Evaluating Usefulness of C Comments using SVM and Naïve Bayes Classifier

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 Support Vector Machine (SVM) and Naïve Bayes Classifier. 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 intricate machine learning models can be created that can improve the accuracy achieved in performing said task.

Keywords

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

1. Introduction

Comments are an integral part of code development [1], and a lot of time is used to write the comments to make the code more readable [2]. But, not all comments are useful in helping the cause, and with coding becoming more and more commonplace, novice coders are ignoring the art of commenting, and the quality and quantity of comments are degrading [3]. A lot of comments are useless. But reading through a long comment only to find out that it is useless is frustrating, and a wastage of time.

The quantity of comments can be increased using various automatic commenting models based off of deep learning [4]. But, unfortunately there is a lack of research done when it comes to address the bad quality of these comments. However, finally these problems are being addressed, and to improve the quality of the human-written comments, machine learning models are being developed to recognize and label the comments based on their usefulness.

The author has explored various Machine Learning (ML) models to approach this problem. In this paper, the author tries to find the answer to the following questions as a part of a shared task, called the Information Retrieval in Software Engineering (IRSE) at Forum for Information Retrieval Evaluation (FIRE) 2022 [5], which was completed with the team name FaultySegment:

Forum for Information Retrieval Evaluation, December 9-13, 2022, India

*Corresponding author.

aritramitra2002@gmail.com (A. Mitra)

https://cse.iitkgp.ac.in/~aritra.mitra/ (A. Mitra)

- How much complex does a Machine Learning model need to be for being able to reliably separate the useful comments from the useless ones?
- How do models like SVM and Naïve Bayes Classifier, two models which are known to even the most novice Machine Learning students, fair in this problem?

The paper aims to show that models like these can be good starting points in approaching a problem like this, and further complex models can be built upon these, with, of course, keeping the factor of overfitting in mind. The paper

2. Related Works

Several research works have been done to classify code comments, and different papers have also been published.
Yoann Padioleau et. al. showed that there exist many different classes of comments, each serving a different purpose [6].

Luca Pascarella et. al. worked on classifying Java comments based on their use cases and types [7] where they used random forests and multinomial naïve bayes classifier to distinguish between the different commonly occuring types of code comments.
Similarly, Jingyi Zhang et. al. classified comments based on their use cases and types [8] where they used decision trees and multinomial naïve bayes classifier to distinguish between the different commonly occuring types of code comments.

Yusuke Shinyama et. al. have analysed different type of code comments in Java and Python to get details about the working of the code at a microscopic level using methods like decision tree to identify explanatory comments with 60% precision and 80% percent recall [9].

Srijoni Majumdar et. al. have worked on evaluating the quality and usefulness of comments in being able to make the relevant code more comprehensible [10] where they use neural networks to achieve precision and recall scores of 86.27% and 86.42%, respectively.

Mohammed Masudur Rahman, et. al. worked on evaluating usefulness of comments in code review [11] where they used textual features along with developer experience.

Pooja Rani et. al. have looked at classifying comments of multiple languages, and the differences that appear among the types of comments in those different languages [12].

3. Task and Dataset Description

In this section, a description of the task at hand and the dataset provided are given. The task at IRSE, FIRE 2022 was as follows:

A binary classification task to classify source code comments as Useful or Not Useful for a given comment and associated code pair as input.

The corresponding dataset was split into two:

- The training dataset with 8048 data points, and
- the testing dataset with 1001 data points.

The training dataset was randomly 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. System Description

4.1. Text Preprocessing

All the links, punctuations, numbers and stop words have been removed. Lemmatization is used for grouping together the different forms of a word into a single word. NLTK wordnet [13] is used for lemmatization. Both training and testing datasets use same preprocessing steps.

4.2. Feature Extraction

TfidfVectorizer [14] is used for converting the text into numerical features. Tokenizer by Keras [15] library is used, along with TfidfVectorizer that was used from scikit-learn library.

