A study of the impact of generative AI-based data augmentation on software metadata classification

Tripti Kumari[1,*], Chakali Sai Charan[1] and Ayan Das[1]

Department of Computer Science and Engineering
Indian Institute of Technology (ISM) Dhanbad, Jharkhand, 826004, India

Abstract

This paper presents the system submitted by the team from IIT(ISM) Dhanbad in FIRE IRSE 2023 shared task 1 on the automatic usefulness prediction of code-comment pairs as well as the impact of Large Language Model(LLM) generated data on original base data towards an associated source code. We have developed a framework where we train a machine learning-based model using the neural contextual representations of the comments and their corresponding codes to predict the usefulness of code-comments pair and performance analysis with LLM-generated data with base data. In the official assessment, our system achieves a 4% increase in F1-score from baseline and quality of generated data.

Keywords

Comment-code pairs, LLM-generated data, Support vector machine, ELMO

1. Introduction

In the rapidly developing world of software development, comments play a crucial role in enhancing code readability and maintainability of the corresponding codes in the code bases [1]. Before executing any software maintenance-related task, or doing any kind of modification and enhancement, developers usually spend a significant amount of time reading and understanding the codes. This process is very time-consuming, particularly, in the case of source codes that implement complex functionalities. So, it is common practice among developers to write comments for code snippets to enhance the comprehensibility of the code. The comments are expected to be helpful in capturing the complete structure and functionality of the codes. This makes commenting one of the most commonly employed documentation methods for software maintenance tasks [2] on condition that the comments are elaborate and expressive enough to capture the functionality of the programs and that the quality of the comments is maintained throughout the code base.

However, sometimes the comments themselves may be incomplete, inconsistent, and difficult to relate to the source code [3]. Such comments may result in a waste of effort in the interpretation of the corresponding code and even may result in a complete misinterpretation of the purpose of the program.

Thus understanding the relevance of a comment to a piece of code is crucial before actually

Forum for Information Retrieval Evaluation, December 15-18, 2023, India

[*] Corresponding author.

22dr0264@iitism.ac.in (T. Kumari); 22mt0348@iitism.ac.in (C. S. Charan); ayandas@iitism.ac.in (A. Das)

Table 1
Samples of original base data

Comment	Surrounding code context	Label	Explanation
/*deal with it later*/	-1. /*deal with it later*/1.	Not Useful	The code does not exist for this comment. So, comment is Not Useful
/*switch on*/	-1.f(toggle) /*switch on*/1.else	Useful	The comment correctly describes the code and hence Useful

using it to understand the purpose of the program. However, given the volume of source code in a standard software project, it is a laborious task to manually verify the usefulness of each comment to their corresponding code. Thus, a system that can automatically predict the usefulness of a comment to its related code snippet may significantly speed up the process of source code analysis. Furthermore, the comment may be rewritten to make it more relevant and informative in case the system predicts the comment to be unuseful.

Recently, artificial intelligence-based interactive systems, such as ChatGPT [4] are being widely used to generate texts for different real-time purposes. These systems are also being used by programmers to generate comments for their programs to save time and effort. However, no work has been reported in the literature on the quality of the comments generated by such systems. Given a code-comment pair, these systems may also be used to predict the usefulness of the comment to the corresponding code. However, the accuracy of the predictions of the AI-based systems is not reported in the literature. Thus, it is an open and interesting research area to explore the efficacy of such AI-based systems in automatic comment generation or prediction of the usefulness of a comment for a given code snippet.

The Task-1 of FIRE 2023 IRSE shared task[5], mainly focuses on two subtasks. The first subtask is *comment classification*. It involves automatically predicting the usefulness of a given comment to the corresponding source code snippet. It is a binary classification task, that requires us to develop a system, which takes a source code snippet and their associated comment as input. The proposed system automatically classifies whether the comment corresponding to the source code is ”Useful” or ”Not Useful”. The overview paper of IRSE2022 contains information about the shared task in detail[6]. We have proposed a system, which takes a code-comment pair as input and generates their representations using a pre-trained neural encoder uses these representations to predict whether the comment is relevant to the code.

