

# Large Language Models for Software Engineering: A Reproducibility Crisis

Mohammed Latif Siddiq<sup>1</sup>, Arvin Islam-Gomes<sup>1</sup>, Natalie Sekerak<sup>1</sup>,  
Joanna C. S. Santos<sup>1\*</sup>

<sup>1</sup>\*Computer Science and Engineering, University of Notre Dame, Holy Cross Drive, Notre Dame, 46556, IN, USA.

\*Corresponding author(s). E-mail(s): [joannacss@nd.edu](mailto:joannacss@nd.edu);

Contributing authors: [msiddiq3@nd.edu](mailto:msiddiq3@nd.edu); [aislamg2@nd.edu](mailto:aislamg2@nd.edu);  
[mnsekerak@nd.edu](mailto:mnsekerak@nd.edu);

## Abstract

Reproducibility is a cornerstone of scientific progress, yet its state in large language model (LLM)-based software engineering (SE) research remains poorly understood. This paper presents the first large-scale, empirical study of reproducibility practices in LLM-for-SE research. We systematically mined and analyzed **640 papers** published between 2020 and 2025 across premier software engineering, machine learning, and natural language processing venues, extracting structured metadata from publications, repositories, and documentation. Guided by four research questions, we examine (i) the prevalence of reproducibility smells, (ii) how reproducibility has evolved over time, (iii) whether artifact evaluation badges reliably reflect reproducibility quality, and (iv) how publication venues influence transparency practices. Using a taxonomy of seven smell categories: ***Code and Execution, Data, Documentation, Environment and Tooling, Versioning, Model, and Access and Legal***, we manually annotated all papers and associated artifacts. Our analysis reveals persistent gaps in artifact availability, environment specification, versioning rigor, and documentation clarity, despite modest improvements in recent years and increased adoption of artifact evaluation processes at top SE venues. Notably, we find that badges often signal artifact presence but do not consistently guarantee execution fidelity or long-term reproducibility. Motivated by these findings, we provide actionable recommendations to mitigate reproducibility smells and introduce a

**Reproducibility Maturity Model (RMM)** to move beyond binary artifact certification toward multi-dimensional, progressive evaluation of reproducibility rigor.

**Keywords:** Large Language Models (LLMs), Software Reproducibility, Reproducibility Smell, Artifact Evaluation, Reproducibility Maturity Model

## 1 Introduction

**Reproducibility** is a core part of scientific inquiry, enabling independent verification and extension of research findings [1, 2]. In software engineering (SE) research, reproducibility enables independent verification of results, speeds up cumulative knowledge building, and prompts trust in empirical evidence [3]. However, despite its importance, reproducibility remains a persistent challenge in modern machine learning (ML) and natural language processing (NLP) research [4, 5]. This challenge is further exacerbated in the context of large language models (LLMs), where rapidly evolving frameworks, model dependencies, and legal or access restrictions can limit experimental replicability [6, 7].

The integration of LLMs into SE workflows, ranging from code generation and bug repair to test synthesis and program comprehension, has grown rapidly since the emergence of foundation models such as *GPT-4*, *Codex*, and *Code Llama* [8–10]. This widespread use has produced hundreds of publications exploring LLM-driven software engineering techniques [11]. However, unlike traditional empirical SE research, LLM-based studies often rely on closed-source models, volatile APIs, and non-standardized prompts or hyperparameters, making experimental reproduction particularly difficult [12]. As a result, both the SE and ML communities have raised concerns about a looming “*reproducibility crisis*” [4, 5, 13].

Reproducibility has long been a topic of concern in empirical SE research, well before the rise of LLMs. Multiple studies have highlighted gaps between claimed and actual reproducibility of SE experiments, stemming from inadequate artifact sharing, inconsistent environment specifications, lack of standardized benchmarks, and insufficient methodological documentation [12, 14]. Unlike controlled laboratory sciences, SE research often involves complex, multi-layered software ecosystems where dependency mismatches, library deprecations, and undocumented preprocessing steps can lead to results that are impossible to replicate, even when the research’s source code is available. This has led to community initiatives such as artifact evaluation tracks, reproducibility badges, and standardized packaging practices aimed at improving transparency and reusability of research artifacts [15]. However, as argued by Sallou *et al.* [12], true reproducibility for work with LLMs requires not just artifact availability but also explicit methodological guidance, and proper information disclosure.

Despite this growing concern, there is still no comprehensive, large-scale study systematically examining how reproducibility is practiced and enforced in LLM-for-SE

research. Prior work has either focused on reproducibility in general ML research [2, 4] or on specific subareas of software engineering [15], leaving a gap at the intersection of these two domains. Moreover, while artifact evaluation initiatives in premier SE venues have made notable progress [16, 17], their impact on LLM-based research remains poorly understood.

This manuscript addresses this gap by presenting the *first systematic, data-driven investigation of reproducibility practices in LLM-for-SE research*. We thoroughly analyzed **640 papers** published between 2020 and 2025 across top SE, ML, and NLP venues. Through a mixed-method approach, we designed and applied a *reproducibility smell taxonomy* encompassing seven key categories: *Code and Execution*, *Data*, *Documentation*, *Environment and Tooling*, *Versioning*, *Model*, and *Access and Legal*. We then manually annotated each paper to assess the prevalence of reproducibility smells and compared patterns across time periods. Additionally, we evaluated the extent to which artifact evaluation badges serve as reliable indicators of reproducibility quality. We additionally conducted a manual analysis of how reproducibility practices evolved across publication venues over the years.

## 1.1 Contributions

The contributions of our work are:

- We provide the first comprehensive analysis of reproducibility practices in LLM-for-SE research, revealing persistent reproducibility smells, such as *access and legal* smells and *code and execution* smells.
- We examine how reproducibility has evolved over time and the co-occurrence of the reproducibility smells.
- We investigate whether artifact evaluation badges reliably indicate reproducibility quality.
- We analyze how publication venues and artifact evaluation processes influence transparency and reproducibility.
- We provide actionable recommendations to mitigate reproducibility smells and introduce a ***Reproducibility Maturity Model (RMM)*** to move beyond binary artifact certification.

Our scripts and data associated with this work are publicly available in our replication package [18].

## 1.2 Manuscript Organization

This manuscript is organized as follows. Section 2 provides background on Large Language Models (LLMs) and their role in Software Engineering research. Section 3 outlines our research questions and the methodology used to address them. Section 4 presents the results for each research question. Section 6 offers an in-depth discussion of temporal trends in reproducibility smells and examines how model openness contributes to these challenges. Section 5 delivers actionable recommendations for authors, reviewers, and publication venues to mitigate reproducibility issues, and introduces the Reproducibility Maturity Model (RMM). Section 7 discusses threats

to validity. Section 8 reviews related work and compares our contributions within the broader literature. Finally, Section 9 concludes the paper.

**Note:** Throughout this manuscript, we discuss widespread reproducibility problems observed across the 640 LLM-for-SE papers analyzed in our study. These issues reflect systemic patterns in the broader LLM-for-SE literature. However, we intentionally do not name or single out any specific papers from our corpus when illustrating these problems. Similar to prior critical reviews [19, 20], our goal is to merely highlight structural challenges and foster community-wide improvements, not to criticize individual researchers. However, we included the detailed data in our reproducibility package.

## 2 Background

In this section, we provide a conceptual and technical overview of Large Language Models (LLMs) and their role in Software Engineering (SE) research to understand the challenges and opportunities surrounding their reproducibility.

### 2.1 Large Language Models (LLMs)

**Large Language Models (LLMs)** are deep neural networks trained on vast text corpora using transformer-based architectures [21]. These models, such as GPT-4 [8], LLaMA [22], and DeepSeek-V2 [23], demonstrate strong capabilities in understanding and generating human-like language, enabling a wide range of tasks through few-shot or zero-shot prompting [24]. LLMs have rapidly become foundational technologies in natural language processing (NLP), powering advances in text generation, summarization, question answering, and even program synthesis.

Several open-source and commercial LLMs have been further trained and fine-tuned for *code-related* tasks, such as code generation. These LLMs, also known as **code LLMs**, are large language models specialized for understanding and generating source code, enabling a wide range of programming and software engineering applications. Notably, OpenAI’s Codex [9] was trained on GitHub code to power tools like GitHub Copilot, enabling intelligent code completion and generation. Meta’s Code Llama series [25] and Salesforce’s CodeGen [26] have followed similar trajectories, offering LLMs optimized for programming languages.

### 2.2 LLMs in Software Engineering Research

The integration of LLMs into Software Engineering (SE) research has given rise to a new field of inquiry (“**LLM-for-SE**”) where models are applied to automate, support, or improve SE tasks [11]. Applications span the entire software lifecycle: requirements engineering (*e.g.*, intent extraction [27]), software design (*e.g.*, UML generation [28]), implementation (*e.g.*, code synthesis [29]), testing (*e.g.*, test case generation [30] and summarization [31]), maintenance (*e.g.*, bug fixing and documentation [32]), and even project management (*e.g.*, issue triage [33]).

Recent studies have shown promising results using LLMs for tasks such as code completion [34–36], code search [37], code summarization [38], and code generation [9]. However, these capabilities often depend heavily on prompt engineering, data quality, and fine-tuning strategies, all of which can vary significantly across implementations, making it challenging to reliably reproduce research results.

## 3 Methodology

In this section, we outline the methodology we followed to answer our research questions. Figure 1 provides an overview of the methodology of our work.

### 3.1 Research Questions

We answer four research questions in this work:

**RQ1** *What types of reproducibility smells are most prevalent in LLM-for-SE research?*

This RQ investigates the common patterns of poor reproducibility practices (henceforth, “**reproducibility smells**”) present in published LLM-for-SE studies. Identifying prevalent reproducibility smells provides a foundation for understanding systemic weaknesses in current reproducibility practices.

**RQ2** *How has the reproducibility of LLM-for-SE research evolved over time?*

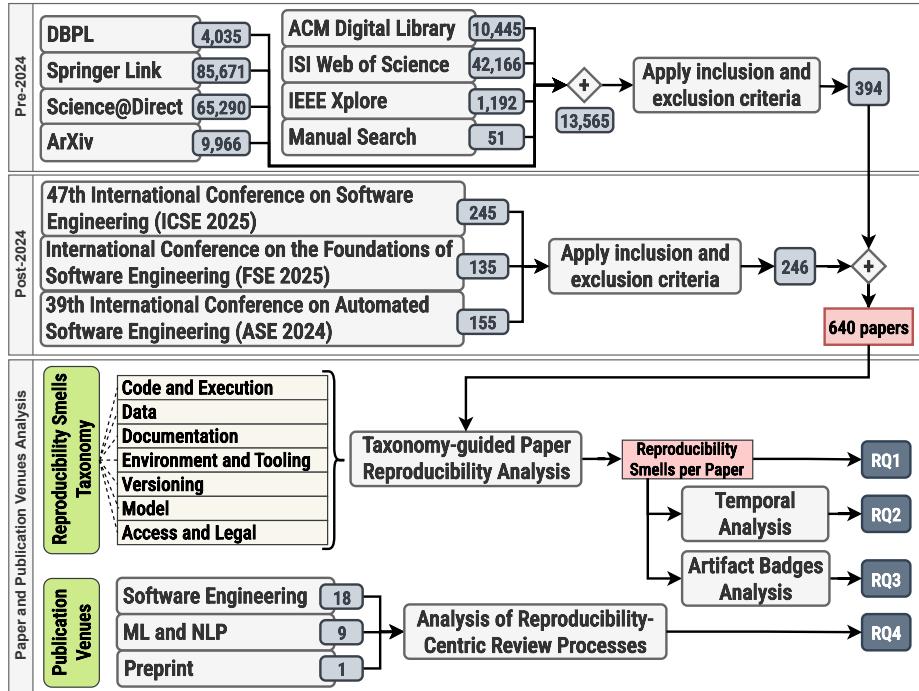
This research question examines temporal trends in reproducibility, focusing on whether more recent studies adopt improved practices compared to earlier work. We examine changes in documentation quality, artifact availability, and adherence to reproducibility checklists to determine whether the field is moving toward more open and verifiable research.

**RQ3** *Do artifact evaluation badges reliably reflect the reproducibility of LLM-for-SE research?*

This RQ investigates whether papers awarded artifact evaluation badges exhibit stronger reproducibility characteristics in practice. Specifically, we compare the prevalence of reproducibility smells in papers with badges, examining whether badges serve as a reliable proxy for replication readiness.

**RQ4** *How do software engineering research venues ensure that published studies are reproducible?*

This RQ investigates the role of conferences and journals in enforcing reproducibility standards. It examines the presence and effectiveness of artifact evaluation tracks, reproducibility badges, mandatory dataset or code submission requirements, review guidelines, and the evolution of review guidelines over time.



**Fig. 1** Methodology Overview.

### 3.2 Paper Selection

To answer our RQs, we need to collect papers on large language models in software engineering (*i.e.*, works on LLM-for-SE). To establish a foundation, we first reviewed the set of papers analyzed in the recent systematic literature review by Hou *et al.* [11]. Since that review covered studies published between January 2017 and January 2024, we further expanded the corpus to include more recent work, ensuring that emerging trends in LLM-for-SE research were represented. The resulting dataset comprises a comprehensive collection of relevant papers spanning both **pre-2024** (*i.e.*, those included in Hou *et al.* [11]) and **post-2024** publications. Table 2 presents the year-wise distribution of the LLM-for-SE papers collected for this study.

#### 3.2.1 Pre 2024 Papers

To understand reproducibility practices in published LLM-for-SE, we examined **394** papers<sup>1</sup> from a systematic literature review on the use of LLMs in software engineering research [11]. This literature review collected papers published from **January 2017** to **January 2024** from a combination of manual searches in top software engineering venues (*e.g.*, ICSE, ESEC/FSE, ASE, ISSTA, TOSEM, and TSE) and automated

<sup>1</sup>Originally, there were 395 papers and one of the papers was retracted.

searches across major digital libraries, including IEEE Xplore, ACM Digital Library, ScienceDirect, Web of Science, Springer, arXiv, and DBLP. Moreover, this prior literature review also conducted backward and forward snowballing to identify additional relevant studies through reference lists and citations, ensuring both breadth and depth of coverage. Although their initial search covered papers published between 2017 and 2024, the final set of included papers spanned 2020 to 2024, after applying the inclusion and exclusion criteria summarized in Table 1.

**Table 1** Summary of the Inclusion and Exclusion Criteria Used by Hou *et al.* [11]

Inclusion Criteria	Exclusion Criteria
<b>I1</b> Written between January 2017 – January 2024	<b>E1</b> Short articles (less than 8 pages)
<b>I2</b> Focused on a Software Engineering (SE) task	<b>E2</b> Duplicated articles
<b>I3</b> An LLM is used for an SE task	<b>E3</b> Position papers, tool demo papers, keynotes, editorials, reviews, tutorials, panel discussions, grey publications, workshop papers, or doctoral symposium papers.
<b>I4</b> Full text is available	<b>E4</b> Studies not in English
	<b>E5</b> Article uses LLMs but does not describe the employed techniques.

### 3.2.2 Post 2024 Papers

For studies published in or after 2024, we focused on the papers published in the research track of the top three (A\*) software engineering conferences ranked by the CORE Conference Ranking<sup>2</sup>: International Conference on Software Engineering (**ICSE**), International Conference on Automated Software Engineering (**ASE**), and International Conference on the Foundations of Software Engineering (**FSE**). This selection guarantees that all included papers meet the public accessibility requirement (inclusion criterion **I4**) and do not meet any of the exclusion criteria listed in Table 1. To verify the remaining inclusion criteria (*i.e.*, **I2** and **I3**), we manually reviewed the full texts of the technical-track papers accepted at **ICSE 2025**, **FSE 2025**, and **ASE 2024**. Through this process, we identified **102**, **67**, and **77** papers, respectively, that employed LLMs for software engineering tasks.

**Table 2** Year-wise distribution of papers.

Year	Number of Papers	List of Papers
<b>2020</b>	7	[39–45]
<b>2021</b>	13	[46–58]
<b>2022</b>	56	[59–114]
<b>2023</b>	272	[115–386]
<b>2024</b>	123	[387–509]
<b>2025</b>	169	[510–678]

<sup>2</sup><https://portal.core.edu.au/conf-ranks/>

### 3.3 Reproducibility Smell Analysis

After merging the papers collected pre-2024 and post-2024, we obtained a total of **640** papers. To systematically evaluate the reproducibility of LLM-for-SE works, we conducted a structured analysis of each of these 640 selected papers, focusing on the key dimensions that affect experimental replication and verification. As part of this process, we developed *a reproducibility smell taxonomy* to categorize and quantify recurring issues that hinder reproducibility in LLM-for-SE research. Our taxonomy draws conceptual inspiration from Hassan *et.al.*'s systematic literature review of reproducibility debt in scientific software [679], which identified seven broad categories of issues (data-centric, code-centric, documentation-centric, process-centric, human-centric, legal, and version-centric) and provided a high-level organization of reproducibility-related technical debt across computational sciences.

However, Hassan *et.al.*'s framework was designed to characterize reproducibility challenges in *general* scientific software, predating modern LLM-based workflows. Moreover, it does not cover LLM-specific sources of reproducibility flaws, such as undocumented prompts, stochastic inference behavior, model drift, closed-source models and APIs, missing temperature/seed settings, or the unavailability of training data and model weights [12]. To adapt their conceptual structure to the LLM-for-SE context, we replaced their *Human-Centric Issues* category with a new category (**Model Smells**) because human factors could not be reliably inferred from our corpus and because LLM-driven research introduces fundamentally new reproducibility risks tied to model behavior rather than researcher actions. Furthermore, we refined the definitions of remaining categories (*e.g.*, documentation-, data-, and tool-centric smells) to reflect LLM-specific artifacts such as prompt templates, fine-tuning configurations, API parameters, and model sourcing.

Overall, our taxonomy extends Hassan *et al.* 's foundation by translating a domain-agnostic reproducibility debt categories into an LLM-aware, SE-specific taxonomy that captures emerging reproducibility pitfalls unique to LLM-for-SE research. In doing so, we provide the first systematic, domain-tailored characterization of reproducibility smells for the rapidly evolving LLM-for-SE landscape. In the next subsection, we elaborate on this taxonomy and the categories it contains.

#### 3.3.1 A Taxonomy of Reproducibility Smells

Our taxonomy has seven categories, each of which is described below, along with the corresponding detection approach used to identify the smell in papers.

##### *Code and Execution Smells.*

This type of smell occurs when the paper's experimental artifacts lack a runnable, complete, or accessible implementation needed to replicate results [679]. This includes missing code repositories, private or inactive links, incomplete scripts (*e.g.*, missing entry points), or a lack of runnable pipelines. These smells reduce *construct validity* by preventing external verification of claims and are exacerbated when models or APIs are closed-source or evolve over time [680].

- *Identifying code and execution smells:* For this category, we first check whether they provided any link to the source code. If no link was present, we flagged the paper as containing this smell. If the source code is available, then we checked whether the repository is private, closed-source, or inactive, which also contributed to this category. If the reproducibility links were accessible, we specifically looked for training or inference scripts, shell commands, or Makefiles that could allow an independent researcher to replicate the experiments described in the paper. We flagged the paper as having this smell when the replication package was non-executable (*e.g.*, missing entry points), incomplete (*e.g.*, missing core scripts, configuration files, or dependency specifications), or entirely absent. This information was collected by manually opening each repository and inspecting directory structures and execution instructions.

#### ***Data Smells.***

A ***data smell*** arises when the paper poorly documents the datasets it used or when these used datasets are unavailable, proprietary, or lack stable references such as DOIs, dataset URLs, or version numbers [679]. Missing preprocessing details, vague dataset descriptions, or reliance on evaluation data with potential leakage from pretraining exacerbate this issue [680].

- *Identifying data smells:* To detect data-related reproducibility issues, we examined whether the papers provided clear references to the datasets used, including persistent identifiers (*e.g.*, DOIs, dataset URLs), version numbers, or repository links. We also checked if the dataset was publicly accessible or proprietary. When preprocessing steps were mentioned, we recorded whether they were fully documented or only described vaguely. Missing references, inaccessible datasets, or a lack of preprocessing instructions were labeled as data smells. We systematically extracted this information from both the paper’s main text and the linked artifacts, prioritizing README files, data preparation scripts, and supplementary materials.

#### ***Documentation Smells.***

A ***documentation smell*** is present when methodological descriptions or replication instructions are insufficient, vague, or fragmented. This includes missing or incomplete README files, lack of usage examples, or absence of clear guidance for replicating experiments [679].

- *Identifying documentation smells:* These smells were identified by analyzing both the paper’s methodology section and its associated documentation (*e.g.*, README files, Wiki pages, or tutorials). We examined whether details about the file structure, execution steps, or dependencies were missing. We labeled a documentation smell if methodological descriptions were vague, incomplete, or scattered across multiple places without clear guidance for replication.

### *Environment and Tooling Smells.*

An *environment and tooling smell* occurs when system dependencies, software environments, or hardware requirements are missing, incomplete, or ambiguous [679]. This includes lack of `requirements.txt`, `environment.yml`, or `Dockerfile`, unspecified CUDA or Python versions, or absence of containerization information.

- *Identifying environment and tooling smells:* To detect environment and tooling smells, we investigated whether the papers or repositories specified their software and hardware dependencies. We checked for the presence of `requirements.txt`, `environment.yml`, or `Dockerfile` files, and whether specific CUDA, Python, or library versions *etc.* were documented. Absence of dependency files, incomplete environment configurations, or missing information about containerization were considered environment smells. Additionally, we noted whether hardware configurations (e.g., GPU type) were mentioned, as a lack of this information can make study replication difficult.

### *Versioning Smells.*

A *versioning smell* is detected when datasets, models, or software dependencies are not associated with explicit version identifiers [679]. This includes unpinned package versions, untagged commits, use of floating versions like “latest,” or absent version metadata for models. Lack of versioning severely impacts reproducibility, especially in rapidly evolving LLM environments where model behavior can drift over time [680].

- *Identifying versioning smells:* We classified a reproducibility smell under versioning if datasets, models, or software dependencies lacked clear version information. This included untagged dataset references, unpinned package versions, or ambiguous model identifiers (e.g., latest rather than a specific commit or release). We checked the repository, dependency files, and paper text to determine whether versioning was handled rigorously. When version control was missing or inconsistent, we marked it as a versioning smell. It is worth mentioning that when a dataset was used without specifying its version, we tagged the paper with both a data smell (due to insufficient referencing or documentation) and a versioning smell (due to the absence of explicit versioning or stable identifiers).

### *Model Smells.*

As our focus is on using LLM in software engineering, we introduced the category of model smells to capture issues unique to LLM research. A *model smell* captures a lack of access to model weights or checkpoints, missing prompt templates, missing details on inference parameters (e.g., temperature, top-k, top-p), or missing fine-tuning hyperparameters.

- *Identifying model smells:* We checked whether model weights or checkpoints were available, whether prompt templates were provided, and whether sufficient information about inference parameters (e.g., temperature, top-k, top-p) and fine-tuning parameters (e.g., learning rate, number of epochs, optimizer) was

included. If models were inaccessible, not released, or only vaguely described, the paper was labeled as having a model smell. This information was extracted mainly from the paper. If they were not mentioned, we checked the repository.

#### *Access and Legal Smells.*

An *access and legal smell* occurs when experimental artifacts (code, datasets, models) are restricted by licenses, behind approval walls, or have ambiguous usage terms [679]. This includes proprietary APIs, restrictive model licenses, institutional access barriers, or a lack of clear licensing statements.

- *Identifying access and legal smells* We investigated legal and accessibility barriers by checking whether the artifacts (datasets, models, code) were subject to restrictive licenses or hosted behind access requests. We recorded whether licensing information was explicitly stated, whether usage was restricted to specific institutions, or whether artifacts required approval for access. Papers that relied on proprietary tools, private models, or unclear licensing terms were labeled as having access and legal smells. We systematically extracted this data from license files, model cards, and dataset hosting pages.

### 3.4 Answering RQ1 & RQ2: Taxonomy-Guided Paper Analysis

Each paper was manually coded according to the taxonomy, and smell frequencies were aggregated to identify the most prevalent reproducibility barriers in current LLM-for-SE research to answer **RQ1**. We further analyzed the temporal evolution of these smells on a yearly basis to answer **RQ2**, enabling us to observe how specific reproducibility barriers have changed over time. This manual analysis was performed by the first and last authors, both of whom have extensive experience serving on artifact evaluation committees at premier software engineering venues.

### 3.5 Answering RQ3: Artifact Badge and Reproducibility Assessment

To answer **RQ3**, we systematically examined whether artifact evaluation badges are indicative of stronger reproducibility practices in LLM-for-SE research. Many premier software engineering venues (*e.g.*, ICSE, FSE, ASE) award artifact badges in accordance with ACM or IEEE standards. For example, the ACM’s reproducibility badge system [681] has the following badges: *i.e.*, Artifact Available, Artifact Evaluated–Functional, Artifact Evaluated–Reusable, and Results Reproduced. IEEE follows a nearly similar badging system [682, 683], including badges such as Open Research Objects (ORO) (Available), Research Objects Reviewed (ROR), Results Reproduced (ROR-R), and Results Replicated (RER). These badges are intended to signal robustness and accessibility of research artifacts.

For each of the 640 papers in our study, we manually inspected official conference proceedings websites, artifact evaluation pages, and paper web pages to record whether an artifact evaluation badge was awarded. When applicable, we also recorded the specific badge type (*e.g.*, Available, Functional, Reusable).

We mapped badge information to reproducibility smells to compare their distributions across all categories (*e.g.*, code and execution, environment and tooling, model, versioning, data, documentation, access, and legal). We inspected whether badge-awarded papers exhibit fewer or less severe smells, and whether particular smell categories are still present despite artifact recognition.

### 3.6 Answering RQ4: Publication Venues

We examine reproducibility practices in the call for papers to understand how venues are enforcing (or not) replication artifacts in published LLM-for-SE works.

#### 3.6.1 Selecting Publication Venues for Reproducibility Analysis

To ensure broad and representative coverage of the LLM-for-SE research landscape, we selected a diverse set of publication venues spanning top-tier software engineering, machine learning, and natural language processing communities to answer RQ4. The venue selection was guided by three key principles: (i) inclusion of **high-impact venues** based on the CORE ranking system, (ii) representation of both **conference and journal publications**, and (iii) inclusion of **preprint platform** to capture rapidly evolving research.

We categorized venues into three main groups: SE venues, ML & NLP Venues, and pre-print platforms.

##### *Software Engineering Venues*

Core SE publication venues were prioritized, including both A and A\*-ranked conferences and journals. These include:

- **Journals:** ACM Transactions on Software Engineering and Methodology (**TOSEM**), IEEE Transactions on Software Engineering (**TSE**), Journal of Systems and Software (**JSS**), Empirical Software Engineering (**EMSE**), Information and Software Technology (**IST**).
- **Conferences:** International Conference on Software Engineering (**ICSE**), International Conference on Automated Software Engineering (**ASE**), International Conference on the Foundations of Software Engineering (**FSE**), IEEE International Conference on Software Maintenance and Evolution (**ICSME**), International Symposium on Software Testing and Analysis (**ISSTA**), International Conference on Software Analysis, Evolution and Reengineering (**SANER**), International Conference on Program Comprehension (**ICPC**), Mining Software Repositories Conference (**MSR**), International Symposium on Empirical Software Engineering and Measurement (**ESEM**), International Conference on Evaluation and Assessment in Software Engineering (**EASE**), IEEE International Conference on Software Testing, Verification and Validation (**ICST**), IEEE International Symposium on Software Reliability Engineering (**ISSRE**), IEEE International Conference on Software Architecture (**ICSA**).

These venues represent the main outlets for software engineering research and are frequently used for reporting empirical studies, tool evaluations, and engineering-focused LLM applications.

#### ***ML and NLP Venues***

To capture research at the intersection of LLMs and SE, we also included leading ML and NLP conferences:

- **Conferences:** Conference on Neural Information Processing Systems (**NeurIPS**), International Conference on Machine Learning (**ICML**), Conference on Empirical Methods in Natural Language Processing (**EMNLP**), AAAI Conference on Artificial Intelligence (**AAAI**), International Conference on Learning Representations (**ICLR**), Conference on Learning Theory (**COLT**), International Joint Conference on Artificial Intelligence (**IJCAI**), International Conference on Principles of Knowledge Representation and Reasoning (**KR**), Annual Meeting of the Association for Computational Linguistics (**ACL**).

These venues often publish foundational work on LLM architectures, training methods, prompt engineering, and model evaluation that can be applied to SE tasks.

#### ***Pre-print Platforms***

Given the rapid pace of development in LLM research, many influential works appear first as pre-prints before formal peer review. We therefore included **arXiv** as an additional source of recent and emerging studies.

#### **3.6.2 Reproducibility Practices Analysis**

To answer RQ4, we systematically analyzed the aforementioned publication venues' calls for papers and websites from 2020 to 2025 (*i.e.*, the same year range as our 640 studied papers) to understand how institutional policies and review processes shape reproducibility practices in LLM-for-SE research. For each venue, we checked whether it provides a formal artifact evaluation track or equivalent mechanism, supports or requires reproducibility badges (*e.g.*, *artifact available*, *artifact evaluated*, *results reproduced*), mandates code or dataset availability as part of the publication process, and includes explicit review guidelines or checklists addressing reproducibility and transparency.