4.3. Machine Learning Models

Two runs have been submitted 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'

Table 2
Results of Classifier Runs

Run	Macro F1 Score	Macro Precision	Macro Recall	Accuracy%
SVM	0.771718	0.772345	0.771381	77.2670
Naïve Bayes	0.599571	0.609193	0.644289	63.9751

5. Findings

With these parameters set for the SVM model, the validation set gives a 77.26708074534162% accuracy score, along with an F1 score of 0.786464410735123.
Also, with the Naïve Bayes Classifier, the validation set gives a 60.993788819875775% accuracy score, along with an F1 score of 0.699233716475096.

6. Conclusion

The tasks have been completed using elementary machine learning models like SVM and Naïve Bayes Classifier, and the results for the SVM classifier shows that this model can be improved upon and more complex models can be created, which will better suit the problem statement, and will give a better result. Srijoni Majumdar, et. al. have already gotten better results using neural networks [10], and the author hopes that these results will only improve over time.

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] B. Fluri, M. Würsch, H. Gall, Do code and comments co-evolve? on the relation between source code and comment changes, 2007, pp. 70–79. doi:10.1109/WCRE.2007.21.
[2] M. Kajko-Mattsson, A survey of documentation practice within corrective maintenance, Empirical Software Engineering 10 (2005) 31–55. URL: https://doi.org/10.1023/B:LIDA.0000048322.42751.ca. doi:10.1023/B:LIDA.0000048322.42751.ca.
[3] J. Raskin, Comments are more important than code, ACM Queue 3 (2005) 64–. doi:10.1145/1053331.1053354.
[4] 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.
[5] S. Majumdar, A. Bandyopadhyay, P. P. Das, P. D Clough, S. Chattopadhyay, P. Majumder, Overview of the IRSE subtrack at FIRE 2022: Information Retreival in Software Engineering, in: Working Notes of FIRE 2022 - Forum for Information Retrieval Evaluation, ACM, 2022.

[6] Y. Padioleau, L. Tan, Y. Zhou, Listening to programmers — taxonomies and characteristics of comments in operating system code, in: 2009 IEEE 31st International Conference on Software Engineering, 2009, pp. 331–341. doi:10.1109/ICSE.2009.5070533.

[7] L. Pascarella, M. Bruntink, A. Bacchelli, Classifying code comments in java software systems, Empirical Software Engineering 24 (2019) 1499–1537. URL: https://doi.org/10.1007/s10664-019-09694-w. doi:10.1007/s10664-019-09694-w.

[8] J. Zhang, L. Xu, Y. Li, Classifying python code comments based on supervised learning, in: X. Meng, R. Li, K. Wang, B. Niu, X. Wang, G. Zhao (Eds.), Web Information Systems and Applications, Springer International Publishing, Cham, 2018, pp. 39–47.

[9] Y. Shinyama, Y. Arahori, K. Gondow, Analyzing code comments to boost program comprehension, in: 2018 25th Asia-Pacific Software Engineering Conference (APSEC), 2018, pp. 325–334. doi:10.1109/APSEC.2018.00047.

[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. 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.

[11] M. M. Rahman, C. Roy, R. Kula, Predicting usefulness of code review comments using textual features and developer experience, 2017. doi:10.1109/MSR.2017.17.

[12] P. Rani, S. Panichella, M. Leuenberger, A. Di Sorbo, O. Nierstrasz, How to identify class comment types? a multi-language approach for class comment classification, Journal of Systems and Software 181 (2021) 111047. URL: https://www.sciencedirect.com/science/article/pii/S0164121221001448. doi:https://doi.org/10.1016/j.jss.2021.111047.

[13] E. Loper, S. Bird, Nltk: The natural language toolkit, 2002. URL: https://arxiv.org/abs/cs/0205028. doi:10.48550/ARXIV.CS/0205028.

[14] 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.

[15] N. Ketkar, Introduction to Keras, 2017, pp. 95–109. doi:10.1007/978-1-4842-2766-4_7.