Generative AI for Software Metadata: Overview of the Information Retrieval in Software Engineering Track at FIRE 2023

Srijoni Majumdar[1,2,*,†], Soumen Paul[1,*,†], Bhargav Dave[6,*,†], Debjyoti Paul[8], Ayan Bandyopadhyay[4], Samiran Chattopadhyay[7], Partha Pratim Das[1], Paul D Clough[4,5] and Prasenjit Majumder[3,6]

[1]*IIT Kharagpur, West-Bengal, India*
[2]*University of Leeds, UK*
[3]*TCG CREST, West-Bengal, India*
[4]*TPXimpact London, UK*
[5]*Sheffield University, Sheffield, UK*
[6]*DA-IICT Gandhinagar, Gujarat, India*
[7]*Jadavpur University, West-Bengal, India*
[8]*Indian Statistical Institute, Kolkata India*

Abstract

The Information Retrieval in Software Engineering (IRSE) track aims to develop solutions for automated evaluation of code comments in a machine learning framework based on human and large language model generated labels. In this track, there is a binary classification task to classify comments as useful and not useful. The dataset consists of 9048 code comments and surrounding code snippet pairs extracted from open source *github* C based projects and an additional dataset generated individually by teams using large language models. Overall 56 experiments have been submitted by 17 teams from various universities and software companies. The submissions have been evaluated quantitatively using the F1-Score and qualitatively based on the type of features developed, the supervised learning model used and their corresponding hyper-parameters. The labels generated from large language models increase the bias in the prediction model but lead to less over-fitted results.

Keywords

bert, GPT-2, Stanford POS Tagging, neural networks, abstract syntax tree

Forum for Information Retrieval Evaluation (FIRE)- 2023, Indian Statistical Institute, Kolkata, India, $15^{th} - 18^{th}$ *December, 2023*

*Corresponding author.
†These authors contributed equally.
majumdar.srijoni@gmail.com (S. Majumdar); soumenpaul165@gmail.com (S. Paul); bhargavdave1@gmail.com (B. Dave); debjyoti93.paul@gmail.com (D. Paul); bandyopadhyay.ayan@gmail.com (A. Bandyopadhyay); samiran.chattopadhyay@jadavpuruniversity.in (S. Chattopadhyay); ppd@cse.iitkgp.ac.in (P. P. Das); p.d.clough@sheffield.ac.uk (P. D. Clough); prasenjit.majumder@gmail.com (P. Majumder)

1. Introduction

Assessing comment quality can help to de-clutter code bases and subsequently improve code maintainability. Comments can significantly help to read and comprehend code if they are consistent and informative.

The perception of quality in terms of the 'usefulness' of the information contained in comments is relative and hence is perceived differently based on the context. Bosu et al. [1] attempted to assess code review comments (logged in a separate tool) in the context of their utility in helping developers write better code through a detailed survey at Microsoft. A similar quality assessment model is important to analyse the type of source code comments that can help for standard maintenance tasks but is largely missing. Majumdar et al. [2] proposed a comment quality evaluation framework wherein comments were assessed as 'useful', 'partially useful', and 'not useful' based on whether they increase the readability of the surrounding code snippets. The authors analyse comments for concepts that aid in code comprehension and also the redundancies or inconsistencies of these concepts with the related code constructs in a machine learning framework for an overall assessment. The concepts are derived through exploratory studies with developers across 7 companies and from a larger community using crowd-sourcing.

The first edition of the IRSE track of FIRE 2023, extends the work in [2] and empirically investigates comment quality with a larger set of machine learning solvers and features. The task is based on the quality evaluation of comments into two clusters - 'useful' and 'not useful'. A 'useful' comment (refer Table 1) contains relevant concepts that are not evident from the surrounding code design, and thus increases the comprehensibility of the code. The suitability of analysing comment quality using various vector space representations of code and comment pairs along with standard textual features and code comment correlation links are evaluated.

