Rethinking Training Data for Generating Code Review Comments

1st Leonardo Centellas-Claros
Department of Computer Science, School of Engineering
Pontificia Universidad Católica de Chile
Santiago, Chile
lcentellas@uc.cl

2nd Estefania Pakarati-Cofre
Department of Computer Science, School of Engineering
Pontificia Universidad Católica de Chile
Santiago, Chile
estefania.pakarati@uc.cl

3rd Juan Pablo Sandoval Alcocer
Department of Computer Science, School of Engineering
Pontificia Universidad Católica de Chile
Santiago, Chile
juanpablo.sandoval@uc.cl

4th Diego Elias Costa
REALISE Lab
Concordia University
Montreal, Canada
diego.costa@concordia.ca

***Abstract*—Generating code review comments has become a prominent research direction in automated code review, commonly formulated as a text generation task over diff-comment pairs. Despite advances in learning-based approaches, generated review comments are often generic, weakly grounded, or non-actionable. Recent studies have also shown that review comment datasets contain noisy or unsuitable training instances, motivating LLM-based dataset cleaning approaches. In this paper, we argue that problematic training instances are not homogeneous and that some limitations stem from deeper issues in the task formulation itself. Through an empirical inspection of a widely used review comment dataset, we identify *misaligned training pairs*: instances where the relationship between the code change and the review comment does not provide a reliable learning signal for generating actionable review feedback from localized inputs. We derive a taxonomy of misalignment capturing three recurring sources: *semantic ambiguity*, *lack of actionability*, and *context dependence*. We further explore whether incorporating this taxonomy into LLM-based filtering improves the identification of problematic training instances, observing that detecting misaligned training pairs remains challenging. Based on these observations, we argue that improving review comment generation requires more than dataset cleaning alone, motivating explicit validity criteria, richer contextual inputs, and evaluation practices aligned with review intent and actionability.**

***Index Terms*—Software Engineering, Code Review Automation, Dataset Cleaning**

I. Introduction

Automated code review aims to support developers by either providing feedback similar to that produced during human code review [1] or transforming code into a reviewed version [2]. With the increasing adoption of learning-based approaches and large language models, generating review comments has become a prominent research direction in software engineering. Most existing approaches formulate this task as a supervised text generation problem over diff-comment pairs mined from pull request reviews, where localized representations of code changes, such as diff hunks or modified methods, are used to predict human-written review comments [3]–[5].

Despite substantial progress in modeling techniques, the effectiveness of automatically generated review comments remains limited [6]. Prior studies report that generated comments are often *generic, weakly grounded, or non-actionable* [7]. To address these limitations, most prior work has focused on improving model architectures, scaling learning approaches, or refining prompting strategies.

More recently, researchers have started questioning the quality of review comment datasets themselves. Tufano et al. [6] and Liu et al. [8] reported that a substantial portion of review comment datasets correspond to noisy or unsuitable training instances. Liu et al. [8] further explored LLM-based dataset cleaning, showing that automatically identifying valid training instances remains challenging and that cleaning alone only partially improves review comment generation performance. Taken together, these observations suggest that the limitations of review comment generation may stem not only from modeling challenges, but also from how the task itself is instantiated through current training datasets.

However, we argue that problematic training instances are not homogeneous. While some correspond to conventional forms of dataset noise, others may reflect a deeper structural issue related to the task formulation itself. We hypothesize that some review comments are legitimate review interactions whose interpretation depends on contextual information not available in localized input representations, leading to a *structural misalignment* between review comments, training inputs, and the intended learning objective.

In this paper, we empirically investigate these problematic training instances through an inspection of the [3] dataset. We derive a taxonomy of *misaligned training pairs*: instances where the relationship between the code change and the review comment does not provide a reliable learning signal for generating actionable review feedback from localized inputs. Our taxonomy identifies three recurring sources of misalignment: *semantic ambiguity*, *lack of actionability*, and

context dependence. We further investigate whether incorporating our taxonomy into LLM-based filtering can improve the identification of misaligned training pairs, observing that distinguishing these cases remains challenging even when explicit misalignment categories are provided.