## **4 Results**

This section presents the findings of our study, highlighting the prevalence, characteristics, and temporal evolution of reproducibility smells in LLM-for-SE research. Furthermore, we discuss artifact evaluation badges' reliability in terms of reproducibility smells as well as the reproducibility practices for publication venues.

## 4.1 RQ1: Prevalence of Reproducibility Smells

### 4.1.1 Reproducibility Smells Distribution

Table 3 summarizes the overall distribution of reproducibility smells across all the 640 analyzed LLM-for-SE papers. From our analysis, we found that *access and legal smells* (35.9%) and *code and execution smells* (35.5%) are the most frequent issues, indicating that studies either lack runnable artifacts or impose legal and licensing barriers that complicate replication. *Versioning smells* are also widespread (32.2%), reflecting common problems such as unpinned dependency versions, missing model or dataset identifiers, and ambiguous versioning practices.

**Table 3** Reproducibility Smells Distribution.

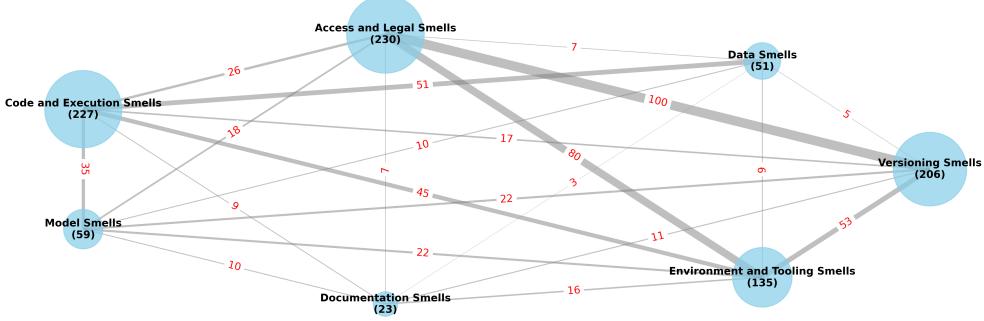
Reproducibility Smell	# Papers	% Papers
Access and Legal	230	35.9%
Code and Execution	227	35.5%
Versioning	206	32.2%
Environment and Tooling	135	21.1%
Model	59	9.2%
Data	51	8.0%
Documentation	23	3.6%
<i>No Smells</i>	85	13.3%

*Environment and tooling smells* appear in 21.1% of papers, pointing to gaps in specifying computational dependencies, environment configuration files, or hardware details. In contrast, *data* (8%) and *model smells* (9.2%) occur less frequently, but they still signal gaps in dataset accessibility, model checkpoint availability, or missing prompt and inference parameter details. *Documentation smells* are the least pervasive category (3.6%), suggesting that while methodological descriptions are often present, other factors, such as execution or access barriers, remain more dominant. Finally, only 13.3% of papers did not have any reproducibility smells, indicating that reproducibility challenges are widespread in LLM-for-SE literature. These findings show that while LLM-for-SE research often provides some documentation, there remain open challenges in reproducing these works related to executable artifacts, access constraints, and transparency in versioning.

### 4.1.2 Reproducibility Smells Co-occurrence

Our co-occurrence analysis of the reproducibility smells reveals a structural coupling among several smell categories, as shown in Figure 2, indicating that reproducibility debts are often interconnected. For example, among 230 papers with *Access and Legal* smells, 43.5% (100 papers) also exhibit *Versioning* smells. This reflects a common pattern in LLM-for-SE work, where restricted or proprietary APIs coincide with undocumented or unstable software versions, making it more difficult to reproduce experiments.

A similar co-occurrence is observed between the *Access and Legal* and *Environment and Tooling* smells. 34.8% (80 of 230) of papers with access-related issues also



**Fig. 2** Co-occurrence Network of Reproducibility Smells.

lack sufficient environment specifications, suggesting that institutional barriers (*e.g.*, licensing, rate limits, or unavailable datasets) propagate into incomplete environment descriptions in practice.

**Code and Execution** smells also demonstrate several meaningful co-occurrences. For example, **22.5%** (51 of 227) of papers with code issues also suffer from **data smells**, while **19.8%** (45 of 227) co-occur with environment and tooling issues. These results show an interplay between runnable pipelines, data availability, and environment specification—three ingredients essential for end-to-end reproducibility.

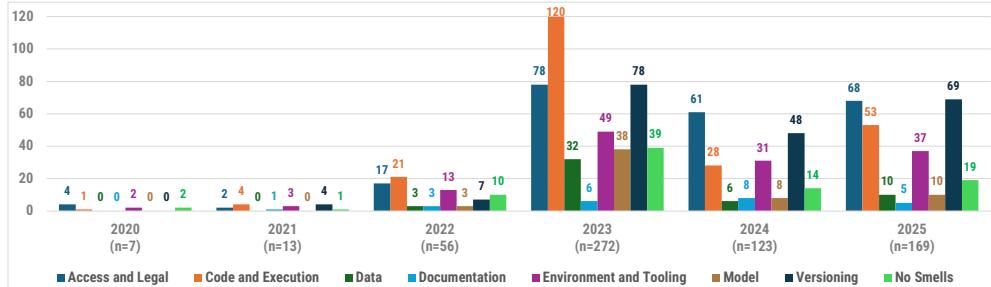
To quantify the strength of these relationships, we computed pairwise lift (*i.e.*, measures how much more frequently two smells co-occur than would be expected if they were independent) and  $\phi$ -coefficients (*i.e.*, captures the correlation strength between two binary smell variables) for all smell combinations. The strongest structural associations were observed between **Documentation** and **Model** smells (lift  $\approx 4.72$ ,  $\phi \approx 0.23$ ) and between **Documentation** and **Environment and Tooling** smells (lift  $\approx 3.30$ ,  $\phi \approx 0.23$ ), followed by **Code and Execution** and **Data** smells (lift  $\approx 2.82$ ,  $\phi \approx 0.40$ ), indicating **pronounced, non-random co-dependencies**. We also find elevated associations among infrastructure-related smells, specifically, **Access and Legal**, **Environment and Tooling**, and **Versioning** (lift up to  $\approx 1.65$ ,  $\phi$  up to  $\approx 0.25$ ). This reinforces that reproducibility debt is organized into coherent clusters rather than isolated issues.

These patterns demonstrate that reproducibility debt in LLM-for-SE research is *structural*. If we make improvements to any single artifact dimension (*e.g.*, documentation), we may not fully resolve the problem unless we address other issues like access constraints, code executability, dependency management, and model availability.

#### 4.2 RQ2: Temporal Trends in Reproducibility Over Time

Figure 3 presents the temporal distribution of reproducibility smells across publication years from 2020 to 2025. Across the analyzed period, we found that **code and execution smells** remain the most prevalent category overall. In 2020, only 1 out

of 7 papers that year had this smell (14.3%), rising to 120 papers in 2023 (44.1% of papers in that year), before declining to 28 papers (22.8%) in 2024 and 53 papers (31.4%) in 2025. This downward trend in recent years suggests incremental improvements in the availability and clarity of experimental pipelines. A chi-square test of independence confirms that the distribution of reproducibility smells varies significantly across years ( $\chi^2 = 75.34, p = 0.000089 < 0.001$ ), providing statistical support for the temporal trends observed in Figure 3.



**Fig. 3** Temporal distribution of reproducibility smells across years. The total number of papers with reproducibility smells may exceed the total number of papers, as multiple smells can co-occur within a single paper.

**Access and legal smells** show an increase over time. Starting at 4 papers (57.1%) in 2020, their prevalence rises to 61 papers (49.6%) in 2024 and 68 papers (40.2%) in 2025, making this category one of the frequent reproducibility barriers in recent research. **Versioning smells** follow a similar pattern, increasing from 0 papers in 2020 to 48 (39.0%) in 2024 and 69 (40.8%) in 2025. These two categories show the sharpest rise over the six-year span we analyzed, reflecting growing challenges related to model version pinning, licensing, and access restrictions.

**Environment and tooling smells** appear consistently across the years. There were 2 papers (28.6%) in 2020, 49 (18.0%) in 2023, and 37 (21.9%) in 2025, indicating persistent, unresolved issues in dependency specification and computational reproducibility. Moreover, **data smells** and **documentation smells** are relatively infrequent. Data smells peak at 32 papers (11.8%) in 2023 but remain below 6% in other years. Documentation smells never exceed 8 papers in any year, with percentages ranging from 0.0% to 7.7%, suggesting that these issues, while present, are less widespread compared to other categories.

**Model smells** emerge more prominently in and after 2022. There were 38 papers (14.0%) in 2023 and 10 papers (5.9%) in 2025 with these smells, highlighting growing dependencies on closed-source checkpoints or unreleased model components. Finally, the proportion of papers with **no detected reproducibility smells** declines slightly over time: from 28.6% in 2020 to 11.2% in 2025. This indicates that as the LLM-for-SE research landscape expands, reproducibility issues remain pervasive, particularly those related to legal, versioning, and environment concerns.

### 4.3 RQ3: Artifact Evaluation Badges Reliability

Table 4 and Table 5 summarize the evolution of artifact evaluation badges and reproducibility smells from **2020** to **2025**. We found 63 papers (9.8%) with badges. It is worth mentioning that papers from 2024 and 2025 in our dataset primarily originate from the top three software engineering venues (ICSE, FSE, ASE), whereas earlier years include a broader mix of journals, conferences, and pre-prints, and the number of badged papers in our corpus is small (especially before 2023). For this reason, our badge analysis is intentionally associational rather than causal. We stratify the results descriptively by year and badge combination (Tables 4 and 5), and we interpret badges as *proxies* for stronger artifact scrutiny rather than as direct causal interventions.

It is also worth noting that only two papers used the IEEE badging system [104, 213] and had only the Available badge. Two other papers used adapted ACM badging systems with similar badge categories [42, 353]. For this reason, Table 4 and Table 5 list ACM-style badging categories *i.e.*, Artifact Available, Artifact Evaluated—Functional, and Artifact Evaluated—Reusable. Artifact Available indicates that the authors have uploaded their artifacts (*e.g.*, code, data, models) in a stable, publicly accessible repository, while Artifact Evaluated—Functional certifies that reviewers were able to execute the artifact and verify its basic correctness. Artifact Evaluated—Reusable reflects a higher standard, recognizing artifacts that are not only functional but also well-documented, modular, and easy for others to adapt or extend.

**Table 4** Temporal distribution of badges.

Category	2020 (n=7)		2021 (n=13)		2022 (n=56)		2023 (n=272)		2024 (n=123)		2025 (n=169)	
	#	%	#	%	#	%	#	%	#	%	#	%
At least one badge	1	14.3	1	7.7	4	7.1	11	4.0	12	9.8	34	20.1
Available	1	14.3	1	7.7	3	5.4	11	4.0	12	9.8	34	20.1
Functional	0	0.0	1	7.7	2	3.6	5	1.8	2	1.6	22	13.0
Reusable	1	14.3	0	0.0	0	0.0	1	0.4	7	5.7	17	10.1

Badge adoption remained sparse until recently: fewer than **8%** of papers prior to 2023 earned badges, increasing to **9.8%** in 2024 and **20.1%** in 2025. This uptick aligns with stronger enforcement and visibility of artifact evaluation processes at flagship SE venues. The Artifact Available badge was consistently the most common, while Functional and Reusable badges were less frequent until 2025, where both incidences rose to **13.0%** and **10.1%**, respectively. This upward trend reflects growing expectations not only for availability but also for usability and reusability of research artifacts.

Despite these positive developments, reproducibility smells persist even among badged papers. Across years, the most common smells among badge recipients were *versioning issues* (reaching **50%** in 2025) and *code and execution smells*. Notably, some earlier badged papers (with very small sample sizes) exhibited high smell incidence rates, highlighting that early badges may have focused primarily on artifact presence

**Table 5** Mapping between badge combinations and reproducibility smells in badged papers from 2020 – 2025. Each row represents papers sharing the same year and badge combination (**A** means available, **F** means functional, **R** means Reusable). Each cell shows the count of papers exhibiting each smell, followed by the percentage relative to the group size.

Year	Badge	#	Legal	Code	Data	Doc	Env	Model	Version
2020	A+R	1	0 (0.0)	1 (100.0)	0 (0.0)	0 (0.0)	1 (100.0)	0 (0.0)	0 (0.0)
2021	A+F	1	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	1 (100.0)
2022	A	2	1 (50.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	1 (50.0)
2022	A+F	1	0 (0.0)	1 (100.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
2022	F	1	0 (0.0)	1 (100.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
2023	A	6	0 (0.0)	2 (33.3)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	1 (16.7)
2023	A+F	4	1 (25.0)	1 (25.0)	1 (25.0)	0 (0.0)	2 (50.0)	1 (25.0)	1 (25.0)
2023	A+F+R	1	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	1 (100.0)
2024	A	3	0 (0.0)	0 (0.0)	0 (0.0)	1 (33.3)	1 (33.3)	0 (0.0)	2 (66.7)
2024	A+F	2	1 (50.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
2024	A+R	7	2 (28.6)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	3 (42.9)
2025	A	12	2 (16.7)	1 (8.3)	0 (0.0)	0 (0.0)	2 (16.7)	1 (8.3)	6 (50.0)
2025	A+F	5	2 (40.0)	1 (20.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	3 (60.0)
2025	A+F+R	17	2 (11.8)	5 (29.4)	1 (5.9)	1 (5.9)	2 (11.8)	2 (11.8)	8 (47.1)

rather than quality or usability. Even in 2024–2025, when SE venues set higher reproducibility standards, badged papers frequently lacked explicit dependency versions or stable execution pipelines. For example, **41.7%** of badged papers from 2024 still exhibited *versioning smells*.

#### 4.4 RQ4: Reproducibility Practices Across Publication Venues

To address RQ4, we analyzed reproducibility practices across 28 publication venues spanning the software engineering (SE), machine learning (ML), and natural language processing (NLP) communities. Our analysis focused on four key dimensions: **(i)** presence of artifact evaluation tracks, **(ii)** support for reproducibility badges, **(iii)** policies requiring or encouraging sharing of code and datasets, and **(iv)** use of formal review checklists addressing reproducibility.

- **SE Journals.** Among the SE journals, TOSEM offers the most structured support, providing optional artifact submission via Replicated Computational Results (RCR) reports and endorsing ACM badges. However, this was introduced in 2022 and implemented from 2023 onwards [684]. EMSE actively promotes reproducibility through its Open Science initiative, requiring data availability statements and encouraging artifact sharing with informal review by its Open Science Board from 2019 [685]. JSS also promotes reproducibility through the JSS Open Science Initiative [686]. For papers where authors indicate participation in the initiative, the manuscript is automatically forwarded to the JSS Open Science Board after acceptance. Following a successful review of availability and usability, the publisher will add a statement acknowledging validation of the Open Science material. IST encourages artifact sharing and requires data availability statements, though they lack formal evaluation or badging [687]. TSE, as an IEEE journal, follows IEEE’s general policy of encouraging (but not requiring) artifact availability and maintains this policy from 2020 to 2025.

**Table 6** Reproducibility support across publication venues based on the latest available data on November 2025. Full black circle = present/required, half black circle = partial/encouraged, empty black circle = absent

Venue	Artifact Evaluation Track	Badges	Code/Data	Checklist
<i>SE Journals</i>				
TOSEM	●	●	●	○
EMSE	○	○	●●	○
JSS	○	○	●●●	○
IST	○	○	●●	○
TSE	○	○	●○	○
<i>SE Conferences (ACM)</i>				
ICSE	●	●	●	○
FSE	●●	●●	●●	○
ASE	●●	●●	●●	○
ISSTA	●●	●●	●●	○
<i>SE Conferences (IEEE)</i>				
ICSME	●●	●●	●●	○
ISSRE	●●	●●	●●	○
ICSA	●●	●●	●●	○
<i>Other SE Conferences</i>				
SANER	○	○	●	○
MSR	○	○	●●	○
ESEM	○	○	●●	○
ICPC	○	○	●●	○
ICST	○	○	●○	○
EASE	○	○	●○	○
<i>ML Conferences</i>				
NeurIPS	○	○	●	●
ICML	○	○	●●	●●
ICLR	○	○	●●	●●
AAAI	○	○	●●	●●
IJCAI	○	○	●●	●●
COLT	○	○	○	○
KR	○	○	●○	○
<i>NLP Conferences</i>				
ACL	○	○	●●	●●
EMNLP	○	○	●●	●●
<i>Preprints</i>				
arXiv	○	○	●○	○

- **SE Conferences.** ACM-affiliated SE conferences such as ICSE, FSE, ASE, and ISSTA feature dedicated artifact evaluation tracks throughout 2020 to 2025. These venues apply ACM’s reproducibility badge system, introduced in 2016 and updated to version 1.1 in 2020 [681]. This badging system includes the following badge categories: *Artifact Available*, *Artifact Evaluated – Reusable*, and *Results Reproduced* to qualifying papers. IEEE venues, including ICSME, ISSRE, and ICSA, have adopted similar practices, offering artifact tracks and reproducibility badges based on the IEEE-endorsed *Research Object Reviewed (ROR)/Open Research Object (ORO)* framework. However, ICSME introduced artifact badging in pilot mode in 2020, awarding ACM badges [688], and adopted the current IEEE-endorsed badging scheme starting in 2021. Similarly, ISSRE initiated its artifact badging process in 2023. ICSA launched its first Artifact Evaluation track in 2021 and expanded

it in 2022 [689]. The ICSA process is built on the National Information Standards Organization (NISO) reproducibility badge definitions (supported by IEEE) and mirrors ACM’s criteria [682].

Other SE conferences, such as SANER, MSR, ESEM, and ICPC, do not employ formal artifact evaluation but enforce strong open science policies that require or strongly encourage code/data sharing and mandate data availability statements at the time of paper submission. Moreover, SANER hosts specialized tracks introduced in 2023 for reproducibility and replication studies, providing structured incentives without formal badging [690]. ICST and EASE promote transparency and artifact sharing informally, without specific evaluation mechanisms.

- **ML and NLP Conferences.** Leading ML conferences such as NeurIPS, ICML, and ICLR enforce reproducibility through detailed checklists that authors must complete upon submission. These checklists cover code and data availability, computational details, and reporting standards. Although these venues do not issue badges, reproducibility is integrated into the review criteria, and post-publication reproducibility challenges further incentivize transparency. NeurIPS introduced this reproducibility program in 2019, requiring authors to complete a detailed checklist with each submission. In 2021, NeurIPS expanded it into a broader “responsible research” checklist (covering reproducibility, transparency, ethics, and societal impact) [691]. In 2020, ICML introduced a similar reproducibility checklist for authors (modeled on NeurIPS’s) [692]. ICLR began urging authors to include a “Reproducibility Statement” in their submissions around 2021 and to submit code; by 2025, ICLR’s author guide explicitly stated that it “strongly encourages” a reproducibility section in each paper [693].

AAAI 2021 was the first AAAI conference to require a reproducibility checklist with each submission [694]. This checklist, similar to NeurIPS’s, became mandatory for AAAI submissions. IJCAI has followed a similar suit, requiring checklists and reproducibility assessments by reviewers by explicitly incorporating reproducibility scores into the review form.

From 2020 to 2025, COLT maintained a strict focus on theoretical reproducibility through complete proofs, without introducing artifact sharing policies, reproducibility checklists, or badges [695]. Experimental components were optional and not subject to formal reproducibility expectations throughout this period. Between 2020 and 2025, KR gradually expanded support for reproducibility by allowing supplementary materials (*e.g.*, proofs, code, or data), and by 2025 explicitly encouraged artifact sharing, particularly in applied tracks. However, it did not adopt artifact evaluation or badging [696, 697].

In the NLP domain, EMNLP and ACL enforce reproducibility through the Responsible NLP Checklist, mandatory limitations sections, and expectations for artifact availability statements in camera-ready submissions from 2020 [694]. While these venues do not operate formal artifact tracks, community norms strongly encourage public release of code and datasets.

- **Preprints.** arXiv, as a preprint platform, does not enforce any reproducibility requirements. Artifact availability is left to the discretion of authors, with no peer review or badging mechanisms. There is a separate tab, introduced in 2020 [698],

in each of the Machine Learning arXiv papers to provide the link to data and code using third-party services like Hugging Face, Papers with Code *etc.*

## 5 Recommendations

The findings from RQ1–RQ4 show that LLM-for-SE research could benefit from stronger, more systematic reproducibility practices. In this section, we present actionable recommendations for each reproducibility smell category, offering concrete steps that authors, reviewers, and venues could adopt to prevent persistent barriers from becoming entrenched. We also introduce a **Reproducibility Maturity Model (RMM)** to move beyond binary notions of artifact quality and provide a structured, multi-dimensional framework for assessing the durability, rigor, and long-term verifiability of research outputs.

### 5.1 Smells Prevalence and Actionable Recommendations

Our analysis across 640 LLM-for-SE papers reveals that reproducibility challenges remain a problem. The most common problem areas are **Access and Legal** and **Code and Execution** smells (each affecting roughly one-third of the studies), followed closely by **Versioning** issues (32.2%). **Environment and Tooling** smells (21.1%) is less frequent but notable. Other smells like **Data**, **Model**, and **Documentation** smells occur below 10%. In the next sections, we discuss each smell category with recommendations.

#### 5.1.1 Access and Legal Smells

**Access and Legal Smells** is one of most pervasive issues, affecting **35.9%** of studies overall, with a sharp rise from **28.7%** in 2023 to nearly half of all papers (**49.6%**) in 2024. This rise parallels the widespread adoption of closed APIs and gated model endpoints. Common signs of this smell are replication packages with missing or non-permissive licenses and reliance on commercial APIs that cannot be redistributed. The absence of clear licensing and access documentation creates legal uncertainty for reuse and hinders downstream replication.

##### *Recommendations:*

- **Mandatory Licensing:** Every artifact must specify an explicit license (*e.g.*, MIT, Apache 2.0, CC-BY-4.0). Artifact Evaluation (AE) forms should be rejected if licensing is absent or ambiguous.
- **Persistent Hosting:** Encourage repositories such as Zenodo or Figshare for long-term archival access to code, data, and model weights beyond GitHub links.
- **Transparent Proprietary Usage:** For closed APIs, papers should clearly document API version, model identifier, and date of access, and provide request/response examples to enable approximate reproduction.

### 5.1.2 Code and Execution Smells

**Code and Execution Smells** affect **35.5%** of papers but show a modest improvement post-2023, dropping from 44.1% in 2023 to 31.4% in 2025. However, this issue still persists because inactive or private repositories, missing entry points, and incomplete scripts remain frequent. Out of 477 replication links examined, 19 were broken or private, and 64 contained incomplete or non-runnable artifacts. This confirms that the mere existence of shared code does not guarantee reproducibility.

*Recommendations:*

- **Runnable Entry Points:** AE guidelines should require a single executable entry (*e.g.*, `run.sh`, `Makefile` target, or Jupyter notebook) verified during artifact review.
- **Reproducible Pipelines:** Encourage standardized workflows (Make, Snake-make, or bash scripts) that automate preprocessing, training, and evaluation.
- **Code Completeness:** All scripts tied to reported results must be present and documented within the replication package.

### 5.1.3 Versioning Smells

**Versioning Smells** were found in **32.2%** of papers, rising from 12.5% in 2022 to over 40% in 2024–2025. The most common issues include missing dependency specifications, unpinned library versions, and vague references such as “latest model release”.

*Recommendations:*

- **Pin All Dependencies:** Require exact version locking using environmental files (*e.g.*, `requirements.txt` or `environment.yml`) with explicit equality operators.
- **Model Version Disclosure:** Papers must specify model commit hashes, release tags, or dataset identifiers for all LLMs used.
- **Data Versioning:** Track derived datasets via Data Version Control (DVC) or Digital Object Identifier (DOI)-based repositories to ensure stable data lineage.

### 5.1.4 Environment and Tooling Smells

Roughly **21.1%** of studies have **Environment and Tooling Smells**, a proportion that has remained stable across years (around 18–25%). Missing or incompatible environment settings frequently cause replication failures, and only 33 papers include Docker support. Furthermore, **572 papers** lacked any mention of hardware setup, obscuring critical GPU and driver dependencies.

*Recommendations:*

- **Containerization:** Require a runnable `Dockerfile` or similar container specification to guarantee reproducible environments.
- **Hardware Disclosure:** Explicitly report GPU/CPU models, CUDA versions, and OS details, especially for fine-tuning experiments.

- **Executable Environment Files:** Ensure environment specifications are directly runnable and reflect all transitive dependencies.

### 5.1.5 Model Smells

*Model Smells* occur in **9.2%** of papers and are tied to ambiguous model configurations. Missing prompt templates, undocumented inference parameters, and absent training details cause this smell. For instance, 199 papers failed to specify their exact prompt templates, and 88 lacked documentation of their inference parameters.

*Recommendations:*

- **Release Fine-tuned Weights:** When feasible, publish fine-tuned checkpoints or scripts to regenerate them.
- **Parameter Transparency:** Report all inference and training parameters, including sampling temperature, top- $k$ , and learning rates.
- **Prompt Publication:** Archive all prompt templates used in evaluation to ensure consistent task reproduction.

### 5.1.6 Data Smells

Detected in **7.9%** of studies, *Data Smells* typically occur from missing preprocessing documentation or inaccessible dataset versions. Even when datasets were cited in the paper, the artifact often lacked persistent links or transformation scripts.

*Recommendations:*

- **Stable Data References:** Use DOIs or permanent URLs for datasets and their processed variants.
- **Include Preprocessing Scripts:** Provide runnable scripts that transform raw data into the experiment-ready format.
- **Dataset Cards:** Summarize data provenance, filtering, and versioning in concise dataset documentation.

### 5.1.7 Documentation Smells

Although *Documentation Smells* is the least frequent (**3.6%**), it can critically block replication when other artifacts are ambiguous. Even when the code is complete, missing README files or disjointed instructions often make it practically unusable.

*Recommendations:*

- **Unified README:** Include a single, step-by-step README.md that maps each experiment to paper results.
- **Minimal Documentation Principle:** Centralize all essential instructions in one or two top-level documents.
- **Self-contained Guidance:** Documentation should enable replication without author contact or guesswork.

## 5.2 Toward a Reproducibility Maturity Model (RMM)

As discussed in RQ3 and RQ4, ACM and IEEE provide artifact badges *i.e.*, “available”, “functional”, or “reusable”, but they do not offer guidance on the *durability*, *rigor*, or *long-term verifiability* of research outputs. Our empirical analysis shows that these systems primarily evaluate *momentary functionality*. That means they check whether artifacts worked during review, but not whether they are engineered for long-term reproducibility. Over 40% of “functional” artifacts in our corpus from 2024 – 2025 fail within months due to drifting dependencies, unpinned versions, incomplete environments, or unclear licensing. Moreover, existing badges do not assess reproducibility holistically across the multiple smell dimensions we identified. A paper can earn a badge even when substantial barriers remain, as shown in RQ3.

To address this gap, we propose a **Reproducibility Maturity Model (RMM)**, a multi-dimensional framework that evaluates the *durability*, *rigor*, and *legal openness* of artifacts. It is designed to help authors, reviewers, and conference organizers systematically evaluate and enforce the reproducibility of LLM-for-SE research.

The RMM framework evaluates reproducibility along five orthogonal axes identified through our taxonomy and empirical observations:

- A1. Accessibility.** Availability, persistence, and openness of datasets, code, and models.
- A2. Environment Specification.** Clarity and completeness of environment, dependency, and system-level configuration.
- A3. Versioning Rigor.** Explicit, pinned versions for datasets, models, pipelines, libraries, and tools.
- A4. Execution Fidelity.** Presence of runnable pipelines, automated scripts, or containers enabling end-to-end replication.
- A5. Legal Openness.** Licensing clarity for data, models, and code to legally permit reuse and redistribution.

### 5.2.1 Four Maturity Levels (RMM-Tiers)

Based on these axes, we distill four maturity tiers that reflect the empirical distribution of our corpus and provide concrete review criteria.

#### *RMM-0: Minimal Reproducibility (Low Maturity)*

Artifacts are missing, inaccessible, or only partially available. Environment specifications are vague or absent, versions are unpinned, reproduction depends heavily on implicit assumptions. Multiple reproducibility smells can co-occur.

*Reviewer checklist.*

- Are any artifacts inaccessible or missing?
- Are the environmental details insufficient to recreate the setup?
- Are versions unspecified or floating?
- Does legal ambiguity prevent reuse?

### **RMM-1: Operational Reproducibility (Intermediate Maturity)**

Artifacts are functional at publication time but vulnerable to dependency drift. Instructions exist, but require manual reconstruction. Containers or scripts may exist, but are incomplete or rely on deprecated APIs.

*Reviewer checklist.*

- Are installation steps documented but potentially brittle?
- Are versions given but not pinned at all hierarchy levels?
- Are pipelines runnable but not fully automated?
- Are licenses present but incomplete or unclear?

### **RMM-2: Durable Reproducibility (High Maturity)**

Artifacts are fully runnable, containerized, versioned, and legally open. All five axes are satisfied. Outputs can be reproduced end-to-end with no manual inference. These exemplars (13.3% of our corpus) demonstrate reproducibility engineered for long-term integrity.

*Reviewer checklist.*

- Is a containerized or automated pipeline provided?
- Are all dependencies pinned, including OS, libraries, and models?
- Are datasets and models persistently hosted with stable identifiers?
- Are all components licensed for reuse?