The second subtask is *to study the impact of large language models in comments*. In this subtask, the participants are required to augment the base data provided for Subtask 1 with additional data and to carry out a comparative study of the performance of the models trained using the base data and augmented data. The additional data for augmentation is expected to comprise the code-comment pairs obtained from different sources with their usefulness labels predicted using large language models (LLMs)[7]. For this purpose, we manually collect code-comment pairs from different data resources such as GitHub, stack overflow, computer vision, Curl, etc., and then queried the ChatGPT[4] with each code-comment pair to get the usefulness label. We augmented this data with the original seed data and trained some models using different combinations of the additional dataset. We carried out a set of experiments to study the effect of data augmentation on the system performance.

This paper reported the comprehensive explanation of the proposed system submitted to FIRE IRSE2023 for task 1[5]. We have conducted some experiments and trained the machine learning models, which take the representations of a snippet of source code and their corresponding comments as input. These trained models made predictions about the relevance of the comment to the associated source code with original base data and augmented datasets.

The remaining sections are arranged as follows. Section 2, presented the related works where we have done some literature surveys on previous work. In this section 3, we have presented a description of different types of LLM-generated datasets and the data made available for the shared task. We have also reported a brief description of data representation and system specification of the system submitted for the shared task. Section 4, presented a comprehensive analysis of the results on different runs with the dataset. In Section 5, we have concluded our work on shared tasks.

2. Related work

We have done some surveys on the usefulness of code-comment pairs as well as the impact of ChatGPT[4] generated comments. We found out some important studies.

Majumdar et al.[8], proposed a survey paper, which is based on the IRSE track (FIRE 2022), and developed solutions for automated evaluation of code comments and classifying comments as useful or not Useful. Rahman et al.[9] did a comparative study on usefulness and developed a RevHelper for automatic usefulness prediction. Soni et al.[10] developed an automatic text classifier to identify ChatGPT-generated summaries. Shinyama et al.[1]propose a model i.e.C4.5 for code-comments analysis. Naili et al.[11]developed a generator network with a coverage mechanism. Pre-trained ELMo contextual embedding was used to generate the highlights of this research paper. Majumdar et al.[12] have proposed a COMMENT-MINE semantic search architecture. this architecture is mainly used to extract knowledge based on the design, implementation, and development of software in the form of a knowledge graph. Majumdar et al.[13], developed features to semantically analyze the comments to concepts based on categories of usefulness. They have used Neural networks(NN) to know the usefulness of code comment pairs. Majumdar et al.[14] search for contextualized embeddings for code search and classification and developed a system for generating contextualized representations for codes and comments by training ELMo from scratch.

3. Experiment design

This section presents a detailed discussion of the proposed system developed for automatically predicting the relevance of code-comment pairs and the experiments carried out on the different combinations of the data sets. In Subsection 3.1 we present the details of the prediction system. The details for the datasets used for the experiments are reported in Subsection 3.2.

3.1. System description

Our prediction system is a supervised machine-learning-based system that consists of a support vector machine (SVM) [15] trained on the distributed representations of the code-comment pairs. It takes the representations of the code-comment pair as input and predicts whether the comment is relevant to the corresponding code snippet.

The distributed representations of code-comment pairs are obtained from a pre-trained ELMO-based model [16]. We have used the ELMO code[1] provided by the Information Retrieval in Software Engineering (IRSE) team. For a given code comment pair, we separately pass the code and the comment to the ELMO model[16] as input as a sequence of tokens. For an input sequence, the ELMO model[16] generates 200-dimensional contextual embeddings for each space-separated token in the input sequence. The representation for an input sequence is then obtained by taking the mean of all the token representations. So, the representation of a given input sequence is a 200-dimensional embedding. The 200-dimensional representations of the code and the comment sequences are then concatenated into a joint 400-dimensional representation.