Taken together, these observations suggest that improving review comment generation requires more than dataset cleaning: some problematic cases stem from deeper limitations in how the task is formulated through localized-input representations. Progress therefore requires *explicit validity criteria* for training instances, *richer contextual inputs* that expose review dependencies, and evaluation practices aligned with *review intent and actionability* rather than surface-level textual similarity.

II. Automated Review Comment Generation

Code review serves as both a quality assurance mechanism and a means of knowledge sharing [9], with reviewers providing feedback through natural-language comments that suggest improvements or identify issues. Automating this feedback has attracted increasing research attention [10].

Most existing approaches to review comment generation formulate the task as a supervised text generation problem. Training data is typically constructed by mining pull request reviews, where each instance consists of a code change paired with a corresponding review comment [3]. Depending on the approach, the input is operationalized using different localized views of the change, including the entire code diff associated with the comment [7], [11], a diff hunk [3], or only the modified method after the change [1], [4].

Given a localized representation of a code change as input, models are trained to generate review comments that resemble those written by human reviewers. This formulation has been widely adopted across learning-based approaches, including encoder-decoder architectures [5] and, more recently, large language models [11].

While intuitive, this formulation implicitly defines what it means to "generate a review comment". By treating localized code representations as self-contained training instances, datasets assume that the available input is sufficient to infer review intent and actionability, and that all reviewer comments constitute appropriate generation targets. As a result, the task is effectively reduced to reproducing the surface form of review comments, rather than modeling when and how actionable feedback should be provided. This task definition has direct implications. When relevant information lies outside the localized input, models are forced to rely on generic patterns observed in the data, which can yield fluent but weakly grounded comments and limit actionability.

III. Empirical Observations on Review Comment Datasets

Rather than evaluating model performance or dataset quality in isolation, we focus on characterizing misaligned training pairs in datasets for review comment generation and on understanding how these misalignments shape the learning objective induced by current task formulations.

A. Methodology

We follow a four-step methodology:

1) *Dataset selection.* To enable comparison with related studies such as [8], we ground our empirical observations in a widely used public dataset for review comment generation, whose training split contains 117,739 diff-comment pairs [3]. We selected this dataset because it is representative of current practice in automated code review [6] and has served as the basis for numerous studies, comparisons, and models since its publication [7], [12], [13].
2) *Sample collection.* As analyzing all training pairs is not feasible, we determined the sample size using standard random sampling with a 95% confidence level and a 5% margin of error, resulting in a required sample of 383 training pairs. These pairs were drawn uniformly at random and independently inspected by two reviewers. Each sampled pair consists of a code diff and its associated review comment.
3) *Instance screening.* Two reviewers independently assessed whether each sampled diff-comment pair constituted actionable review feedback that a model would be expected to generate, having only received the diff as input. Both reviewers are postgraduate researchers in software engineering, with prior experience in code review and software quality analysis. Inter-rater reliability was substantial ($\kappa = 0.758$). Disagreements were subsequently discussed to reach a consensus, resulting in a final set of 184 misaligned training pairs.
4) *Taxonomy construction.* Following the screening step, we obtained a set of 184 training pairs identified as *misaligned* with the objective of generating actionable review comments from change-local inputs. We based the taxonomy construction on an open card sorting approach, following the methodology proposed by [14]. For each training pair, reviewers assigned descriptive category labels capturing the underlying source of the misalignment, and recorded short notes to clarify category boundaries. As the analysis progressed, categories were iteratively revised, merged, or expanded.