Table 1
Useful and Not-Useful comments in context of code comprehension

#	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); ...}	U
2	/* serial bus is locked before use */	static int bus_reset (. . .) /* serial bus is locked before use*/ { .. update_serial_bus_lock (bus * busR); }	NU
3	// integer variable	int Delete_Vendor; // integer variable	NU

U: Useful; NU: Not Useful

The 2023 IRSE track extends this challenge to understand the feasibility of using silver standard quality labels generated from the Large Language Models (LLMs) and understand how it augments the classification model in terms of prediction. Developing the gold industry

standard for analysing the usefulness of comments that can be relevant for code comprehension in legacy systems can be challenging and time-consuming. However, to scale the model and use it on different languages, it is important to generate more data which we attempt to do with the large language models. The performance of these modes in the context of understating the relations between code and comment can provide an approximation of the data quality generated and how it can be used to scale the existing classification mode. This approach can also be further generalised to any classification model based on software metadata.

2. Related Work

Software metadata is integral to code maintenance and subsequent comprehension. A significant number of tools [3, 4, 5, 6, 7, 8] have been proposed to aid in extracting knowledge from software metadata like runtime traces or structural attributes of codes.

In terms of mining code comments and assessing the quality, authors [9, 10, 11] compare the similarity of words in code-comment pairs using the Levenshtein distance and length of comments to filter out trivial and non-informative comments. Rahman et al. [12] detect useful and non-useful code review comments (logged-in review portals) based on attributes identified from a survey conducted with developers of Microsoft [1]. Majumdar et al. [2, 13] proposed a framework to evaluate comments based on concepts that are relevant for code comprehension. They developed textual and code correlation features using a knowledge graph for semantic interpretation of information contained in comments.

These approaches use semantic and structural features to design features to set up a prediction problem for useful and not useful comments that can be subsequently integrated into the process of decluttering codebases.

With the advent of large language models [14], it is important to compare the quality assessment of code comments by the standard models like GPT 3.5 or llama with the human interpretation. The IRSE track at FIRE 2023 extends the approach proposed in [2] to explore various vector space models [15] and features for binary classification and evaluation of comments in the context of their use in understanding the code. This track also compares the performance of the prediction model with the inclusion of the GPT-generated labels for the quality of code and comment snippets extracted from open-source software.

3. IRSE Track Overview and Data Set

The following section outlines the task descriptions and the characteristics of the dataset.

3.1. Task Description

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.
`Input`: A code comment with surrounding code snippet (written in C)
`Output`: A label (Useful or Not Useful) that characterizes whether the comment helps developers comprehend the associated code

Therefore, in this classification task, the output is based on whether the information contained in the comment is relevant *and* would help to comprehend the surrounding code, i.e., it is *useful*.

Useful: Comments have sufficient software development concept → Comment is Relevant, and these concepts are not mostly present in the surrounding code → Comment is not Redundant, hence the comment is *Useful*

Not Useful: Comments have sufficient software development concept → Comment is Relevant, and these concepts are mostly present in the surrounding code → Comment is Redundant, hence the comment is *Not Useful*

It may also be the case that comments do not contain sufficient software development concepts → Comment is Not Relevant, hence the comment is *Not Useful*.

It is left to the participants to decide on the threshold value for how many concepts retrieved make a comment relevant or how many matches with surrounding code make a comment redundant.

The notion of *relevant* comments refers to those that developers perceive as important in comprehending the associated or surrounding lines of code. These concepts are related to the outline of the algorithm, data-structure descriptions, mapping to user interface details, possible exceptions, version details, etc. In the below examples, the comments highlight useful details about the input data to the function, which is not evident from the associated code itself.

Dataset: For the IRSE track, we use a set of 9048 comments (from Github) with comment text, surrounding code snippets, and a label that specifies whether the comment is useful or not. Sample data has been characterised in Table 1.

- The development dataset contains 8048 rows of comment text, surrounding code snippets, and labels (Useful and Not useful).
- The test dataset contains 1,000 rows of comment text, surrounding code snippets, and labels (Useful and Not useful).