During training, we generate the representations for all the code-comment pairs in the training data and use them to train the support vector machine using the radial basis function (RBF) kernel[17]. During testing, the model saved during the training phase takes the representation of the code-comment pair as input and predicts the usefulness of the comment.

The details of the working of our prediction system is presented in Figure 1.

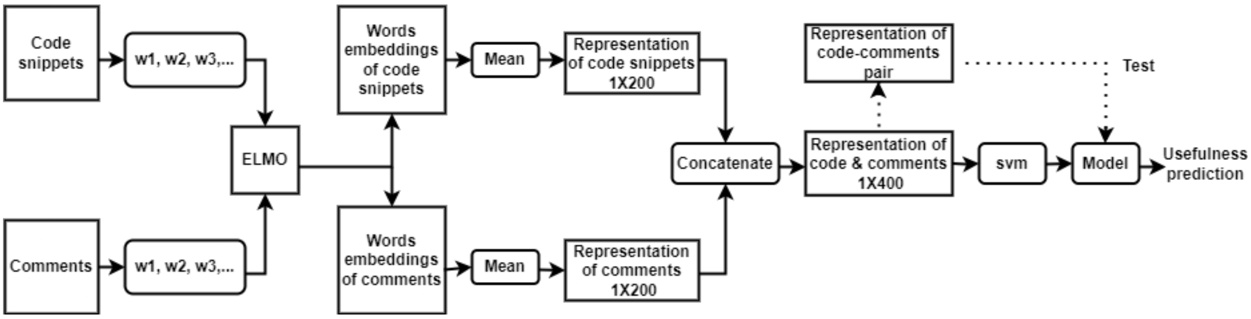
Code snippets
w1, w2, w3,...
ELMO
Comments
w1, w2, w3,...
Words embeddings of code snippets
Words embeddings of comments
Mean
Mean
Representation of code snippets 1X200
Representation of comments 1X200
Concatenate
Representation of code-comments pair
Test
Representation of code & comments 1X400
svm
Model
Usefulness prediction

Figure 1: Block diagram of proposed model

3.2. Data description

Here we present a description of different combinations of the used datasets for our experiments.

3.2.1. Original data

The original data for task 1[5] was shared by the FIRE IRSE 2023. It contained 11,452 pairs of comments, surrounding code snippets, and their class labels. i.e. if a comment is relevant to the corresponding source code then the corresponding pair is labeled as ”Useful” otherwise it is

[1] ELMO Code link to generate word embeddings-https://github.com/SMARTKT/WordEmbeddings

labeled as ”Not Useful”. A total of 11,452 rows of comments were written in text format and their surrounding source codes. A total of 4,389 code-comment pairs are labeled as ”Not Useful” and 7,063 code-comment pairs are labeled as ”Useful”, which is mentioned in a sample example in Table 1 and Table 2.

Table 2
Sample of Original base dataset

Comment	Surrounding code context	Label
/*upper 8 bit CLASS*/	-7.if(dot) -6.host p++;	Useful
/*need expand*/	-1.png set background fixed(png ptr,c;	Not Useful

3.2.2. LLM-generated data

For Subtask 2, we have manually collected a total of 510 code-comments pairs from different data resources such as GitHub, stack overflow, computer vision, curl, etc., and then query the ChatGPT[4] with each code-comments pair to get the usefulness label. Then we augment this data with the original base data seen in Table 3 and we re-trained the model with the augmented data.