B. Results: Taxonomy of Misaligned Training Pairs

Table I summarizes the resulting taxonomy of *misaligned training pairs*, which groups these cases into three high-level categories: *semantic ambiguity*, *lack of actionability*, and *context dependence*. Rather than representing isolated noise, these categories capture systematic patterns that reflect different review-related intents or dependencies on information that is not available in the provided change-local input.

- *Semantic ambiguity (25/184).* This category includes comments whose intent is unclear or under-specified, preventing the inference of a concrete action. This category covers cases where reviewers explicitly express uncertainty about whether an issue exists, as well as comments whose phrasing is incomplete or vague. Although such

TABLE I
TAXONOMY OF MISALIGNED TRAINING PAIRS IN REVIEW COMMENT DATASETS.

Category	Freq.	Description
Semantic Ambiguity		
Unsure	5	Expresses uncertainty about whether an issue exists, without proposing a concrete action.
Unclear	20	Intent is ambiguous or incomplete, preventing inference of a concrete action by the developer.
Lack of Actionability		
Inquiry	56	Asks for clarification, but does not suggest or justify any modification.
Comment	9	Refers to the pull request without requesting changes or providing a clear purpose.
Chit-chat	1	Casual or conversational remark with no technical relevance.
Remark	32	Descriptive or acknowledging statement that does not imply action and often reflects self-commentary.
Context Dependence		
Context-dependent	44	Although the comment may be useful, it is not possible for a model or reviewer to infer this comment from the information present in the diff alone.
Another change	1	Refers to a change located in another diff of the same commit.
Other line	14	Refers to code outside the displayed diff within the same file.
Previous comment	1	Explicitly refers to an earlier review comment.
Broken URL	1	Discusses a URL whose validity cannot be determined from the available data.

comments are part of real review conversations, they do not provide a clear learning signal for generating actionable feedback grounded in the code.

- *Lack of actionability (98/184).* This category encompasses comments that serve conversational, coordinative, or descriptive purposes rather than proposing or justifying code changes. This includes inquiries that ask for clarification without suggesting a modification, as well as remarks and acknowledgments that comment on the pull request without implying any required action. While these comments are legitimate artifacts of the review process, they are far from the actionable review feedback expected of a model's output.
- *Context dependence (61/184).* This category captures comments that are potentially actionable, but whose correctness depends on information not present in the localized input. These include references to other changes within the same commit, code outside the displayed diff, prior review comments, or external artifacts such as URLs. For such instances, generating appropriate feedback requires access to additional context beyond the diff-comment representation. We argue that a model trained on these types of instances would learn to guess a review rather than to infer it from its input.

While some degree of noise is expected in large-scale datasets, these cases go beyond incidental dataset noise. **Our analysis shows that 184 out of 383 sampled training pairs (48%) are misaligned training pairs**, with lack of actionability and missing contextual information being the two most frequent sources of misalignment. This indicates that such cases constitute a substantial portion of the dataset rather than isolated anomalies. This proportion is also in line with prior work reporting large numbers of problematic review-comment pairs, including 38% [6] and 25% noisy instances [8]. Note that, although related, our analysis goes beyond dataset noise and focuses on *misalignment*. A replication package with all experimental data is available online [15].

System Prompt

Role and Task
Your task, as an experienced code reviewer is ...
<TASK_INSTRUCTION>

You are an expert code reviewer evaluating whether a human-written review comment is suitable training data ...

Definition
<SUITABLE_INSTANCE_DEFINITION>

Evaluation Criteria
Your evaluation should be guided by the following criteria:
1) Relevance to Code Change: ...
2) Clarity and Constructiveness: ...
3) Focus on Improvement: ...

Categories
Consider the following categories:
1) <CATEGORY_DEFINITION_1>
2) <CATEGORY_DEFINITION_2>
3) ...

User Prompt

Below is a code diff and review comment.
Please evaluate whether this comment is Valid or Noisy.

Context Code Changes:
<CODE_DIFF>

Input Review Comment:
<REVIEW_COMMENT>

Answer Format:
valid or noisy

Fig. 1. Prompt structure used in our study. Yellow-highlighted sections indicate the modifications introduced over the prompt of Liu et al. [8].

