CEUR-WS.org/Vol-3395/T1-2.pdf

Prediction of Useless and irrelevant Comments in C Language as per Surrounding Code Context

Bikram Ghosh[2], Pankaj Chowdhury[1*] and Utpal Sarkar[1]

[1]Department of Computer Science and Engineering, Jadavpur University, India
[2]Department of Mechanical Engineering, Jadavpur University, India

Abstract

Irrelevant and Useless Comments is a prevalent problem that Programmers/Coders generally struggle with everyday.Though comments increases the size of the code file, but it is very useful for a new programmer to understand which work has already been done in the past. The programmer can easily use a function without understanding the logic with the help of comments. However there are some unnecessary or out-of-context comments that not only decreases the readability of the codebase but also, sometimes, confuses the programmer. This presents the need of automatic detection of such comments on a codebase to ease the programmer with understanding code context, adding new features or debugging the entire code for a bug.

In this paper, we have presented the machine learning models that can detect Useless Comments as per the Surrounding code context. Specifically, we described the model submitted for the shared task on Comment Classification in C Language at FIRE 2022.The problem concentrates on binary classification of comments into two classes, namely: Useful and Useless Comments. Overall, our performance is good but it needs some improvement that can be done by some more data pre-processing techniques.Our scores are encouraging enough to work for better results in future.

Keywords

Naive-Bayes Classification, SVM, TFIDF, Logistic regression

1. Introduction

Commenting involves placing **Human Readable Descriptions** inside computer programs detailing what the Code is doing. Proper use of commenting can make code maintenance much easier, as well as helping to find bugs faster.Further, commenting is very important when writing functions so that other people will use the function just by understanding its utilization. Remember, a well-documented code is as important as a correctly working code.

In C Language, the syntax of writing comments is:

use // for a single-line comment.
use /* for multiple lines */
However, when a codebase is filled with irrelevant or useless comments, it creates problems. Sometimes it even leads to misinterpreting the code when a programmer believes the comment. Also, a comment is definitely useless if it matches the signature of a function. Such useless comments reduce the clarity of a well-expressed code, take time for the pre-processor to remove, and also occupy screen space without any valuable contribution.
Artificial Intelligence and different Machine Learning techniques can be implemented to remove such useless comments from the codebase. In this paper, we have implemented several Machine Learning models to classify the comments from the provided dataset into two classes: Useful(1) and Not-useful(0). Out of all the algorithms implemented, we were eventually able to achieve an F1 Accuracy Score of 0.83.

Forum for Information Retrieval Evaluation, December 9-13, 2022, India
*Corresponding authors
bikramghosh9547@gmail.com (B. Ghosh); pankajchowdhury497@gmail.com (P. Chowdhury); sutpal872@gmail.com (U.Sarkar)
https://github.com/bikramghosh-ux/IRSE_FIRE_2022_Submission

© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

2. Related Work

This task can be considered as a problem requiring a suitable implementation of Text Classification Algorithms as, herein, comments/text are being put into binary classification. Several Works have been proposed that can be effectively used to classify text into two classes based on data set. *Mukesh Zaveri[1]* proposed anautomatic text classification model that was implemented over the content of Blog posts to classify them as structured or unstructured. Moreover, *Kamel Alreshedy[2]* proposed a research paper where he showed his team work on machine learning algorithms to classify Code Snippets into programming Languages they belong.
Also, *Kriti Kumari[3]* participated in HASOC-2019 task 1 i.e. Identification of Abusive Content in a text where they used Glove embedding and fastText embedding techniques to classify comments whether they are abusive or not. The problems that the references discussed above, were not the same as ours. However, these problems shared many similarities and hence were very useful to gain some intuition about the approach.