### **RMM-3: Independently Verified Reproducibility (Very High Maturity)**

This tier reflects a level of reproducibility that extends beyond the authors' own artifacts. An external reviewer or independent research group has successfully re-executed the full experimental pipeline and reproduced the key findings using the provided artifacts without modification. Because this tier requires substantial effort outside the standard publication workflow, we treat it as an *optional*, evaluation-driven extension analogous to the ACM *Results Reproduced* badge rather than a requirement for authors at submission time.

*Reviewer checklist.*

- Can the results be reproduced *exactly* using only the provided artifacts and instructions?
- Was the reproduction performed independently, without author intervention?
- Are all outputs (tables, figures, metrics) consistent with those reported in the paper?
- Does the reproduction process complete successfully in a clean environment?

#### **5.2.2 Axes by Tier Criteria**

Table 7 summarizes how each axis is instantiated at different maturity levels. **RMM-0** and **RMM-1** roughly distinguish between missing or brittle artifacts and those that are merely “momentarily functional”, while **RMM-2** and **RMM-3** represent durable, independently verifiable reproducibility.

**Table 7** Summary of RMM axes and indicative criteria per maturity level. RMM-3 assumes all RMM-2 criteria plus independent re-execution by a third party.

Axis	RMM-0: Minimal	RMM-1: Operational	RMM-2: Durable
Accessibility	No or fragmented artifacts; links missing, private, or ephemeral.	Artifacts hosted but with fragile links (personal cloud, ad-hoc URLs); partial coverage of code/data/models.	Artifacts versioned and persistently hosted (e.g., DOI, archival repository) with clear mapping to experiments.
Environment Specification	Environment details largely absent; no explicit hardware/software requirements.	Basic installation steps documented, but incomplete dependency lists and implicit assumptions about the platform.	Complete, machine-readable environment specification (e.g., lockfiles, container specs) covering OS, libraries, and hardware.
Versioning Rigor	Floating versions or unspecified model-/dataset snapshots; reproducibility depends on latest defaults.	Some versions documented (e.g., major library versions), but not consistently pinned across the stack.	Pinned versions at all hierarchy levels (datasets, models, frameworks, CUDA, etc.), with change logs or manifests.
Execution Fidelity	No runnable scripts; only high-level prose or pseudo-code.	Pipelines runnable with manual effort (e.g., several shell commands, manual downloads, ad-hoc scripts).	End-to-end, automated pipelines or containers that re-create results from scratch with minimal manual steps.
Legal Openness	Licenses absent, incompatible, or ambiguous; reuse is legally risky.	Some components licensed, but coverage incomplete or terms unclear for key artifacts (e.g., models, datasets).	Explicit, compatible licenses for all artifacts required to reproduce results, enabling legal reuse and redistribution.

By treating reproducibility as a set of measurable, enforceable criteria, the RMM provides a structured path toward sustainable, verifiable LLM-for-SE research and extends current badging systems with a more holistic maturity spectrum.

## 6 Discussion

In this section, we discuss the temporal trends and policy implications of the reproducibility smells. We also discuss how using open and proprietary models is contributing to the issue.

### 6.1 Temporal Trends and Policy Implications

The temporal data (Figure 3) reveal a mixed trajectory. Some reproducibility issues, especially code executability, have improved since 2023, suggesting growing community awareness. However, the occurrence of **Access and Legal** and **Versioning** smells increased, driven by the usage of proprietary APIs and the absence of fixed model identifiers. In parallel, **Environment and Tooling** issues persist at a steady baseline, showing that environment reproducibility remains an issue.

These patterns indicate a structural rather than individual challenge. As LLM-for-SE research increasingly depends on commercial ecosystems, even diligent authors face obstacles in releasing reusable artifacts. This shift from open academic tooling to closed proprietary services calls for policy interventions at the venue level. Artifact-evaluation and reproducibility-badging programs should adapt to this evolving landscape by requiring disclosure of API version, access date, and decoding parameters for any proprietary model, inclusion of at least one open-source baseline to preserve replicability even if commercial endpoints disappear. Moreover, archival of prompt templates, inference settings, and sample interaction logs for LLM-based experiments should also be made a requirement alongside with explicit licensing statements for all code, data, and model artifacts.

## 6.2 Open vs. Proprietary Model Regimes

We observed that the reproducibility gap mirrors the divide between **open** and **proprietary** model regimes. Open models (*e.g.*, LLaMA [699], Mistral [700]) enable complete replication pipelines but require robust environment control and version pinning. Proprietary models (*e.g.*, GPT [701], Claude [702]) offer convenience but introduce opacity and temporal drift, as API outputs can change silently. Our work confirms that access/legal and versioning smells are overwhelmingly concentrated in the latter period (post-2023).

To mitigate this, a ***dual-track reproducibility framework*** can be adopted, in which proprietary-model studies should include at least one open-source baseline. In contrast, open-model studies must ensure full environmental encapsulation and version transparency. Journals and conferences should adapt review rubrics to recognize these distinct reproducibility periods rather than enforcing a uniform standard.

## 7 Threats to Validity

### *Internal validity*

During the manual data collection and annotation process for reproducibility smells, some information may have been overlooked due to the large scale of the study involving 640 papers and their corresponding repositories. To minimize potential omissions and ensure consistency, two authors with prior experience in artifact evaluation cross-checked the annotations. This expertise and verification process help reduce the likelihood of missing relevant reproducibility information.

### *External validity*

Although the dataset analyzed in this study is large and representative of top-tier venues, it may not fully capture reproducibility practices across the broader research landscape. In particular, our focus on highly visible academic venues such as ICSE, FSE, ASE, and major ML/NLP conferences might bias the results toward research that already follows more formalized artifact and documentation practices. Reproducibility behaviors in non-top-tier venues, workshops, industry-focused conferences, or grey literature (*e.g.*, preprints, technical reports) may differ significantly, potentially

involving fewer formal artifact evaluation processes, less consistent documentation, or different types of legal and access barriers. However, our findings from top-tier venues provide a strong and conservative baseline. If reproducibility issues are prevalent even in venues that typically enforce stronger standards and review processes, similar or more severe issues are likely to exist in less formal publication settings.

## 8 Related Work

### 8.1 Large Language Models for Software Engineering (LLM-for-SE)

The rapid evolution of large language models (LLMs) has catalyzed a new wave of research in software engineering (SE), giving rise to the emerging field commonly referred to as LLM-for-SE. Unlike earlier program analysis or deep learning for SE approaches, LLM-for-SE leverages general-purpose foundation models such as GPT-4, Codex, and Code Llama to automate or assist in a broad spectrum of SE tasks [680, 703]. These tasks span the entire software development lifecycle, including code generation, test synthesis, debugging, program comprehension, and software documentation.

Early empirical studies demonstrated that pretrained language models could learn code semantics and structure sufficiently to outperform traditional baselines on code summarization [694], code search [37], and completion tasks [34–36]. This has encouraged the integration of LLM-powered tools into mainstream development workflows (*e.g.*, GitHub Copilot). However, these models differ significantly from earlier task-specific ML approaches: they are often hosted behind APIs, trained on massive and largely opaque datasets, and fine-tuned continuously, which introduces significant reproducibility challenges [680]. Our work focuses on the current situation of the reproducibility of software engineering tasks using large language models.

### 8.2 Reproducibility in Software Engineering and Machine Learning

Reproducibility has emerged as a critical concern across science and engineering in recent years. Surveys indicate that over 70% of researchers have failed to reproduce another group’s findings, highlighting a cross-disciplinary “reproducibility crisis” [704]. In software engineering (SE), reproducibility is foundational for building trust in empirical results and enabling independent verification [705]. Krishnamurthi and Vitek famously argued that repeatability is the “real software crisis,” urging the community to treat reproducibility as a core value in SE research. Early studies of SE experiments uncovered substantial barriers to repetition. For example, González-Barahona and Robles identified difficulties in reusing data retrieval tools, prompting calls for better reproducibility practices [706], and their follow-up work a decade later confirmed persistent gaps [707].

Top SE venues have responded by institutionalizing artifact sharing and evaluation processes to improve reproducibility. Conferences such as ICSE, FSE, ASE, and

ISSTA now regularly include Artifact Evaluation Committees and award reproducibility badges for papers that provide accessible code and data [708–710]. This cultural shift has led to a steady rise in artifact availability over the past decade. Qualitative studies have explored the impact of these initiatives. Hermann *et al.* examined community expectations for artifacts [708], while Timperley *et al.* analyzed artifact evolution trends [709]. Still, domain-specific hurdles remain; for example, replication in NLP-for-Requirements-Engineering required significant re-implementation effort due to under-specified models and data preparation steps [711].

The machine learning (ML) and natural language processing (NLP) communities face similar concerns. Reproducibility issues in ML are often linked to missing code, opaque experimental setups, and insufficient reporting [4, 694]. Even when code is shared, it may lack critical preprocessing instructions or fixed hyperparameters, leading to inconsistent results across environments. Liu *et al.* demonstrated that many SE deep learning results were inflated or unstable across runs, highlighting the need for multiple runs and better reporting [712]. To address such problems, venues like NeurIPS and ICLR introduced reproducibility checklists, increasing the proportion of papers sharing code and methodological details [4], while ACL introduced its Responsible NLP Checklist to improve reporting standards [694].

Hassan *et al.* [679] conducted a systematic literature review on reproducibility debt in scientific software, introducing a seven-dimensional taxonomy encompassing data, code, documentation, processes, human factors, legal constraints, and versioning. Their work provides a high-level conceptual framework to understand reproducibility breakdowns in computational sciences. In contrast, our study targets a rapidly changing and highly specialized domain, LLM-for-SE and develops a reproducibility smell taxonomy grounded in empirical evidence drawn from 640 papers published between 2020 and 2025. While inspired by Hassan *et al.*'s structure, our taxonomy is to capture domain-specific reproducibility pitfalls in LLM workflows, evaluation practices, and software engineering tooling. Furthermore, our analysis is not limited to conceptual mapping but includes a systematic quantification of reproducibility smells, offering actionable insights for both researchers and publication venues in the LLM-for-SE space.

### 8.3 Reproducibility Challenges in LLM-for-SE Research

LLM-for-SE research inherits reproducibility challenges from both domains. A key issue is the reliance on closed-source or continuously evolving models (e.g., GPT-4 or Codex), where researchers cannot control or replicate the exact model state [680]. Experiments relying on such APIs are vulnerable to model updates and access restrictions, limiting repeatability. Further, complex dependencies and environmental configurations introduce additional failure points, especially when model or dataset versioning is incomplete. Prompt design and generation parameters significantly impact results, with small variations yielding divergent outcomes [713].

Addressing these challenges requires both cultural and technical interventions. In SE, artifact evaluation and open science initiatives have been widely adopted [710],

while in ML/NLP, reproducibility checklists and data-sharing incentives have shown measurable impact [4, 694]. Beyond policies, tooling such as containerization (*e.g.*, Docker) and data/model versioning (*e.g.*, DVC) has become central to reproducible experimentation. Emerging guidelines for LLM-based SE studies advocate documenting prompts, model versions, and intermediate artifacts [703, 714] to ensure verifiable and transparent research practices. Building on these insights, our work provides a large-scale empirical assessment of reproducibility in LLM-for-SE papers and proposes actionable guidelines informed by both SE artifact evaluation culture and ML/NLP reproducibility initiatives.

Williams *et al.* [715] provide a high-level assessment of LLM-based SE research at ICSE, focusing on model and benchmark usage, contamination risks, and broad replicability indicators such as whether artifacts or badges are present. In contrast, our study conducts a cross-venue, multi-year analysis and introduces a fine-grained taxonomy of reproducibility smells, quantifying their prevalence, co-occurrence, and evolution.

## 8.4 Critical Reviews of Software Engineering Research Practices

A growing body of meta-research within empirical software engineering has highlighted foundational methodological weaknesses and reporting deficiencies. Stol *et al.* [20] examine 98 SE articles that mention or claim to use the Grounded Theory (GT) method. They find that only 16 of the 98 provide detailed accounts of their research procedures, and many studies show “method-slurring” (mixing GT techniques without coherent adherence). The paper offers guidelines for the conduct and reporting of GT in SE. Baltes *et al.* [19] perform a critical review of sampling in recent high-quality SE empirical studies and report three major findings: (1) random sampling is rare; (2) sophisticated sampling strategies are extremely rare; (3) sampling, representativeness, and randomness are often misunderstood. They argue that SE research suffers from a generalisability crisis. Huang *et al.* [716] review how qualitative research synthesis (QRS) is carried out in SE, identifying several methodological deficiencies (*e.g.*, inconsistent definitions, weak reporting of how primary studies were selected and synthesised). While these works focus on critical reviews of various practices in software engineering research, our work focuses on the reproducibility of LLM-for-SE research.

## 9 Conclusion

Reproducibility remains a critical and under-addressed challenge in large language model (LLM)-based software engineering (SE) research. Through a systematic analysis of **640 papers** across premier SE, ML, and NLP venues, we observed persistent reproducibility smells, particularly in code execution pipelines, versioning rigor, and access and legal constraints. While traditional SE research has benefited from structured artifact evaluation and open-science initiatives, LLM-for-SE introduces domain-specific barriers, including dependence on proprietary APIs, opaque prompt

designs, non-deterministic inference behaviors, and rapidly evolving model and package ecosystems.

Our temporal results reveal modest progress in recent years, driven in part by reproducibility policies at top-tier venues. However, even as artifact evaluation badges become more common, they do not consistently guarantee execution fidelity or long-term replicability. These findings highlight a gap between *artifact availability* and *artifact maturity*. To bridge this gap, we introduce a **Reproducibility Maturity Model (RMM)** that moves beyond binary badge systems toward multi-dimensional assessments encompassing accessibility, environment specification, version control, execution fidelity, and legal openness.

We release our dataset, taxonomy, and analysis framework to support future benchmarking and automated smell detection efforts. By embracing maturity-oriented reproducibility practices, the LLM-for-SE community can ensure that research remains verifiable, buildable, cumulative, and trustworthy.

## References

- [1] Stodden, V.: The scientific method in practice: Reproducibility in the computational sciences. MIT Sloan Research Paper (2010)
- [2] Stodden, V., Guo, P., Ma, Z.: Toward reproducible computational research: an empirical analysis of data and code policy adoption by journals. *PLoS one* **8**(6), 67111 (2013)
- [3] Anda, B.C., Sjøberg, D.I., Mockus, A.: Variability and reproducibility in software engineering: A study of four companies that developed the same system. *IEEE Transactions on Software Engineering* **35**(3), 407–429 (2008)
- [4] Pineau, J., Vincent-Lamarre, P., Sinha, K., Larivière, V., Beygelzimer, A., d’Alché-Buc, F., Fox, E., Larochelle, H.: Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). *Journal of Machine Learning Research* **22**(1), 1–20 (2021)
- [5] Lucic, A., Bleeker, M., Bhargav, S., Forde, J., Sinha, K., Dodge, J., Luccioni, S., Stojnic, R.: Towards reproducible research in natural language processing: A position paper. arXiv preprint arXiv:2302.04858 (2023)
- [6] Gudibande, A., Wallace, E., Snell, C., Geng, X., Liu, H., Abbeel, P., Levine, S., Song, D.: The false promise of imitating proprietary llms. arXiv preprint arXiv:2305.15717 (2023)
- [7] Liu, Y., Yao, Y., Ton, J.-F., Zhang, X., Guo, R., Cheng, H., Klochkov, Y., Taufiq, M.F., Li, H.: Trustworthy large language models: A survey. arXiv preprint arXiv:2308.05374 (2023)

- [8] OpenAI: GPT-4 Technical Report. CoRR **abs/2303.08774** (2023) <https://doi.org/10.48550/arXiv.2303.08774> arXiv:2303.08774 [cs.CL]
- [9] Chen, M., Tworek, J., Jun, H., Yuan, Q., Oliveira Pinto, H.P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F.P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W.H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A.N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., Zaremba, W.: Evaluating large language models trained on code. In: Advances in Neural Information Processing Systems (NeurIPS) (2021)
- [10] Roziere, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X.E., Adi, Y., Liu, J., Remez, T., Rapin, J., et al.: Code Llama: Open Foundation Models for Code (2024)
- [11] Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., Wang, H.: Large language models for software engineering: A systematic literature review. ACM Transactions on Software Engineering and Methodology **33**(8), 1–79 (2024)
- [12] Sallou, J., Durieux, T., Panichella, A.: Breaking the silence: the threats of using LLMs in software engineering. In: ACM/IEEE 46th International Conference on Software Engineering - New Ideas and Emerging Results. ACM/IEEE, ??? (2024)
- [13] Liu, C., Gao, C., Xia, X., Lo, D., Grundy, J., Yang, X.: On the reproducibility and replicability of deep learning in software engineering. ACM Transactions on Software Engineering and Methodology **31**(1), 1–46 (2021) <https://doi.org/10.1145/3477535>
- [14] Moraila, G., Shankaran, A., Shi, Z., Warren, A.M.: Measuring reproducibility in computer systems research. PLoS Comput Biol **9**, 37 (2014)
- [15] Collberg, C., Proebsting, T., Warren, A.M.: Repeatability and benefaction in computer systems research (2015)
- [16] Liu, M., Huang, X., He, W., Xie, Y., Zhang, J.M., Jing, X., Chen, Z., Ma, Y.: Research Artifacts in Software Engineering Publications: Status and Trends (2024). <https://arxiv.org/abs/2404.06852>
- [17] Merkel, D., et al.: Docker: lightweight linux containers for consistent development and deployment. Linux j **239**(2), 2 (2014)

- [18] Siddiq, M.L., Islam-Gomes, A., Sekerak, N., Santos, J.C.S.: LLM4SE\_Reproducibility\_Smells. <https://doi.org/10.5281/zenodo.17496408>. Zenodo, accessed 31 Oct 2025 (2025)
- [19] Baltes, S., Ralph, P.: Sampling in software engineering research: A critical review and guidelines. *Empirical Software Engineering* **27**(4), 94 (2022)
- [20] Stol, K.-J., Ralph, P., Fitzgerald, B.: Grounded theory in software engineering research: a critical review and guidelines. In: Proceedings of the 38th International Conference on Software Engineering, pp. 120–131 (2016)
- [21] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., ??? (2017)
- [22] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G.: Llama: Open and Efficient Foundation Language Models. CoRR <abs/2302.13971> (2023) [arXiv:2302.13971](https://arxiv.org/abs/2302.13971) [cs.CL]
- [23] DeepSeek-AI, Liu, A., Feng, B., Wang, B., Wang, B., Liu, B., Zhao, C., Dengr, C., Ruan, C., Dai, D., Guo, D., Yang, D., Chen, D., Ji, D., Li, E., Lin, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Xu, H., Yang, H., Zhang, H., Ding, H., Xin, H., Gao, H., Li, H., Qu, H., Cai, J.L., Liang, J., Guo, J., Ni, J., Li, J., Chen, J., Yuan, J., Qiu, J., Song, J., Dong, K., Gao, K., Guan, K., Wang, L., Zhang, L., Xu, L., Xia, L., Zhao, L., Zhang, L., Li, M., Wang, M., Zhang, M., Zhang, M., Tang, M., Li, M., Tian, N., Huang, P., Wang, P., Zhang, P., Zhu, Q., Chen, Q., Du, Q., Chen, R.J., Jin, R.L., Ge, R., Pan, R., Xu, R., Chen, R., Li, S.S., Lu, S., Zhou, S., Chen, S., Wu, S., Ye, S., Ma, S., Wang, S., Zhou, S., Yu, S., Zhou, S., Zheng, S., Wang, T., Pei, T., Yuan, T., Sun, T., Xiao, W.L., Zeng, W., An, W., Liu, W., Liang, W., Gao, W., Zhang, W., Li, X.Q., Jin, X., Wang, X., Bi, X., Liu, X., Wang, X., Shen, X., Chen, X., Chen, X., Nie, X., Sun, X., Wang, X., Liu, X., Xie, X., Yu, X., Song, X., Zhou, X., Yang, X., Lu, X., Su, X., Wu, Y., Li, Y.K., Wei, Y.X., Zhu, Y.X., Xu, Y., Huang, Y., Li, Y., Zhao, Y., Sun, Y., Li, Y., Wang, Y., Zheng, Y., Zhang, Y., Xiong, Y., Zhao, Y., He, Y., Tang, Y., Piao, Y., Dong, Y., Tan, Y., Liu, Y., Wang, Y., Guo, Y., Zhu, Y., Wang, Y., Zou, Y., Zha, Y., Ma, Y., Yan, Y., You, Y., Liu, Y., Ren, Z.Z., Ren, Z., Sha, Z., Fu, Z., Huang, Z., Zhang, Z., Xie, Z., Hao, Z., Shao, Z., Wen, Z., Xu, Z., Zhang, Z., Li, Z., Wang, Z., Gu, Z., Li, Z., Xie, Z.: DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model (2024). <https://arxiv.org/abs/2405.04434>
- [24] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G.: Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language

Processing (2021). <https://arxiv.org/abs/2107.13586>

- [25] Rozière, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X.E., Adi, Y., Liu, J., Sauvestre, R., Remez, T., Rapin, J., Kozhevnikov, A., Evtimov, I., Bitton, J., Bhatt, M., Ferrer, C.C., Grattafiori, A., Xiong, W., Défossez, A., Copet, J., Azhar, F., Touvron, H., Martin, L., Usunier, N., Scialom, T., Synnaeve, G.: Code llama: Open foundation models for code. arXiv preprint arXiv:2308.12950 (2023)
- [26] Nijkamp, E., Hayashi, H., Xiong, C., Savarese, S., Zhou, Y.: Codegen2: Lessons for training llms on programming and natural languages. arXiv preprint arXiv:2305.01186 (2023)
- [27] Hemmat, A., Sharbaf, M., Kolahdouz-Rahimi, S., Lano, K., Tehrani, S.Y.: Research directions for using llm in software requirement engineering: A systematic review. *Frontiers in Computer Science* **7**, 1519437 (2025)
- [28] Conrardy, A., Cabot, J.: From image to uml: First results of image based uml diagram generation using llms. arXiv preprint arXiv:2404.11376 (2024)
- [29] Ságodi, Z., Siket, I., Ferenc, R.: Methodology for code synthesis evaluation of llms presented by a case study of chatgpt and copilot. *Ieee Access* **12**, 72303–72316 (2024)
- [30] Siddiq, M.L., Da Silva Santos, J.C., Tanvir, R.H., Ulfat, N., Al Rifat, F., Carvalho Lopes, V.: Using large language models to generate junit tests: An empirical study. In: Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering. EASE '24, pp. 313–322. Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3661167.3661216> . <https://doi.org/10.1145/3661167.3661216>
- [31] Sun, W., Miao, Y., Li, Y., Zhang, H., Fang, C., Liu, Y., Deng, G., Liu, Y., Chen, Z.: Source code summarization in the era of large language models. arXiv preprint arXiv:2407.07959 (2024)
- [32] Meng, X., Ma, Z., Gao, P., Peng, C.: An empirical study on llm-based agents for automated bug fixing. arXiv preprint arXiv:2411.10213 (2024)
- [33] Yu, Z., Ma, M., Feng, X., Ding, R., Zhang, C., Li, Z., Chintalapati, M., Zhang, X., Wang, R., Bansal, C., *et al.*: Triangle: Empowering incident triage with multi-llm-agents. In: Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering. ACM (2025)
- [34] Izadi, M., Gismondi, R., Gousios, G.: Codefill: Multi-token code completion by jointly learning from structure and naming sequences. In: 44th International Conference on Software Engineering (ICSE) (2022)

- [35] Kim, S., Zhao, J., Tian, Y., Chandra, S.: Code prediction by feeding trees to transformers. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), pp. 150–162 (2021). IEEE
- [36] Svyatkovskiy, A., Lee, S., Hadjitofti, A., Riechert, M., Franco, J.V., Allamanis, M.: Fast and memory-efficient neural code completion. In: 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), pp. 329–340 (2021). IEEE
- [37] Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., Zhou, M.: CodeBERT: A pre-trained model for programming and natural languages. In: Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 1536–1547. Association for Computational Linguistics, Online (2020). <https://doi.org/10.18653/v1/2020.findings-emnlp.139>
- [38] Gao, Y., Lyu, C.: M2ts: Multi-scale multi-modal approach based on transformer for source code summarization. In: Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension. ICPC ’22, pp. 24–35. Association for Computing Machinery, New York, NY, USA (2022). <https://doi.org/10.1145/3524610.3527907>
- [39] Tian, H., Liu, K., Kaboreé, A.K., Koyuncu, A., Li, L., Klein, J., Bissyandé, T.F.: Evaluating Representation Learning of Code Changes for Predicting Patch Correctness in Program Repair. arXiv (2020). <https://doi.org/10.48550/ARXIV.2008.02944> . <https://arxiv.org/abs/2008.02944>
- [40] Liu, F., Li, G., Zhao, Y., Jin, Z.: Multi-task learning based pre-trained language model for code completion. In: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering. ASE ’20, pp. 473–485. ACM, ??? (2020). <https://doi.org/10.1145/3324884.3416591> . <http://dx.doi.org/10.1145/3324884.3416591>
- [41] Kanade, A., Maniatis, P., Balakrishnan, G., Shi, K.: Learning and Evaluating Contextual Embedding of Source Code (2020). <https://arxiv.org/abs/2001.00059>
- [42] Hey, T., Keim, J., Koziolek, A., Tichy, W.F.: Norbert: Transfer learning for requirements classification. In: 2020 IEEE 28th International Requirements Engineering Conference (RE), pp. 169–179. IEEE, ??? (2020). <https://doi.org/10.1109/re48521.2020.00028> . <http://dx.doi.org/10.1109/RE48521.2020.00028>
- [43] Zhang, T., Xu, B., Thung, F., Haryono, S.A., Lo, D., Jiang, L.: Sentiment analysis for software engineering: How far can pre-trained transformer models go? In: 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 70–80. IEEE, ??? (2020). <https://doi.org/10.1109/icsme46990.2020.00017> . <http://dx.doi.org/10.1109/ICSME46990.2020.00017>

- [44] Biswas, E., Karabulut, M.E., Pollock, L., Vijay-Shanker, K.: Achieving reliable sentiment analysis in the software engineering domain using bert. In: 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 162–173. IEEE, ??? (2020). <https://doi.org/10.1109/icsme46990.2020.00025> . <http://dx.doi.org/10.1109/ICSME46990.2020.00025>
- [45] Wang, Y., Shi, L., Li, M., Wang, Q., Yang, Y.: A deep context-wise method for coreference detection in natural language requirements. In: 2020 IEEE 28th International Requirements Engineering Conference (RE). IEEE, ??? (2020). <https://doi.org/10.1109/re48521.2020.00029> . <http://dx.doi.org/10.1109/RE48521.2020.00029>
- [46] Prenner, J.A., Robbes, R.: Making the most of small Software Engineering datasets with modern machine learning. arXiv (2021). <https://doi.org/10.48550/ARXIV.2106.15209> . <https://arxiv.org/abs/2106.15209>
- [47] Pearce, H., Tan, B., Ahmad, B., Karri, R., Dolan-Gavitt, B.: Examining Zero-Shot Vulnerability Repair with Large Language Models. arXiv (2021). <https://doi.org/10.48550/ARXIV.2112.02125> . <https://arxiv.org/abs/2112.02125>
- [48] Mastropaoletti, A., Scalabrino, S., Cooper, N., Nader Palacio, D., Poshyvanyk, D., Oliveto, R., Bavota, G.: Studying the usage of text-to-text transfer transformer to support code-related tasks. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), pp. 336–347. IEEE, ??? (2021). <https://doi.org/10.1109/icse43902.2021.00041> . <http://dx.doi.org/10.1109/ICSE43902.2021.00041>
- [49] Mastropaoletti, A., Aghajani, E., Pascarella, L., Bavota, G.: An empirical study on code comment completion. In: 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 159–170. IEEE, ??? (2021). <https://doi.org/10.1109/icsme52107.2021.00021> . <http://dx.doi.org/10.1109/ICSME52107.2021.00021>
- [50] Lin, J., Liu, Y., Zeng, Q., Jiang, M., Cleland-Huang, J.: Traceability transformed: Generating more accurate links with pre-trained bert models. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), pp. 324–335. IEEE, ??? (2021). <https://doi.org/10.1109/icse43902.2021.00040> . <http://dx.doi.org/10.1109/ICSE43902.2021.00040>
- [51] Li, J., Huang, R., Li, W., Yao, K., Tan, W.: Toward less hidden cost of code completion with acceptance and ranking models. In: 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 195–205. IEEE, ??? (2021). <https://doi.org/10.1109/icsme52107.2021.00024> . <http://dx.doi.org/10.1109/ICSME52107.2021.00024>
- [52] Khan, J.Y., Tawkat Islam Khondaker, M., Uddin, G., Iqbal, A.: Automatic detection of five api documentation smells: Practitioners' perspectives. In: 2021

- IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 318–329. IEEE, ??? (2021). <https://doi.org/10.1109/saner50967.2021.00037> . <http://dx.doi.org/10.1109/SANER50967.2021.00037>
- [53] Isotani, H., Washizaki, H., Fukazawa, Y., Nomoto, T., Oiji, S., Saito, S.: Duplicate bug report detection by using sentence embedding and fine-tuning. In: 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 535–544. IEEE, ??? (2021). <https://doi.org/10.1109/icsme52107.2021.00054> . <http://dx.doi.org/10.1109/ICSME52107.2021.00054>
- [54] Henkel, J., Silva, D., Teixeira, L., d’Amorim, M., Reps, T.: Shipwright: A human-in-the-loop system for dockerfile repair. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), pp. 1148–1160. IEEE, ??? (2021). <https://doi.org/10.1109/icse43902.2021.00106> . <http://dx.doi.org/10.1109/ICSE43902.2021.00106>
- [55] Hendrycks, D., Basart, S., Kadavath, S., Mazeika, M., Arora, A., Guo, E., Burns, C., Puranik, S., He, H., Song, D., Steinhardt, J.: Measuring Coding Challenge Competence With APPS. arXiv (2021). <https://doi.org/10.48550/ARXIV.2105.09938> . <https://arxiv.org/abs/2105.09938>
- [56] Ghadhab, L., Jenhani, I., Mkaouer, M.W., Ben Messaoud, M.: Augmenting commit classification by using fine-grained source code changes and a pre-trained deep neural language model. Information and Software Technology **135**, 106566 (2021) <https://doi.org/10.1016/j.infsof.2021.106566>
- [57] Ciniselli, M., Cooper, N., Pascarella, L., Mastropaoletti, A., Aghajani, E., Poshyvanyk, D., Di Penta, M., Bavota, G.: An empirical study on the usage of transformer models for code completion. IEEE Transactions on Software Engineering, 1–1 (2021) <https://doi.org/10.1109/tse.2021.3128234>
- [58] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.d.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F.P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W.H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A.N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., Zaremba, W.: Evaluating Large Language Models Trained on Code. arXiv (2021). <https://doi.org/10.48550/ARXIV.2107.03374> . <https://arxiv.org/abs/2107.03374>
- [59] Tufano, R., Masiero, S., Mastropaoletti, A., Pascarella, L., Poshyvanyk, D., Bavota, G.: Using Pre-Trained Models to Boost Code Review Automation. arXiv (2022). <https://doi.org/10.48550/ARXIV.2201.06850> . <https://arxiv.org/abs/2201.06850>

- [60] Thapa, C., Jang, S.I., Ahmed, M.E., Camtepe, S., Pieprzyk, J., Nepal, S.: Transformer-Based Language Models for Software Vulnerability Detection. arXiv (2022). <https://doi.org/10.48550/ARXIV.2204.03214> . <https://arxiv.org/abs/2204.03214>
- [61] Su, H., Kasai, J., Wu, C.H., Shi, W., Wang, T., Xin, J., Zhang, R., Ostdorf, M., Zettlemoyer, L., Smith, N.A., Yu, T.: Selective Annotation Makes Language Models Better Few-Shot Learners. arXiv (2022). <https://doi.org/10.48550/ARXIV.2209.01975> . <https://arxiv.org/abs/2209.01975>
- [62] Shi, Z., Xiong, Y., Zhang, X., Zhang, Y., Li, S., Zhu, Y.: Cross-modal contrastive learning for code search. In: 2022 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 94–105. IEEE, ??? (2022). <https://doi.org/10.1109/icsme55016.2022.00017> . <http://dx.doi.org/10.1109/ICSME55016.2022.00017>
- [63] Shen, D., Chen, X., Wang, C., Sen, K., Song, D.: Benchmarking Language Models for Code Syntax Understanding. arXiv (2022). <https://doi.org/10.48550/ARXIV.2210.14473> . <https://arxiv.org/abs/2210.14473>
- [64] Sharma, R., Chen, F., Fard, F., Lo, D.: An Exploratory Study on Code Attention in BERT. arXiv (2022). <https://doi.org/10.48550/ARXIV.2204.10200> . <https://arxiv.org/abs/2204.10200>
- [65] Workshop, B., :; Scao, T.L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A.S., Yvon, F., Gallé, M., Tow, J., Rush, A.M., Biderman, S., Webson, A., Ammanamanchi, P.S., Wang, T., Sagot, B., Muenninghoff, N., Moral, A.V., Ruwase, O., Bawden, R., Bekman, S., McMillan-Major, A., Beltagy, I., Nguyen, H., Saulnier, L., Tan, S., Suarez, P.O., Sanh, V., Laurençon, H., Jernite, Y., Launay, J., Mitchell, M., Raffel, C., Gokaslan, A., Simhi, A., Soroa, A., Aji, A.F., Alfassy, A., Rogers, A., Nitzav, A.K., Xu, C., Mou, C., Emezue, C., Klamm, C., Leong, C., Strien, D., Adelani, D.I., Radev, D., Ponferrada, E.G., Levkovizh, E., Kim, E., Natan, E.B., De Toni, F., Dupont, G., Kruszewski, G., Pistilli, G., Elsahar, H., Benyamina, H., Tran, H., Yu, I., Abdulkummin, I., Johnson, I., Gonzalez-Dios, I., Rosa, J., Chim, J., Dodge, J., Zhu, J., Chang, J., Frohberg, J., Tobing, J., Bhattacharjee, J., Almubarak, K., Chen, K., Lo, K., Von Werra, L., Weber, L., Phan, L., allal, L.B., Tanguy, L., Dey, M., Muñoz, M.R., Masoud, M., Grandury, M., Saško, M., Huang, M., Coavoux, M., Singh, M., Jiang, M.T.-J., Vu, M.C., Jauhar, M.A., Ghaleb, M., Subramani, N., Kassner, N., Khamis, N., Nguyen, O., Espejel, O., Gibert, O., Villegas, P., Henderson, P., Colombo, P., Amuok, P., Lhoest, Q., Harliman, R., Bommasani, R., López, R.L., Ribeiro, R., Osei, S., Pyysalo, S., Nagel, S., Bose, S., Muhammad, S.H., Sharma, S., Longpre, S., Nikpoor, S., Silberberg, S., Pai, S., Zink, S., Torrent, T.T., Schick, T., Thrush, T., Danchev, V., Nikoulina, V., Laippala, V., Lepercq, V., Prabhu, V., Alyafeai, Z., Talat, Z., Raja, A., Heinzerling, B., Si, C., Taşar, D.E., Salesky, E., Mielke, S.J., Lee, W.Y., Sharma, A.,

Santilli, A., Chaffin, A., Stiegler, A., Datta, D., Szczechla, E., Chhablani, G., Wang, H., Pandey, H., Strobelt, H., Fries, J.A., Rozen, J., Gao, L., Sutawika, L., Bari, M.S., Al-shaibani, M.S., Manica, M., Nayak, N., Teehan, R., Albanie, S., Shen, S., Ben-David, S., Bach, S.H., Kim, T., Bers, T., Fevry, T., Neeraj, T., Thakker, U., Raunak, V., Tang, X., Yong, Z.-X., Sun, Z., Brody, S., Uri, Y., Tojarieh, H., Roberts, A., Chung, H.W., Tae, J., Phang, J., Press, O., Li, C., Narayanan, D., Bourfoune, H., Casper, J., Rasley, J., Ryabinin, M., Mishra, M., Zhang, M., Shoeybi, M., Peyrounette, M., Patry, N., Tazi, N., Sanseviero, O., Platen, P., Cornette, P., Lavallée, P.F., Lacroix, R., Rajbhandari, S., Gandhi, S., Smith, S., Requena, S., Patil, S., Dettmers, T., Baruwa, A., Singh, A., Cheveleva, A., Ligozat, A.-L., Subramonian, A., Névéol, A., Lovering, C., Garrette, D., Tunuguntla, D., Reiter, E., Taktasheva, E., Voloshina, E., Bogdanov, E., Winata, G.I., Schoelkopf, H., Kalo, J.-C., Novikova, J., Forde, J.Z., Clive, J., Kasai, J., Kawamura, K., Hazan, L., Carpuat, M., Clinciu, M., Kim, N., Cheng, N., Serikov, O., Antverg, O., Wal, O., Zhang, R., Zhang, R., Gehrmann, S., Mirkin, S., Pais, S., Shavrina, T., Scialom, T., Yun, T., Limisiewicz, T., Rieser, V., Protasov, V., Mikhailov, V., Pruksachatkun, Y., Belinkov, Y., Bamberger, Z., Kasner, Z., Rueda, A., Pestana, A., Feizpour, A., Khan, A., Faranak, A., Santos, A., Hevia, A., Unldreaj, A., Aghagol, A., Abdollahi, A., Tammour, A., HajiHosseini, A., Behroozi, B., Ajibade, B., Saxena, B., Ferrandis, C.M., McDuff, D., Contractor, D., Lansky, D., David, D., Kiela, D., Nguyen, D.A., Tan, E., Baylor, E., Ozoani, E., Mirza, F., Ononiwu, F., Rezanejad, H., Jones, H., Bhattacharya, I., Solaiman, I., Sedenko, I., Nejadgholi, I., Passmore, J., Seltzer, J., Sanz, J.B., Dutra, L., Samagaio, M., Elbadri, M., Mieskes, M., Gerchick, M., Akinlolu, M., McKenna, M., Qiu, M., Ghauri, M., Burynok, M., Abrar, N., Rajani, N., Elkott, N., Fahmy, N., Samuel, O., An, R., Kromann, R., Hao, R., Alizadeh, S., Shubber, S., Wang, S., Roy, S., Viguier, S., Le, T., Oyebade, T., Le, T., Yang, Y., Nguyen, Z., Kashyap, A.R., Palasciano, A., Callahan, A., Shukla, A., Miranda-Escalada, A., Singh, A., Beilharz, B., Wang, B., Brito, C., Zhou, C., Jain, C., Xu, C., Fourrier, C., Periñán, D.L., Molano, D., Yu, D., Manjavacas, E., Barth, F., Fuhrmann, F., Altay, G., Bayrak, G., Burns, G., Vrabec, H.U., Bello, I., Dash, I., Kang, J., Giorgi, J., Golde, J., Posada, J.D., Sivaraman, K.R., Bulchandani, L., Liu, L., Shinzato, L., Bykhovetz, M.H., Takeuchi, M., Pàmies, M., Castillo, M.A., Nezhurina, M., Sänger, M., Samwald, M., Cullan, M., Weinberg, M., De Wolf, M., Mihaljevic, M., Liu, M., Freidank, M., Kang, M., Seelam, N., Dahlberg, N., Broad, N.M., Muellner, N., Fung, P., Haller, P., Chandrasekhar, R., Eisenberg, R., Martin, R., Canalli, R., Su, R., Su, R., Cahyawijaya, S., Garda, S., Deshmukh, S.S., Mishra, S., Kiblawi, S., Ott, S., Sang-aroonsiri, S., Kumar, S., Schweter, S., Bharati, S., Laud, T., Gigant, T., Kainuma, T., Kusa, W., Labrak, Y., Bajaj, Y.S., Venkatraman, Y., Xu, Y., Xu, Y., Xu, Y., Tan, Z., Xie, Z., Ye, Z., Bras, M., Belkada, Y., Wolf, T.: BLOOM: A 176B-Parameter Open-Access Multilingual Language Model. arXiv (2022). <https://doi.org/10.48550/ARXIV.2211.05100> . <https://arxiv.org/abs/2211.05100>

[66] Salza, P., Schwizer, C., Gu, J., Gall, H.C.: On the effectiveness of transfer

- learning for code search. *IEEE Trans. Softw. Eng.* **49**(4), 1804–1822 (2023) <https://doi.org/10.1109/TSE.2022.3192755>
- [67] Niu, C., Li, C., Ng, V., Ge, J., Huang, L., Luo, B.: SPT-Code: Sequence-to-Sequence Pre-Training for Learning Source Code Representations. arXiv (2022). <https://doi.org/10.48550/ARXIV.2201.01549> . <https://arxiv.org/abs/2201.01549>
- [68] Moharil, A., Sharma, A.: Identification of intra-domain ambiguity using transformer-based machine learning. In: Proceedings of the 1st International Workshop on Natural Language-based Software Engineering. ICSE '22, pp. 51–58. ACM, ??? (2022). <https://doi.org/10.1145/3528588.3528651> . <http://dx.doi.org/10.1145/3528588.3528651>
- [69] Mastropaoletti, A., Pascarella, L., Bavota, G.: Using deep learning to generate complete log statements. In: Proceedings of the 44th International Conference on Software Engineering. ICSE '22, pp. 2279–2290. ACM, ??? (2022). <https://doi.org/10.1145/3510003.3511561> . <http://dx.doi.org/10.1145/3510003.3511561>
- [70] Mastropaoletti, A., Cooper, N., Palacio, D.N., Scalabrino, S., Poshyvanyk, D., Oliveto, R., Bavota, G.: Using transfer learning for code-related tasks. *IEEE Transactions on Software Engineering* **49**(4), 1580–1598 (2023) <https://doi.org/10.1109/tse.2022.3183297>
- [71] Madaan, A., Zhou, S., Alon, U., Yang, Y., Neubig, G.: Language Models of Code are Few-Shot Commonsense Learners. arXiv (2022). <https://doi.org/10.48550/ARXIV.2210.07128> . <https://arxiv.org/abs/2210.07128>
- [72] Luo, X., Xue, Y., Xing, Z., Sun, J.: Prbert: Prompt learning for requirement classification using bert-based pretrained language models. In: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. ASE '22, pp. 1–13. ACM, ??? (2022). <https://doi.org/10.1145/3551349.3560417> . <http://dx.doi.org/10.1145/3551349.3560417>
- [73] Li, Y., Zhang, T., Luo, X., Cai, H., Fang, S., Yuan, D.: Do pretrained language models indeed understand software engineering tasks? *IEEE Transactions on Software Engineering* **49**(10), 4639–4655 (2023) <https://doi.org/10.1109/tse.2023.3308952>
- [74] Li, D., Shen, Y., Jin, R., Mao, Y., Wang, K., Chen, W.: Generation-Augmented Query Expansion For Code Retrieval. arXiv (2022). <https://doi.org/10.48550/ARXIV.2212.10692> . <https://arxiv.org/abs/2212.10692>
- [75] Li, L., Yang, L., Jiang, H., Yan, J., Luo, T., Hua, Z., Liang, G., Zuo, C.: 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. ESEC/FSE '22,

- pp. 1009–1021. ACM, ??? (2022). <https://doi.org/10.1145/3540250.3549099> . <http://dx.doi.org/10.1145/3540250.3549099>
- [76] Lee, J., Han, K., Yu, H.: A light bug triage framework for applying large pre-trained language model. In: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. ASE '22, pp. 1–11. ACM, ??? (2022). <https://doi.org/10.1145/3551349.3556898> . <http://dx.doi.org/10.1145/3551349.3556898>
- [77] Lajkó, M., Csuvik, V., Vidács, L.: Towards javascript program repair with generative pre-trained transformer (gpt-2). In: Proceedings of the Third International Workshop on Automated Program Repair. ICSE '22, pp. 61–68. ACM, ??? (2022). <https://doi.org/10.1145/3524459.3527350> . <http://dx.doi.org/10.1145/3524459.3527350>
- [78] Lahiri, S.K., Fakhouri, S., Naik, A., Sakkas, G., Chakraborty, S., Musuvathi, M., Choudhury, P., Veh, C., Inala, J.P., Wang, C., Gao, J.: Interactive Code Generation via Test-Driven User-Intent Formalization. arXiv (2022). <https://doi.org/10.48550/ARXIV.2208.05950> . <https://arxiv.org/abs/2208.05950>
- [79] Kuznia, K., Mishra, S., Parmar, M., Baral, C.: Less is more: Summary of long instructions is better for program synthesis. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, ??? (2022). <https://doi.org/10.18653/v1/2022.emnlp-main.301> . <http://dx.doi.org/10.18653/v1/2022.emnlp-main.301>
- [80] Khan, J.Y., Uddin, G.: Automatic detection and analysis of technical debts in peer-review documentation of r packages. In: 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 765–776. IEEE, ??? (2022). <https://doi.org/10.1109/saner53432.2022.00094> . <http://dx.doi.org/10.1109/SANER53432.2022.00094>
- [81] Kang, S., Yoon, J., Yoo, S.: Large language models are few-shot testers: Exploring llm-based general bug reproduction. In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), pp. 2312–2323. IEEE, ??? (2023). <https://doi.org/10.1109/icse48619.2023.00194> . <http://dx.doi.org/10.1109/ICSE48619.2023.00194>
- [82] Jones, E., Steinhardt, J.: Capturing Failures of Large Language Models via Human Cognitive Biases (2022). <https://arxiv.org/abs/2202.12299>
- [83] Jesse, K., Devanbu, P.T., Sawant, A.: Learning to predict user-defined types. IEEE Transactions on Software Engineering **49**(4), 1508–1522 (2023) <https://doi.org/10.1109/tse.2022.3178945>
- [84] Jain, N., Vaidyanath, S., Iyer, A., Natarajan, N., Parthasarathy, S., Rajamani,

- S., Sharma, R.: Jigsaw: large language models meet program synthesis. In: Proceedings of the 44th International Conference on Software Engineering. ICSE '22, pp. 1219–1231. ACM, ??? (2022). <https://doi.org/10.1145/3510003.3510203> . <http://dx.doi.org/10.1145/3510003.3510203>
- [85] Izadi, M., Gismondi, R., Gousios, G.: Codefill: multi-token code completion by jointly learning from structure and naming sequences. In: Proceedings of the 44th International Conference on Software Engineering. ICSE '22, pp. 401–412. ACM, ??? (2022). <https://doi.org/10.1145/3510003.3510172> . <http://dx.doi.org/10.1145/3510003.3510172>
- [86] Zhu, J., Xiao, G., Zheng, Z., Sui, Y.: Enhancing traceability link recovery with unlabeled data. In: 2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE), pp. 446–457. IEEE, ??? (2022). <https://doi.org/10.1109/issre55969.2022.00050> . <http://dx.doi.org/10.1109/ISSRE55969.2022.00050>
- [87] Zhang, J., Mytkowicz, T., Kaufman, M., Piskac, R., Lahiri, S.K.: Using pre-trained language models to resolve textual and semantic merge conflicts (experience paper). In: Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis. ISSTA '22, pp. 77–88. ACM, ??? (2022). <https://doi.org/10.1145/3533767.3534396> . <http://dx.doi.org/10.1145/3533767.3534396>
- [88] He, J., Xu, B., Yang, Z., Han, D., Yang, C., Lo, D.: PtM4tag: sharpening tag recommendation of stack overflow posts with pre-trained models. In: Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension. ICPC '22, pp. 1–11. ACM, ??? (2022). <https://doi.org/10.1145/3524610.3527897> . <http://dx.doi.org/10.1145/3524610.3527897>
- [89] Zhang, J., Panthaplatzel, S., Nie, P., Li, J.J., Gligoric, M.: Coditt5: Pretraining for source code and natural language editing. In: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. ASE '22, pp. 1–12. ACM, ??? (2022). <https://doi.org/10.1145/3551349.3556955> . <http://dx.doi.org/10.1145/3551349.3556955>
- [90] Gu, J., Salza, P., Gall, H.C.: Assemble foundation models for automatic code summarization. In: 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, ??? (2022). <https://doi.org/10.1109/saner53432.2022.00112> . <http://dx.doi.org/10.1109/SANER53432.2022.00112>
- [91] Zhang, J., Liu, S., Gong, L., Zhang, H., Huang, Z., Jiang, H.: Beqain: An effective and efficient identifier normalization approach with bert and the question answering system. IEEE Transactions on Software Engineering **49**(4), 2597–2620 (2023) <https://doi.org/10.1109/tse.2022.3227559>
- [92] Fu, M., Tantithamthavorn, C.: Gpt2sp: A transformer-based agile story point

- estimation approach. *IEEE Transactions on Software Engineering* **49**(2), 611–625 (2023) <https://doi.org/10.1109/tse.2022.3158252>
- [93] Fatima, S., Ghaleb, T.A., Briand, L.: Flakify: A black-box, language model-based predictor for flaky tests. *IEEE Transactions on Software Engineering* **49**(4), 1912–1927 (2023) <https://doi.org/10.1109/tse.2022.3201209>
- [94] Fan, Z., Gao, X., Mirchev, M., Roychoudhury, A., Tan, S.H.: Automated repair of programs from large language models. In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), pp. 1469–1481. IEEE, ??? (2023). <https://doi.org/10.1109/icse48619.2023.00128> . <http://dx.doi.org/10.1109/ICSE48619.2023.00128>
- [95] Ezzini, S., Abualhaija, S., Arora, C., Sabetzadeh, M.: Automated handling of anaphoric ambiguity in requirements: a multi-solution study. In: Proceedings of the 44th International Conference on Software Engineering. ICSE '22, pp. 187–199. ACM, ??? (2022). <https://doi.org/10.1145/3510003.3510157> . <http://dx.doi.org/10.1145/3510003.3510157>
- [96] Zeng, Z., Tan, H., Zhang, H., Li, J., Zhang, Y., Zhang, L.: An extensive study on pre-trained models for program understanding and generation. In: Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis. ISSTA '22, pp. 39–51. ACM, ??? (2022). <https://doi.org/10.1145/3533767.3534390> . <http://dx.doi.org/10.1145/3533767.3534390>
- [97] Döderlein, J.-B., Kouadio, N.H., Acher, M., Khelladi, D.E., Combemale, B.: Piloting copilot, codex, and starcoder2: Hot temperature, cold prompts, or black magic? *Journal of Systems and Software* **230**, 112562 (2025) <https://doi.org/10.1016/j.jss.2025.112562>
- [98] Dibia, V., Fourney, A., Bansal, G., Poursabzi-Sangdeh, F., Liu, H., Amershi, S.: Aligning Offline Metrics and Human Judgments of Value for Code Generation Models. arXiv (2022). <https://doi.org/10.48550/ARXIV.2210.16494> . <https://arxiv.org/abs/2210.16494>
- [99] Ciborowska, A., Damevski, K.: Fast changeset-based bug localization with bert. In: Proceedings of the 44th International Conference on Software Engineering. ICSE '22, pp. 946–957. ACM, ??? (2022). <https://doi.org/10.1145/3510003.3510042> . <http://dx.doi.org/10.1145/3510003.3510042>
- [100] Chochlov, M., Aftab Ahmed, G., Vincent Patten, J., Lu, G., Hou, W., Gregg, D., Buckley, J.: Using a nearest-neighbour, bert-based approach for scalable clone detection. In: 2022 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 582–591. IEEE, ??? (2022). <https://doi.org/10.1109/icsme55016.2022.00080> . <http://dx.doi.org/10.1109/ICSME55016.2022.00080>
- [101] Chen, F., Fard, F.H., Lo, D., Bryksin, T.: On the transferability of pre-trained

- language models for low-resource programming languages. In: Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension. ICPC '22, pp. 401–412. ACM, ??? (2022). <https://doi.org/10.1145/3524610.3527917> . <http://dx.doi.org/10.1145/3524610.3527917>
- [102] Zan, D., Chen, B., Lin, Z., Guan, B., Wang, Y., Lou, J.-G.: When Language Model Meets Private Library. arXiv (2022). <https://doi.org/10.48550/ARXIV.2210.17236> . <https://arxiv.org/abs/2210.17236>
- [103] Bareiš, P., Souza, B., d'Amorim, M., Pradel, M.: Code Generation Tools (Almost) for Free? A Study of Few-Shot, Pre-Trained Language Models on Code. arXiv (2022). <https://doi.org/10.48550/ARXIV.2206.01335> . <https://arxiv.org/abs/2206.01335>
- [104] Alhamed, M., Storer, T.: Evaluation of context-aware language models and experts for effort estimation of software maintenance issues. In: 2022 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 129–138. IEEE, ??? (2022). <https://doi.org/10.1109/icsme55016.2022.00020> . <http://dx.doi.org/10.1109/ICSME55016.2022.00020>
- [105] Zan, D., Chen, B., Yang, D., Lin, Z., Kim, M., Guan, B., Wang, Y., Chen, W., Lou, J.-G.: CERT: Continual Pre-Training on Sketches for Library-Oriented Code Generation. arXiv (2022). <https://doi.org/10.48550/ARXIV.2206.06888> . <https://arxiv.org/abs/2206.06888>
- [106] Yuan, W., Zhang, Q., He, T., Fang, C., Hung, N.Q.V., Hao, X., Yin, H.: CIR-CLE: Continual Repair across Programming Languages. arXiv (2022). <https://doi.org/10.48550/ARXIV.2205.10956> . <https://arxiv.org/abs/2205.10956>
- [107] Yang, c., Xu, B., Khan, J.y., Uddin, G., Han, D., Yang, Z., Lo, D.: Aspect-Based API Review Classification: How Far Can Pre-Trained Transformer Model Go? arXiv (2022). <https://doi.org/10.48550/ARXIV.2201.11327> . <https://arxiv.org/abs/2201.11327>
- [108] Xu, F.F., Alon, U., Neubig, G., Hellendoorn, V.J.: A Systematic Evaluation of Large Language Models of Code. arXiv (2022). <https://doi.org/10.48550/ARXIV.2202.13169> . <https://arxiv.org/abs/2202.13169>
- [109] Xia, C.S., Wei, Y., Zhang, L.: Automated program repair in the era of large pre-trained language models. In: Proceedings of the 45th International Conference on Software Engineering. ICSE '23, pp. 1482–1494. IEEE Press, ??? (2023). <https://doi.org/10.1109/ICSE48619.2023.00129> . <https://doi.org/10.1109/ICSE48619.2023.00129>
- [110] Wei, M., Harzevili, N.S., Huang, Y., Wang, J., Wang, S.: Clear: &lt;u&gt;c&lt;/u&gt;ontrastive &lt;u&gt;le&lt;/u&gt;arning for &lt;u&gt;a&lt;/u&gt;pi &lt;u&gt;r&lt;/u&gt;ecommendation. In: Proceedings

of the 44th International Conference on Software Engineering. ICSE '22, pp. 376–387. ACM, ??? (2022). <https://doi.org/10.1145/3510003.3510159> . <http://dx.doi.org/10.1145/3510003.3510159>

- [111] Wang, Y., Wang, J., Zhang, H., Ming, X., Shi, L., Wang, Q.: Where is Your App Frustrating Users? . In: 2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE), pp. 2427–2439. IEEE Computer Society, Los Alamitos, CA, USA (2022). <https://doi.ieeecomputersociety.org/>
- [112] Wang, S., Li, Z., Qian, H., Yang, C., Wang, Z., Shang, M., Kumar, V., Tan, S., Ray, B., Bhatia, P., Nallapati, R., Ramanathan, M.K., Roth, D., Xiang, B.: ReCode: Robustness Evaluation of Code Generation Models. arXiv (2022). <https://doi.org/10.48550/ARXIV.2212.10264> . <https://arxiv.org/abs/2212.10264>
- [113] Wan, Y., Zhao, W., Zhang, H., Sui, Y., Xu, G., Jin, H.: What Do They Capture? – A Structural Analysis of Pre-Trained Language Models for Source Code. arXiv (2022). <https://doi.org/10.48550/ARXIV.2202.06840> . <https://arxiv.org/abs/2202.06840>
- [114] Mosel, J., Trautsch, A., Herbold, S.: On the validity of pre-trained transformers for natural language processing in the software engineering domain. arXiv (2021). <https://doi.org/10.48550/ARXIV.2109.04738> . <https://arxiv.org/abs/2109.04738>
- [115] Zhuo, T.Y., Du, X., Xing, Z., Sun, J., Quan, H., Li, L., Zhu, L.: Pop Quiz! Do Pre-trained Code Models Possess Knowledge of Correct API Names? arXiv (2023). <https://doi.org/10.48550/ARXIV.2309.07804> . <https://arxiv.org/abs/2309.07804>
- [116] Kocmi, T., Federmann, C.: Large Language Models Are State-of-the-Art Evaluators of Translation Quality. arXiv (2023). <https://doi.org/10.48550/ARXIV.2302.14520> . <https://arxiv.org/abs/2302.14520>
- [117] Zhao, J., Rong, Y., Guo, Y., He, Y., Chen, H.: Understanding Programs by Exploiting (Fuzzing) Test Cases. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.13592> . <https://arxiv.org/abs/2305.13592>
- [118] Tufano, M., Chadel, S., Agarwal, A., Sundaresan, N., Clement, C.: Predicting Code Coverage without Execution. arXiv (2023). <https://doi.org/10.48550/ARXIV.2307.13383> . <https://arxiv.org/abs/2307.13383>
- [119] Tu, H., Zhou, Z., Jiang, H., Yusuf, I.N.B., Li, Y., Jiang, L.: Isolating compiler bugs by generating effective witness programs with large language models. IEEE Trans. Softw. Eng. **50**(7), 1768–1788 (2024) <https://doi.org/10.1109/TSE.2024.3397822>

