Comments in Source Code: A classification approach

Amisha Shingala[1], Namrata Shroff[2]

[1] *CHARUSAT University, CMPICA, Changa, Gujarat, India*
[2] *Gujarat Technological University, Ahmedabad, Gujarat, India*

Abstract

Primary and Secondary result was submitted in Information Retrieval of Software Engineering(IRSE) – FIRE-2022. We propose using a domain specific Supervised learning models and classifiers to classify the comment text as Useful and Not-Useful. The study aims to evaluate the usefulness of comments whether they increase code comprehensibility of software maintenance or not. A total 8657 records were given as training dataset which consists of comments, surrounding code and class given and also the test dataset which contains 1000 records was released by FIRE 2022. We developed features to semantically analyze the comments. Using various supervised learning model, the comments are classified as useful or not useful and the best run we achieved with Precision and Recall score of 0.83% and 0.84% respectively using Random Forest model. Also we used Adam optimization algorithm for training Deep Neural Network in our study. The study will help the research community and developer to classify the comments text for larger code based data.

Keywords

Supervised Learning, Deep Learning, Comment code, Classification, FIRE

1. Introduction

The journey of thought process sparks with reading a paper [2], which motivates us to do work on the task assigned by FIRE team. To perform any modification or enhancement in software code, developer struggle a lot to understand the code and this leads to time consuming for a maintenance task to be performed. Also, we all are aware that reading comment along with source code, will aid to comprehend the design of code and can locate the relevant dependency for errors. The comments can be noisy, inconsistent and may not evolve with source code, they are still semantically rich and easier to follow the documentation tasks [2]. The quality of comment written helps to develop guidelines for the developer in maintenance of the code. Accessing quality of comments in terms of usefulness of information will based on context of the code. Several matrix and features has been proposed by many researchers to access quality of information contained in comment line in terms of its importance for the code. Thus we can say that identifying comment category will form the basis for comment classification and quality assessment of the code. We analyze the code and comment given as an input to our model, pre-process the code and comment, extract different lexical and non-lexical features from comment and code text and finally using supervised learning model, we predict the usefulness of comments for given test data, also, we try to evaluate the results using Deep learning techniques also and the overall performance is good but still needs improvement, our results are encouraging enough to work for better results in future. The exploratory study and experimental results from other developer team of FIRE 2022 are available in paper [1].

"Forum for Information Retrieval Evaluation, December 9-13, 2022, India"
EMAIL: amishashingala.mca@charusat.ac.in (A.Shingala); nam.shroff@gmail.com (N.Shroff);
ORCID: 0000-0001-9304-8246 (A.Shingala); 0000-0001-6343-6205 (N.Shroff);

2. Tasks

Our work discusses to automate the approach to classify the code comments into two classes. We explore the use of several supervised machine learning algorithm and deep learning algorithm to design our code in order to do the following tasks. The two classes and its input and output are described below:

Comment Classification: A binary classification task to classify source code comments as Useful or Not Useful for a given comment and associated code pair as input and a sample is shown in figure-1.

Input: A code comment with associated lines of code (written in C)

Output: A label (Useful or Not Useful) in helping developers comprehend the associated code

#	Comment	Code	Label
1	/* uses png_calloc defined in pngriv.h*/	/* uses png_calloc defined in pngriv.h*/ PNG_FUNCTION(png_const_structrp png_ptr) { if (png_ptr == NULL || info_ptr == NULL) return; png_calloc(png_ptr); ...}	Useful
2	/* serial bus is locked before use */	static int bus_reset (. . .) /* serial bus is locked before use*/ { .. update_serial_bus_lock (bus * busR); }	Not Useful
3	// integer variable	int Delete_Vendor; // integer variable	Not Useful

Figure-1: Sample of task classification

3. Related Work

In [2] author collected total of 20,206 comments from GitHub and from industry experts by conducting the survey. They design the Comment Probe framework, extract the features from comments, developed SD Ontology, developed knowledge domain, did pilot study, taken developer interviews by making questionnaires'', identify the comment categories, Survey the directed examples, built Knowledge Graph and using Neural Network- classified the comments as Useful, Not Useful and Partially Useful. Also, they validate the model, finding overall quality scores, distribute the important comment categories, automatically classify the comment quality using LSTM and ANN combined architecture and analysis of false positive in Comment Probe framework. In future, the vector scape model can be enrich using transformer based model-BERT. Also, correlation to run-time behavior can be analyze in future.

In [4] author work for toxic comment classification. They compare the public multi-label dataset of more than 200,000 user comments. They apply the supervised machine learning classifier to train the data and validate the results. Also, they apply two different pre-trained word embedding for user comments and tweets. Also, they compare the classifier predictions and make different errors as measured by Person correlation coefficients and F1-measures.

4. Dataset

The development dataset provided during the track which contains set of 8657 comments (from GitHub) is released on 1-June-2022. It contains comment text, surrounding code snippets and class label (Useful and Not Useful). The test dataset contains 1000 records which was released on 1-July-2022. It contains comment text, surrounding code snippets and class label (Useful and Not Useful).

