upon in the IRSE track at FIRE 2023 [14]. Particularly in relation to their function in understanding code, it explores several vector space models [15] and features for binary classification and evaluation of comments. Additionally, this track compares the performance of the prediction model with GPT-generated code and comment quality labels that were taken from open-source software.

3. Task and Dataset Description

In this section, a description of the task at hand and the dataset provided are given. *A binary code comment quality classification model needs to be augmented with generated code and comment pairs that can improve the accuracy of the model.*
The corresponding dataset was split into two:

- The training dataset with 8048 entries, and
- The testing dataset with 1000 entries.

The training dataset was shuffled, and split into 70% for training the models, and 30% for cross-validation. The data was labelled as follows:

- **Useful**: Comments that are useful for code comprehension
- **Not Useful**: Comments that are not useful for code comprehension

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 enhance the current dataset, data produced by the potent language model GPT-3.5-turbo was included as part of the dataset augmentation process. To increase the dataset's diversity and size, more comment data was produced by utilizing GPT-3.5-turbo's natural language generation capabilities. The goal of this augmentation technique was to include comments that covered a wider range of writing styles, formats, and subject matter. In order to evaluate its potential for improving the training of machine learning models for comment quality evaluation, GPT-generated data was incorporated. This method made it possible to investigate how AI-generated content might enhance human-written data, creating a more complete and reliable dataset that would enhance model performance.

5. System Description

5.1. Text Preprocessing

First, all stop words, punctuation, digits, and links are eliminated. Next, every word with a POS tag that isn't a noun, verb, adverb, or adjective is eliminated. Lemmatization is the process of combining a word's several forms into a single term. For lemmatization, NLTK wordnet is utilized. The same preparation procedures are used for the training and testing datasets.

5.2. Feature Extraction

TfidfVectorizer is used for converting the text into numerical features. Tokenizer by Keras library is used, along with TfidfVectorizer that was used from SciKit-Learn library.

5.3. Machine Learning Models

Two models have been used for the task: one using Support Vector Machine (**SVM**) model, and another with **Naïve Bayes classifer** model. We have used the SciKit-Learn library for both of the models, with the parameters for the SVM model as follows:

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

6. Findings

6.1. Without Augmentations

With these parameters set for the SVM model, the validation set gives a 77.27% accuracy score, along with an F1 score of 0.786.
Also, with the Naïve Bayes Classifier, the validation set gives a 60.99% accuracy score, along with an F1 score of 0.699.

Table 2
Results of Classifier Runs

Run	Macro F1 Score	Macro Precision	Macro Recall	Accuracy%
SVM	0.771	0.785	0.758	77.27
Naïve Bayes	0.686	0.736	0.642	60.99

6.2. With Augmentation

With these parameters set for the SVM model, the validation set gives a 77.65% accuracy score, along with an F1 score of 0.783.
Also, with the Naïve Bayes Classifier, the validation set gives a 64.03% accuracy score, along with an F1 score of 0.695.

Table 3
Results of Classifier Runs

Run	Macro F1 Score	Macro Precision	Macro Recall	Accuracy%
SVM	0.778	0.791	0.765	77.65
Naïve Bayes	0.730	0.736	0.652	64.03

7. Conclusion

Basic machine learning models such as SVM and the Naïve Bayes Classifier were used to complete the tasks. The SVM classifier's results show that there is potential for improvement, allowing for the creation of more complex models that better fit the issue statement and produce better outcomes. Notably, Srijoni Majumdar et al. [16] have already used neural networks to produce excellent outcomes, and the author expects these findings to continue to improve over time.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[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. 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.