- [120] Tihanyi, N., Bisztray, T., Jain, R., Ferrag, M.A., Cordeiro, L.C., Mavroeidis, V.: The formai dataset: Generative ai in software security through the lens of formal verification. In: Proceedings of the 19th International Conference on Predictive Models and Data Analytics in Software Engineering. PROMISE 2023, pp. 33–43. Association for Computing Machinery, New York, NY, USA (2023). <https://doi.org/10.1145/3617555.3617874> . <https://doi.org/10.1145/3617555.3617874>
- [121] Tian, Z., Chen, J., Zhang, X.: Fixing Large Language Models’ Specification Misunderstanding for Better Code Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2309.16120> . <https://arxiv.org/abs/2309.16120>
- [122] Tian, H., Lu, W., Li, T.O., Tang, X., Cheung, S.-C., Klein, J., Bissyandé, T.F.: Is ChatGPT the Ultimate Programming Assistant – How far is it? arXiv (2023). <https://doi.org/10.48550/ARXIV.2304.11938> . <https://arxiv.org/abs/2304.11938>
- [123] Tian, H., Liu, K., Li, Y., Kaboré, A.K., Koyuncu, A., Habib, A., Li, L., Wen, J., Klein, J., Bissyandé, T.F.: The Best of Both Worlds: Combining Learned Embeddings with Engineered Features for Accurate Prediction of Correct Patches. arXiv (2022). <https://doi.org/10.48550/ARXIV.2203.08912> . <https://arxiv.org/abs/2203.08912>
- [124] Zhao, Z., Xu, Z., Zhu, J., Di, P., Yao, Y., Ma, X.: The Right Prompts for the Job: Repair Code-Review Defects with Large Language Model. arXiv (2023). <https://doi.org/10.48550/ARXIV.2312.17485> . <https://arxiv.org/abs/2312.17485>
- [125] Thakur, S., Ahmad, B., Pearce, H., Tan, B., Dolan-Gavitt, B., Karri, R., Garg, S.: VeriGen: A Large Language Model for Verilog Code Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2308.00708> . <https://arxiv.org/abs/2308.00708>
- [126] Tarassow, A.: The potential of LLMs for coding with low-resource and domain-specific programming languages. arXiv (2023). <https://doi.org/10.48550/ARXIV.2307.13018> . <https://arxiv.org/abs/2307.13018>
- [127] Tang, X., Chen, Z., Kim, K., Tian, H., Ezzini, S., Klein, J.: Just-in-Time Detection of Silent Security Patches. arXiv (2023). <https://doi.org/10.48550/ARXIV.2312.01241> . <https://arxiv.org/abs/2312.01241>
- [128] Tang, Z., Ge, J., Liu, S., Zhu, T., Xu, T., Huang, L., Luo, B.: Domain Adaptive Code Completion via Language Models and Decoupled Domain Databases. arXiv (2023). <https://doi.org/10.48550/ARXIV.2308.09313> . <https://arxiv.org/abs/2308.09313>
- [129] Tang, W., Tang, M., Ban, M., Zhao, Z., Feng, M.: Csgvd: A deep learning approach combining sequence and graph embedding for source code vulnerability detection. Journal of Systems and Software **199**, 111623 (2023) <https://doi.org/10.1016/j.jss.2023.111623>

<https://doi.org/10.1016/j.jss.2023.111623>

- [130] Tang, Y., Liu, Z., Zhou, Z., Luo, X.: ChatGPT vs SBST: A Comparative Assessment of Unit Test Suite Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2307.00588> . <https://arxiv.org/abs/2307.00588>
- [131] Tan, C.W., Guo, S., Wong, M.F., Hang, C.N.: Copilot for Xcode: Exploring AI-Assisted Programming by Prompting Cloud-based Large Language Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2307.14349> . <https://arxiv.org/abs/2307.14349>
- [132] Sun, J., Xing, Z., Lu, Q., Xu, X., Zhu, L., Hoang, T., Zhao, D.: Silent vulnerable dependency alert prediction with vulnerability key aspect explanation. In: Proceedings of the 45th International Conference on Software Engineering. ICSE '23, pp. 970–982. IEEE Press, ??? (2023). <https://doi.org/10.1109/ICSE48619.2023.00089> . <https://doi.org/10.1109/ICSE48619.2023.00089>
- [133] Zhao, J.X., Xie, Y., Kawaguchi, K., He, J., Xie, M.Q.: Automatic Model Selection with Large Language Models for Reasoning. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.14333> . <https://arxiv.org/abs/2305.14333>
- [134] Xu, T., Miao, Y., Fang, C., Qian, H., Feng, X., Chen, Z., Wang, C., Zhang, J., Sun, W., Chen, Z., Liu, Y.: A Prompt Learning Framework for Source Code Summarization. arXiv (2023). <https://doi.org/10.48550/ARXIV.2312.16066> . <https://arxiv.org/abs/2312.16066>
- [135] Sun, Y., Wu, D., Xue, Y., Liu, H., Wang, H., Xu, Z., Xie, X., Liu, Y.: Gptscan: Detecting logic vulnerabilities in smart contracts by combining gpt with program analysis. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. ICSE '24. Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3597503.3639117> . <https://doi.org/10.1145/3597503.3639117>
- [136] Sun, T., Allix, K., Kim, K., Zhou, X., Kim, D., Lo, D., Bissyandé, T.F., Klein, J.: Dexbert: Effective, task-agnostic and fine-grained representation learning of android bytecode. IEEE Transactions on Software Engineering **49**(10), 4691–4706 (2023) <https://doi.org/10.1109/tse.2023.3310874>
- [137] Sun, C., Sheng, Y., Padon, O., Barrett, C.: Clover: Closed-Loop Verifiable Code Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2310.17807> . <https://arxiv.org/abs/2310.17807>
- [138] Sun, W., Fang, C., You, Y., Miao, Y., Liu, Y., Li, Y., Deng, G., Huang, S., Chen, Y., Zhang, Q., Qian, H., Liu, Y., Chen, Z.: Automatic Code Summarization via ChatGPT: How Far Are We? arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.12865> . <https://arxiv.org/abs/2305.12865>

- [139] Steenhoek, B., Tufano, M., Sundaresan, N., Svyatkovskiy, A.: Reinforcement Learning from Automatic Feedback for High-Quality Unit Test Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2310.02368> . <https://arxiv.org/abs/2310.02368>
- [140] Sridhara, G., R.H., Mazumdar, S.: ChatGPT: A Study on its Utility for Ubiquitous Software Engineering Tasks. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.16837> . <https://arxiv.org/abs/2305.16837>
- [141] Sobania, D., Briesch, M., Hanna, C., Petke, J.: An Analysis of the Automatic Bug Fixing Performance of ChatGPT. arXiv (2023). <https://doi.org/10.48550/ARXIV.2301.08653> . <https://arxiv.org/abs/2301.08653>
- [142] Singla, A.: Evaluating chatgpt and gpt-4 for visual programming. In: Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 2. ICER 2023, pp. 14–15. ACM, ??? (2023). <https://doi.org/10.1145/3568812.3603474> . <http://dx.doi.org/10.1145/3568812.3603474>
- [143] Silva, A., Fang, S., Monperrus, M.: RepairLLaMA: Efficient Representations and Fine-Tuned Adapters for Program Repair. arXiv (2023). <https://doi.org/10.48550/ARXIV.2312.15698> . <https://arxiv.org/abs/2312.15698>
- [144] Siddiq, M.L., Casey, B., Santos, J.C.S.: Franc: A lightweight framework for high-quality code generation. In: 2024 IEEE International Conference on Source Code Analysis and Manipulation (SCAM), pp. 106–117. IEEE, ??? (2024). <https://doi.org/10.1109/scam63643.2024.00020> . <http://dx.doi.org/10.1109/SCAM63643.2024.00020>
- [145] Siddiq, M.L., Da Silva Santos, J.C., Tanvir, R.H., Ulfat, N., Al Rifat, F., Carvalho Lopes, V.: Using large language models to generate junit tests: An empirical study. In: Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering. EASE 2024, pp. 313–322. ACM, ??? (2024). <https://doi.org/10.1145/3661167.3661216> . <http://dx.doi.org/10.1145/3661167.3661216>
- [146] Shypula, A., Madaan, A., Zeng, Y., Alon, U., Gardner, J., Hashemi, M., Neubig, G., Ranganathan, P., Bastani, O., Yazdanbakhsh, A.: Learning Performance-Improving Code Edits. arXiv (2023). <https://doi.org/10.48550/ARXIV.2302.07867> . <https://arxiv.org/abs/2302.07867>
- [147] Shirafuji, A., Watanobe, Y., Ito, T., Morishita, M., Nakamura, Y., Oda, Y., Suzuki, J.: Exploring the Robustness of Large Language Models for Solving Programming Problems. arXiv (2023). <https://doi.org/10.48550/ARXIV.2306.14583> . <https://arxiv.org/abs/2306.14583>
- [148] Shin, J., Tang, C., Mohati, T., Nayebi, M., Wang, S., Hemmati, H.: Prompt Engineering or Fine-Tuning: An Empirical Assessment of LLMs for Code. arXiv

- (2023). <https://doi.org/10.48550/ARXIV.2310.10508> . <https://arxiv.org/abs/2310.10508>
- [149] Shin, J., Hashtroudi, S., Hemmati, H., Wang, S.: Domain Adaptation for Code Model-based Unit Test Case Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2308.08033> . <https://arxiv.org/abs/2308.08033>
- [150] Shi, E., Wang, Y., Zhang, H., Du, L., Han, S., Zhang, D., Sun, H.: Towards Efficient Fine-tuning of Pre-trained Code Models: An Experimental Study and Beyond. arXiv (2023). <https://doi.org/10.48550/ARXIV.2304.05216> . <https://arxiv.org/abs/2304.05216>
- [151] Shi, E., Zhang, F., Wang, Y., Chen, B., Du, L., Zhang, H., Han, S., Zhang, D., Sun, H.: SoTaNa: The Open-Source Software Development Assistant. arXiv (2023). <https://doi.org/10.48550/ARXIV.2308.13416> . <https://arxiv.org/abs/2308.13416>
- [152] Sheng, Y., Zheng, L., Yuan, B., Li, Z., Ryabinin, M., Fu, D.Y., Xie, Z., Chen, B., Barrett, C., Gonzalez, J.E., Liang, P., Ré, C., Stoica, I., Zhang, C.: FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU. arXiv (2023). <https://doi.org/10.48550/ARXIV.2303.06865> . <https://arxiv.org/abs/2303.06865>
- [153] Shapkin, A., Litvinov, D., Zharov, Y., Bogomolov, E., Galimzyanov, T., Bryksin, T.: Dynamic Retrieval-Augmented Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2312.08976> . <https://arxiv.org/abs/2312.08976>
- [154] Sghaier, O.B., Sahraoui, H.: A multi-step learning approach to assist code review. In: 2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 450–460. IEEE, ??? (2023). <https://doi.org/10.1109/saner56733.2023.00049> . <http://dx.doi.org/10.1109/SANER56733.2023.00049>
- [155] Schroder, M.: AutoScrum: Automating Project Planning Using Large Language Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2306.03197> . <https://arxiv.org/abs/2306.03197>
- [156] Schlag, I., Sukhbaatar, S., Celikyilmaz, A., Yih, W.-t., Weston, J., Schmidhuber, J., Li, X.: Large Language Model Programs. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.05364> . <https://arxiv.org/abs/2305.05364>
- [157] Schäfer, M., Nadi, S., Eghbali, A., Tip, F.: An Empirical Evaluation of Using Large Language Models for Automated Unit Test Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2302.06527> . <https://arxiv.org/abs/2302.06527>
- [158] Schäfer, M., Nadi, S., Eghbali, A., Tip, F.: An Empirical Evaluation of

- Using Large Language Models for Automated Unit Test Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2302.06527> . <https://arxiv.org/abs/2302.06527>
- [159] Sakib, F.A., Khan, S.H., Karim, A.H.M.R.: Extending the Frontier of ChatGPT: Code Generation and Debugging. arXiv (2023). <https://doi.org/10.48550/ARXIV.2307.08260> . <https://arxiv.org/abs/2307.08260>
- [160] Saieva, A., Chakraborty, S., Kaiser, G.: REINFOREST: Reinforcing Semantic Code Similarity for Cross-Lingual Code Search Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.03843> . <https://arxiv.org/abs/2305.03843>
- [161] Sadik, A.R., Ceravola, A., Joublin, F., Patra, J.: Analysis of ChatGPT on Source Code. arXiv (2023). <https://doi.org/10.48550/ARXIV.2306.00597> . <https://arxiv.org/abs/2306.00597>
- [162] Saberi, I., Fard, F., Chen, F.: Utilization of Pre-trained Language Model for Adapter-based Knowledge Transfer in Software Engineering. arXiv (2023). <https://doi.org/10.48550/ARXIV.2307.08540> . <https://arxiv.org/abs/2307.08540>
- [163] Saberi, I., Esmaeili, A., Fard, F., Chen, F.: AdvFusion: Adapter-based Knowledge Transfer for Code Summarization on Code Language Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2307.07854> . <https://arxiv.org/abs/2307.07854>
- [164] Ronanki, K., Cabrero-Daniel, B., Berger, C.: ChatGPT as a tool for User Story Quality Evaluation: Trustworthy Out of the Box? arXiv (2023). <https://doi.org/10.48550/ARXIV.2306.12132> . <https://arxiv.org/abs/2306.12132>
- [165] Ren, X., Ye, X., Zhao, D., Xing, Z., Yang, X.: From Misuse to Mastery: Enhancing Code Generation with Knowledge-Driven AI Chaining. arXiv (2023). <https://doi.org/10.48550/ARXIV.2309.15606> . <https://arxiv.org/abs/2309.15606>
- [166] Rao, N., Tsay, J., Kate, K., Hellendoorn, V.J., Hirzel, M.: AI for Low-Code for AI. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.20015> . <https://arxiv.org/abs/2305.20015>
- [167] Rahmani, S., Naghshzhan, A., Guerrouj, L.: Improving Code Example Recommendations on Informal Documentation Using BERT and Query-Aware LSH: A Comparative Study. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.03017> . <https://arxiv.org/abs/2305.03017>
- [168] Quan, V.L.A., Phat, C.T., Van Nguyen, K., Duy, P.T., Pham, V.-H.: XGV-BERT: Leveraging Contextualized Language Model and Graph Neural Network for Efficient Software Vulnerability Detection. arXiv (2023). <https://doi.org/10.48550/ARXIV.2309.14677> . <https://arxiv.org/abs/2309.14677>

- [169] Qian, C., Liu, W., Liu, H., Chen, N., Dang, Y., Li, J., Yang, C., Chen, W., Su, Y., Cong, X., Xu, J., Li, D., Liu, Z., Sun, M.: ChatDev: Communicative Agents for Software Development. arXiv (2023). <https://doi.org/10.48550/ARXIV.2307.07924> . <https://arxiv.org/abs/2307.07924>
- [170] Qi, M., Huang, Y., Wang, M., Yao, Y., Liu, Z., Gu, B., Clement, C., Sundaresan, N.: SUT: Active Defects Probing for Transcompiler Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2310.14209> . <https://arxiv.org/abs/2310.14209>
- [171] Pudari, R., Ernst, N.A.: From Copilot to Pilot: Towards AI Supported Software Development. arXiv (2023). <https://doi.org/10.48550/ARXIV.2303.04142> . <https://arxiv.org/abs/2303.04142>
- [172] Poudel, A., Lin, J., Cleland-Huang, J.: Leveraging Transformer-based Language Models to Automate Requirements Satisfaction Assessment. arXiv (2023). <https://doi.org/10.48550/ARXIV.2312.04463> . <https://arxiv.org/abs/2312.04463>
- [173] Zhang, C., Zheng, Y., Bai, M., Li, Y., Ma, W., Xie, X., Li, Y., Sun, L., Liu, Y.: How effective are they? exploring large language model based fuzz driver generation. In: Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis. ISSTA 2024, pp. 1223–1235. Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3650212.3680355> . <https://doi.org/10.1145/3650212.3680355>
- [174] Plein, L., Ouédraogo, W.C., Klein, J., Bissyandé, T.F.: Automatic Generation of Test Cases based on Bug Reports: a Feasibility Study with Large Language Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2310.06320> . <https://arxiv.org/abs/2310.06320>
- [175] Piya, S., Sullivan, A.: LLM4TDD: Best Practices for Test Driven Development Using Large Language Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2312.04687> . <https://arxiv.org/abs/2312.04687>
- [176] Pei, K., Bieber, D., Shi, K., Sutton, C., Yin, P.: Can large language models reason about program invariants? JMLR.org (2023). <https://doi.org/0.5555/3618408.3619552>
- [177] Pegolotti, T., Frantar, E., Alistarh, D., Püschel, M.: QIGen: Generating Efficient Kernels for Quantized Inference on Large Language Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2307.03738> . <https://arxiv.org/abs/2307.03738>
- [178] Paul, R., Hossain, M.M., Siddiq, M.L., Hasan, M., Iqbal, A., Santos, J.C.S.: Enhancing Automated Program Repair through Fine-tuning and Prompt Engineering. arXiv (2023). <https://doi.org/10.48550/ARXIV.2304.07840> . <https://arxiv.org/abs/2304.07840>
- [179] Patil, S.G., Zhang, T., Wang, X., Gonzalez, J.E.: Gorilla: Large Language Model

Connected with Massive APIs. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.15334> . <https://arxiv.org/abs/2305.15334>

- [180] Patel, A., Reddy, S., Bahdanau, D., Dasigi, P.: Evaluating In-Context Learning of Libraries for Code Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2311.09635> . <https://arxiv.org/abs/2311.09635>
- [181] Paranjape, B., Lundberg, S., Singh, S., Hajishirzi, H., Zettlemoyer, L., Ribeiro, M.T.: ART: Automatic multi-step reasoning and tool-use for large language models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2303.09014> . <https://arxiv.org/abs/2303.09014>
- [182] Zhang, K., Zhang, H., Li, G., Li, J., Li, Z., Jin, Z.: ToolCoder: Teach Code Generation Models to use API search tools. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.04032> . <https://arxiv.org/abs/2305.04032>
- [183] Pan, R., Ibrahimzada, A.R., Krishna, R., Sankar, D., Wassi, L.P., Merler, M., Sobolev, B., Pavuluri, R., Sinha, S., Jabbarvand, R.: Lost in translation: A study of bugs introduced by large language models while translating code. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. ICSE '24. Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3597503.3639226> . <https://doi.org/10.1145/3597503.3639226>
- [184] Pan, J., Sadé, A., Kim, J., Soriano, E., Sole, G., Flamant, S.: SteloCoder: a Decoder-Only LLM for Multi-Language to Python Code Translation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2310.15539> . <https://arxiv.org/abs/2310.15539>
- [185] Ouyang, S., Zhang, J.M., Harman, M., Wang, M.: An empirical study of the non-determinism of chatgpt in code generation. ACM Trans. Softw. Eng. Methodol. **34**(2) (2025) <https://doi.org/10.1145/3697010>
- [186] Olausson, T.X., Inala, J.P., Wang, C., Gao, J., Solar-Lezama, A.: Is Self-Repair a Silver Bullet for Code Generation? arXiv (2023). <https://doi.org/10.48550/ARXIV.2306.09896> . <https://arxiv.org/abs/2306.09896>
- [187] Ochs, M., Narasimhan, K., Mezini, M.: Evaluating and improving transformers pre-trained on asts for code completion. In: 2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 834–844. IEEE, ??? (2023). <https://doi.org/10.1109/saner56733.2023.00096> . <http://dx.doi.org/10.1109/SANER56733.2023.00096>
- [188] Noever, D.: Can Large Language Models Find And Fix Vulnerable Software? arXiv (2023). <https://doi.org/10.48550/ARXIV.2308.10345> . <https://arxiv.org/abs/2308.10345>

- [189] Nijkamp, E., Hayashi, H., Xiong, C., Savarese, S., Zhou, Y.: CodeGen2: Lessons for Training LLMs on Programming and Natural Languages. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.02309> . <https://arxiv.org/abs/2305.02309>
- [190] Ni, A., Iyer, S., Radev, D., Stoyanov, V., Yih, W.-t., Wang, S.I., Lin, X.V.: LEVER: Learning to Verify Language-to-Code Generation with Execution. arXiv (2023). <https://doi.org/10.48550/ARXIV.2302.08468> . <https://arxiv.org/abs/2302.08468>
- [191] Zhang, Y., Jin, Z., Xing, Y., Li, G.: STEAM: Simulating the InTeractive BEhavior of ProgrAMmers for Automatic Bug Fixing. arXiv (2023). <https://doi.org/10.48550/ARXIV.2308.14460> . <https://arxiv.org/abs/2308.14460>
- [192] Ni, A., Yin, P., Zhao, Y., Riddell, M., Feng, T., Shen, R., Yin, S., Liu, Y., Yavuz, S., Xiong, C., Joty, S., Zhou, Y., Radev, D., Cohan, A.: L2CEval: Evaluating Language-to-Code Generation Capabilities of Large Language Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2309.17446> . <https://arxiv.org/abs/2309.17446>
- [193] Nguyen, P.T., Di Rocco, J., Di Sipio, C., Rubei, R., Di Ruscio, D., Di Penta, M.: Is this Snippet Written by ChatGPT? An Empirical Study with a CodeBERT-Based Classifier. arXiv (2023). <https://doi.org/10.48550/ARXIV.2307.09381> . <https://arxiv.org/abs/2307.09381>
- [194] Nasir, M.U., Earle, S., Togelius, J., James, S., Cleghorn, C.: Llmatic: Neural architecture search via large language models and quality diversity optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference. GECCO '24, pp. 1110–1118. Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3638529.3654017> . <https://doi.org/10.1145/3638529.3654017>
- [195] Nascimento, N., Alencar, P., Cowan, D.: Comparing Software Developers with ChatGPT: An Empirical Investigation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.11837> . <https://arxiv.org/abs/2305.11837>
- [196] Nam, D., Macvean, A., Hellendoorn, V., Vasilescu, B., Myers, B.: Using an LLM to Help With Code Understanding. arXiv (2023). <https://doi.org/10.48550/ARXIV.2307.08177> . <https://arxiv.org/abs/2307.08177>
- [197] Murali, V., Maddila, C., Ahmad, I., Bolin, M., Cheng, D., Ghorbani, N., Fernandez, R., Nagappan, N., Rigby, P.C.: AI-assisted Code Authoring at Scale: Fine-tuning, deploying, and mixed methods evaluation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.12050> . <https://arxiv.org/abs/2305.12050>
- [198] Mukherjee, M., Hellendoorn, V.J.: Skill over scale: The case for medium,

- domain-specific models for se. In: 2025 IEEE/ACM Second International Conference on AI Foundation Models and Software Engineering (Forge), pp. 212–223. IEEE, ??? (2025). <https://doi.org/10.1109/forge66646.2025.00031> . <http://dx.doi.org/10.1109/Forge66646.2025.00031>
- [199] Mu, F., Shi, L., Wang, S., Yu, Z., Zhang, B., Wang, C., Liu, S., Wang, Q.: Clarifygpt: A framework for enhancing llm-based code generation via requirements clarification. Proceedings of the ACM on Software Engineering 1(FSE), 2332–2354 (2024) <https://doi.org/10.1145/3660810>
- [200] Moon, S., Chae, H., Song, Y., Kwon, T., Kang, D., Ong, K.T.-i., Hwang, S.-w., Yeo, J.: Coffee: Boost Your Code LLMs by Fixing Bugs with Feedback. arXiv (2023). <https://doi.org/10.48550/ARXIV.2311.07215> . <https://arxiv.org/abs/2311.07215>
- [201] Zhu, J., Li, L., Yang, L., Ma, X., Zuo, C.: Automating Method Naming with Context-Aware Prompt-Tuning. arXiv (2023). <https://doi.org/10.48550/ARXIV.2303.05771> . <https://arxiv.org/abs/2303.05771>
- [202] Zhang, K., Li, Z., Li, J., Li, G., Jin, Z.: Self-Edit: Fault-Aware Code Editor for Code Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.04087> . <https://arxiv.org/abs/2305.04087>
- [203] Moharil, A., Sharma, A.: Tabasco: A transformer based contextualization toolkit. Science of Computer Programming 230, 102994 (2023) <https://doi.org/10.1016/j.scico.2023.102994>
- [204] Mohajer, M.M., Aleithan, R., Harzevili, N.S., Wei, M., Belle, A.B., Pham, H.V., Wang, S.: SkipAnalyzer: A Tool for Static Code Analysis with Large Language Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2310.18532> . <https://arxiv.org/abs/2310.18532>
- [205] Mastropaoletti, A., Di Penta, M., Bavota, G.: Towards automatically addressing self-admitted technical debt: How far are we? In: 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 585–597. IEEE, ??? (2023). <https://doi.org/10.1109/ase56229.2023.00103> . <http://dx.doi.org/10.1109/ASE56229.2023.00103>
- [206] Mastropaoletti, A., Pascarella, L., Guglielmi, E., Ciniselli, M., Scalabrino, S., Oliveto, R., Bavota, G.: On the robustness of code generation techniques: An empirical study on github copilot. In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), pp. 2149–2160. IEEE, ??? (2023). <https://doi.org/10.1109/icse48619.2023.00181> . <http://dx.doi.org/10.1109/ICSE48619.2023.00181>
- [207] Nguyen, D., Nam, L., Dau, A., Nguyen, A., Nghiem, K., Guo, J., Bui, N.:

- The vault: A comprehensive multilingual dataset for advancing code understanding and generation. In: Findings of the Association for Computational Linguistics: EMNLP 2023, pp. 4763–4788. Association for Computational Linguistics, ??? (2023). <https://doi.org/10.18653/v1/2023.findings-emnlp.316> . <http://dx.doi.org/10.18653/v1/2023.findings-emnlp.316>
- [208] Zhang, T., Irsan, I.C., Thung, F., Lo, D.: Revisiting Sentiment Analysis for Software Engineering in the Era of Large Language Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2310.11113> . <https://arxiv.org/abs/2310.11113>
- [209] Mandal, S., Chethan, A., Janfaza, V., Mahmud, S.M.F., Anderson, T.A., Turek, J., Tithi, J.J., Muzahid, A.: Large Language Models Based Automatic Synthesis of Software Specifications. arXiv (2023). <https://doi.org/10.48550/ARXIV.2304.09181> . <https://arxiv.org/abs/2304.09181>
- [210] Ma, W., Liu, S., Lin, Z., Wang, W., Hu, Q., Liu, Y., Zhang, C., Nie, L., Li, L., Liu, Y.: LMs: Understanding Code Syntax and Semantics for Code Analysis. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.12138> . <https://arxiv.org/abs/2305.12138>
- [211] Luo, Z., Xu, C., Zhao, P., Sun, Q., Geng, X., Hu, W., Tao, C., Ma, J., Lin, Q., Jiang, D.: WizardCoder: Empowering Code Large Language Models with Evol-Instruct. arXiv (2023). <https://doi.org/10.48550/ARXIV.2306.08568> . <https://arxiv.org/abs/2306.08568>
- [212] Luitel, D., Hassani, S., Sabetzadeh, M.: Improving requirements completeness: automated assistance through large language models. Requirements Engineering **29**(1), 73–95 (2024) <https://doi.org/10.1007/s00766-024-00416-3>
- [213] Lu, J., Yu, L., Li, X., Yang, L., Zuo, C.: Llama-reviewer: Advancing code review automation with large language models through parameter-efficient fine-tuning. In: 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE), pp. 647–658. IEEE, ??? (2023). <https://doi.org/10.1109/isre59848.2023.00026> . <http://dx.doi.org/10.1109/ISSRE59848.2023.00026>
- [214] Zhang, C., Liu, H., Zeng, J., Yang, K., Li, Y., Li, H.: Prompt-Enhanced Software Vulnerability Detection Using ChatGPT. arXiv (2023). <https://doi.org/10.48550/ARXIV.2308.12697> . <https://arxiv.org/abs/2308.12697>
- [215] Liu, J., Xia, C.S., Wang, Y., Zhang, L.: Is Your Code Generated by Chat-GPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.01210> . <https://arxiv.org/abs/2305.01210>
- [216] Liu, Z., Chen, C., Wang, J., Chen, M., Wu, B., Tian, Z., Huang, Y., Hu, J., Wang, Q.: Testing the limits: Unusual text inputs generation for mobile app crash detection with large language model. In: Proceedings of

the IEEE/ACM 46th International Conference on Software Engineering. ICSE '24, pp. 1–12. ACM, ??? (2024). <https://doi.org/10.1145/3597503.3639118> . <http://dx.doi.org/10.1145/3597503.3639118>

- [217] Liu, T., Xu, C., McAuley, J.: RepoBench: Benchmarking Repository-Level Code Auto-Completion Systems. arXiv (2023). <https://doi.org/10.48550/ARXIV.2306.03091> . <https://arxiv.org/abs/2306.03091>
- [218] Liu, Y., Le-Cong, T., Widyasari, R., Tantithamthavorn, C., Li, L., Le, X.-B.D., Lo, D.: Refining chatgpt-generated code: Characterizing and mitigating code quality issues. ACM Transactions on Software Engineering and Methodology **33**(5), 1–26 (2024) <https://doi.org/10.1145/3643674>
- [219] Liu, H., Wang, Y., Wei, Z., Xu, Y., Wang, J., Li, H., Ji, R.: Refbert: A two-stage pre-trained framework for automatic rename refactoring. In: Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis. ISSTA '23, pp. 740–752. ACM, ??? (2023). <https://doi.org/10.1145/3597926.3598092> . <http://dx.doi.org/10.1145/3597926.3598092>
- [220] Liu, Z., Tang, Y., Luo, X., Zhou, Y., Zhang, L.F.: No need to lift a finger anymore? assessing the quality of code generation by chatgpt. IEEE Transactions on Software Engineering **50**(6), 1548–1584 (2024) <https://doi.org/10.1109/tse.2024.3392499>
- [221] Liu, C., Bao, X., Zhang, H., Zhang, N., Hu, H., Zhang, X., Yan, M.: Guiding chatgpt for better code generation: An empirical study. In: 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 102–113. IEEE, ??? (2024). <https://doi.org/10.1109/saner60148.2024.00018> . <http://dx.doi.org/10.1109/SANER60148.2024.00018>
- [222] Liu, P., Sun, C., Zheng, Y., Feng, X., Qin, C., Wang, Y., Xu, Z., Li, Z., Di, P., Jiang, Y., Sun, L.: Llm-powered static binary taint analysis. ACM Transactions on Software Engineering and Methodology **34**(3), 1–36 (2025) <https://doi.org/10.1145/3711816>
- [223] Liu, Z., Chen, C., Wang, J., Che, X., Huang, Y., Hu, J., Wang, Q.: Fill in the blank: Context-aware automated text input generation for mobile gui testing. In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), pp. 1355–1367. IEEE, ??? (2023). <https://doi.org/10.1109/icse48619.2023.00119> . <http://dx.doi.org/10.1109/ICSE48619.2023.00119>
- [224] Zhang, Y., Li, G., Jin, Z., Xing, Y.: Neural Program Repair with Program Dependence Analysis and Effective Filter Mechanism. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.09315> . <https://arxiv.org/abs/2305.09315>
- [225] Liu, S., Wu, B., Xie, X., Meng, G., Liu, Y.: Contrabert: Enhancing code pre-trained models via contrastive learning. In: 2023 IEEE/ACM

