Identification of the Relevance of Comments in Codes Using Bag of Words and Transformer Based Models

Sruthi S[1,*], Tanmay Basu[1]

[1]*Department of Data Science and Engineering, Indian Institute of Science Education and Research, Bhopal*

Abstract

The Forum for Information Retrieval (FIRE) started a shared task this year for classification of comments of different code segments. This is binary text classification task where the objective is to identify whether comments given for certain code segments are relevant or not. The BioNLP-IISERB group at the Indian Institute of Science Education and Research Bhopal (IISERB) participated in this task and submitted five runs for five different models. The paper presents the overview of the models and other significant findings on the training corpus. The methods involve different feature engineering schemes and text classification techniques. The performance of the classical bag of words model and transformer-based models were explored to identify significant features from the given training corpus. We have explored different classifiers viz., random forest, support vector machine and logistic regression using the bag of words model. Furthermore, the pre-trained transformer based models like BERT, RoBERT and ALBERT were also used by fine-tuning them on the given training corpus. The performance of different such models over the training corpus were reported and the best five models were implemented on the given test corpus. The empirical results show that the bag of words model outperforms the transformer based models, however, the performance of our runs are not reasonably well in both training and test corpus. This paper also addresses the limitations of the models and scope for further improvement.

Keywords

code comment classification, information retrieval, text classification

1. Introduction

A crucial part in the software development and management industry is program comprehension. It is a usual practice to add new functionalities to the existing programs to meet the dynamic requirements. Therefore understanding the relevant parts and functions of the code is extremely important to avoid redundancy and inefficiency [1]. Developers often act themselves as end users to identify any malfunctions [2] and to mine the process. Even though qualified personnel or peer code reviewer can handle this cumbersome process, the resources needed are quite high. Over the years, researchers formalized structured process for code inspection, the need for inspection and debugging and its benefits [3]. Industries like Microsoft, Google and some open source platforms use another version of the peer code review supported with

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

*Corresponding author.

sruthisudheer1214@gmail.com (S. S); welcometanmay@gmail.com (T. Basu)
https://sites.google.com/view/tanmaybasu/ (T. Basu)

tool-based approaches known as modern code review or contemporary code review [4]. The most useful block in a program review are comments. They are direct and descriptive rather than the source code. This helps reviewers to analyse, add feedback and suggestions which, in a sense, influence the effectiveness of such practices. Nevertheless, some comments can be irrelevant or redundant which make this task even more complicated. So assessing the quality of comments is also necessary as they act as a guide to the reviewer [5].

The Forum for Information Retrieval (FIRE) 2022 started a shared-task this year, named as Information Retrieval in Software Engineering (IRSE)[6], to evaluate the relevance of comments with respect to its surrounding codes. The objective of this shared task is to build a reusable benchmark for evaluating the models to classify the relevance of the comments of given source codes [6]. This is a binary text classification problem to categorize the source code comments as useful or not useful, where the given codes were written in C programming language. We, the BioNLP research group at Indian Institute of Science Education and Research Bhopal (IISERB) participated in this shared task and explored the performance of various feature engineering and classification techniques to categorize the source code comments.

The proposed framework generates features from the given training corpus by using the classical bag of words (BOW) model [7] and transformer architecture based deep learning models [8, 9, 10]. The term frequency (TF) and inverse document frequency (IDF) i.e., TF-IDF [7] and Entropy based method [11] were used as the term weighting schemes. Eventually, the performance of the classifiers like support vector machine, logistic regression and random forest that uses BOW features on the training corpus have been reported . Furthermore, three different attention layer based deep learning models, viz. BERT [8], ALBERT[10], and RoBERTa[9] were implemented to generate semantic features from the given training corpus and then those features were used for comments classification. The top five frameworks that have best performance on the training corpus were chosen based on the F1 score and accuracy. Consequently, these models were implemented on the test data and submitted.