4.1 Trends in Dataset

Based on given and collected information, the following trends were observed in the dataset.

- Training dataset includes 46% comments Not Useful and 54% comments are Useful.
- The duplicate rows in training dataset are 4653.

- The null value in code is 699 and comments are 4141 for training dataset.
- For testing dataset, total 75% comments are Not Useful and 25% comments are Useful. The duplicate rows in testing dataset are 203.
- Most common words in dataset includes ‘static’, ‘constant’, ‘return’, ‘int’, ’char’, ’null’, ‘include’ etc. These words are extracted using vocabulary package in python.
- Cosine similarity between comments and code words from training dataset are 0.48 and for testing dataset it was 0.79.

5. Methodology

5.1 Pre-Processing

Following the prior experience of NLP tasks [4], we pre-processed the comments in order to improve the quality of word embedding produced by supervised learning algorithms. The pre-processing done on both the column of training and testing dataset. The columns are surrounding-code-contents and comments.

- Remove stop words: A stop word is commonly used words such as ‘a’, ’an’, ’the’, ’in’ etc., which do not provide any valuable information to the model, so we remove such words and focuses on important information. We imported the stop words library from NLTK corpus.
- Convert words to lowercase: To enhance the text analysis, and Part of speech tagging, we convert all the sentences from upper case to lowercase.
- Expand contractions to text: In order to standardize the text, each contraction is converted to its expanded, original form. For example, words like “don’t” to do not.
- Remove non-alphanumeric characters: For enhancing the text analysis, we remove all non-letter characters such as brackets, colon, semi-colon, special characters, extra white space characters., etc.
- Remove URLs: URL do not help in our analysis, so we removed it using regular expression from text.

5.2 Feature selection

After pre-processing of the text, we have to make data understandable by machine learning model. For that, various text representation schemes have been developed. Text representation is for numerical representation of document so that it can be feed as an input to the classifier. The numerical representation is in the form of vectors that together form a matrix [5]. The main objective of this paper is an identification of best text representation scheme to model the comment probe for classification of its usefulness or not. We use two types of text representation schemes: (i) Bag-of-words (BOW) and (ii) word embedding.

The Bag-of-words(BOW) is the simple method to represent the text or document in the vector form and is very common feature extraction method. Word count or TF/IDF (Term Frequency – Inverse Document Frequency) weight of each n-gram word used as a feature. Another feature can be word embedding, which is text representation techniques to represent word in low dimensional space so that similar words can be represented in semantically format. Major word embedding technique such as Word2Vec is considered. It maps words into vectors of real numbers, convert text corpus into output as a set of vectors.

5.3 Experimental setup

After preprocessing and vectorization of features, we merged the training and testing data, shuffled into training and validation sets in the ratio of 70:30 such that the percentage of instances of each class were preserved. We used cross-validation to split train data into random training and validation sets for 10 k-fold stratified splits provided by sklearn model.

5.4 Model selection

After preprocessing, vectorization and splitting of dataset, we feed our results into the classifier which encodes the comments label as useful or not useful based on the predictions after learning from training model. We experimented with five classifiers such as Decision Tree, Random Forest, Logistic Regression, KNN and SVM. Thus in our study, we applied supervised learning algorithm as well as deep learning methods. Due to its high success, deep learning method is in-cooperated in our study. We implemented the model using Keras API that is running on top of tensor flow platform using python programming. We used sequential modelling – Adam algorithm for our deep learning experimental study.

5.5 Prediction of Results from Model

For the prediction over available test and validation data, we used various supervised and deep learning model to generate the usefulness and not usefulness of comment classification and prediction of probability score of each comment against both the classes. The accuracy and F1-score calculated for all model was submitted as a primary file for task 1. The final prediction file containing the comment score was submitted as a secondary file for task 2.

5.6 Evaluation and Discussion of Results

The Task 1 track results are evaluated using overall accuracy and macro-F1 score on two classes of comments. Standard Information Retrieval measures such as Precision, Recall, MAP and F-Score were used to evaluate the runs. Higher credit was given to runs that identified more number of comments. For task-1, the run submission was evaluated against their standards and measures like accuracy and macro-F1 scores are used for evaluation of runs which is shown in table-1. The parameters for supervised learning models and its test classification given in table-2. The training and testing data loss epoch wise is shown in figure-2.

