Check for updates

ACM DIGITAL LIBRARY | Association for Computing Machinery | acm open

Latest updates: https://dl.acm.org/doi/10.1145/3734947.3735667

SHORT-PAPER

Overview of the "Information Retrieval in Software Engineering" (IRSE) track at Forum for Information Retrieval 2024

SOUMEN PAUL, Indian Institute of Technology Kharagpur, Kharagpur, WB, India

SRIJONI MAJUMDAR, University of Leeds, Leeds, West Yorkshire, U.K.

RAJ SHAH, Indian Institute of Technology Goa, Ponda, GA, India

SUSMITA DAS, University of Glasgow, Glasgow, Scotland, U.K.

MADHUSUDAN GHOSH, Indian Association for the Cultivation of Science, Kolkata, WB, India

DEBASIS GANGULY, University of Glasgow, Glasgow, Scotland, U.K.

View all

Open Access Support provided by:

University of Glasgow

University of Leeds

Techno India University, West Bengal

Indian Institute of Technology Kharagpur

Indian Association for the Cultivation of Science

Indian Institute of Technology Goa

View all

PDF Download
3734947.3735667.pdf
05 February 2026
Total Citations: 1
Total Downloads: 336

Published: 12 December 2024

Citation in BibTeX format

FIRE 2024: Proceedings of the 16th Annual Meeting of the Forum for Information Retrieval Evaluation
December 12 - 15, 2024
Gandhinagar, India

https://doi.org/10.1145/3734947.3735667
ISBN: 9798400713187

Overview of the "Information Retrieval in Software Engineering" (IRSE) track at Forum for Information Retrieval 2024

Soumen Paul
IIT Kharagpur
Kharagpur, West Bengal, India
soumenpaul165@gmail.com

Srijoni Majumdar
University of Leeds
Leeds, United Kingdom
s.majumdar@leeds.ac.uk

Raj Shah
Indian Institute of Technology Goa
Ponda, India
raj.shah.21031@iitgoa.ac.in

Susmita Das
University of Glasgow
Glasgow, United Kingdom
2956827d@student.gla.ac.uk

Madhusudan Ghosh
Indian Association of Cultivation of Science
Kolkata, India
madhusuda.iacs@gmail.com

Debasis Ganguly
University of Glasgow
Glasgow, United Kingdom
debasis.ganguly@glasgow.ac.uk

Gul Calikli
University of Glasgow
Glasgow, United Kingdom
handangul.calikli@glasgow.ac.uk

Debarshi Sanyal
Indian Association of Cultivation of Science
Kolkata, India
debarshisanyal@gmail.com

Partha Pratim Das
Ashoka University
Haryana, India
ppd@ashoka.edu.in

Paul D. Clough
University of Sheffield
Sheffield, United Kingdom
p.d.clough@sheffield.ac.uk

Ayan Bandyopadhyay
Techno India University
Kolkata, India
bandyopadhyay.ayan@gmail.com

Samiran Chattopadhyay
Techno India University
Kolkata, India
samiran.chattopadhyay@jadavpuruniversity.in

Abstract

The "Software Engineering Information Retrieval" (IRSE) track aims to devise solutions for the automated evaluation of code comments within a machine learning framework, with labels generated by both humans and large language models. Within this track, we offered a total of two tasks this year - i) a *comment usefulness prediction* task, and ii) a *code quality estimation* task.

The comment classification task involves discerning comments as either useful or not useful. The dataset includes 9,048 pairs of code comments and surrounding code snippets drawn from open-source C-based projects on GitHub and an additional dataset generated by teams employing large language models. In total, 12 teams representing various universities have contributed their experiments. These experiments were assessed through quantitative metrics, primarily the F1-Score, and qualitative evaluations based on the features developed, the supervised learning models employed, and their respective hyper-parameters. It is worth noting that labels generated by large language models introduce bias into the prediction model but lead to less over-fitted results.

The sub-track pertaining to code quality estimation was introduced this year. Given a problem description, and a list of large language model (LLM) generated software code, the objective of the task is to automatically estimate the functional correctness of each generated code. For the purpose of evaluation, each problem-solution pair is then ranked by these estimated probabilities of functional correctness, the quality of which is then reported with standard ranking performance measures.

