Summary Review: Too Noisy To Learn: Enhancing Data Quality for Code Review Comment Generation

Xiangrui Ke
x3ke@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

Review : Too Noisy To Learn: Enhancing Data Quality for Code Review Comment Generation

1 Summary

This work presents a groundbreaking exploration into improving the data quality of automated code review systems through large language models (LLMs). Recognizing that existing code review datasets are often contaminated with noisy comments which is vague, or non-actionable, and current cleaning methods using heuristics and ML approaches still incur noisy and lead models to generate low-quality review comments because it lacks a complex semantic understanding of both code changes and natural language comments. The authors point out three key research questions in LLM-driven code review comment generation.

By answering these questions, authors propose semantic detect and cleaning approach to distinguish valid review comments from noisy ones. Their empirical evaluation, conducted on the widely used CodeReviewer benchmark, demonstrates that LLMs such as GPT-3.5 and Llama 3 can classify the valid comments with high precision. Furthermore, fine-tuning existing code review models on the cleaned datasets leads to substantial generation performance improvement — both in currency and quality in several different metrics.

This work not only validates the feasibility of leveraging LLMs for semantic data labeling in software engineering field but also underscores a critical insight: quality outweigh quantity in terms of data when enhancing model performance. The work represents an advanced step toward LLM-assisted software engineering, offering both methodological innovation and practical implications for building reliable, high-utility automated code review systems.

2 Strength

- It is the first study to employ large language models (LLMs) for semantic cleaning of code review datasets, addressing a long-standing challenge of noise in code review dataset. This contribution is both novel and practically significant, as it introduces a new research direction for enhancing dataset quality in software engineering.
- This work conduct a comprehensive empirical evaluation involving multiple LLMs, diverse prompt designs, and multiple evaluation metrics including precision, recall, BLEU-4, informativeness, and relevance. This experimentation provides strong empirical grounding for their claims.
- The study delivers insightful results demonstrating that improved data quality can substantially enhance model performance, even when the dataset size is reduced — quality over quantity.
- This paper pioneeringly investigates the feasibility of using LLMs as data curators in software engineering, extending their applicability beyond traditional text generation tasks to the realm of dataset refinement and quality assurance.

3 Weakness

- This work relies on a human-labeled subset of valid and noisy comments to guide and evaluate the LLM-based classification. This manual annotation process, although necessary, may introduce human bias and subjectivity in defining what constitutes a valid review comment, potentially influencing both the model evaluation and the downstream cleaning quality.
- Despite leveraging LLMs for semantic labeling, the overall framework still requires a non-trivial amount of human effort, including manual labeling for training and evaluation as well as prompt tuning and qualitative analysis. This limits the scalability and full automation potential of the proposed approach.
- The experimental scope is confined to a single benchmark dataset–CodeReviewer. The generalizability of the findings to other datasets, programming languages, or industrial code review environments remains to be validated. A broader empirical evaluation would strengthen the external validity of the conclusions.

4 Future Work

Future research can further extend and strengthen this study in several directions. First, this paper focuses on assessing the validity of review comments, future work could also evaluate their technical correctness and usefulness, thereby providing a more holistic assessment of comment quality. Incorporating correctness-oriented metrics would help ensure that the generated comments are not only actionable but also technically sound.

Second, the development pipeline could be enhanced by integrating more LLM-driven components, such as automated labeling refinement, dynamic prompt optimization, or self-consistency checks, to further reduce human involvement and improve scalability.

Third, it would be valuable to explore domain generalization, testing the proposed cleaning and generation approaches across diverse datasets, programming languages, and industrial code review environments. Such extensions would help validate the robustness and general applicability of the LLM-based semantic cleaning framework.

5 Discussion on Class

We have started several discussion on the class.

- One classmate brought up that this work didn't implement the model selection of LLMs, or even considered it as one of the methodological flaws of this paper. He pointed out that LLMs selected by this paper are: GPT-3.5, CodeLlama, and Llama3. And all of these three are decoder-only LLMs. Normally, LLMs with an encoder-only architecture are considered better at understanding code, and decoder-only architectures are considered better at generating code. The task for this paper is to learn the code and review comments, and decide whether they are valid or not. The classmate suggests that at least they should include some encoder-only and encoder-decoder models for comparison.
  While I responded to the student's question and explained that this paper mainly focus on the investigation on the feasibility of LLM-assist code review, more likely serves as a lower bound to the problem, in the sense that if randomly selected models could work well, more strengthened models are more likely to work better than the baseline.
- One classmate asked why it is important to only have clean comments, and why any comments asking for clarifications are not good comments. Code review comments that ask for more clarification may indicate that the code snippet is not readable or that the documentation is missing. It's an indicator that something is wrong and cannot be simply removed.
  One classmate was skeptical about how they break down the valid comments and invalid comments. Many people second it, and another person pointed out that sometimes vague comments make sense to human beings, in the sense that they know the context of the code snippet, but they do not make sense to LLMs.
  I totally agreed with that we could act more smartly to deal with these datasets. On the one hand, valid ones will always help improve the model's performance, and on the other hand, we could refine the vague ones to make them clear enough for LLMs to learn, instead of simply getting rid of them so as to avoid leaving out important information.
- One student linked the paper to her current project and highlighted that LLMs are surprisingly poor at identifying sarcasm. Deciding whether or not one comment is valid is manually done in this paper. Though in most cases, we normally don't have time to do so much manual validation, it's definitely not the best practice to just throw that data away.
- Professor concluded that we could leave the nuance and sarcasm identification work for future directions, and right now, we should focus on letting it walk before running. People should be more tolerant of it and focus on what it can do, instead of judging it at the moment.

6 Rating

4/5 Overall, this study presents an advanced LLM-assisted exploration in software engineering, demonstrating that LLM-assisted data cleaning can significantly improve the quality and performance of automated code review models. The work delivers a key insight for future research — data quality matters more than the quantity — highlighting that cleaned datasets can achieve better, more reliable outcomes.