Section 2 of the paper presents the related works in code comment classification. Section 3 describes the proposed frameworks. The experimental results are recorded and analyzed in section 4. Ultimately, the work is concluded in section 5.

2. Related Works

This section briefly describes the related works in developing an architecture that can be used for comment classification. As comments are written in natural language while codes are programming language, detecting inconsistencies of codes with comments are often difficult. Tan et al. [12] developed a framework known as iComment which combines Natural Language Processing (NLP), Machine Learning, Statistics and Program Analysis techniques to overcome this issue. They experimented on four large code bases, namely, Linux, Mozilla, Wine and Apache. The framework has an accuracy of 90.8-100% which also detects bad comments.

Similarly, the base research work of this challenge, named as Commentprobe [13], was implemented on C codebases. First, a developer survey to study the commenting behaviour among the programmers was conducted and then the ground truth for the comment classification task was generated by manual annotation. They developed pretrained embeddings known as SWVec using the data from the posts in Stack Overflow and literature works. These features are trained using neural network architecture like LSTM and ANN to classify the comments as useful or not, or partially useful and could achieve an F1 score of 86.34%.

Apart from the comment classification, many other attempts have been done to understand and transform the code of different programming languages. Some of the works on different types of embeddings are explored in the paper *A Literature Study of Embeddings on Source Code* [14].

3. Experimental Design

The given training and test corpora contain comments, corresponding C code snippets and the class labels, which denotes whether the given comments are useful or not useful. Both the training and test data have these three information and were released in csv format. We extracted the codes and comments to make two different corpora as one containing only comments and the other containing both the code and comments.

The logistic regression (LR) [15], random forest (RF) [16] and support vector machine (SVM)[17, 18] classifiers were implemented using both TF-IDF based term weighting scheme [7] and Entropy based term weighting scheme [19] following the bag of words model. In Entropy based term weighting scheme, the weight[1] of a term in a document is determined by the entropy of term frequency of the term in that document [11, 19]. We implemented the χ^2-statistic and mutual information [20] based term selection techniques to identify a predefined number of top terms from the bag of words. We had done the experiments by using different numbers as threshold for the χ^2-statistic and mutual information based term selection method and then reported the best result for each model. These models were trained only using the comments of the given training corpus and were implemented in Scikit-learn[2], a ML tool in Python. The parameters of the classifiers were tuned using 10-fold cross validation scheme on the training corpus. We did not use the codes to train these models as the BOW model cannot identify relevant characteristics of the code snippets.

Moreover pre-trained transformer based models viz. BERT[3] [8], RoBERT[4] [9] and ALBERT[5] [10] were used and fine-tuned on the given training corpus using both the code and comments following 10-fold cross validation scheme. For ALBERT and ROBERTa models, the length of the tokenized text is fixed as 432 and were trained over 18 and 38 epochs respectively. Subsequently, the best setting of each of these models were tested on the test corpus. The top five models which perform better than the other on the test corpus were submitted to the organizers for

[1] https://radimrehurek.com/gensim/models/logentropy_model.html
[2] http://scikit-learn.org/stable/supervised_learning.html
[3] https://huggingface.co/bert-base-uncased
[4] https://huggingface.co/roberta-base
[5] https://huggingface.co/albert-base-v1

Table 1
Performance of Different Frameworks on the Training Corpus

Feature Types	Classifier	Accuracy	Precision	Recall	F1 Score
Entropy Based Features (using only comments)	Logistic Regression	0.66	0.71	0.62	0.67
	Random Forest	0.68	0.73	0.65	0.69
	Support Vector Machine	0.67	0.72	0.63	0.67
TF-IDF Based Features (using only comments)	Logistic Regression	0.69	0.71	0.73	**0.72**
	Random Forest	0.67	0.72	0.65	0.68
	Support Vector Machine	0.69	0.71	0.70	**0.71**
Transformer Based Features (using both code and comments)	BERT	0.54	0.56	0.66	0.61
	RoBERTa	0.52	0.54	0.84	**0.66**
	ALBERT	0.53	0.54	0.87	**0.67**

final evaluation. The code and data set that are used to implement the proposed framework are available on Github[6].