CCS Concepts

• **Code Comment Quality**; • **Transformers**; • **Large Language Models**; • **Code Coherence**;

Keywords

Large Language Models, Comment Usefulness Prediction, Code Quality Estimation

ACM Reference Format:
Soumen Paul, Srijoni Majumdar, Raj Shah, Susmita Das, Madhusudan Ghosh, Debasis Ganguly, Gul Calikli, Debarshi Sanyal, Partha Pratim Das, Paul D. Clough, Ayan Bandyopadhyay, and Samiran Chattopadhyay. 2024. Overview of the "Information Retrieval in Software Engineering" (IRSE) track at Forum for Information Retrieval 2024. In *Proceedings of the 16th Annual Meeting of the Forum for Information Retrieval Evaluation (FIRE '24), December 12–15, 2024, Gandhinagar, India.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3734947.3735667

This work is licensed under a Creative Commons Attribution International 4.0 License.

FIRE '24, Gandhinagar, India
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1318-7/24/12
https://doi.org/10.1145/3734947.3735667

1 Introduction

Assessing the quality of comments plays a crucial role in streamlining codebases, which, in turn, enhances code maintainability. Well-structured comments can significantly aid in the readability

and comprehension of code, provided they are consistent and informative. Perceptions of comment quality, particularly regarding the "usefulness" of the information they convey, are context-dependent and can vary from one situation to another. Bosu et al. [1] endeavored to evaluate code review comments (from a separate tool) in terms of their utility in assisting developers in writing better code. This assessment was based on an extensive survey conducted at Microsoft. However, there needs to be a similar quality assessment model specifically tailored to analyze source code comments that are valuable for routine maintenance tasks.

Majumdar et al. [5] introduced a framework for evaluating comment quality, categorizing comments as "useful", "partially useful" or "not useful". This categorization is determined by whether the comments enhance the understandability of the surrounding code snippets. The authors employ a machine learning framework to assess comments, considering concepts that aid in code comprehension and identifying redundancies or inconsistencies in relation to the code constructs. These concepts were derived through exploratory studies involving developers from seven companies and a broader community, using crowd-sourcing as a valuable resource.

In the first iteration of the IRSE track at FIRE 2022, the work in [5] is extended, and an empirical investigation into comment quality is conducted with an expanded set of machine learning techniques and features. In 2023, the IRSE track takes a step further by introducing a challenge focused on evaluating the viability of incorporating silver standard quality labels generated by Large Language Models (LLMs)[6, 8]. The objective is to assess how this addition enhances the predictive capabilities of the classification model[7]. Establishing a gold industry standard for assessing the usefulness of comments that aid in understanding code, especially in legacy systems, can be a demanding and time-consuming activity. Nevertheless, generating a larger dataset becomes crucial to expand the model's applicability to various programming languages. This expansion is being pursued through the utilization of large language models. The performance of these models, particularly in comprehending the relationships between code and comments, can offer insights into the quality of the generated data and its potential for scaling the existing classification model. Furthermore, this approach can be generalized to apply to any classification model rooted in software metadata.

In addition to the core track of comment usefulness detection, this year - as a pilot track, we introduced a new sub-track pertaining to quality estimation of large language model (LLM)-generated code. More precisely speaking, given a problem description comprised of a natural language description of a programming task along with a partially written code, e.g., the function prototype, a common practice is to employ generative AI models (instruction tuned LLMs specifically fine-tuned on software code, e.g., CodeLlama, Codestral etc.) to automatically generate code to solve the given task. Given a pair comprised of a problem description along with an LLM-generated solution the objective is to estimate the likelihood that the code is functionally correct, i.e., it provides a correct solution to the problem. As analogy, this task is somewhat similar to the task of query performance prediction (QPP) [2, 4] in IR, where the objective is to estimate the quality of a retrieved list of documents in terms of relevance (which is substituted by the notion of 'functional correctness' in the context of our problem).

Table 1: Test data predictions of the submitted systems.

	Seed			Seed + LLM-augmented		
Affiliation	P	R	F1	P	R	F1
IIT KGP 1	0.8426	0.8576	0.8428	0.8462	0.8582	0.8573
IIT KGP 2	0.8100	0.8600	0.7923	0.8221	0.8241	0.8164
IIT KGP 3	0.7738	0.7233	0.7863	0.7900	0.7802	0.8046
IIT KGP 4	0.8100	0.8103	0.8212	0.8321	0.8121	0.8195
IIT KGP 5	0.7916	0.8446	0.8172	0.7886	0.8470	0.8167
IIT Goa 1	0.7901	0.8043	0.7942	0.7976	0.8017	0.7987
IIT Goa 2	0.8621	0.8750	0.8530	0.8900	0.8940	0.8920
IIT Goa 3	0.8350	0.8520	0.8340	0.8730	0.8710	0.8800
IIT Goa 4	0.7983	0.8040	0.7841	0.7922	0.8086	0.7985
IIT Goa 5	0.8283	0.8040	0.8141	0.8178	0.7906	0.8013
SRM Chennai 1	0.8120	0.7930	0.8000	0.8231	0.8500	0.8320
SRM Chennai 2	0.8143	0.8231	0.8000	0.8213	0.8423	0.8287

2 Tasks and Data Sets

We now describe the task and the dataset details of the two subtracks (ST) for IRSE.