4. Participation and Evaluation

IRSE 2023 received a total of 56 experiments from 17 teams for the two tasks. As this track is related to software maintenance, we received participation from companies like Microsoft, Amazon, American Express, Bosch Research along with several research labs of educational institutes.

The various teams with the details of their submissions are characterised in Table 2.

Evaluation Procedure: Candidates submitted the prediction metrics (precision, recall, F1-Score) for the classification model with the Gold labels dataset (referred to as the Seed Dataset) and combined dataset (Seed + LLM generated labels - Silver labels dataset). The difference in the F1 score was evaluated by us.

Features: Apart from evaluating the prediction metrics, we analysed the types of features the teams have used to devise the machine learning pipeline. The teams have performed routine pre-processing and have retained the significant words or letters only for both the code and comment pairs. Further, some of the teams have also used morphological features of a comment

Table 2
Characterizations of the Submissions: test Data Predictions

Affiliation	Seed data			Seed data + LLM-generated data		
	Precision	Recall	F1Score	Precision	Recall	F1Score
DSTI, France	0.8326	0.8626	0.8473	0.844	0.8682	0.8559
	0.8948	0.8738	0.884	0.9	0.8707	0.885
	0.8807	0.8822	0.8813	0.8871	0.8839	0.8854
SSN-1 (RAM)	0.8	0.8	0.8	0.8021	0.81	0.73
	0.72	0.71	0.74	0.7	0.73	0.74
SSN-2 (Aloy)	0.788	0.7363	0.7613	0.89	0.8802	0.8846
	0.7994	0.7994	0.7994	0.89	0.8795	0.8841
	0.7993	0.9352	0.8619	0.839	0.9199	0.8776
	0.7842	0.8453	0.8136	0.8154	0.8823	0.8475
	0.7572	0.8637	0.807	0.7785	0.9003	0.835
IIT (ISM) Dhanbad	0.92	0.96	0.94	0.92	0.97	0.97
SSN-3 (Black)	0.7916	0.8446	0.8172	0.7886	0.847	0.8167
	0.763	0.8696	0.813	0.7655	0.8724	0.8154
	0.705	0.9387	0.8052	0.6994	0.9041	0.7887
	0.7292	0.856	0.7875	0.7374	0.8533	0.7911
Microsoft-American Express	0.7902	0.8016	0.7949	0.7908	0.8014	0.7952
DDU-1	0.895	0.891	0.893	0.890	0.894	0.892
DDU-2	0.875	0.872	0.874	0.870	0.875	0.880
IIT KGP-1	0.8283	0.804	0.8141	0.8322	0.8086	0.8185
SRM	0.8283	0.804	0.8141	0.8178	0.7906	0.8013
IIT KGP-2	0.78	0.85	0.8	0.77	0.85	0.8
DA-IICT	0.81	0.8	0.8	0.58	0.58	0.58
IIT Goa	0.6087	0.6526	0.6321	0.6114	0.6598	0.6403
TCS	0.778	0.753	0.74	0.645	0.6598	0.650
IIT KGP-3	0.631	0.645	0.639	0.6114	0.6598	0.631
Amazon	0.659	0.672	0.666	0.656	0.635	0.645

like a length, significant words ratio, parts of speech characteristics, or occurrence of words from an enumerated set as textual features. To correlate code and comment and detect redundancies, the teams mostly used grep-like string match to find similar words.

Vector Space Representations: Code and comments belong- to different semantic granularity which is unified by a vector space representation. The participants have used various pre-trained embeddings to generate vectors for the words like those based on one hot encoding, tf-idf based, word2vec or context aware like ELMo and BERT. Each of the employed embedding models are trained or finetuned using software development corpora.

Results: The participants are able to achieve a slight increase (in the range of 2%-4%) in the test prediction metrics and in many cases the performance decrease. The statistics of LLM generated data submitted by each team is illustrated in Table 3. However, the increase in bias due to the incorporation of silver standard data reduces the over-fitting of the models.