- 45th International Conference on Software Engineering (ICSE), pp. 2476–2487. IEEE, ??? (2023). <https://doi.org/10.1109/icse48619.2023.00207> . <http://dx.doi.org/10.1109/ICSE48619.2023.00207>
- [226] Liu, J., Tang, X., Li, L., Chen, P., Liu, Y.: Which is a better programming assistant? A comparative study between chatgpt and stack overflow. arXiv (2023). <https://doi.org/10.48550/ARXIV.2308.13851> . <https://arxiv.org/abs/2308.13851>
- [227] Lin, Y.-C., Kumar, A., Chang, N., Zhang, W., Zakir, M., Apte, R., He, H., Wang, C., Jang, J.-S.R.: Novel preprocessing technique for data embedding in engineering code generation using large language model. In: 2024 IEEE LLM Aided Design Workshop (LAD), pp. 1–5. IEEE, ??? (2024). <https://doi.org/10.1109/lad62341.2024.10691715> . <http://dx.doi.org/10.1109/LAD62341.2024.10691715>
- [228] Li, X.-Y., Xue, J.-T., Xie, Z., Li, M.: Think Outside the Code: Brainstorming Boosts Large Language Models in Code Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.10679> . <https://arxiv.org/abs/2305.10679>
- [229] Li, J., Li, G., Li, Y., Jin, Z.: Structured chain-of-thought prompting for code generation. ACM Transactions on Software Engineering and Methodology **34**(2), 1–23 (2025) <https://doi.org/10.1145/3690635>
- [230] Zhang, J., Nie, P., Li, J.J., Gligoric, M.: Multilingual Code Co-Evolution Using Large Language Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2307.14991> . <https://arxiv.org/abs/2307.14991>
- [231] Li, Z., Li, C., Tang, Z., Huang, W., Ge, J., Luo, B., Ng, V., Wang, T., Hu, Y., Zhang, X.: PtM-apirec: Leveraging pre-trained models of source code in api recommendation. ACM Transactions on Software Engineering and Methodology **33**(3), 1–30 (2024) <https://doi.org/10.1145/3632745>
- [232] Li, T.-O., Zong, W., Wang, Y., Tian, H., Wang, Y., Cheung, S.-C., Kramer, J.: Nuances are the key: Unlocking chatgpt to find failure-inducing tests with differential prompting. In: 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 14–26. IEEE, ??? (2023). <https://doi.org/10.1109/ase56229.2023.00089> . <http://dx.doi.org/10.1109/ASE56229.2023.00089>
- [233] Li, Y., Shi, J., Zhang, Z.: A Novel Approach for Rapid Development Based on ChatGPT and Prompt Engineering. arXiv (2023). <https://doi.org/10.48550/ARXIV.2312.13115> . <https://arxiv.org/abs/2312.13115>
- [234] Li, J., Chen, P., Xia, B., Xu, H., Jia, J.: MoTCoder: Elevating Large Language Models with Modular of Thought for Challenging Programming Tasks. arXiv (2023). <https://doi.org/10.48550/ARXIV.2312.15960> . <https://arxiv.org/abs/2312.15960>

- [235] Li, Y., Huo, Y., Jiang, Z., Zhong, R., He, P., Su, Y., Briand, L.C., Lyu, M.R.: Exploring the effectiveness of llms in automated logging statement generation: An empirical study. *IEEE Transactions on Software Engineering* **50**(12), 3188–3207 (2024) <https://doi.org/10.1109/tse.2024.3475375>
- [236] Li, J., Li, G., Li, Y., Jin, Z.: Structured chain-of-thought prompting for code generation. *ACM Transactions on Software Engineering and Methodology* **34**(2), 1–23 (2025) <https://doi.org/10.1145/3690635>
- [237] Li, P., Sun, T., Tang, Q., Yan, H., Wu, Y., Huang, X., Qiu, X.: CodeIE: Large Code Generation Models are Better Few-Shot Information Extractors. *arXiv* (2023). <https://doi.org/10.48550/ARXIV.2305.05711> . <https://arxiv.org/abs/2305.05711>
- [238] Li, J., Li, G., Li, Z., Jin, Z., Hu, X., Zhang, K., Fu, Z.: <scp>codeeditor</scp> : Learning to edit source code with pre-trained models. *ACM Transactions on Software Engineering and Methodology* **32**(6), 1–22 (2023) <https://doi.org/10.1145/3597207>
- [239] Li, Z., Wang, C., Liu, Z., Wang, H., Chen, D., Wang, S., Gao, C.: Cctest: Testing and repairing code completion systems. In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), pp. 1238–1250. IEEE, ??? (2023). <https://doi.org/10.1109/icse48619.2023.00110> . <http://dx.doi.org/10.1109/ICSE48619.2023.00110>
- [240] Zhang, Q., Fang, C., Zhang, T., Yu, B., Sun, W., Chen, Z.: GAMMA: Revisiting Template-based Automated Program Repair via Mask Prediction. *arXiv* (2023). <https://doi.org/10.48550/ARXIV.2309.09308> . <https://arxiv.org/abs/2309.09308>
- [241] Le-Cong, T., Luong, D.-M., Le, X.B.D., Lo, D., Tran, N.-H., Quang-Huy, B., Huynh, Q.-T.: Invalidator: Automated patch correctness assessment via semantic and syntactic reasoning. *IEEE Transactions on Software Engineering*, 1–20 (2023) <https://doi.org/10.1109/tse.2023.3255177>
- [242] Le, H., Chen, H., Saha, A., Gokul, A., Sahoo, D., Joty, S.: CodeChain: Towards Modular Code Generation Through Chain of Self-revisions with Representative Sub-modules. *arXiv* (2023). <https://doi.org/10.48550/ARXIV.2310.08992> . <https://arxiv.org/abs/2310.08992>
- [243] Laskar, M.T.R., Bari, M.S., Rahman, M., Bhuiyan, M.A.H., Joty, S., Huang, J.: A systematic study and comprehensive evaluation of chatgpt on benchmark datasets. In: Findings of the Association for Computational Linguistics: ACL 2023. Association for Computational Linguistics, ??? (2023). <https://doi.org/10.18653/v1/2023.findings-acl.29> . <http://dx.doi.org/10.18653/v1/2023.findings-acl.29>

- [244] Lai, Y., Li, C., Wang, Y., Zhang, T., Zhong, R., Zettlemoyer, L., Yih, S.W.-t., Fried, D., Wang, S., Yu, T.: DS-1000: A Natural and Reliable Benchmark for Data Science Code Generation (2022). <https://doi.org/10.48550/arXiv.2211.11501> . <https://arxiv.org/abs/2211.11501>
- [245] Zhang, T., Irsan, I.C., Thung, F., Lo, D., Sharma, A., Jiang, L.: Evaluating Pre-trained Language Models for Repairing API Misuses. arXiv (2023). <https://doi.org/10.48550/ARXIV.2310.16390> . <https://arxiv.org/abs/2310.16390>
- [246] Kou, B., Chen, S., Wang, Z., Ma, L., Zhang, T.: Do large language models pay similar attention like human programmers when generating code? Proceedings of the ACM on Software Engineering **1**(FSE), 2261–2284 (2024) <https://doi.org/10.1145/3660807>
- [247] Kolthoff, K., Bartelt, C., Ponzetto, S.P.: Data-driven prototyping via natural-language-based gui retrieval. Automated Software Engineering **30**(1) (2023) <https://doi.org/10.1007/s10515-023-00377-x>
- [248] Koide, T., Nakano, H., Chiba, D.: Chatphishdetector: Detecting phishing sites using large language models. IEEE Access **12**, 154381–154400 (2024) <https://doi.org/10.1109/access.2024.3483905>
- [249] Khare, A., Dutta, S., Li, Z., Solko-Breslin, A., Alur, R., Naik, M.: Understanding the effectiveness of large language models in detecting security vulnerabilities. In: 2025 IEEE Conference on Software Testing, Verification and Validation (ICST), pp. 103–114. IEEE, ??? (2025). <https://doi.org/10.1109/icst62969.2025.10988968> . <http://dx.doi.org/10.1109/ICST62969.2025.10988968>
- [250] Khanfir, A., Degiovanni, R., Papadakis, M., Traon, Y.L.: Efficient Mutation Testing via Pre-Trained Language Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2301.03543> . <https://arxiv.org/abs/2301.03543>
- [251] Khan, M.A.M., Bari, M.S., Long, D., Wang, W., Parvez, M.R., Joty, S.: Xcodeeval: An execution-based large scale multilingual multitask benchmark for code understanding, generation, translation and retrieval. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 6766–6805. Association for Computational Linguistics, ??? (2024). <https://doi.org/10.18653/v1/2024.acl-long.367> . <http://dx.doi.org/10.18653/v1/2024.acl-long.367>
- [252] Khan, M.F.A., Ramsdell, M., Falor, E., Karimi, H.: Assessing the Promise and Pitfalls of ChatGPT for Automated Code Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2311.02640> . <https://arxiv.org/abs/2311.02640>
- [253] Zhang, T., Irsan, I.C., Thung, F., Lo, D.: CUPID: Leveraging ChatGPT for More Accurate Duplicate Bug Report Detection. arXiv (2023). <https://doi.org/10.48550/ARXIV.2308.10022> . <https://arxiv.org/abs/2308.10022>

- [254] Khakhar, A., Mell, S., Bastani, O.: PAC Prediction Sets for Large Language Models of Code (2023). <https://arxiv.org/abs/2302.08703>
- [255] Li, K., Hong, S., Fu, C., Zhang, Y., Liu, M.: Discriminating human-authored from chatgpt-generated code via discernable feature analysis. In: 2023 IEEE 34th International Symposium on Software Reliability Engineering Workshops (ISSREW), pp. 120–127. IEEE, ??? (2023). <https://doi.org/10.1109/issrew60843.2023.00059> . <http://dx.doi.org/10.1109/ISSREW60843.2023.00059>
- [256] Kannan, J.: Can LLMs Configure Software Tools. arXiv (2023). <https://doi.org/10.48550/ARXIV.2312.06121> . <https://arxiv.org/abs/2312.06121>
- [257] Kang, S., An, G., Yoo, S.: A quantitative and qualitative evaluation of llm-based explainable fault localization. Proceedings of the ACM on Software Engineering 1(FSE), 1424–1446 (2024) <https://doi.org/10.1145/3660771>
- [258] Kang, S., Chen, B., Yoo, S., Lou, J.-G.: Explainable automated debugging via large language model-driven scientific debugging. Empirical Software Engineering 30(2) (2024) <https://doi.org/10.1007/s10664-024-10594-x>
- [259] Kang, S., Yoon, J., Askarbekkyzy, N., Yoo, S.: Evaluating diverse large language models for automatic and general bug reproduction. IEEE Transactions on Software Engineering 50(10), 2677–2694 (2024) <https://doi.org/10.1109/tse.2024.3450837>
- [260] Zhang, T., Yu, T., Hashimoto, T.B., Lewis, M., Yih, W.-t., Fried, D., Wang, S.I.: Coder Reviewer Reranking for Code Generation. arXiv (2022). <https://doi.org/10.48550/ARXIV.2211.16490> . <https://arxiv.org/abs/2211.16490>
- [261] Kabir, M.M.A., Hassan, S.A., Wang, X., Wang, Y., Yu, H., Meng, N.: An empirical study of ChatGPT-3.5 on question answering and code maintenance. arXiv (2023). <https://doi.org/10.48550/ARXIV.2310.02104> . <https://arxiv.org/abs/2310.02104>
- [262] Jin, M., Shahriar, S., Tufano, M., Shi, X., Lu, S., Sundaresan, N., Svyatkovskiy, A.: Inferfix: End-to-end program repair with llms. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE ’23, pp. 1646–1656. ACM, ??? (2023). <https://doi.org/10.1145/3611643.3613892> . <http://dx.doi.org/10.1145/3611643.3613892>
- [263] Jin, X., Larson, J., Yang, W., Lin, Z.: Binary Code Summarization: Benchmarking ChatGPT/GPT-4 and Other Large Language Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2312.09601> . <https://arxiv.org/abs/2312.09601>
- [264] Jin, P., Zhang, S., Ma, M., Li, H., Kang, Y., Li, L., Liu, Y., Qiao, B., Zhang, C., Zhao, P., He, S., Sarro, F., Dang, Y., Rajmohan, S., Lin, Q., Zhang, D.: Assess

- and summarize: Improve outage understanding with large language models. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE '23, pp. 1657–1668. ACM, ??? (2023). <https://doi.org/10.1145/3611643.3613891> . <http://dx.doi.org/10.1145/3611643.3613891>
- [265] Jimenez, C.E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., Narasimhan, K.: SWE-bench: Can Language Models Resolve Real-World GitHub Issues? arXiv (2023). <https://doi.org/10.48550/ARXIV.2310.06770> . <https://arxiv.org/abs/2310.06770>
- [266] Jiang, S., Wang, Y., Wang, Y.: SelfEvolve: A Code Evolution Framework via Large Language Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2306.02907> . <https://arxiv.org/abs/2306.02907>
- [267] Jiang, X., Dong, Y., Wang, L., Fang, Z., Shang, Q., Li, G., Jin, Z., Jiao, W.: Self-planning code generation with large language models. ACM Transactions on Software Engineering and Methodology **33**(7), 1–30 (2024) <https://doi.org/10.1145/3672456>
- [268] Jiang, N., Wang, C., Liu, K., Xu, X., Tan, L., Zhang, X., Babkin, P.: Nova: Generative Language Models for Assembly Code with Hierarchical Attention and Contrastive Learning. arXiv (2023). <https://doi.org/10.48550/ARXIV.2311.13721> . <https://arxiv.org/abs/2311.13721>
- [269] Jiang, N., Liu, K., Lutellier, T., Tan, L.: Impact of code language models on automated program repair. In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), pp. 1430–1442. IEEE, ??? (2023). <https://doi.org/10.1109/icse48619.2023.00125> . <http://dx.doi.org/10.1109/ICSE48619.2023.00125>
- [270] Zhang, Q., Fang, C., Sun, W., Liu, Y., He, T., Hao, X., Chen, Z.: APPT: Boosting Automated Patch Correctness Prediction via Fine-tuning Pre-trained Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2301.12453> . <https://arxiv.org/abs/2301.12453>
- [271] Ji, Z., Ma, P., Li, Z., Wang, S.: Benchmarking and Explaining Large Language Model-based Code Generation: A Causality-Centric Approach. arXiv (2023). <https://doi.org/10.48550/ARXIV.2310.06680> . <https://arxiv.org/abs/2310.06680>
- [272] Jana, P., Jha, P., Ju, H., Kishore, G., Mahajan, A., Ganesh, V.: CoTran: An LLM-Based Code Translator Using Reinforcement Learning with Feedback from Compiler and Symbolic Execution, pp. 4011–4018. IOS Press, ??? (2024). <https://doi.org/10.3233/faia240968> . <http://dx.doi.org/10.3233/FAIA240968>
- [273] Jain, N., Zhang, T., Chiang, W.-L., Gonzalez, J.E., Sen, K., Stoica, I.:

- LLM-Assisted Code Cleaning For Training Accurate Code Generators. arXiv (2023). <https://doi.org/10.48550/ARXIV.2311.14904> . <https://arxiv.org/abs/2311.14904>
- [274] Jain, A., Adiole, C., Chaudhuri, S., Reps, T., Jermaine, C.: Coarse-Tuning Models of Code with Reinforcement Learning Feedback. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.18341> . <https://arxiv.org/abs/2305.18341>
- [275] Zhang, K., Wang, D., Xia, J., Wang, W.Y., Li, L.: ALGO: Synthesizing Algorithmic Programs with LLM-Generated Oracle Verifiers. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.14591> . <https://arxiv.org/abs/2305.14591>
- [276] Ibrahimzada, A.R., Chen, Y., Rong, R., Jabbarvand, R.: Challenging bug prediction and repair models with synthetic bugs. In: 2025 IEEE International Conference on Source Code Analysis & Manipulation (SCAM), pp. 133–144. IEEE, ??? (2025). <https://doi.org/10.1109/scam67354.2025.00021> . <http://dx.doi.org/10.1109/SCAM67354.2025.00021>
- [277] Huang, D., Bu, Q., Qing, Y., Cui, H.: CodeCoT: Tackling Code Syntax Errors in CoT Reasoning for Code Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2308.08784> . <https://arxiv.org/abs/2308.08784>
- [278] Huang, Q., Zhu, J., Li, Z., Xing, Z., Wang, C., Xu, X.: Pcr-chain: Partial code reuse assisted by hierarchical chaining of prompts on frozen copilot. In: 2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), pp. 1–5. IEEE, ??? (2023). <https://doi.org/10.1109/icse-companion58688.2023.00013> . <http://dx.doi.org/10.1109/ICSE-Companion58688.2023.00013>
- [279] Huang, Q., Sun, Y., Xing, Z., Yu, M., Xu, X., Lu, Q.: Api entity and relation joint extraction from text via dynamic prompt-tuned language model. ACM Transactions on Software Engineering and Methodology **33**(1), 1–25 (2023) <https://doi.org/10.1145/3607188>
- [280] Huang, D., Nan, Z., Hu, X., Jin, P., Peng, S., Wen, Y., Zhang, R., Du, Z., Guo, Q., Pu, Y., Chen, Y.: ANPL: Towards Natural Programming with Interactive Decomposition (2023). <https://arxiv.org/abs/2305.18498>
- [281] Huang, Q., Zou, Z., Xing, Z., Zuo, Z., Xu, X., Lu, Q.: AI Chain on Large Language Model for Unsupervised Control Flow Graph Generation for Statically-Typed Partial Code. arXiv (2023). <https://doi.org/10.48550/ARXIV.2306.00757> . <https://arxiv.org/abs/2306.00757>
- [282] Huang, D., Zhang, J.M., Luck, M., Bu, Q., Qing, Y., Cui, H.: AgentCoder: Multi-Agent-based Code Generation with Iterative Testing and Optimisation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2312.13010> . <https://arxiv.org/abs/2312.13010>

- [283] Huang, Q., Wu, Y., Xing, Z., Jiang, H., Cheng, Y., Jin, H.: Adaptive Intellect Unleashed: The Feasibility of Knowledge Transfer in Large Language Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2308.04788> . <https://arxiv.org/abs/2308.04788>
- [284] Hu, J., Zhang, Q., Yin, H.: Augmenting Greybox Fuzzing with Generative AI. arXiv (2023). <https://doi.org/10.48550/ARXIV.2306.06782> . <https://arxiv.org/abs/2306.06782>
- [285] Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Zhang, C., Wang, J., Wang, Z., Yau, S.K.S., Lin, Z., Zhou, L., Ran, C., Xiao, L., Wu, C., Schmidhuber, J.: MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. arXiv (2023). <https://doi.org/10.48550/ARXIV.2308.00352> . <https://arxiv.org/abs/2308.00352>
- [286] Helmeczi, R.K., Cevik, M., Yildirim, S.: Few-shot learning for sentence pair classification and its applications in software engineering (2023). <https://arxiv.org/abs/2306.08058>
- [287] He, J., Zhou, X., Xu, B., Zhang, T., Kim, K., Yang, Z., Thung, F., Irsan, I.C., Lo, D.: Representation learning for stack overflow posts: How far are we? ACM Transactions on Software Engineering and Methodology **33**(3), 1–24 (2024) <https://doi.org/10.1145/3635711>
- [288] Hao, Y., Chen, W., Zhou, Z., Cui, W.: E&V: Prompting Large Language Models to Perform Static Analysis by Pseudo-code Execution and Verification. arXiv (2023). <https://doi.org/10.48550/ARXIV.2312.08477> . <https://arxiv.org/abs/2312.08477>
- [289] Hajali, P., Budvytis, I.: Function-constrained Program Synthesis. arXiv (2023). <https://doi.org/10.48550/ARXIV.2311.15500> . <https://arxiv.org/abs/2311.15500>
- [290] Gupta, P., Khare, A., Bajpai, Y., Chakraborty, S., Gulwani, S., Kanade, A., Radhakrishna, A., Soares, G., Tiwari, A.: Grace: Language models meet code edits. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE ’23, pp. 1483–1495. ACM, ??? (2023). <https://doi.org/10.1145/3611643.3616253> . <http://dx.doi.org/10.1145/3611643.3616253>
- [291] Guo, D., Xu, C., Duan, N., Yin, J., McAuley, J.: LongCoder: A Long-Range Pre-trained Language Model for Code Completion (2023). <https://arxiv.org/abs/2306.14893>
- [292] Grishina, A., Hort, M., Moonen, L.: The earlybird catches the bug: On exploiting early layers of encoder models for more efficient code classification. In:

Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE '23, pp. 895–907. ACM, ??? (2023). <https://doi.org/10.1145/3611643.3616304> . <http://dx.doi.org/10.1145/3611643.3616304>

- [293] Gong, L., Zhang, J., Wei, M., Zhang, H., Huang, Z.: What is the intended usage context of this model? an exploratory study of pre-trained models on various model repositories. *ACM Transactions on Software Engineering and Methodology* **32**(3), 1–57 (2023) <https://doi.org/10.1145/3569934>
- [294] Gomes, L., Silva Torres, R., Córtes, M.L.: Bert- and tf-idf-based feature extraction for long-lived bug prediction in floss: A comparative study. *Information and Software Technology* **160**, 107217 (2023) <https://doi.org/10.1016/j.infsof.2023.107217>
- [295] Gilbert, H., Sandborn, M., Schmidt, D.C., Spencer-Smith, J., White, J.: Semantic compression with large language models. In: 2023 Tenth International Conference on Social Networks Analysis, Management and Security (SNAMS). IEEE, ??? (2023). <https://doi.org/10.1109/snams60348.2023.10375400> . <http://dx.doi.org/10.1109/SNAMS60348.2023.10375400>
- [296] Gao, Z., Wang, H., Zhou, Y., Zhu, W., Zhang, C.: How Far Have We Gone in Vulnerability Detection Using Large Language Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2311.12420> . <https://arxiv.org/abs/2311.12420>
- [297] Gao, S., Wen, X.-C., Gao, C., Wang, W., Zhang, H., Lyu, M.R.: What makes good in-context demonstrations for code intelligence tasks with llms? In: 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 761–773. IEEE, ??? (2023). <https://doi.org/10.1109/ase56229.2023.00109> . <http://dx.doi.org/10.1109/ASE56229.2023.00109>
- [298] Gandhi, A., Nguyen, T.Q., Jiao, H., Steen, R., Bhatawdekar, A.: Natural Language Commanding via Program Synthesis. arXiv (2023). <https://doi.org/10.48550/ARXIV.2306.03460> . <https://arxiv.org/abs/2306.03460>
- [299] First, E., Rabe, M.N., Ringer, T., Brun, Y.: Baldur: Whole-proof generation and repair with large language models. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE '23, pp. 1229–1241. ACM, ??? (2023). <https://doi.org/10.1145/3611643.3616243> . <http://dx.doi.org/10.1145/3611643.3616243>
- [300] Feng, S., Chen, C.: Prompting is all you need: Automated android bug replay with large language models. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. ICSE '24, pp. 1–13. ACM, ??? (2024). <https://doi.org/10.1145/3597503.3608137> . <http://dx.doi.org/10.1145/3597503.3608137>

- [301] Fakhouri, S., Chakraborty, S., Musuvathi, M., Lahiri, S.K.: Nl2fix: Generating functionally correct code edits from bug descriptions. In: Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings. ICSE-Companion '24, pp. 410–411. ACM, ??? (2024). <https://doi.org/10.1145/3639478.3643526> . <http://dx.doi.org/10.1145/3639478.3643526>
- [302] Endres, M., Fakhouri, S., Chakraborty, S., Lahiri, S.K.: Can large language models transform natural language intent into formal method postconditions? Proceedings of the ACM on Software Engineering **1**(FSE), 1889–1912 (2024) <https://doi.org/10.1145/3660791>
- [303] El-Hajjami, A., Fafin, N., Salinesi, C.: Which AI Technique Is Better to Classify Requirements? An Experiment with SVM, LSTM, and ChatGPT. arXiv (2023). <https://doi.org/10.48550/ARXIV.2311.11547> . <https://arxiv.org/abs/2311.11547>
- [304] Du, X., Liu, M., Wang, H., Li, J., Peng, X., Lou, Y.: Exploring Large Language Models in Resolving Environment-Related Crash Bugs: Localizing and Repairing. arXiv (2023). <https://doi.org/10.48550/ARXIV.2312.10448> . <https://arxiv.org/abs/2312.10448>
- [305] Du, Y., Yu, Z.: Pre-training code representation with semantic flow graph for effective bug localization. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE '23, pp. 579–591. ACM, ??? (2023). <https://doi.org/10.1145/3611643.3616338> . <http://dx.doi.org/10.1145/3611643.3616338>
- [306] Du, X., Liu, M., Wang, K., Wang, H., Liu, J., Chen, Y., Feng, J., Sha, C., Peng, X., Lou, Y.: ClassEval: A Manually-Crafted Benchmark for Evaluating LLMs on Class-level Code Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2308.01861> . <https://arxiv.org/abs/2308.01861>
- [307] Dou, S., Shan, J., Jia, H., Deng, W., Xi, Z., He, W., Wu, Y., Gui, T., Liu, Y., Huang, X.: Towards Understanding the Capability of Large Language Models on Code Clone Detection: A Survey. arXiv (2023). <https://doi.org/10.48550/ARXIV.2308.01191> . <https://arxiv.org/abs/2308.01191>
- [308] Dong, Y., Jiang, X., Jin, Z., Li, G.: Self-collaboration code generation via chatgpt. ACM Transactions on Software Engineering and Methodology **33**(7), 1–38 (2024) <https://doi.org/10.1145/3672459>
- [309] Dong, Y., Ding, J., Jiang, X., Li, G., Li, Z., Jin, Z.: Codescore: Evaluating code generation by learning code execution. ACM Transactions on Software Engineering and Methodology **34**(3), 1–22 (2025) <https://doi.org/10.1145/3695991>
- [310] Dong, G., Yuan, H., Lu, K., Li, C., Xue, M., Liu, D., Wang, W., Yuan, Z., Zhou,

- C., Zhou, J.: How abilities in large language models are affected by supervised fine-tuning data composition. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 177–198. Association for Computational Linguistics, ??? (2024). <https://doi.org/10.18653/v1/2024.acl-long.12> . <http://dx.doi.org/10.18653/v1/2024.acl-long.12>
- [311] Ding, H., Kumar, V., Tian, Y., Wang, Z., Kwiatkowski, R., Li, X., Ramanathan, M.K., Ray, B., Bhatia, P., Sengupta, S.: A static evaluation of code completion by large language models. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track), pp. 347–360. Association for Computational Linguistics, ??? (2023). <https://doi.org/10.18653/v1/2023.acl-industry.34> . <http://dx.doi.org/10.18653/v1/2023.acl-industry.34>
- [312] Deng, Y., Xia, C.S., Peng, H., Yang, C., Zhang, L.: Large language models are zero-shot fuzzers: Fuzzing deep-learning libraries via large language models. In: Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis. ISSTA ’23, pp. 423–435. ACM, ??? (2023). <https://doi.org/10.1145/3597926.3598067> . <http://dx.doi.org/10.1145/3597926.3598067>
- [313] Deng, Y., Xia, C.S., Yang, C., Zhang, S.D., Yang, S., Zhang, L.: Large language models are edge-case generators: Crafting unusual programs for fuzzing deep learning libraries. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. ICSE ’24, pp. 1–13. ACM, ??? (2024). <https://doi.org/10.1145/3597503.3623343> . <http://dx.doi.org/10.1145/3597503.3623343>
- [314] Zelikman, E., Lorch, E., Mackey, L., Kalai, A.T.: Self-Taught Optimizer (STOP): Recursively Self-Improving Code Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2310.02304> . <https://arxiv.org/abs/2310.02304>
- [315] Deligiannis, P., Lal, A., Mehrotra, N., Poddar, R., Rastogi, A.: Rustassistant: Using llms to fix compilation errors in rust code. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 3097–3109. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00022> . <http://dx.doi.org/10.1109/ICSE55347.2025.00022>
- [316] Dakhel, A.M., Nikanjam, A., Majdinasab, V., Khomh, F., Desmarais, M.C.: Effective test generation using pre-trained large language models and mutation testing. Information and Software Technology **171**, 107468 (2024) <https://doi.org/10.1016/j.infsof.2024.107468>
- [317] Ciborowska, A., Damevski, K.: Too Few Bug Reports? Exploring Data Augmentation for Improved Changeset-based Bug Localization. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.16430> . <https://arxiv.org/abs/2305.16430>
- [318] Cheng, L., Li, X., Bing, L.: Is GPT-4 a Good Data Analyst? arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.15038> . <https://arxiv.org/abs/2305.15038>