IV. REVISITING LLMS AS A DATA FILTER

Liu et al. [8] previously explored the use of LLMs to filter problematic review-comment pairs, reporting limited effectiveness. In this section, we revisit this idea to evaluate whether our taxonomy helps LLMs better distinguish aligned from misaligned training pairs, and whether the choice of model and input representation influences filtering performance.

A. Methodology

We follow a five-step methodology to evaluate whether taxonomy-guided prompts can improve the filtering of unsuitable review-comment pairs for automated code review generation.

1) *Datasets.* We conducted experiments using two manually annotated datasets. First, we used the dataset released by Liu et al. [8], which contains 270 review-comment pairs sampled from the CodeReviewer dataset and annotated according to the suitability of review comments for automated code review generation. Second, we used our manually curated dataset introduced in this paper, which contains 383 review-comment pairs labeled as either *misaligned* or *aligned*.
2) *LLMs under Study.* We evaluate *GPT-3.5-turbo*, matching the original setup on [8], and *GPT-4o-mini*, a more recent lightweight model, to test whether newer LLMs better identify unsuitable pairs, particularly with taxonomy-derived prompts.
3) *System Prompt (Task Instruction).* We evaluated three prompting strategies that progressively extend the information provided through the *system prompt* to guide the LLM during the filtering task.
   - SP_D (*Definition*): prompt based on Liu et al. [8] that includes only the reviewer role description together with the definitions of valid and noisy review comments used during manual annotation.
   - SP_{DC} (*Definition+Criteria*): extension of SP_D, also developed by Liu et al. [8], that additionally incorporates evaluation criteria and explicit guidance rules to help the model assess the relevance and quality of review comments.
   - SP_{DCT} (*Definition+Criteria+Taxonomy*): our proposed prompt, based on SP_{DC} by incorporating taxonomy-derived categories and criteria related to contextual alignment, actionability, and observable relation with the provided code diff.

   Figure 1 shows the prompt structure proposed originally by Liu et al. and highlights in yellow the changes or additions we made to it. In all configurations, the *task instruction* asks the model to determine whether the provided sample is Valid or Noisy. Our prompt adds to this framing by asking whether the pair is a suitable training instance for automated code review generation.
4) *User Prompt Representations.* We evaluated two *user prompt* input representations that differ in the contextual information provided to the LLM.
   - UP_{NL}: includes only the natural language review comment.
   - $UP_{NL+Diff}$: includes both the review comment and the associated code diff.
5) *Procedure and Metrics* We evaluated all possible combinations of system prompting strategies (3), LLMs (2), and user prompt representations (2), resulting in a total of 12 experimental configurations. For each configuration, the corresponding prompt setup was executed over the full dataset, and the LLM classified each review-comment pair as either a suitable or unsuitable training instance for automated code review generation. We additionally included a random baseline to test whether the prompt-based configurations outperformed chance-level labeling. We then compared the predicted labels against the ground truth annotations and computed precision, recall, and F1-score for both classes to evaluate how effectively each configuration distinguishes suitable from unsuitable review-comment pairs.

B. Results

Table II reports the results across the evaluated prompting configurations. UP_{NL} and $UP_{NL+Diff}$ produced similar results, with UP_{NL} slightly more consistent across datasets. GPT-3.5-turbo consistently outperformed GPT-4o-mini. In several configurations, GPT-4o-mini classified most samples as unsuitable instances, resulting in low recall for the Valid class.

Differences between our taxonomy-guided prompt (SP_{DCT}) and the prompts derived from Liu et al. [8] (SP_D and SP_{DC}) were relatively small. Although SP_{DCT} achieved slightly better results in some configurations, the improvements were marginal and inconsistent across datasets.