3.2.3. Extra-generated data

We have experimented with another set of data where we have randomly extracted a subset of 250 ”Useful” and 250 ”Not Useful” code-comment pairs from the original seed data and altered their labels using the following strategy. We converted the ”Useful” pairs into ”Not Useful” by randomly shuffling the comments and we ensured that at the end of the shuffling none of the codes had their original comments. We labeled such pairs as ”Not useful”. To convert the ”Not Useful” comments to ”Useful”, we queried the ChatGPT[4] with the code snippets and got the comments synthetically generated. This set of code and synthetically generated comment pairs were labeled as ”Useful”. Table 3 gives an explanation of the datasets.

For the sake of convenience, we referred to the original data, LLM-generated data, and extra-generated data as Data1, Data2, and Data3 respectively as shown in Table 3.

Table 3
Different types of data with their size after train test split

Dataset description	Train dataset size	Test dataset size
Original data: Data1	9162	2290
LLM-generated data: Data2	408	102
Extra-generated data: Data3	400	100

We have followed the following steps to split the original and the LLM-generated[7] data: (i) We have separated out the ”Useful” and ”Not Useful” code-comment pairs from the data into two groups. (ii) We then split each group in an 80:20 ratio.

Thus, the training data comprised a combination of 80% of the code-comment pairs with ”Useful” labels and 80% of the code-comment pairs with ”Not Useful” labels. The selection of the 80% of the samples in both cases was done randomly. The test data consisted of the remaining 20% of the samples from both groups.

We followed the same splitting procedure for the ”LLM-generated data”[7] and ”Extra generated data” as well. The train and test split size of the dataset is shown in Table 3.

3.3. Combination of all datasets

We have created different combinations of datasets by combining different types of datasets. The dataset description is given in this subsection (3.2.1, 3.2.2, and 3.2.3). Here, we have combined the different data as shown in Table 4 with train test split sizes of data. The purpose is to create new datasets to understand the impact of system performance on original data, LLM-generated data[7], extra-generated data, and different combinations among them shown in Table 4.

Table 4
Different combination of datasets:

Datasets	Train dataset size	Test dataset size
Dataset1: Data1	9162	2290
Dataset2: Data1+Data2	9570	2392
Dataset3: Data1+Data3	9562	2390
Dataset4: Data1+Data2+Data3	9970	2492

4. Result Analysis

In this section, we present a discussion of our results. We performed four different experiments with different combinations of test datasets as shown in Table 5, and Table 6.

4.1. Run1: Original data (Dataset1)

We performed experiments with the original base data of size 11,452. Original data is split into train and test of sizes 9162 and 2290 in the ratio of 80:20 shown in Table 4. We used only the test data of the original base data (Dataset1) for usefulness prediction.

4.2. Run2: Combination of original data and LLM-generated data (Dataset2)

Our second experiment was carried out with original data and LLM-generated data[7]. A total of 510 LLM-generated data are split in the ratio of 80:20 into train and test data sizes are 408 and 102 respectively. Now, the total sum of training data and test data of Dataset2 sizes are 9570 and 2392. To analyze the impact of LLM-generated data[7] on proposed system performance, we augmented test data of original data and LLM-generated data[7] (Dataset2) are shown in Table 4.

Table 5
Experiment analysis part-1

Experiments	Datasets	Algorithm	Accuracy
Run1	Dataset1	ELMO, SVM	92.18
Run2	Dataset2	ELMO, SVM	92.76
Run3	Dataset3	ELMO, SVM	90.696
Run4	Dataset4	ELMO, SVM	92.47

4.3. Run3: Original data and extra-generated data (Dataset3)

Our third experiment is with the combination of original base data and extra generated data. A total of 500 extra-generated data are split in the ratio of 80:20 into train and test data sizes are 400 and 100 respectively. Now, the total sum of training data and test data of Dataset3 sizes are 9562 and 2390.

4.4. Run4: Original data, LLM-generated data, and extra-generated data (Dataset4)

We did one more experiment with the combination of original data, LLM-generated data[7], and extra-generated data. The total sum of training data and test data of Dataset4 sizes are 9970 and 2492.