Table 3
Characterizations of the LLM Generated datasets

Team name	Total entry	Useful entry	Not useful entry
DSTI, France	421	412	9
SSN-1 (RAM)	1238	740	497
SSN-2 (Alloy)	1510	24	1486
IIT (ISM) Dhanbad	199	182	17
SSN 3 (Black)	738	80	658
Microsoft - American Express	233	92	141
DDU-1	8588	4649	3939
DDU-2	332	311	21
IIT KGP-1	334	309	25
SRM	217	196	21
IITKGP-2	263	130	133
DA-IICT	150	65	85
IIT-Goa	543	460	83
TCS	282	61	221
IITKGP-3	570	450	120
IITKGP-3	412	345	67

5. Conclusions

The IRSE 2023 track empirically investigates the feasibility of augmenting existing classification models using datasets with labels generated from LLM's. A total of 17 teams participated and submitted 56 experiments that used various types of machine learning models, embedding spaces, features and different LLMs to generate data. The LLM-generated labels reduce the overfitting of the overall classification model and also improve the F1 score when the combined data from all participants were used to augment the existing data with gold standard labels from industry practitioners.

References

[1] A. Bosu, M. Greiler, C. Bird, Characteristics of useful code reviews: An empirical study at microsoft, Working Conference on Mining Software Repositories, IEEE, 2015, pp. 146–156.

[2] 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.

[3] S. Majumdar, S. Papdeja, P. P. Das, S. K. Ghosh, Smartkt: a search framework to assist program comprehension using smart knowledge transfer, in: 2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS), IEEE, 2019, pp. 97–108.

[4] N. Chatterjee, S. Majumdar, S. R. Sahoo, P. P. Das, Debugging multi-threaded applications using pin-augmented gdb (pgdb), in: International conference on software engineering research and practice (SERP). Springer, 2015, pp. 109–115.

[5] S. Majumdar, N. Chatterjee, S. R. Sahoo, P. P. Das, D-cube: tool for dynamic design discovery

from multi-threaded applications using pin, in: 2016 IEEE International Conference on Software Quality, Reliability and Security (QRS), IEEE, 2016, pp. 25–32.
[6] S. Majumdar, N. Chatterjee, P. P. Das, A. Chakrabarti, A mathematical framework for design discovery from multi-threaded applications using neural sequence solvers, Innovations in Systems and Software Engineering 17 (2021) 289–307.
[7] S. Majumdar, N. Chatterjee, P. Pratim Das, A. Chakrabarti, Dcube_ nn d cube nn: Tool for dynamic design discovery from multi-threaded applications using neural sequence models, Advanced Computing and Systems for Security: Volume 14 (2021) 75–92.
[8] M. P. O'brien, Software comprehension–a review and research direction, Technical Report Technical Report, Department of Computer Science & Information Systems University of Limerick, Ireland, 2003.
[9] D. Steidl, B. Hummel, E. Juergens, Quality analysis of source code comments, International Conference on Program Comprehension (ICPC), IEEE, 2013, pp. 83–92.
[10] 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.
[11] S. Majumdar, A. Bandyopadhyay, S. Chattopadhyay, P. P. Das, P. D. Clough, P. Majumder, Overview of the irse track at fire 2022: Information retrieval in software engineering, in: Forum for Information Retrieval Evaluation, ACM, 2022.
[12] M. M. Rahman, C. K. Roy, R. G. Kula, Predicting usefulness of code review comments using textual features and developer experience, International Conference on Mining Software Repositories (MSR), IEEE, 2017, pp. 215–226.
[13] S. Majumdar, S. Papdeja, P. P. Das, S. K. Ghosh, Comment-mine - a semantic search approach to program comprehension from code comments, in: Advanced Computing and Systems for Security, Springer, 2020, pp. 29–42.
[14] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Advances in neural information processing systems 33 (2020) 1877–1901.
[15] 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.