2.1 ST-1: Comment Usefulness Prediction

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

- *Useful* Comments have sufficient software development concept → Comment is Relevant, and these concepts are not primarily present in the surrounding code → Comment is not Redundant.
- *Not Useful* Comments have sufficient software development concept → Comment is Relevant, and these concepts are mostly present in the surrounding code → Comment is Redundant.

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.

2.2 ST-2: Code Quality Estimation

Task and Evaluation Measures. We scope the code quality estimation task to estimate the *functional correctness* of code snippets generated via LLMs in response to a prompt specifying a programming task. In particular, we make use of the HumanEval[1] dataset for this task, which constitutes of 161 programming problem descriptions. Given a programming task description P, and a list of m solutions $\mathcal{S}^P = \{S_1^P, \ldots, S_m^P\}$ generated by an LLM, a predictor model θ should estimate a likelihood score of the functional correctness of each solution, i.e., $\theta : P, \mathcal{S} \mapsto \mathbb{R}^m$.

An effective model should estimate a high likelihood value for a functionally correct solution (the ground-truth being computed via a set of test-cases), which means that a standard evaluation metric for a ranking task may also be applied here - the only difference being the notion of 'relevance' replaced with that of 'functional correctness' (P being analogous to a query and $\mathcal{S}^P$ to that of a set of top-m retrieved documents). Motivated by this analogy, we report

[1] https://huggingface.co/datasets/openai/openai_humaneval

> Given the problem {Problem} and the solution {Solution}, generate a likelihood score between 0 and 1 indicating how relevant the solution is to the problem. Only state the score.

Figure 1: Prompt used by the participating team for the code quality estimation task via GPT-3.5 0-shot inference.

nDCG@m (in our setting, $m = 10$, i.e., 10 solutions are generated for each problem) as an evaluation measure.

Additionally, we also report a global ranking effectiveness measure to compare across the performance over all problem tasks. Specifically, we use the predicted likelihoods to rank all the P, S_i^P pairs for each $P \in \mathcal{P}$ (the set of all problem tasks in a benchmark), and compute the nDCG value of this set, i.e., nDCG@$(m|\mathcal{P}|)$.

To differentiate the two measures, we call the former local nDCG (**l-nDCG**) and the latter global nDCG (**g-nDCG**). More precisely speaking, to calculate **l-nDCG**, we rank P, S_i^P pairs for each problem P, calculate nDCG and then calculate the average of nDCG values for all $P \in \mathcal{P}$ (i.e., $\sum_{j=1}^{|\mathcal{P}|}(nDCG^j/|\mathcal{P}|)$), whereas to calculate **g-nDCG**, we rank all $m|\mathcal{P}|$ pairs of P, S_i^P for all $P \in \mathcal{P}$ and then calculate the nDCG value.

3 Participation and Evaluation

3.1 ST-1: Comment Usefulness Prediction

IRSE 2024 received a total of 12 experiments from 12 teams for the two tasks. As this track is related to software maintenance, we received participation from several research labs of educational institutes.

The various teams with the details of their submissions are characterized in Table 1. The dataset provided was balanced and had 4015 useful comments and 4033 not useful comments. The participants used various pre-trained embeddings such as one hot encoding, TF-IDF vectorizer, word2vec, or context-aware like ELM or BERT to generate vectors for the word sequence. Teams have used several machine learning models like support vector machine, logistic regression, and deep-learning based models such as BERT, Recurrent neural network, and so on.

Some participants were observed to achieve a slight increase in test accuracy when the model was trained with the addition of an LLM-generated dataset. However, in many cases, the accuracy reduces (2%-4%). This behavior is due to the incorporation of silver standard data that reduces the over-fitting of the models.

3.2 ST-2: Code Quality Estimation

A team from IIT-KGP participated in this task. Similar to the methodology proposed in [9], they employed GPT-3.5 Turbo zero-shot inference on a problem description and a solution pair to estimate how likely is the solution to be functionally correct. They submitted three runs with three different temperature (τ) settings for the GPT decoder (specifically, $\tau = 0.7$, $\tau = 0.8$ and $\tau = 0.9$). The prompt used by the participating team is shown in Figure 1.