4. Results and Analysis

Table 1 shows the performance of different models on the training corpus. It may be noted from table 1 that the logistic regression (LR) and SVM classifiers using the classical TF-IDF based term weighting scheme of the bag of words model respectively achieves the best and the second best F1 scores among all the other models. The ALBERT model outperformed BERT and ROBERTa models in terms of F1 score, however, they could not beat the LR and SVM classifiers. Table2 shows the performance of the best five models on the test corpus. Note that we had run the best setting of all the models individually on the test corpus, but just reported the results of the best five models that we submitted as our final runs. The best settings of individual frameworks are also reported in Table2. It can be observed from Table2 that almost all the models achieve poor results on the test corpus in comparison to their performance on the training corpus.

The entropy based SVM classifier outperforms the other models on test corpus, however, it could not beat the performance of many classifiers using the training corpus. On the other hand, the LR and SVM classifiers using TF-IDF based bag of words features, which performed very well on the training corpus, did not produce a similar performance on the test corpus. We could not find the reasons behind such poor performance of many such models due to time constraint, but in future we plan to investigate the same. All the transformer based models perform poorly on the test corpus. The major reason behind this may be the semantics that were necessary for the comments of the test corpus were not captured during training stage by the transformer based models as the size of the training corpus was insufficient for such models. Moreover, the the pre-trained models that we used were developed using the Books corpus[11] and Wikipedia[12] and hence they could not capture the semantics of the codes from the comments.

[6] https://github.com/SruthiSudheer/Comment-classification-of-C-code
[11] https://huggingface.co/datasets/bookcorpus
[12] https://huggingface.co/datasets/wikipedia

Table 2
Performance of the selected Frameworks on the Test Corpus

Submitted Results	Framework	Significant Parameters	Accuracy	Precision	Recall	F1 Score
Run 1	TF-IDF + RF	sm[7] =information gain, #trees = 50, #terms=3000, χ^2-statistic	0.47	0.34	0.97	0.51
Run 2	Entopy+SVM	linear kernel, C[8] =1, γ =scale, #terms=3000, χ^2-statistic	0.53	0.37	0.94	0.53
Run 3	Entropy+RF	sm[7] =information gain, #trees = 50 #terms=3000, χ^2-statistic	0.41	0.32	0.98	0.49
Run 4	ALBERT	epochs=18,ws[9] =500,len[10] =432 batch size=4,weight decay=0.01	0.57	0.33	0.48	0.39
Run 5	RoBERTa	epochs=38,ws[9] =500, len[10] =432 batch size=4, weight decay=0.01	0.58	0.32	0.42	0.36

[7] Splitting measure. [8] Cost parameter. [9] Number of warmup steps for learning rate scheduler. [10] Length of tokenized text

5. Conclusion

The task offered by IRSE Track on FIRE 2022 highlights various challenges for identifying useful comments and thus removing redundancy and non dependency of comments with the source code. In this perspective, we have implemented different frameworks using various types of text features from the given training corpus to identify the relevance of code comments. From the perspective of empirical analysis none of the models achieve reasonable performance on the test corpus. In future, we need to investigate the reasons to develop novel models to improve the performance. However, we feel that the given training corpus of codes and comments are very small in size and hence cannot capture all the behavioural aspects of code and comments. We barely used software development concepts which indeed, could have developed a new embedding that can significantly identify relevant features useful in the software domain. The proposed text classification based approaches could not capture all the necessary semantics for software development and maintenance, which needs to be addressed in future.