- [319] Chen, X., Lin, M., Schärli, N., Zhou, D.: Teaching Large Language Models to Self-Debug. arXiv (2023). <https://doi.org/10.48550/ARXIV.2304.05128> . <https://arxiv.org/abs/2304.05128>
- [320] Zan, D., Chen, B., Gong, Y., Cao, J., Zhang, F., Wu, B., Guan, B., Yin, Y., Wang, Y.: Private-Library-Oriented Code Generation with Large Language Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2307.15370> . <https://arxiv.org/abs/2307.15370>
- [321] Chen, A., Scheurer, J., Korbak, T., Campos, J.A., Chan, J.S., Bowman, S.R., Cho, K., Perez, E.: Improving Code Generation by Training with Natural Language Feedback. arXiv (2023). <https://doi.org/10.48550/ARXIV.2303.16749> . <https://arxiv.org/abs/2303.16749>
- [322] Gu, X., Chen, M., Lin, Y., Hu, Y., Zhang, H., Wan, C., Wei, Z., Xu, Y., Wang, J.: On the effectiveness of large language models in domain-specific code generation. ACM Transactions on Software Engineering and Methodology **34**(3), 1–22 (2025) <https://doi.org/10.1145/3697012>
- [323] Chen, Y., Ding, Z., Alowain, L., Chen, X., Wagner, D.: Diversevul: A new vulnerable source code dataset for deep learning based vulnerability detection. In: Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses. RAID 2023, pp. 654–668. ACM, ??? (2023). <https://doi.org/10.1145/3607199.3607242> . <http://dx.doi.org/10.1145/3607199.3607242>
- [324] Tihanyi, N., Jain, R., Charalambous, Y., Ferrag, M.A., Sun, Y., Cordeiro, L.C.: A New Era in Software Security: Towards Self-Healing Software via Large Language Models and Formal Verification (2024). <https://arxiv.org/abs/2305.14752>
- [325] Chan, A., Kharkar, A., Moghaddam, R.Z., Mohylevskyy, Y., Helyar, A., Kamal, E., Elkamhawy, M., Sundaresan, N.: Transformer-based Vulnerability Detection in Code at EditTime: Zero-shot, Few-shot, or Fine-tuning? arXiv (2023). <https://doi.org/10.48550/ARXIV.2306.01754> . <https://arxiv.org/abs/2306.01754>
- [326] Cassano, F., Gouwar, J., Nguyen, D., Nguyen, S., Phipps-Costin, L., Pinckney, D., Yee, M.-H., Zi, Y., Anderson, C.J., Feldman, M.Q., Guha, A., Greenberg, M., Jangda, A.: Multipl-e: A scalable and polyglot approach to benchmarking neural code generation. IEEE Transactions on Software Engineering **49**(7), 3675–3691 (2023) <https://doi.org/10.1109/tse.2023.3267446>
- [327] Cao, J., Li, M., Wen, M., Cheung, S.-C.: A study on prompt design, advantages and limitations of chatgpt for deep learning program repair. Automated Software Engineering **32**(1) (2025) <https://doi.org/10.1007/s10515-025-00492-x>
- [328] Buscemi, A.: A Comparative Study of Code Generation using ChatGPT 3.5 across 10 Programming Languages. arXiv (2023). <https://doi.org/10.48550/>

ARXIV.2308.04477 . <https://arxiv.org/abs/2308.04477>

- [329] Bui, N.D.Q., Le, H., Wang, Y., Li, J., Gotmare, A.D., Hoi, S.C.H.: CodeTF: One-stop Transformer Library for State-of-the-art Code LLM. arXiv (2023). <https://doi.org/10.48550/ARXIV.2306.00029> . <https://arxiv.org/abs/2306.00029>
- [330] Bairi, R., Sonwane, A., Kanade, A., C., V.D., Iyer, A., Parthasarathy, S., Rajamani, S., Ashok, B., Shet, S.: Codeplan: Repository-level coding using llms and planning. Proceedings of the ACM on Software Engineering 1(FSE), 675–698 (2024) <https://doi.org/10.1145/3643757>
- [331] Azaria, A., Azoulay, R., Reches, S.: Chatgpt is a remarkable tool—for experts. Data Intelligence 6(1), 240–296 (2024) [https://doi.org/10.1162/dint\\_a\\_00235](https://doi.org/10.1162/dint_a_00235)
- [332] Arakelyan, S., Das, R.J., Mao, Y., Ren, X.: Exploring Distributional Shifts in Large Language Models for Code Analysis. arXiv (2023). <https://doi.org/10.48550/ARXIV.2303.09128> . <https://arxiv.org/abs/2303.09128>
- [333] Alam, A.I., Roy, P.R., Al-Omari, F., Roy, C.K., Roy, B., Schneider, K.A.: Gptclonebench: A comprehensive benchmark of semantic clones and cross-language clones using gpt-3 model and semanticclonebench. In: 2023 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 1–13. IEEE, ??? (2023). <https://doi.org/10.1109/icsme58846.2023.00013> . <http://dx.doi.org/10.1109/ICSME58846.2023.00013>
- [334] Al-Kaswan, A., Ahmed, T., Izadi, M., Sawant, A.A., Devanbu, P., Deursen, A.: Extending source code pre-trained language models to summarise decompiled binaries. In: 2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 260–271. IEEE, ??? (2023). <https://doi.org/10.1109/saner56733.2023.00033> . <http://dx.doi.org/10.1109/SANER56733.2023.00033>
- [335] Xia, C.S., Zhang, L.: Conversational Automated Program Repair. arXiv (2023). <https://doi.org/10.48550/ARXIV.2301.13246> . <https://arxiv.org/abs/2301.13246>
- [336] Mao, Y., Wan, C., Jiang, Y., Gu, X.: Self-supervised query reformulation for code search. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE '23, pp. 363–374. ACM, ??? (2023). <https://doi.org/10.1145/3611643.3616306> . <http://dx.doi.org/10.1145/3611643.3616306>
- [337] Kou, B., Chen, M., Zhang, T.: Automated summarization of stack overflow posts. In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), pp. 1853–1865. IEEE, ??? (2023). <https://doi.org/10.1109/icse48619.2023.00158> . <http://dx.doi.org/10.1109/ICSE48619.2023.00158>

- [338] Zhou, W., Zhang, S., Gu, Y., Chen, M., Poon, H.: UniversalNER: Targeted Distillation from Large Language Models for Open Named Entity Recognition. arXiv (2023). <https://doi.org/10.48550/ARXIV.2308.03279> . <https://arxiv.org/abs/2308.03279>
- [339] Yuan, Z., Lou, Y., Liu, M., Ding, S., Wang, K., Chen, Y., Peng, X.: No More Manual Tests? Evaluating and Improving ChatGPT for Unit Test Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.04207> . <https://arxiv.org/abs/2305.04207>
- [340] Yu, S., Wu, Y., Li, Z., He, P., Chen, N., Liu, C.: Log parsing with generalization ability under new log types. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE '23, pp. 425–437. ACM, ??? (2023). <https://doi.org/10.1145/3611643.3616355> . <http://dx.doi.org/10.1145/3611643.3616355>
- [341] Yu, H., Shen, B., Ran, D., Zhang, J., Zhang, Q., Ma, Y., Liang, G., Li, Y., Wang, Q., Xie, T.: Codereval: A benchmark of pragmatic code generation with generative pre-trained models. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. ICSE '24. Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3597503.3623316> . <https://doi.org/10.1145/3597503.3623316>
- [342] Yoon, J., Feldt, R., Yoo, S.: Autonomous Large Language Model Agents Enabling Intent-Driven Mobile GUI Testing. arXiv (2023). <https://doi.org/10.48550/ARXIV.2311.08649> . <https://arxiv.org/abs/2311.08649>
- [343] Yetişiren, B., Özsoy, I., Ayerdem, M., Tüzün, E.: Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT. arXiv (2023). <https://doi.org/10.48550/ARXIV.2304.10778> . <https://arxiv.org/abs/2304.10778>
- [344] Yen, R., Zhu, J., Suh, S., Xia, H., Zhao, J.: CoLadder: Supporting Programmers with Hierarchical Code Generation in Multi-Level Abstraction. arXiv (2023). <https://doi.org/10.48550/ARXIV.2310.08699> . <https://arxiv.org/abs/2310.08699>
- [345] Ye, J., Li, C., Kong, L., Yu, T.: Generating Data for Symbolic Language with Large Language Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.13917> . <https://arxiv.org/abs/2305.13917>
- [346] Yang, C., Deng, Y., Lu, R., Yao, J., Liu, J., Jabbarvand, R., Zhang, L.: Whitefox: White-box compiler fuzzing empowered by large language models. Proc. ACM Program. Lang. **8**(OOPSLA2) (2024) <https://doi.org/10.1145/3689736>
- [347] Zhou, Y., Muresanu, A.I., Han, Z., Paster, K., Pitis, S., Chan, H., Ba, J.: Large Language Models Are Human-Level Prompt Engineers. arXiv (2022). <https://arxiv.org/abs/2206.05405>

//doi.org/10.48550/ARXIV.2211.01910 . https://arxiv.org/abs/2211.01910

- [348] Yang, G., Zhou, Y., Chen, X., Zhang, X., Xu, Y., Han, T., Chen, T.: A Syntax-Guided Multi-Task Learning Approach for Turducken-Style Code Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2303.05061> . <https://arxiv.org/abs/2303.05061>
- [349] Yang, G., Zhou, Y., Zhang, X., Chen, X., Han, T., Chen, T.: Assessing and Improving Syntactic Adversarial Robustness of Pre-trained Models for Code Translation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2310.18587> . <https://arxiv.org/abs/2310.18587>
- [350] Yang, C., Liu, J., Xu, B., Treude, C., Lyu, Y., He, J., Li, M., Lo, D.: API-DocBooster: An Extract-Then-Abstract Framework Leveraging Large Language Models for Augmenting API Documentation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2312.10934> . <https://arxiv.org/abs/2312.10934>
- [351] Yan, W., Tian, Y., Li, Y., Chen, Q., Wang, W.: CodeTransOcean: A Comprehensive Multilingual Benchmark for Code Translation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2310.04951> . <https://arxiv.org/abs/2310.04951>
- [352] Yan, D., Gao, Z., Liu, Z.: A closer look at different difficulty levels code generation abilities of chatgpt. In: 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 1887–1898. IEEE, ??? (2023). <https://doi.org/10.1109/ase56229.2023.00096> . <http://dx.doi.org/10.1109/ASE56229.2023.00096>
- [353] Xu, X., Zhang, Z., Su, Z., Huang, Z., Feng, S., Ye, Y., Jiang, N., Xie, D., Cheng, S., Tan, L., Zhang, X.: Symbol Preference Aware Generative Models for Recovering Variable Names from Stripped Binary. arXiv (2023). <https://doi.org/10.48550/ARXIV.2306.02546> . <https://arxiv.org/abs/2306.02546>
- [354] Xu, Z., Li, Q., Tan, S.H.: Guiding ChatGPT to Fix Web UI Tests via Explanation-Consistency Checking. arXiv (2023). <https://doi.org/10.48550/ARXIV.2312.05778> . <https://arxiv.org/abs/2312.05778>
- [355] Zhou, S., Alon, U., Agarwal, S., Neubig, G.: CodeBERTScore: Evaluating Code Generation with Pretrained Models of Code. arXiv (2023). <https://doi.org/10.48550/ARXIV.2302.05527> . <https://arxiv.org/abs/2302.05527>
- [356] Xiong, W., Guo, Y., Chen, H.: The Program Testing Ability of Large Language Models for Code. arXiv (2023). <https://doi.org/10.48550/ARXIV.2310.05727> . <https://arxiv.org/abs/2310.05727>
- [357] Xie, D., Yoo, B., Jiang, N., Kim, M., Tan, L., Zhang, X., Lee, J.S.: How Effective are Large Language Models in Generating Software Specifications? arXiv (2023). <https://doi.org/10.48550/ARXIV.2306.03324> . <https://arxiv.org/abs/2306.03324>

2306.03324

- [358] Chen, Y., Hu, Z., Zhi, C., Han, J., Deng, S., Yin, J.: ChatUniTest: A Framework for LLM-Based Test Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.04764> . <https://arxiv.org/abs/2305.04764>
- [359] Xia, C.S., Zhang, L.: Keep the Conversation Going: Fixing 162 out of 337 bugs for \$0.42 each using ChatGPT. arXiv (2023). <https://doi.org/10.48550/ARXIV.2304.00385> . <https://arxiv.org/abs/2304.00385>
- [360] Wu, Y., Li, Z., Zhang, J.M., Papadakis, M., Harman, M., Liu, Y.: Large Language Models in Fault Localisation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2308.15276> . <https://arxiv.org/abs/2308.15276>
- [361] Wu, Y., Jiang, N., Pham, H.V., Lutellier, T., Davis, J., Tan, L., Babkin, P., Shah, S.: How effective are neural networks for fixing security vulnerabilities. In: Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis. ISSTA 2023, pp. 1282–1294. Association for Computing Machinery, New York, NY, USA (2023). <https://doi.org/10.1145/3597926.3598135> . <https://doi.org/10.1145/3597926.3598135>
- [362] Wu, F., Liu, X., Xiao, C.: DeceptPrompt: Exploiting LLM-driven Code Generation via Adversarial Natural Language Instructions. arXiv (2023). <https://doi.org/10.48550/ARXIV.2312.04730> . <https://arxiv.org/abs/2312.04730>
- [363] Zhong, L., Wang, Z.: Can ChatGPT replace StackOverflow? A Study on Robustness and Reliability of Large Language Model Code Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2308.10335> . <https://arxiv.org/abs/2308.10335>
- [364] Ma, Q., Wu, T., Koedinger, K.: Is AI the better programming partner? Human-Human Pair Programming vs. Human-AI pAIr Programming. arXiv (2023). <https://doi.org/10.48550/ARXIV.2306.05153> . <https://arxiv.org/abs/2306.05153>
- [365] Wong, M.-F., Guo, S., Hang, C.-N., Ho, S.-W., Tan, C.-W.: Natural language generation and understanding of big code for ai-assisted programming: A review. Entropy **25**(6), 888 (2023)
- [366] Widyasari, R., Zhang, T., Bouraffa, A., Maalej, W., Lo, D.: Explaining Explanation: An Empirical Study on Explanation in Code Reviews. arXiv (2023). <https://doi.org/10.48550/ARXIV.2311.09020> . <https://arxiv.org/abs/2311.09020>
- [367] Widjojo, P., Treude, C.: Addressing Compiler Errors: Stack Overflow or Large Language Models? arXiv (2023). <https://doi.org/10.48550/ARXIV.2307.10793> . <https://arxiv.org/abs/2307.10793>

- [368] White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., Schmidt, D.C.: A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. arXiv (2023). <https://doi.org/10.48550/ARXIV.2302.11382> . <https://arxiv.org/abs/2302.11382>
- [369] White, J., Hays, S., Fu, Q., Spencer-Smith, J., Schmidt, D.C.: ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design. arXiv (2023). <https://doi.org/10.48550/ARXIV.2303.07839> . <https://arxiv.org/abs/2303.07839>
- [370] Weyssow, M., Zhou, X., Kim, K., Lo, D., Sahraoui, H.: On the usage of continual learning for out-of-distribution generalization in pre-trained language models of code. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE 2023, pp. 1470–1482. Association for Computing Machinery, New York, NY, USA (2023). <https://doi.org/10.1145/3611643.3616244> . <https://doi.org/10.1145/3611643.3616244>
- [371] Weyssow, M., Zhou, X., Kim, K., Lo, D., Sahraoui, H.: Exploring Parameter-Efficient Fine-Tuning Techniques for Code Generation with Large Language Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2308.10462> . <https://arxiv.org/abs/2308.10462>
- [372] Wei, Y., Wang, Z., Liu, J., Ding, Y., Zhang, L.: Magicoder: Empowering Code Generation with OSS-Instruct. arXiv (2023). <https://doi.org/10.48550/ARXIV.2312.02120> . <https://arxiv.org/abs/2312.02120>
- [373] Wei, Y., Xia, C.S., Zhang, L.: Copiloting the copilots: Fusing large language models with completion engines for automated program repair. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE '23, pp. 172–184. ACM, ??? (2023). <https://doi.org/10.1145/3611643.3616271> . <http://dx.doi.org/10.1145/3611643.3616271>
- [374] Zheng, W., Sharan, S.P., Jaiswal, A.K., Wang, K., Xi, Y., Xu, D., Wang, Z.: Outline, Then Details: Syntactically Guided Coarse-To-Fine Code Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.00909> . <https://arxiv.org/abs/2305.00909>
- [375] Wang, W., Wang, Y., Joty, S., Hoi, S.C.H.: RAP-Gen: Retrieval-Augmented Patch Generation with CodeT5 for Automatic Program Repair. arXiv (2023). <https://doi.org/10.48550/ARXIV.2309.06057> . <https://arxiv.org/abs/2309.06057>
- [376] Wang, D., Chen, B., Li, S., Luo, W., Peng, S., Dong, W., Liao, X.: One Adapter for All Programming Languages? Adapter Tuning for Code Search and Summarization. arXiv (2023). <https://doi.org/10.48550/ARXIV.2303.15822> .

<https://arxiv.org/abs/2303.15822>

- [377] Wang, S., Geng, M., Lin, B., Sun, Z., Wen, M., Liu, Y., Li, L., Bissyandé, T.F., Mao, X.: Natural language to code: How far are we? In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE '23, pp. 375–387. ACM, ??? (2023). <https://doi.org/10.1145/3611643.3616323> . <http://dx.doi.org/10.1145/3611643.3616323>
- [378] Wang, S., Jean, S., Sengupta, S., Gung, J., Pappas, N., Zhang, Y.: Measuring and Mitigating Constraint Violations of In-Context Learning for Utterance-to-API Semantic Parsing. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.15338> . <https://arxiv.org/abs/2305.15338>
- [379] Wang, X., Peng, H., Jabbarvand, R., Ji, H.: LeTI: Learning to Generate from Textual Interactions. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.10314> . <https://arxiv.org/abs/2305.10314>
- [380] Wang, J., Liu, S., Xie, X., Li, Y.: Evaluating AIGC Detectors on Code Content. arXiv (2023). <https://doi.org/10.48550/ARXIV.2304.05193> . <https://arxiv.org/abs/2304.05193>
- [381] Zheng, Q., Xia, X., Zou, X., Dong, Y., Wang, S., Xue, Y., Wang, Z., Shen, L., Wang, A., Li, Y., Su, T., Yang, Z., Tang, J.: CodeGeeX: A Pre-Trained Model for Code Generation with Multilingual Benchmarking on HumanEval-X. arXiv (2023). <https://doi.org/10.48550/ARXIV.2303.17568> . <https://arxiv.org/abs/2303.17568>
- [382] Wang, Y., Le, H., Gotmare, A.D., Bui, N.D.Q., Li, J., Hoi, S.C.H.: CodeT5+: Open Code Large Language Models for Code Understanding and Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2305.07922> . <https://arxiv.org/abs/2305.07922>
- [383] Wang, Z., Li, J., Li, G., Jin, Z.: ChatCoder: Chat-based Refine Requirement Improves LLMs’ Code Generation. arXiv (2023). <https://doi.org/10.48550/ARXIV.2311.00272> . <https://arxiv.org/abs/2311.00272>
- [384] Wang, C., Liu, J., Peng, X., Liu, Y., Lou, Y.: Boosting Static Resource Leak Detection via LLM-based Resource-Oriented Intention Inference. arXiv (2023). <https://doi.org/10.48550/ARXIV.2311.04448> . <https://arxiv.org/abs/2311.04448>
- [385] Wadhwa, N., Pradhan, J., Sonwane, A., Sahu, S.P., Natarajan, N., Kanade, A., Parthasarathy, S., Rajamani, S.: Frustrated with Code Quality Issues? LLMs can Help! arXiv (2023). <https://doi.org/10.48550/ARXIV.2309.12938> . <https://arxiv.org/abs/2309.12938>

- [386] Vikram, V., Lemieux, C., Sunshine, J., Padhye, R.: Can Large Language Models Write Good Property-Based Tests? arXiv (2023). <https://doi.org/10.48550/ARXIV.2307.04346> . <https://arxiv.org/abs/2307.04346>
- [387] Jiang, Z., Shi, L., Yang, G., Wang, Q.: Patuntrack: Automated generating patch examples for issue reports without tracked insecure code. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1–13. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3694982> . <http://dx.doi.org/10.1145/3691620.3694982>
- [388] Guo, A., Zhou, Y., Tian, H., Fang, C., Sun, Y., Sun, W., Gao, X., Luu, A.T., Liu, Y., Chen, Z.: Sovar: Build generalizable scenarios from accident reports for autonomous driving testing. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 268–280. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695037> . <http://dx.doi.org/10.1145/3691620.3695037>
- [389] Wang, C., Gao, S., Gao, C., Wang, W., Chong, C.Y., Gao, S., Lyu, M.R.: A systematic evaluation of large code models in api suggestion: When, which, and how. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 281–293. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695004> . <http://dx.doi.org/10.1145/3691620.3695004>
- [390] Yang, C., Hong, Y., Lewis, G., Wu, T., Kästner, C.: What is wrong with my model? identifying systematic problems with semantic data slicing. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 306–318. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695033> . <http://dx.doi.org/10.1145/3691620.3695033>
- [391] Zhang, Y., Liu, Z., Feng, Y., Xu, B.: Leveraging large language model to assist detecting rust code comment inconsistency. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 356–366. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695010> . <http://dx.doi.org/10.1145/3691620.3695010>
- [392] Wu, Y., Wen, M., Yu, Z., Guo, X., Jin, H.: Effective vulnerable function identification based on cve description empowered by large language models. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 393–405. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695013> . <http://dx.doi.org/10.1145/3691620.3695013>
- [393] Wu, G., Cao, W., Yao, Y., Wei, H., Chen, T., Ma, X.: Llm meets bounded model checking: Neuro-symbolic loop invariant inference. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 406–417. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695014> . <http://dx.doi.org/10.1145/3691620.3695014>

- [394] Lahiri, S., Kalita, P.K., Chittora, A.K., Vankudre, V., Roy, S.: Program synthesis meets visual what-comes-next puzzles. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 418–429. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695015> . <http://dx.doi.org/10.1145/3691620.3695015>
- [395] Cheng, B., Zhang, C., Wang, K., Shi, L., Liu, Y., Wang, H., Guo, Y., Li, D., Chen, X.: Semantic-enhanced indirect call analysis with large language models. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 430–442. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695016> . <http://dx.doi.org/10.1145/3691620.3695016>
- [396] Liu, Y., Yu, J., Sun, H., Shi, L., Deng, G., Chen, Y., Liu, Y.: Efficient detection of toxic prompts in large language models. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 455–467. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695018> . <http://dx.doi.org/10.1145/3691620.3695018>
- [397] Zhao, J., Yang, Z., Zhang, L., Lian, X., Yang, D., Tan, X.: Drminer: Extracting latent design rationale from jira issue logs. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 468–480. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695019> . <http://dx.doi.org/10.1145/3691620.3695019>
- [398] Gao, X., Xiong, Y., Wang, D., Guan, Z., Shi, Z., Wang, H., Li, S.: Preference-guided refactored tuning for retrieval augmented code generation. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 65–77. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3694987> . <http://dx.doi.org/10.1145/3691620.3694987>
- [399] She, X., Zhao, Y., Wang, H.: Wade: Decompiling webassembly using large language model. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 481–492. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695020> . <http://dx.doi.org/10.1145/3691620.3695020>
- [400] Chen, Z., Jiang, L.: Promise and peril of collaborative code generation models: Balancing effectiveness and memorization. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 493–505. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695021> . <http://dx.doi.org/10.1145/3691620.3695021>
- [401] Anandayuvraj, D., Campbell, M., Tewari, A., Davis, J.C.: Fail: Analyzing software failures from the news using llms. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 506–518. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695022> . <http://dx.doi.org/10.1145/3691620.3695022>

- [402] Liao, Y., Xu, M., Lin, Y., Teoh, X., Xie, X., Feng, R., Liaw, F., Zhang, H., Dong, J.S.: Detecting and explaining anomalies caused by web tamper attacks via building consistency-based normality. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 531–543. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695024> . <http://dx.doi.org/10.1145/3691620.3695024>
- [403] Huang, Z., Huang, Y., Chen, X., Zhou, X., Yang, C., Zheng, Z.: An empirical study on learning-based techniques for explicit and implicit commit messages generation. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 544–556. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695025> . <http://dx.doi.org/10.1145/3691620.3695025>
- [404] Xu, C., Chen, S., Wu, J., Cheung, S.-C., Terragni, V., Zhu, H., Cao, J.: Mr-adopt: Automatic deduction of input transformation function for metamorphic testing. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 557–569. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3696020> . <http://dx.doi.org/10.1145/3691620.3696020>
- [405] Liu, W., Yu, A., Zan, D., Shen, B., Zhang, W., Zhao, H., Jin, Z., Wang, Q.: Graphcoder: Enhancing repository-level code completion via coarse-to-fine retrieval based on code context graph. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 570–581. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695054> . <http://dx.doi.org/10.1145/3691620.3695054>
- [406] Wu, C., Chen, J., Wang, Z., Liang, R., Du, R.: Semantic sleuth: Identifying ponzi contracts via large language models. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 582–593. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695055> . <http://dx.doi.org/10.1145/3691620.3695055>
- [407] Cao, S., Sun, X., Wu, X., Lo, D., Bo, L., Li, B., Liu, X., Lin, X., Liu, W.: Snopy: Bridging sample denoising with causal graph learning for effective vulnerability detection. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 606–618. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695057> . <http://dx.doi.org/10.1145/3691620.3695057>
- [408] Muttillo, V., Di Sipio, C., Rubei, R., Berardinelli, L., Dehghani, M.: Towards synthetic trace generation of modeling operations using in-context learning approach. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 619–630. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695058> . <http://dx.doi.org/10.1145/3691620.3695058>

<http://dx.doi.org/10.1145/3691620.3695058>

- [409] Chen, T., Jiang, Y., Fan, F., Liu, B., Liu, H.: A position-aware approach to decomposing god classes. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 129–140. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3694992> . <http://dx.doi.org/10.1145/3691620.3694992>
- [410] Yang, W., Gao, C., Liu, X., Li, Y., Xue, Y.: Rust-twins: Automatic rust compiler testing through program mutation and dual macros generation. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 631–642. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695059> . <http://dx.doi.org/10.1145/3691620.3695059>
- [411] Zhang, Z., Bai, W., Li, Y., Meng, M.H., Wang, K., Shi, L., Li, L., Wang, J., Wang, H.: Glitchprober: Advancing effective detection and mitigation of glitch tokens in large language models. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 643–655. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695060> . <http://dx.doi.org/10.1145/3691620.3695060>
- [412] Yu, X., Zhang, Z., Niu, F., Hu, X., Xia, X., Grundy, J.: What makes a high-quality training dataset for large language models: A practitioners' perspective. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 656–668. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695061> . <http://dx.doi.org/10.1145/3691620.3695061>
- [413] Liu, F., Liu, Z., Zhao, Q., Jiang, J., Zhang, L., Sun, Z., Li, G., Li, Z., Ma, Y.: Fastfixer: An efficient and effective approach for repairing programming assignments. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 669–680. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695062> . <http://dx.doi.org/10.1145/3691620.3695062>
- [414] Yu, X., Li, C., Pan, M., Li, X.: Droidcoder: Enhanced android code completion with context-enriched retrieval-augmented generation. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 681–693. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695063> . <http://dx.doi.org/10.1145/3691620.3695063>
- [415] Li, G., Zhi, C., Chen, J., Han, J., Deng, S.: Exploring parameter-efficient fine-tuning of large language model on automated program repair. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 719–731. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695066> . <http://dx.doi.org/10.1145/3691620.3695066>
- [416] Cui, D., Wang, J., Wang, Q., Ji, P., Qiao, M., Zhao, Y., Hu, J., Wang,

- L., Li, Q.: Three heads are better than one: Suggesting move method refactoring opportunities with inter-class code entity dependency enhanced hybrid hypergraph neural network. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 745–757. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695068> . <http://dx.doi.org/10.1145/3691620.3695068>
- [417] Xu, Y., Li, Y., Wang, J., Zhang, X.: Evaluating terminology translation in machine translation systems via metamorphic testing. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 758–769. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695069> . <http://dx.doi.org/10.1145/3691620.3695069>
- [418] Hu, C., Chai, Y., Zhou, H., Meng, F., Zhou, J., Gu, X.: How effectively do code language models understand poor-readability code? In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 795–806. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695072> . <http://dx.doi.org/10.1145/3691620.3695072>
- [419] JianWang, Liu, S., Xie, X., Li, Y.: An empirical study to evaluate aigc detectors on code content. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 844–856. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695468> . <http://dx.doi.org/10.1145/3691620.3695468>
- [420] Dong, J., Sun, J., Lin, Y., Zhang, Y., Ma, M., Dong, J.S., Hao, D.: Revisiting the conflict-resolving problem from a semantic perspective. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 141–152. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3694993> . <http://dx.doi.org/10.1145/3691620.3694993>
- [421] Wei, J., Courbis, A.-L., Lambolais, T., Xu, B., Bernard, P.L., Dray, G., Maalej, W.: Getting inspiration for feature elicitation: App store-vs. llm-based approach. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 857–869. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695591> . <http://dx.doi.org/10.1145/3691620.3695591>
- [422] Cao, J., Chen, Z., Wu, J., Cheung, S.-C., Xu, C.: Javabench: A benchmark of object-oriented code generation for evaluating large language models. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 870–882. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695470> . <http://dx.doi.org/10.1145/3691620.3695470>
- [423] Han, Y., Du, Q., Huang, Y., Wu, J., Tian, F., He, C.: The potential of one-shot failure root cause analysis: Collaboration of the large language model and small classifier. In: Proceedings of the 39th IEEE/ACM