4.5. Result summary

The overall accuracies corresponding to the experiments carried out for Run1 (Subsection 4.1, Run2 (Subsection 4.2), Run3 (Subsection 4.3) and Run4 (Subsection 4.4) are 92.18%, 92.76%, 90.696%, and 92.47% respectively. The results are summarized in Table 5. In Run3 (Dataset3), the accuracy value decreases, and in other Runs (with Dataset1, Dataset2, Dataset3), we are getting almost the same accuracies with slight variation in decimal fractions value.

To evaluate the performance of the system with respect to the ”Useful” class, we have used precision, recall, and F1-score as evaluation metrics. The results are summarized in Table 6. We have carried out different runs using their corresponding useful class dataset and evaluated the Useful precision, recall, and F1-score. In Run1 (Useful dataset size -1465) and Run4 (Useful dataset size- 1578), we are getting the same precision, recall, and F1 score. But, in the case of Run2 (Useful dataset size- 1542), it gets slightly higher recall than other Runs but other evaluation parameters remain the same and in Run2 (Useful dataset size- 1470), all evaluation parameters slightly decrease than other Runs.

5. Conclusion

In this paper, we presented our proposed system submitted for participating in task-1 shared by IRSE FIRE 2023. The first task of shared task-1 is to build a system that takes a code-comment pair as input to the encoder, which generates embedding that is passed to the classifier and the classifier classifies whether the comment that corresponds to the code is ”Useful” or ”Not

Table 6
Experiment analysis part-2 with ”Useful” class

Experiments	Useful dataset size	Useful precision	Useful recall	Useful F1-score
Run1	1465	0.92	0.96	0.94
Run2	1542	0.92	0.97	0.94
Run3	1470	0.89	0.96	0.93
Run4	1578	0.92	0.96	0.94

Useful”. The second task is to make predictions on the augmentation of original seed data and LLM-generated data. We have also done impact analysis and model performance with an augmented dataset(original base data and LLM-generated data). All the performance evaluation metrics parameters are mentioned in Table 5 and Table 6. According to the declared result, our system achieves a 4% increase in F1-score from baseline and quality of data generated.

References

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

[2] S. C. B. de Souza, N. Anquetil, K. M. de Oliveira, A study of the documentation essential to software maintenance, Association for Computing Machinery (2005) 68–75. URL: https://doi.org/10.1145/1085313.1085331. doi:10.1145/1085313.1085331.

[3] 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.

[4] OpenAI, Gpt-4 technical report, 2023. arXiv:2303.08774.

[5] 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.

[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] T. Gao, H. Yen, J. Yu, D. Chen, Enabling large language models to generate text with citations, arXiv preprint arXiv:2305.14627 (2023).

[8] 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.

[9] M. M. Rahman, C. K. Roy, R. G. Kula, Predicting usefulness of code review comments using textual features and developer experience, in: 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), IEEE, 2017, pp. 215–226.

[10] M. Soni, V. Wade, Comparing abstractive summaries generated by chatgpt to

real summaries through blinded reviewers and text classification algorithms, 2023. arXiv:2303.17650.

[11] M. Naili, A. H. Chaibi, H. H. B. Ghezala, Comparative study of word embedding methods in topic segmentation, Procedia computer science 112 (2017) 340–349.

[12] S. Majumdar, S. Papdeja, P. P. Das, S. K. Ghosh, Comment-mine—a semantic search approach to program comprehension from code comments, Advanced Computing and Systems for Security: Volume Twelve (2020) 29–42.

[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] 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.

[15] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (1995) 273–297.

[16] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, Association for Computational Linguistics (2018) 2227–2237. URL: https://aclanthology.org/N18-1202. doi:10.18653/v1/N18-1202.

[17] K. Thurnhofer-Hemsi, E. López-Rubio, M. A. Molina-Cabello, K. Najarian, Radial basis function kernel optimization for support vector machine classifiers, arXiv preprint arXiv:2007.08233 (2020).