References

[1] S. C. B. de Souza, N. Anquetil, K. M. de Oliveira, A study of the documentation essential to software maintenance, in: Proceedings of the 23rd Annual International Conference on Design of Communication: Documenting amp; Designing for Pervasive Information, SIGDOC '05, Association for Computing Machinery, New York, NY, USA, 2005, p. 68–75. URL: https://doi.org/10.1145/1085313.1085331. doi:10.1145/1085313.1085331.

[2] T. Roehm, R. Tiarks, R. Koschke, W. Maalej, How do professional developers comprehend software?, in: Proceedings of the 34th International Conference on Software Engineering, ICSE '12, IEEE Press, 2012, p. 255–265.

[3] M. E. Fagan, Design and code inspections to reduce errors in program development, IBM Systems Journal 15 (1976) 182–211. doi:10.1147/sj.153.0182.

[4] C. Bird, A. Bacchelli, Expectations, outcomes, and challenges of modern code review, in: Proceedings of the International Conference on Software Engineering, IEEE, 2013. URL: https://www.microsoft.com/en-us/research/publication/expectations-outcomes-and-challenges-of-modern-code-review/.

[5] S. Majumdar, S. Papdeja, P. P. Das, S. K. Ghosh, Comment-Mine—A Semantic Search Approach to Program Comprehension from Code Comments, Springer Singapore, Singapore, 2020, pp. 29–42. URL: https://doi.org/10.1007/978-981-15-2930-6_3. doi:10.1007/978-981-15-2930-6_3.

[6] S. Majumdar, A. Bandyopadhyay, P. P. Das, P. D Clough, S. Chattopadhyay, P. Majumder, Overview of the IRSE track at FIRE 2022: Information Retrieval in Software Engineering, in: Forum for Information Retrieval Evaluation, ACM, 2022.

[7] C. D. Manning, P. Raghavan, H. Schutze, Introduction to information retrieval (2008).

[8] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[9] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, arXiv preprint arXiv:1907.11692 (2019).

[10] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, Albert: A lite bert for self-supervised learning of language representations, ArXiv abs/1909.11942 (2020).

[11] T. Sabbah, A. Selamat, M. H. Selamat, F. S. Al-Anzi, E. H. Viedma, O. Krejcar, H. Fujita, Modified frequency-based term weighting schemes for text classification, Applied Soft Computing 58 (2017) 193–206.

[12] L. Tan, D. Yuan, G. Krishna, Y. Zhou, /*icomment: Bugs or bad comments?*/, SIGOPS Oper. Syst. Rev. 41 (2007) 145–158. URL: https://doi.org/10.1145/1323293.1294276. doi:10.1145/1323293.1294276.

[13] 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.

[14] Z. Chen, M. Monperrus, A literature study of embeddings on source code, 2019. URL: https://arxiv.org/abs/1904.03061. doi:10.48550/ARXIV.1904.03061.

[15] A. Genkin, D. D. Lewis, D. Madigan, Large-scale bayesian logistic regression for text categorization, Technometrics 49 (2007) 291–304.

[16] B. Xu, X. Guo, Y. Ye, J. Cheng, An improved random forest classifier for text categorization., JCP 7 (2012) 2913–2920.

[17] S. Tong, D. Koller, Support vector machine active learning with applications to text classification, Journal of machine learning research 2 (2001) 45–66.

[18] T. Basu, S. Kumar, A. Kalyan, P. Jayaswal, P. Goyal, S. Pettifer, S. R. Jonnalagadda, A novel framework to expedite systematic reviews by automatically building information extraction training corpora, arXiv preprint arXiv:1606.06424 (2016).

[19] T. Basu, S. Goldsworthy, G. Gkoutos, A sentence classification framework to identify geometric errors in radiation therapy from relevant literature, Information 12 (2021) 139. doi:10.3390/info12040139.

[20] T. Basu, C. Murthy, A supervised term selection technique for effective text categorization, International Journal of Machine Learning and Cybernetics 7 (2016) 877–892.