Overall, **adding taxonomy-derived categories to the prompt did not substantially improve filtering effectiveness**. Still, LLM-based filtering captured some signal beyond chance, with GPT-3.5-turbo achieving F1 scores between 0.58 and 0.71 for the Valid class across the best-performing configurations. These results suggest that more effective filtering may require clearer validity criteria, richer contextual information, and approaches beyond zero-shot prompting.

V. Discussion: From Noise to Misalignment

Our empirical observations reveal limitations beyond isolated data quality issues. A substantial portion of training instances encode intents misaligned with the goal of producing actionable, change-grounded feedback. Rather than reflecting low-quality data, these instances expose a structural mismatch between the intended task and its instantiation in commonly used datasets and change-local inputs.

Beyond noisy data. At first glance, ambiguous or non-actionable comments may appear to be simple noise that could be addressed through dataset cleaning [6]. However, our categories capture systematic patterns that correspond to legitimate artifacts of real-world code review, including clarification, coordination, and conversational interaction. Treating these instances as noise obscures a deeper issue: the learning objective induced by current datasets conflates actionable review feedback with broader review discourse. Our findings suggest that current review comment datasets capture interactions serving different purposes in human code review, although only a subset naturally aligns with generating actionable review comments from localized code changes.

Task misalignment through training data. By mixing actionable feedback with comments that serve other purposes, datasets implicitly redefine what models are expected to learn. As a result, models are rewarded for producing fluent, review-like text even when a training instance does not support actionable feedback grounded in the provided input. This

TABLE II
RESULTS ACROSS DATASETS AND CONFIGURATIONS.

Prompt	GPT	Input	Datasets															
			Liu et al. Dataset [8]								Our Dataset							
			Valid (172)				Noisy (98)				Valid (199)				Noisy (184)			
			Prec	Rec	F1	#	Prec	Rec	F1	#	Prec	Rec	F1	#	Prec	Rec	F1	#
Random			0.62	0.46	0.53	128	0.35	0.50	0.41	142	0.55	0.55	0.55	200	0.51	0.51	0.51	183
SP_D	3.5	UP_{NL}	0.85	0.44	0.58	89	0.47	0.87	0.61	181	0.76	0.39	0.52	102	0.57	0.87	0.69	281
SP_D	3.5	$UP_{NL+Diff}$	0.74	0.61	0.67	141	0.48	0.63	0.55	129	0.58	0.59	0.59	200	0.55	0.55	0.55	183
SP_D	4o-mini	UP_{NL}	0.94	0.17	0.29	32	0.40	0.98	0.57	238	0.87	0.20	0.32	45	0.53	0.97	0.68	338
SP_D	4o-mini	$UP_{NL+Diff}$	0.94	0.09	0.17	17	0.38	0.99	0.55	253	0.76	0.13	0.22	34	0.50	0.96	0.66	349
SP_{DC}	3.5	UP_{NL}	0.82	0.63	0.71	133	0.54	0.76	0.63	137	0.71	0.60	0.65	167	0.63	0.74	0.68	216
SP_{DC}	3.5	$UP_{NL+Diff}$	0.72	0.57	0.64	136	0.45	0.61	0.52	134	0.60	0.55	0.58	183	0.56	0.60	0.58	200
SP_{DC}	4o-mini	UP_{NL}	0.92	0.26	0.41	49	0.43	0.96	0.59	221	0.85	0.27	0.41	62	0.55	0.95	0.69	321
SP_{DC}	4o-mini	$UP_{NL+Diff}$	1.00	0.09	0.17	16	0.39	1.00	0.56	254	0.79	0.12	0.20	29	0.50	0.97	0.66	354
SP_{DCT}	3.5	UP_{NL}	0.79	0.61	0.69	133	0.51	0.71	0.60	137	0.65	0.61	0.63	186	0.60	0.65	0.62	197
SP_{DCT}	3.5	$UP_{NL+Diff}$	0.74	0.66	0.70	155	0.50	0.58	0.54	115	0.56	0.59	0.57	210	0.53	0.49	0.51	173
SP_{DCT}	4o-mini	UP_{NL}	0.87	0.40	0.55	79	0.46	0.90	0.61	191	0.81	0.41	0.55	101	0.59	0.90	0.71	282
SP_{DCT}	4o-mini	$UP_{NL+Diff}$	0.87	0.15	0.26	30	0.39	0.96	0.56	240	0.77	0.14	0.23	35	0.51	0.96	0.66	348