Participant	Method	Evaluation Metrics l-nDCG	g-nDCG
IIT KGP	GPT-3.5 ($\tau = 0.7$)	0.6595	0.9108
IIT KGP	GPT-3.5 ($\tau = 0.8$)	**0.6616**	**0.9109**
IIT KGP	GPT-3.5 ($\tau = 0.9$)	0.6602	0.9107
CodeBERT-CLS	CodeBERT CLS	0.6401	0.9036

Table 2: Evaluation of the submitted runs for three different temperature settings and the in-house baseline of CodeBERT-based embedding similarities.

To set a reference point for comparison purposes, we employed a relatively simple heuristic baseline which given a problem description and a list of solutions measures the variance across the semantic similarities between each solution pair. For measuring the semantic similarity between a pairs of code solutions, we use the CLS embeddings obtained from CodeBERT [3], a BERT model fine-tuned on large volumes of source code data.

The assumption of using the variance across generated code solutions as an estimate is that topical diversity of the solutions may indicate lack of consistency in the solutions being generated, which could be be associated with a risk of the solutions being incorrect. Making this assumption is appropriate since HumanEval dataset contains the method signature for each problem. Although there can be different ways to provide solution S (i.e., implement code) for a problem P (e.g., by using different data structures or implementing a recursive algorithm instead of an iterative one, the method signature limits the way one can provide a solution S for a given problem P.

Table 2 shows that the GPT-based 0-shot inference produced better results than the in-house heuristic-based baseline of estimating code quality as a measure of the topical diversity between the LLM-generated solutions.

4 Conclusions

The first sub-task of the IRSE track investigated various approaches for automated comment quality evaluation. The quality of comments were evaluated based on whether they contain information that can aid in understanding the surrounding code. A total of 12 teams participated that used various types of machine learning models, embedding spaces, features, and different LLMs to generate data. The best F1-Score of 0.853 was reported by a team that achieved an improved F1-score of 0.892 while adding the LLM-generated data. The LLM-generated labels reduce the overfitting of the classification models and also improve the F1-score when the combined data from all the participants were used to augment the existing data with gold standard labels from the industry practitioners.

The second sub-task of the IRSE track evaluated the effectiveness of predictive models in estimating the functional correctness of LLM-generated code. We observed that an LLM-based solution towards code quality estimation works better than an embedding-based baseline.

References

[1] Amiangshu Bosu, Michaela Greiler, and Christian Bird. 2015. Characteristics of useful code reviews: An empirical study at microsoft *(Working Conference on*

Mining Software Repositories). IEEE, 146–156.

[2] Suchana Datta, Debasis Ganguly, Derek Greene, and Mandar Mitra. 2022. Deep-QPP: A Pairwise Interaction-based Deep Learning Model for Supervised Query Performance Prediction. In *WSDM*. ACM, 201–209.

[3] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. *CoRR* abs/2002.08155 (2020). arXiv:2002.08155 https://arxiv.org/abs/2002.08155

[4] Debasis Ganguly, Suchana Datta, Mandar Mitra, and Derek Greene. 2022. An Analysis of Variations in the Effectiveness of Query Performance Prediction. In *ECIR (1) (Lecture Notes in Computer Science, Vol. 13185)*. Springer, 215–229.

[5] Srijoni Majumdar, Ayush Bansal, Partha Pratim Das, Paul D Clough, Kausik Datta, and Soumya Kanti Ghosh. 2022. Automated evaluation of comments to aid software maintenance. *Journal of Software: Evolution and Process* 34, 7 (2022), e2463.

[6] Srijoni Majumdar, Soumen Paul, Debjyoti Paul, Ayan Bandyopadhyay, Samiran Chattopadhyay, Partha Pratim Das, Paul D Clough, and Prasenjit Majumder. 2023. Generative ai for software metadata: Overview of the information retrieval in software engineering track at fire 2023. *arXiv preprint arXiv:2311.03374* (2023).

[7] Soumen Paul. 2022. Source Code Comment Classification using Logistic Regression and Support Vector Machine.. In *FIRE (Working Notes)*. 53–59.

[8] Soumen Paul, Srijoni Majumdar, Ayan Bandyopadhyay, Bhargav Dave, Samiran Chattopadhyay, Partha Das, Paul D Clough, and Prasenjit Majumder. 2024. Efficiency of Large Language Models to scale up Ground Truth: Overview of the IRSE Track at Forum for Information Retrieval 2023. In *Proceedings of the 15th Annual Meeting of the Forum for Information Retrieval Evaluation* (Panjim, India) *(FIRE '23)*. Association for Computing Machinery, New York, NY, USA, 16–18. doi:10.1145/3632754.3633480

[9] Terry Yue Zhuo. 2024. ICE-Score: Instructing Large Language Models to Evaluate Code. arXiv:2304.14317 [cs.AI]