3. Task and Dataset Description

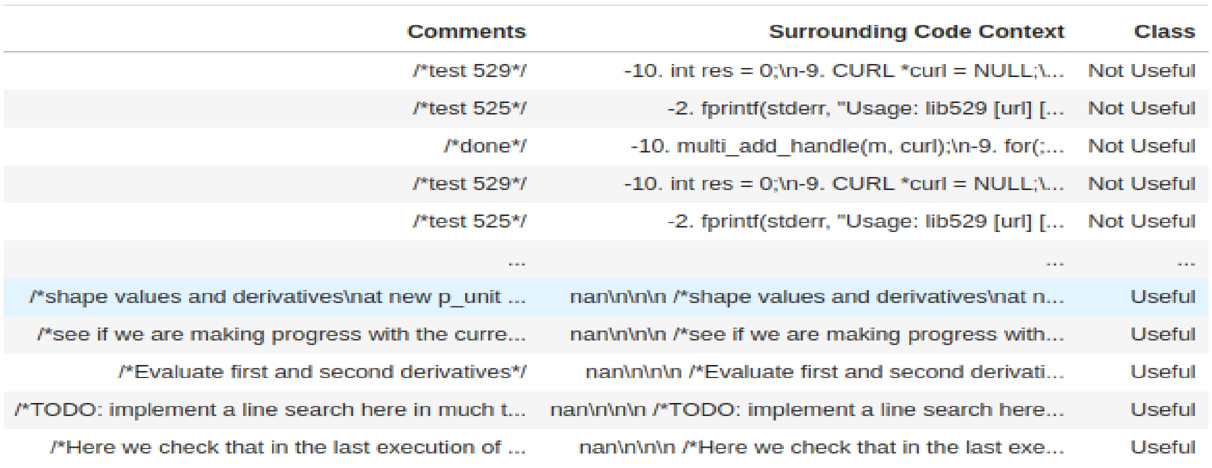
In this section, we have described the Comment Classification shared task and the dataset provided to the participants.

FIRE 2022 shared a task that basically aims to classify comments written in C Languageinto two classes, namely: Useful (U) and Not Useful (N). (shown in Table 1):

1. (U) Useful - This Comment is very useful as gives some vital information needed to be taken care of while running the code context.
2. (N) NotUseful - This Comment is not useful. It is unnecessarily provided or sometimes, even makes no sense in the codebase.

Alongwith the Comments, their respective Surrounding Code Context was also provided.On the basis of which the comments were required to be classified.

Table



Comments	Surrounding Code Context	Class
/*test 529*/	-10. int res = 0;\n-9. CURL *curl = NULL;\...	Not Useful
/*test 525*/	-2. fprintf(stderr, "Usage: lib529 [url] [...	Not Useful
/*done*/	-10. multi_add_handle(m, curl);\n-9. for(;...	Not Useful
/*test 529*/	-10. int res = 0;\n-9. CURL *curl = NULL;\...	Not Useful
/*test 525*/	-2. fprintf(stderr, "Usage: lib529 [url] [...	Not Useful
...	...	...
/*shape values and derivatives\nat new p_unit ...	nan\n\n\n /*shape values and derivatives\nat n...	Useful
/*see if we are making progress with the curre...	nan\n\n\n /*see if we are making progress with...	Useful
/*Evaluate first and second derivatives*/	nan\n\n\n /*Evaluate first and second derivati...	Useful
/*TODO: implement a line search here in much t...	nan\n\n\n /*TODO: implement a line search here...	Useful
/*Here we check that in the last execution of ...	nan\n\n\n /*Here we check that in the last exe...	Useful

We have used the dataset available at IRSE 2022. The dataset consists of 8,047 comments for training provided separately as a training dataset and 1,001 comments for testingprovided as a testing dataset with a balanced distribution of each class. There was no need of splitting the dataset for testing and training purposes.

4. System Description

4.1. Text Pre-processing

We have removed all the punctuations, numbers and stop words from the Surrounding Code context. We also removed the starting two characters (“/*”) and ending two characters (“*/”) from the Comments as they are just basic syntax of writing comments in C Language. We then converted all alphabetic characters present into lowercase. We have also used lemmatization for grouping together the different forms of a word into a single word. NLTK wordnet is used for lemmatization. Both Train and Test data uses same preprocessing.

4.2. Feature Extraction

For Logistic regression,Multinomial Naive Bayes, and Support Vector Machine algorithms we have used *TF-IDF Vectorizer [4]* from the Sci-kit learn library. Tf-IdfVectorizeris used for converting the text into numerical features. Pipeline 1 is used for doing Tf-IDFVectorizer and classification in a pipelined manner.

4.3. Machine Learning Models

We have submitted runs based on three different algorithms, namely- *Logistic Regression [5]*, *Support VectorMachine [6]*,and *Multinomial Naive Bayes [7]*.
We have used the Sci-kit-learn library for logisticregression-based models. We scored a maximum F1 score of 0.83 using SVM and Logistic Regression (both) for the task.
For SVM we have used a Linear kernel with a C=1.0 and gamma= ‘auto’ and degree=3. Proper value of c and gamma need to chosen for optimizing the performance of SVM Classifier.