Note: SP_D and SP_{DC} denote the prompts from Liu et al. [8], while SP_{DCT} extends the original prompt with taxonomy-derived category definitions. UP_{NL} uses only the review comment, whereas $UP_{NL+Diff}$ also includes the code diff. Green cells indicate F1 scores above the random baseline.

helps explain why generated comments can be plausible yet unhelpful: improvements in fluency or stylistic similarity do not necessarily translate into actionable review behavior.

Exploring automatic data filtering. As an exploratory analysis, we investigated whether noisy training instances could be automatically filtered using a large language model. The results suggest that identifying valid training instances is not trivial and requires explicit validity criteria and sensitivity to review intent and missing context. Future work could explore more sophisticated prompting techniques or richer LLM-based configurations, as the straightforward approach we tested leaves considerable room for improvement.

Structural limits of change-local inputs. Context-dependent instances represent a substantial fraction of the misaligned training pairs in our taxonomy. In these cases, the review comment cannot be inferred from change-local inputs alone, as essential information lies outside the diff, such as references to other changes, code outside the snippet, or prior review comments. For these instances, misclassification reflects missing context rather than a failure of the model. Addressing this limitation would require input representations that extend beyond strictly change-local inputs. These context-dependent instances may also provide a useful benchmark for evaluating approaches that retrieve additional context before generating feedback, including emerging agentic review systems.

VI. RETHINKING DATA AND INPUTS FOR REVIEW COMMENT GENERATION

The observations derived from our dataset analysis suggest that advancing review comment generation requires rethinking how the task is defined through data and input representations. Rather than assuming that improvements in model capacity will compensate for misaligned or incomplete data, we argue for a data- and input-centric perspective in which the learning objective is made explicit and better aligned with real review practices.

Validity as a first-class property. Current datasets implicitly treat all diff-comment pairs as equally valid training instances, an assumption our taxonomy shows does not hold in practice. Future datasets should explicitly define what constitutes a valid instance, distinguishing actionable review feedback from other interactions such as discussion, coordination, or clarification. Making instance validity explicit would help ensure models are trained on signals aligned with the intended task, rather than reproducing heterogeneous review behavior.

Context-aware input representations. The prevalence of context-dependent comments highlights the limitations of diff-only inputs. To support actionable review behavior, input representations should incorporate richer contextual signals, such as review-thread history, references to non-local code, project-specific conventions, or signals from continuous integration pipelines. Importantly, context should not be treated as an optional enhancement, but as a necessary component for a substantial class of review interactions.

Explicit handling of missing context. Even with richer inputs, there will remain situations in which the available information is insufficient to generate meaningful review feedback. Rather than forcing models to produce a comment in all cases, future approaches should explicitly model the possibility of insufficient context. Enabling models to abstain, defer, or request additional information may be preferable to generating fluent but weakly grounded comments, and better reflects realistic review behavior.

ACKNOWLEDGMENT

Leonardo Centellas-Claros is supported by the Chilean National Agency for Research and Development (ANID)/Scholarship Program/DOCTORADO NACIONAL/ 2024-21240734 and National Center for Artificial Intelligence (CENIA), Grant/Award Number: BASAL, ANID, FB210017.