Table-1: Model accuracy and F1-Score

Run	Macro F1-Score	Accuracy
Random Forest	0.84	0.83%
Navie Bayer's	0.82	0.83%
SVM	0.80	0.80%
Logistic Regression	0.79	0.82%
Decision Tree	0.79	0.79%
K-Nearest Neighbor	0.74	0.74%

Table 2: Parameters for Model Architecture

Run	Parameters	Test Misclassification Percentage
Random Forest	Min_sample split:2, estimator:100	16.03
Navie Bayer's	-	17.71
SVM	Kernal: Linear, gamma:1e-3 to 1e, C:1000	19.45
Logistic Regression	K-Fold splits: 10, random state:1	20.94
Decision Tree	-	20.29
K-Nearest Neighbor	n-neighbour:5	25.90

Table 3: Deep Learning Model

Run	Parameters	Loss /Mean Square Error
Sequential : Adam model	Kernal : uniform, activation: Relu, sigmoid , learning rate: 0.00009, Dropout : 0.2, batch size: 32, total epoch:20	Epoch: 10/20 : Loss : 0.1547

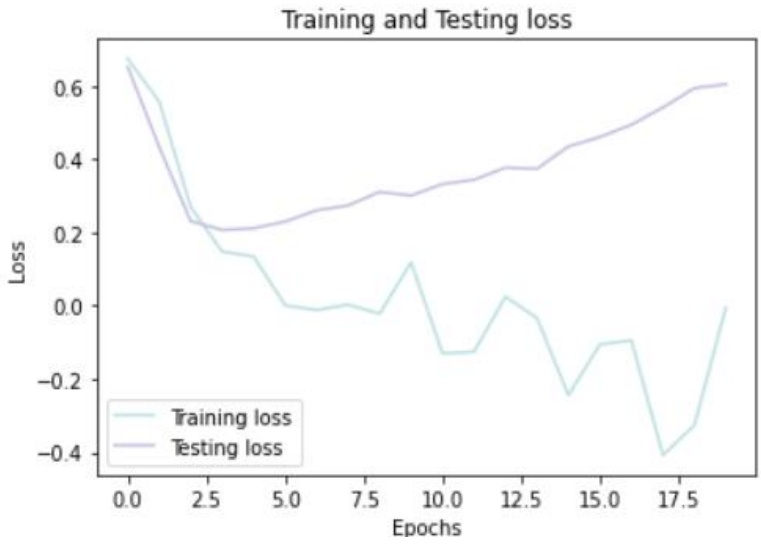

Figure 2: Category loss Vs. No. of Epochs

6. Conclusion and Future Work

This paper uses Supervised Machine learning algorithm and deep neural network to trained on a large corpus of comment text code, to classify comments as Useful or Not Useful. We observed that the transformer-based model outperformed the traditional natural language processing classifier, namely Naive Bayes, SVM and Random Forest etc., as word embedding computed by the former are more expressive and yield better results on the task. We further propose to look into data augmentation strategies for improving the performance of our model since transformer-based models are data-hungry. Another addition could be to adversarial train the model to improve its robustness.

7. Acknowledgements

We thanks to individuals and groups that assisted in the motivation, the identification of task and prompt answering to our queries during our work by Information Retrieval Evaluation Forum, FIRE 2022.

References

[1] Majumdar, Srijoni and Bandyopadhyay, Ayan and Das, Partha Pratim and D Clough, Paul and Chattopadhyay, Samiran and Majumder, Prasenjit, " in processing of majumdar2022overview, Overview of the IRSE track at FIRE 2022: Information Retrieval in Software Engineering.", *Forum for Information Retrieval Evaluation, ACM, December 2022.*

[2] Majumdar, Srijoni, Ayush Bansal, Partha Pratim Das, Paul D. Clough, Kausik Datta, and Soumya Kanti Ghosh. "Automated evaluation of comments to aid software maintenance." *Journal of Software: Evolution and Process* 34, no. 7 (2022): e2463.
[3] Majumdar, Srijoni, Shakti Papdeja, Partha Pratim Das, and Soumya Kanti Ghosh. "Comment-Mine—A Semantic Search Approach to Program Comprehension from Code Comments." In *Advanced Computing and Systems for Security*, pp. 29-42. Springer, Singapore, 2020.
[4] Modha, Sandip Jayantilal. "Microblog processing: summarization and impoliteness detection." PhD diss., Dhirubhai Ambani Institute of Information and Communication Technology, 2019.
[5] Bithel, Shivangi, and Samidha Verma. "VaccineBERT: BERT for COVID-19 Vaccine Tweet Classification." In *Working Notes of FIRE-13th Forum for Information Retrieval Evaluation, FIRE-WN 2021*, pp. 1199-1203. 2021.
[6] Chowdhury, Aayush, Aishwarya Roy, and Aman Choudhary. "Retrieval of Actionable Information during Mass Emergency: A Classification Approach." (2021).
[7] Van Aken, Betty, Julian Risch, Ralf Krestel, and Alexander Löser. "Challenges for toxic comment classification: An in-depth error analysis." *arXiv preprint arXiv:1809.07572* (2018).
[8] Goldani, Mohammad Hadi, Saeedeh Momtazi, and Reza Safabakhsh. "Detecting fake news with capsule neural networks." *Applied Soft Computing* 101 (2021): 106991.
[9] Shah, Darshin Kalpesh, Meet Ashok Sanghvi, Raj Paresh Mehta, Prasham Sanjay Shah, and Artika Singh. "Multilabel Toxic Comment Classification Using Supervised Machine Learning Algorithms." In *Machine Learning for Predictive Analysis*, pp. 23-32. Springer, Singapore, 2021.