5. Results and Discussion

The results of the task are represented in terms of Macro-F1,Macro Precision, Macro Recall, and Accuracy (shown in Table below). The best score is Macro-F1,
we got a maximum Macro Precision of 0.79 from the implementation of the Logistic Regression Model.

Run	Macro F1	Macro Recall	Macro Precision	Accuracy
Logistic Regression	0.8	0.8	0.79	83%
Linear Support Vector Machine(SVM)	0.79	0.8	0.79	83%
Multinomial Naive Bayes Classification	0.63	0.7	0.67	63%

6. Conclusion and Future Work

We have completed the given task using various text classification algorithms and evaluated the performance of different algorithms for Comment Classification.
The task was very interesting and unique. We got to learn a lot from this task and we also got to test our knowledge of Machine Learning Algorithms.
We look forward to experimenting with different advancedalgorithms or neural network models. Also, fine-tuning the parameters of the algorithm can help in the improvement of the overall performance. And the results of more than one classification algorithm can be combined to generate an overall better score.We shall be exploring these tasks in the coming days.

References

[1] Mukesh Zaveri, Mita K Dalal, Automatic Text Classification: A Technical Review: Sardar Vallabhai National Institute of Technology, Surat,India

[2] Kamel Alreshedy, Dhanush Dharmaretnam, Daniel M. German, Venkatesh Srinivasan, T. Aaron Gulliver : SCC : Automatic Classification of Code Snippets: 21 September 2018

[3] Kriti Kumari, Jyotiprakash Singh : Deep Learning Approach for Classification of Abusive Text: National Institute of Technology, Patna: HASOC 2019

[4] V. Kumar, B. Subba, A TF-IDF vectorizer 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.

[5] Logistic regression (2010) 631–631. URL: https://doi.org/10.1007/978-0-387-30164-8_493DOI: 10.1007/978- 0- 387- 30164- 8_493 .

[6] Support Vector Machines. In: Sammut, C., Webb, G.I. (eds) Encyclopedia of Machine Learning and Data Mining. Springer, Boston, MA. https://doi.org/10.1007/978-1-4899-7687-1_810

[7] Multinomial Naive Bayes for Text Categorization Revisited. In: Webb, G.I., Yu, X. (eds) AI 2004: Advances in Artificial Intelligence. AI 2004. Lecture Notes in Computer Science(), vol 3339. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-30549-1_43

[8]@inproceedings{majumdar2022overview,title={{Overview of the IRSE track at FIRE 2022: Information Retrieval in Software Engineering}},author={Majumdar,Srijoni and Bandyopadhyay, Ayan and Das, Partha Pratim and D Clough, Paul and Chattopadhyay, Samiran and Majumder,Prsenjit},booktitle={ForumforInformationRetrievalEvaluation},publisher={ACM},year={2022 },month={December}}

[9] Majumdar, S., Papdeja, S., Das, P.P., Ghosh, S.K. (2020). COMMENT-MINE—A SemantiSearch Approach to Program Comprehension from Code Comments. In: Chaki, R., Cortesi, A., Saeed, K., Chaki, N. (eds) Advanced Computing and Systems for Security. Advances in Intelligent Systems and Computing, vol 1136. Springer, Singapore. https://doi.org/10.1007/978-981-15-2930-6_3

[10] Majumdar, S., Bansal, B., Das, P.P.,Clough, P.D., Dutta, K., Ghosh,S.K. (2022). Automatic Evaluation of Comments to aid software Maintenance.https://doi.org/10.1002/smr.2463

[11] Singer, J., Lethbridge, T., Vinson, N. and Anquetil, N., 2010. An examination of software engineering work practices. In CASCON First Decade High Impact Papers (pp. 174-188).

[12] Etzkorn, Letha H., Carl G. Davis, and Lisa L. Bowen. "The language of comments in computer software: A sublanguage of English." Journal of Pragmatics 33.11 (2001): 1731-1756.

[13] Tilus, Tero, Jussi Koskinen, Jarmo J. Ahonen, Heikki Lintinen, Henna Sivula, and Irja Kankanapää. "Industrial application and evaluation of a software evolution decision model." In Technologies for Business Information Systems, pp. 417-427. Springer, Dordrecht, 2007.

[14] Dehaghani SM, Hajrahimi N. Which factors affect software projects maintenance cost more? Acta Inform Med. 2013 Mar;21(1):63-6. doi: 10.5455/AIM.2012.21.63-66. PMID: 23572866; PMCID: PMC3610582