REFERENCES

[1] R. Tufano, L. Pascarella, M. Tufano, D. Poshyvanyk, and G. Bavota, "Towards automating code review activities," in *Proceedings of the 43rd International Conference on Software Engineering*, ser. ICSE '21. IEEE Press, 2021, p. 163–174. [Online]. Available: https://doi.org/10.1109/ICSE43902.2021.00027

[2] P. Thongtanunam, C. Pornprasit, and C. Tantithamthavorn, "Autotransform: automated code transformation to support modern code review process," in *Proceedings of the 44th International Conference on Software Engineering*, ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 237–248. [Online]. Available: https://doi.org/10.1145/3510003.3510067

[3] Z. Li, S. Lu, D. Guo, N. Duan, S. Jannu, G. Jenks, D. Majumder, J. Green, A. Svyatkovskiy, S. Fu, and N. Sundaresan, "Automating code review activities by large-scale pre-training," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 1035–1047. [Online]. Available: https://doi.org/10.1145/3540250.3549081

[4] Y. Hong, C. Tantithamthavorn, P. Thongtanunam, and A. Aleti, "Commentfinder: a simpler, faster, more accurate code review comments recommendation," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 507–519. [Online]. Available: https://doi.org/10.1145/3540250.3549119

[5] L. Li, L. Yang, H. Jiang, J. Yan, T. Luo, Z. Hua, G. Liang, and C. Zuo, "Auger: automatically generating review comments with pre-training models," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 1009–1021. [Online]. Available: https://doi.org/10.1145/3540250.3549099

[6] R. Tufano, O. Dabić, A. Mastropaolo, M. Ciniselli, and G. Bavota, "Code review automation: Strengths and weaknesses of the state of the art," *IEEE Transactions on Software Engineering*, vol. 50, no. 2, pp. 338–353, 2024.

[7] H. Y. Lin, P. Thongtanunam, C. Treude, and W. Charoenwet, "Improving automated code reviews: Learning from experience," in *Proceedings of the 21st International Conference on Mining Software Repositories*, ser. MSR '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 278–283. [Online]. Available: https://doi.org/10.1145/3643991.3644910

[8] C. Liu, H. Y. Lin, and P. Thongtanunam, " Too Noisy To Learn: Enhancing Data Quality for Code Review Comment Generation ," in *2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*. Los Alamitos, CA, USA: IEEE Computer Society, Apr. 2025, pp. 236–248. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/MSR66628.2025.00043

[9] N. Davila and I. Nunes, "A systematic literature review and taxonomy of modern code review," *Journal of Systems and Software*, vol. 177, p. 110951, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121221000480

[10] R. Tufano and G. Bavota, "Automating code review: A systematic literature review," 2025. [Online]. Available: https://arxiv.org/abs/2503.09510

[11] Y. Yu, G. Rong, H. Shen, H. Zhang, D. Shao, M. Wang, Z. Wei, Y. Xu, and J. Wang, "Fine-tuning large language models to improve accuracy and comprehensibility of automated code review," *ACM Trans. Softw. Eng. Methodol.*, vol. 34, no. 1, Dec. 2024. [Online]. Available: https://doi.org/10.1145/3695993

[12] J. Lu, L. Yu, X. Li, L. Yang, and C. Zuo, "Llama-reviewer: Advancing code review automation with large language models through parameter-efficient fine-tuning," 2023. [Online]. Available: https://arxiv.org/abs/2308.11148

[13] Q. Guo, J. Cao, X. Xie, S. Liu, X. Li, B. Chen, and X. Peng, "Exploring the potential of chatgpt in automated code refinement: An empirical study," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3597503.3623306

[14] J. Zhang, X. Hu, S. Gao, X. Xia, D. Lo, and S. Li, "Less is more: On the importance of data quality for unit test generation," *Proc. ACM Softw. Eng.*, vol. 2, no. FSE, Jun. 2025. [Online]. Available: https://doi.org/10.1145/3715778

[15] "Replication package." [Online]. Available: https://doi.org/10.6084/m9.figshare.32296611