- International Conference on Automated Software Engineering. ASE '24, pp. 931–943. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695475> . <http://dx.doi.org/10.1145/3691620.3695475>
- [424] Chen, J., Zhong, Q., Wang, Y., Ning, K., Liu, Y., Xu, Z., Zhao, Z., Chen, T., Zheng, Z.: Rmcbench: Benchmarking large language models' resistance to malicious code. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 995–1006. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695480> . <http://dx.doi.org/10.1145/3691620.3695480>
- [425] Wu, Y., Xie, X., Peng, C., Liu, D., Wu, H., Fan, M., Liu, T., Wang, H.: Advscanner: Generating adversarial smart contracts to exploit reentrancy vulnerabilities using llm and static analysis. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1019–1031. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695482> . <http://dx.doi.org/10.1145/3691620.3695482>
- [426] Zhao, Y., Chen, Y., Sun, Z., Liang, Q., Wang, G., Hao, D.: Spotting code mutation for predictive mutation testing. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1133–1145. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695491> . <http://dx.doi.org/10.1145/3691620.3695491>
- [427] Huang, Y., Wang, R., Zheng, W., Zhou, Z., Wu, S., Ke, S., Chen, B., Gao, S., Peng, X.: Spiderscan: Practical detection of malicious npm packages based on graph-based behavior modeling and matching. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1146–1158. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695492> . <http://dx.doi.org/10.1145/3691620.3695492>
- [428] Sun, X., Gao, X., Cao, S., Bo, L., Wu, X., Huang, K.: 1+1>2: Integrating deep code behaviors with metadata features for malicious pypi package detection. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1159–1170. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695493> . <http://dx.doi.org/10.1145/3691620.3695493>
- [429] Kim, K., Kim, J., Park, B., Kim, D., Chong, C.Y., Wang, Y., Sun, T., Tang, D., Klein, J., Bissyande, T.F.: Datarecipe — how to cook the data for codelm? In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1206–1218. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695593> . <http://dx.doi.org/10.1145/3691620.3695593>
- [430] Li, Z., Zhang, C., Pan, M., Zhang, T., Li, X.: Aacegen: Attention guided adversarial code example generation for deep code models. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1245–1257. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695500>

- . <http://dx.doi.org/10.1145/3691620.3695500>
- [431] Xiao, Y., Le, V.-H., Zhang, H.: Demonstration-free: Towards more practical log parsing with large language models. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 153–165. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3694994> . <http://dx.doi.org/10.1145/3691620.3694994>
- [432] Wang, Z., Liu, K., Li, G., Jin, Z.: Hits: High-coverage llm-based unit test generation via method slicing. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1258–1268. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695501> . <http://dx.doi.org/10.1145/3691620.3695501>
- [433] Song, Z., Zhou, Y., Dong, S., Zhang, K., Zhang, K.: Typefsl: Type prediction from binaries via inter-procedural data-flow analysis and few-shot learning. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1269–1281. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695502> . <http://dx.doi.org/10.1145/3691620.3695502>
- [434] Huang, J., Guo, D., Wang, C., Gu, J., Lu, S., Inala, J.P., Yan, C., Gao, J., Duan, N., Lyu, M.R.: Contextualized data-wrangling code generation in computational notebooks. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1282–1294. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695503> . <http://dx.doi.org/10.1145/3691620.3695503>
- [435] Qu, M., Liu, J., Kang, L., Wang, S., Ye, D., Huang, T.: Dynamic scoring code token tree: A novel decoding strategy for generating high-performance code. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1308–1318. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695505> . <http://dx.doi.org/10.1145/3691620.3695505>
- [436] Zhang, H., Cheng, W., Wu, Y., Hu, W.: A pair programming framework for code generation via multi-plan exploration and feedback-driven refinement. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1319–1331. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695506> . <http://dx.doi.org/10.1145/3691620.3695506>
- [437] Wu, D., Mu, F., Shi, L., Guo, Z., Liu, K., Zhuang, W., Zhong, Y., Zhang, L.: ismell: Assembling llms with expert toolsets for code smell detection and refactoring. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1345–1357. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695508> . <http://dx.doi.org/10.1145/3691620.3695508>
- [438] Yan, C., Ren, R., Meng, M.H., Wan, L., Ooi, T.Y., Bai, G.: Exploring chatgpt

- app ecosystem: Distribution, deployment and security. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1370–1382. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695510> . <http://dx.doi.org/10.1145/3691620.3695510>
- [439] Corradini, D., Montolli, Z., Pasqua, M., Ceccato, M.: Deeprest: Automated test case generation for rest apis exploiting deep reinforcement learning. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1383–1394. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695511> . <http://dx.doi.org/10.1145/3691620.3695511>
- [440] Pirzada, M.A.A., Reger, G., Bhayat, A., Cordeiro, L.C.: Llm-generated invariants for bounded model checking without loop unrolling. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1395–1407. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695512> . <http://dx.doi.org/10.1145/3691620.3695512>
- [441] Jiang, Z., Wen, M., Cao, J., Shi, X., Jin, H.: Towards understanding the effectiveness of large language models on directed test input generation. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1408–1420. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695513> . <http://dx.doi.org/10.1145/3691620.3695513>
- [442] Lu, J., Wang, H., Liu, Z., Liang, K., Bao, L., Yang, X.: Instructive code retriever: Learn from large language model's feedback for code intelligence tasks. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 191–203. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3694997> . <http://dx.doi.org/10.1145/3691620.3694997>
- [443] Li, X., Meng, G., Liu, S., Xiang, L., Sun, K., Chen, K., Luo, X., Liu, Y.: Attribution-guided adversarial code prompt generation for code completion models. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1460–1471. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695517> . <http://dx.doi.org/10.1145/3691620.3695517>
- [444] Bo, L., Ji, W., Sun, X., Zhang, T., Wu, X., Wei, Y.: Chatbr: Automated assessment and improvement of bug report quality using chatgpt. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1472–1483. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695518> . <http://dx.doi.org/10.1145/3691620.3695518>
- [445] Wang, Y., Chen, Y., Zhao, Y., Gong, Z., Chen, J., Hao, D.: Mutual learning-based framework for enhancing robustness of code models via adversarial training. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1484–1496. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695519> . <http://dx.doi.org/10.1145/3691620.3695519>

- [446] Tang, S., Zhang, Z., Zhou, J., Lei, L., Zhou, Y., Xue, Y.: Legend: A top-down approach to scenario generation of autonomous driving systems assisted by large language models. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1497–1508. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695520> . <http://dx.doi.org/10.1145/3691620.3695520>
- [447] Lu, M., Delaware, B., Zhang, T.: Proof automation with large language models. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1509–1520. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695521> . <http://dx.doi.org/10.1145/3691620.3695521>
- [448] Zhu, M., Karim, M., Lourentzou, I., Yao, D.: Semi-supervised code translation overcoming the scarcity of parallel code data. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1545–1556. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695524> . <http://dx.doi.org/10.1145/3691620.3695524>
- [449] Wang, H., Hu, Y., Wu, H., Liu, D., Peng, C., Wu, Y., Fan, M., Liu, T.: Sky-eye: Detecting imminent attacks via analyzing adversarial smart contracts. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1570–1582. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695526> . <http://dx.doi.org/10.1145/3691620.3695526>
- [450] Mathews, N.S., Nagappan, M.: Test-driven development and llm-based code generation. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1583–1594. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695527> . <http://dx.doi.org/10.1145/3691620.3695527>
- [451] Fan, M., Shi, J., Wang, Y., Yu, L., Zhang, X., Wang, H., Jin, W., Liu, T.: Giving without notifying: Assessing compliance of data transmission in android apps. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1595–1606. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695528> . <http://dx.doi.org/10.1145/3691620.3695528>
- [452] Yang, L., Yang, C., Gao, S., Wang, W., Wang, B., Zhu, Q., Chu, X., Zhou, J., Liang, G., Wang, Q., Chen, J.: On the evaluation of large language models in unit test generation. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1607–1619. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695529> . <http://dx.doi.org/10.1145/3691620.3695529>
- [453] Li, C., Xu, Z., Di, P., Wang, D., Li, Z., Zheng, Q.: Understanding code changes practically with small-scale language models. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 216–228. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3694999> .

<http://dx.doi.org/10.1145/3691620.3694999>

- [454] Zhou, Z., Yang, Y., Wu, S., Huang, Y., Chen, B., Peng, X.: Magneto: A step-wise approach to exploit vulnerabilities in dependent libraries via llm-empowered directed fuzzing. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1633–1644. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695531> . <http://dx.doi.org/10.1145/3691620.3695531>
- [455] Chen, M., Liu, Z., Tao, H., Hong, Y., Lo, D., Xia, X., Sun, J.: B4: Towards optimal assessment of plausible code solutions with plausible tests. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1693–1705. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695536> . <http://dx.doi.org/10.1145/3691620.3695536>
- [456] Zhao, J., Yang, D., Zhang, L., Lian, X., Yang, Z., Liu, F.: Enhancing automated program repair with solution design. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1706–1718. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695537> . <http://dx.doi.org/10.1145/3691620.3695537>
- [457] Zhao, Y., Gong, L., Huang, Z., Wang, Y., Wei, M., Wu, F.: Coding-ptms: How to find optimal code pre-trained models for code embedding in vulnerability detection? In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1732–1744. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695539> . <http://dx.doi.org/10.1145/3691620.3695539>
- [458] Xia, J., Liu, J., Brown, N., Chen, Y., Feng, Y.: Refinement types for visualization. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1871–1881. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695550> . <http://dx.doi.org/10.1145/3691620.3695550>
- [459] Feng, J., Liu, J., Gao, C., Chong, C.Y., Wang, C., Gao, S., Xia, X.: Complex-codeeval: A benchmark for evaluating large code models on more complex code. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1895–1906. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695552> . <http://dx.doi.org/10.1145/3691620.3695552>
- [460] Du, Y., Sun, H., Li, M.: A joint learning model with variational interaction for multilingual program translation. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1907–1918. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695553> . <http://dx.doi.org/10.1145/3691620.3695553>
- [461] Zhang, J., Wang, C., Li, A., Wang, W., Li, T., Liu, Y.: Vuladvisor: Natural language suggestion generation for software vulnerability repair. In: Proceedings of

the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 1932–1944. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695555> . <http://dx.doi.org/10.1145/3691620.3695555>

- [462] Sun, Z., Wan, Y., Li, J., Zhang, H., Jin, Z., Li, G., Lyu, C.: Sifting through the chaff: On utilizing execution feedback for ranking the generated code candidates. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 229–241. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695000> . <http://dx.doi.org/10.1145/3691620.3695000>
- [463] Zhang, Q., Zhou, C., Go, G., Zeng, B., Shi, H., Xu, Z., Jiang, Y.: Imperceptible content poisoning in llm-powered applications. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. ASE '24, pp. 242–254. ACM, ??? (2024). <https://doi.org/10.1145/3691620.3695001> . <http://dx.doi.org/10.1145/3691620.3695001>
- [464] Tian, R., Ye, Y., Qin, Y., Cong, X., Lin, Y., Pan, Y., Wu, Y., Hui, H., Liu, W., Liu, Z., Sun, M.: DebugBench: Evaluating Debugging Capability of Large Language Models. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.04621> . <https://arxiv.org/abs/2401.04621>
- [465] Sun, Z., Du, X., Song, F., Wang, S., Li, L.: When Neural Code Completion Models Size up the Situation: Attaining Cheaper and Faster Completion through Dynamic Model Inference. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.09964> . <https://arxiv.org/abs/2401.09964>
- [466] Sun, Y., Wu, D., Xue, Y., Liu, H., Ma, W., Zhang, L., Liu, Y., Li, Y.: LLM4Vuln: A Unified Evaluation Framework for Decoupling and Enhancing LLMs' Vulnerability Reasoning. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.16185> . <https://arxiv.org/abs/2401.16185>
- [467] Steenhoek, B., Gao, H., Le, W.: Dataflow Analysis-Inspired Deep Learning for Efficient Vulnerability Detection. arXiv (2022). <https://doi.org/10.48550/ARXIV.2212.08108> . <https://arxiv.org/abs/2212.08108>
- [468] Zhang, L., Lu, S., Duan, N.: Selene: Pioneering Automated Proof in Software Verification. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.07663> . <https://arxiv.org/abs/2401.07663>
- [469] Zhang, S., Wang, J., Dong, G., Sun, J., Zhang, Y., Pu, G.: Experimenting a New Programming Practice with LLMs. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.01062> . <https://arxiv.org/abs/2401.01062>
- [470] Shestov, A., Levichev, R., Mussabayev, R., Maslov, E., Cheshkov, A., Zadorozhny, P.: Finetuning Large Language Models for Vulnerability Detection. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.17010> . <https://arxiv.org/abs/2401.17010>

- [471] Zhang, K., Li, J., Li, G., Shi, X., Jin, Z.: CodeAgent: Enhancing Code Generation with Tool-Integrated Agent Systems for Real-World Repo-level Coding Challenges. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.07339> . <https://arxiv.org/abs/2401.07339>
- [472] Ruiz, F.V., Grishina, A., Hort, M., Moonen, L.: Assessing the Latent Automated Program Repair Capabilities of Large Language Models using Round-Trip Translation. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.07994> . <https://arxiv.org/abs/2401.07994>
- [473] Zhang, Q., Fang, C., Sun, W., Liu, Y., He, T., Hao, X., Chen, Z.: APPT: Boosting Automated Patch Correctness Prediction via Fine-tuning Pre-trained Models. arXiv (2023). <https://doi.org/10.48550/ARXIV.2301.12453> . <https://arxiv.org/abs/2301.12453>
- [474] Ridnik, T., Kredo, D., Friedman, I.: Code Generation with AlphaCodium: From Prompt Engineering to Flow Engineering. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.08500> . <https://arxiv.org/abs/2401.08500>
- [475] Peng, Y., Gao, S., Gao, C., Huo, Y., Lyu, M.R.: Domain Knowledge Matters: Improving Prompts with Fix Templates for Repairing Python Type Errors. arXiv (2023). <https://doi.org/10.48550/ARXIV.2306.01394> . <https://arxiv.org/abs/2306.01394>
- [476] Nichols, D., Davis, J.H., Xie, Z., Rajaram, A., Bhatele, A.: Can large language models write parallel code? In: Proceedings of the 33rd International Symposium on High-Performance Parallel and Distributed Computing. HPDC '24, pp. 281–294. Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3625549.3658689> . <https://doi.org/10.1145/3625549.3658689>
- [477] Motger, Q., Miaschi, A., Dell'Orletta, F., Franch, X., Marco, J.: T-frex: A transformer-based feature extraction method from mobile app reviews. In: 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 227–238. IEEE, ??? (2024). <https://doi.org/10.1109/saner60148.2024.00030> . <http://dx.doi.org/10.1109/SANER60148.2024.00030>
- [478] Ma, L., Liu, S., Li, Y., Xie, X., Bu, L.: Specgen: Automated generation of formal program specifications via large language models. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 16–28. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00129> . <http://dx.doi.org/10.1109/ICSE55347.2025.00129>
- [479] Ma, L., Yang, W., Xu, B., Jiang, S., Fei, B., Liang, J., Zhou, M., Xiao, Y.: Knowlog: Knowledge enhanced pre-trained language model for log understanding. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. ICSE '24, pp. 1–13. ACM, ??? (2024). <https://doi.org/10.1145/3597503.3623304> . <http://dx.doi.org/10.1145/3597503.3623304>

- [480] Liu, Y., Tantithamthavorn, C., Liu, Y., Li, L.: On the reliability and explainability of language models for program generation. *ACM Transactions on Software Engineering and Methodology* **33**(5), 1–26 (2024) <https://doi.org/10.1145/3641540>
- [481] Liu, Y., Tao, S., Meng, W., Wang, J., Ma, W., Chen, Y., Zhao, Y., Yang, H., Jiang, Y.: Interpretable online log analysis using large language models with prompt strategies. In: Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension. ICPC '24, pp. 35–46. ACM, ??? (2024). <https://doi.org/10.1145/3643916.3644408> . <http://dx.doi.org/10.1145/3643916.3644408>
- [482] Li, H., Zhou, X., Shen, Z.: Rewriting the Code: A Simple Method for Large Language Model Augmented Code Search. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.04514> . <https://arxiv.org/abs/2401.04514>
- [483] Li, Y., Ren, Z., Wang, Z., Yang, L., Dong, L., Zhong, C., Zhang, H.: Fine-se: Integrating semantic features and expert features for software effort estimation. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. ICSE '24, pp. 1–12. ACM, ??? (2024). <https://doi.org/10.1145/3597503.3623349> . <http://dx.doi.org/10.1145/3597503.3623349>
- [484] Li, J., Li, G., Zhao, Y., Li, Y., Jin, Z., Zhu, H., Liu, H., Liu, K., Wang, L., Fang, Z., Wang, L., Ding, J., Zhang, X., Dong, Y., Zhu, Y., Gu, B., Yang, M.: DevEval: Evaluating Code Generation in Practical Software Projects. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.06401> . <https://arxiv.org/abs/2401.06401>
- [485] Kirinuki, H., Tanno, H.: ChatGPT and Human Synergy in Black-Box Testing: A Comparative Analysis. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.13924> . <https://arxiv.org/abs/2401.13924>
- [486] Kabir, A., Wang, S., Tian, Y., Chen, T.-H.P., Asaduzzaman, M., Zhang, W.: Zs4c: Zero-shot synthesis of compilable code for incomplete code snippets using llms. *ACM Transactions on Software Engineering and Methodology* **34**(4), 1–30 (2025) <https://doi.org/10.1145/3702979>
- [487] Islam, N.T., Khoury, J., Seong, A., Karkevandi, M.B., Parra, G.D.L.T., Bou-Harb, E., Najafirad, P.: LLM-Powered Code Vulnerability Repair with Reinforcement Learning and Semantic Reward. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.03374> . <https://arxiv.org/abs/2401.03374>
- [488] Islam, N.T., Karkevandi, M.B., Najafirad, P.: Code Security Vulnerability Repair Using Reinforcement Learning with Large Language Models. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.07031> . <https://arxiv.org/abs/2401.07031>
- [489] Huang, Y., Wang, J., Liu, Z., Wang, Y., Wang, S., Chen, C., Hu, Y., Wang, Q.:

- Crashtranslator: Automatically reproducing mobile application crashes directly from stack trace. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. ICSE '24, pp. 1–13. ACM, ??? (2024). <https://doi.org/10.1145/3597503.3623298> . <http://dx.doi.org/10.1145/3597503.3623298>
- [490] Hu, X., Kuang, K., Sun, J., Yang, H., Wu, F.: Leveraging Print Debugging to Improve Code Generation in Large Language Models. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.05319> . <https://arxiv.org/abs/2401.05319>
- [491] Guo, Q., Cao, J., Xie, X., Liu, S., Li, X., Chen, B., Peng, X.: Exploring the potential of chatgpt in automated code refinement: An empirical study. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. ICSE '24, pp. 1–13. ACM, ??? (2024). <https://doi.org/10.1145/3597503.3623306> . <http://dx.doi.org/10.1145/3597503.3623306>
- [492] Guo, D., Zhu, Q., Yang, D., Xie, Z., Dong, K., Zhang, W., Chen, G., Bi, X., Wu, Y., Li, Y.K., Luo, F., Xiong, Y., Liang, W.: DeepSeek-Coder: When the Large Language Model Meets Programming – The Rise of Code Intelligence. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.14196> . <https://arxiv.org/abs/2401.14196>
- [493] Geng, M., Wang, S., Dong, D., Wang, H., Li, G., Jin, Z., Mao, X., Liao, X.: Large language models are few-shot summarizers: Multi-intent comment generation via in-context learning. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. ICSE '24, pp. 1–13. ACM, ??? (2024). <https://doi.org/10.1145/3597503.3608134> . <http://dx.doi.org/10.1145/3597503.3608134>
- [494] Gao, S., Mao, W., Gao, C., Li, L., Hu, X., Xia, X., Lyu, M.R.: Learning in the wild: Towards leveraging unlabeled data for effectively tuning pre-trained code models. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. ICSE '24, pp. 1–13. ACM, ??? (2024). <https://doi.org/10.1145/3597503.3639216> . <http://dx.doi.org/10.1145/3597503.3639216>
- [495] Ferreira, I., Rafiq, A., Cheng, J.: Incivility detection in open source code review and issue discussions. Journal of Systems and Software **209**, 111935 (2024) <https://doi.org/10.1016/j.jss.2023.111935>
- [496] Fan, G., Chen, S., Gao, C., Xiao, J., Zhang, T., Feng, Z.: Rapid: Zero-shot domain adaptation for code search with pre-trained models. ACM Transactions on Software Engineering and Methodology **33**(5), 1–35 (2024) <https://doi.org/10.1145/3641542>
- [497] Eghbali, A., Pradel, M.: De-Hallucinator: Mitigating LLM Hallucinations in Code Generation Tasks via Iterative Grounding. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.01701> . <https://arxiv.org/abs/2401.01701>
- [498] Dinh, T., Zhao, J., Tan, S., Negrinho, R., Lausen, L., Zha, S., Karypis, G.:

Large Language Models of Code Fail at Completing Code with Potential Bugs. arXiv (2023). <https://doi.org/10.48550/ARXIV.2306.03438> . <https://arxiv.org/abs/2306.03438>

- [499] Chow, Y.W., Di Grazia, L., Pradel, M.: Pyty: Repairing static type errors in python. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. ICSE '24, pp. 1–13. ACM, ??? (2024). <https://doi.org/10.1145/3597503.3639184> . <http://dx.doi.org/10.1145/3597503.3639184>
- [500] Chen, Y., Gao, C., Zhu, M., Liao, Q., Wang, Y., Xu, G.: Apigen: Generative api method recommendation. In: 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 171–182. IEEE, ??? (2024). <https://doi.org/10.1109/saner60148.2024.00025> . <http://dx.doi.org/10.1109/SANER60148.2024.00025>
- [501] Ahmed, T., Pai, K.S., Devanbu, P., Barr, E.: Automatic semantic augmentation of language model prompts (for code summarization). In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. ICSE '24, pp. 1–13. ACM, ??? (2024). <https://doi.org/10.1145/3597503.3639183> . <http://dx.doi.org/10.1145/3597503.3639183>
- [502] Agarwal, M., Shen, Y., Wang, B., Kim, Y., Chen, J.: Structured Code Representations Enable Data-Efficient Adaptation of Code Language Models. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.10716> . <https://arxiv.org/abs/2401.10716>
- [503] Yang, A.Z.H., Martins, R., Goues, C.L., Hellendoorn, V.J.: Large Language Models for Test-Free Fault Localization. arXiv (2023). <https://doi.org/10.48550/ARXIV.2310.01726> . <https://arxiv.org/abs/2310.01726>
- [504] Xu, J., Cui, Z., Zhao, Y., Zhang, X., He, S., He, P., Li, L., Kang, Y., Lin, Q., Dang, Y., Rajmohan, S., Zhang, D.: Unilog: Automatic logging via llm and in-context learning. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. ICSE '24, pp. 1–12. ACM, ??? (2024). <https://doi.org/10.1145/3597503.3623326> . <http://dx.doi.org/10.1145/3597503.3623326>
- [505] Xia, C.S., Paltenghi, M., Le Tian, J., Pradel, M., Zhang, L.: Fuzz4all: Universal fuzzing with large language models. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. ICSE '24. Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3597503.3639121> . <https://doi.org/10.1145/3597503.3639121>
- [506] Wu, D., Feng, Y., Zhang, H., Xu, B.: Automatic recognizing relevant fragments of apis using api references. Automated Software Engineering **31**(1) (2023) <https://doi.org/10.1007/s10515-023-00401-0>
- [507] Wang, C., Zhang, J., Feng, Y., Li, T., Sun, W., Liu, Y., Peng, X.: Teaching

- Code LLMs to Use Autocompletion Tools in Repository-Level Code Generation. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.06391> . <https://arxiv.org/abs/2401.06391>
- [508] Wang, Y., Huang, Y., Guo, D., Zhang, H., Zheng, Z.: SparseCoder: Identifier-Aware Sparse Transformer for File-Level Code Summarization. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.14727> . <https://arxiv.org/abs/2401.14727>
- [509] Wang, S., Ding, L., Shen, L., Luo, Y., Du, B., Tao, D.: OOP: Object-Oriented Programming Evaluation Benchmark for Large Language Models. arXiv (2024). <https://doi.org/10.48550/ARXIV.2401.06628> . <https://arxiv.org/abs/2401.06628>
- [510] Ye, J., Li, Z., Tang, X., Zou, D., Xu, S., Qiang, W., Jin, H.: A causal learning framework for enhancing robustness of source code models. Proceedings of the ACM on Software Engineering **2**(FSE), 2641–2664 (2025) <https://doi.org/10.1145/3729387>
- [511] Sharmin, S., Zahid, A.H., Bhattacharjee, S., Igwilo, C., Kim, M., Le, W.: Automatically detecting numerical instability in machine learning applications via soft assertions. Proceedings of the ACM on Software Engineering **2**(FSE), 2806–2827 (2025) <https://doi.org/10.1145/3729394>
- [512] Fruntke, L., Krinke, J.: Automatically fixing dependency breaking changes. Proceedings of the ACM on Software Engineering **2**(FSE), 2146–2168 (2025) <https://doi.org/10.1145/3729366>
- [513] Wang, Y., Jiang, T., Liu, M., Chen, J., Mao, M., Liu, X., Ma, Y., Zheng, Z.: Beyond functional correctness: Investigating coding style inconsistencies in large language models. Proceedings of the ACM on Software Engineering **2**(FSE), 690–712 (2025) <https://doi.org/10.1145/3715749>
- [514] Wang, C., Feng, J., Gao, S., Gao, C., Li, Z., Peng, T., Huang, H., Deng, Y., Lyu, M.: Beyond peft: Layer-wise optimization for more effective and efficient large code model tuning. Proceedings of the ACM on Software Engineering **2**(FSE), 1567–1590 (2025) <https://doi.org/10.1145/3729341>
- [515] Li, Y., Dhulipala, H., Yadavally, A., Rong, X., Wang, S., Nguyen, T.N.: Blended analysis for predictive execution. Proceedings of the ACM on Software Engineering **2**(FSE), 2987–3008 (2025) <https://doi.org/10.1145/3729402>
- [516] Li, X., Lei, Y., Jia, Z., Zhang, Y., Liu, H., Chen, L., Dong, W., Li, S.: Bridging operator semantic inconsistencies: A source-level cross-framework model conversion approach. Proceedings of the ACM on Software Engineering **2**(FSE), 2030–2052 (2025) <https://doi.org/10.1145/3729361>

- [517] Virk, Y., Devanbu, P., Ahmed, T.: Calibration of large language models on code summarization. Proceedings of the ACM on Software Engineering **2**(FSE), 2944–2964 (2025) <https://doi.org/10.1145/3729400>
- [518] Levin, K.H., Kempen, N., Berger, E.D., Freund, S.N.: Chatdbg: Augmenting debugging with large language models. Proceedings of the ACM on Software Engineering **2**(FSE), 1892–1913 (2025) <https://doi.org/10.1145/3729355>
- [519] Li, A., Zhang, N., Zou, Y., Chen, Z., Wang, J., Zheng, Z.: Cktyper: Enhancing type inference for java code snippets by leveraging crowdsourcing knowledge in stack overflow. Proceedings of the ACM on Software Engineering **2**(FSE), 176–196 (2025) <https://doi.org/10.1145/3715724>
- [520] Yang, X., Zhu, W., Pacheco, M., Zhou, J., Wang, S., Hu, X., Liu, K.: Code change intention, development artifact, and history vulnerability: Putting them together for vulnerability fix detection by llm. Proceedings of the ACM on Software Engineering **2**(FSE), 489–510 (2025) <https://doi.org/10.1145/3715738>
- [521] Li, Y., Liu, B., Zhang, T., Wang, Z., Lo, D., Yang, L., Lyu, J., Zhang, H.: A knowledge enhanced large language model for bug localization. Proceedings of the ACM on Software Engineering **2**(FSE), 1914–1936 (2025) <https://doi.org/10.1145/3729356>
- [522] Al-Kaswan, A., Deatc, S., Koç, B., Deursen, A., Izadi, M.: Code red! on the harmfulness of applying off-the-shelf large language models to programming tasks. Proceedings of the ACM on Software Engineering **2**(FSE), 2477–2499 (2025) <https://doi.org/10.1145/3729380>
- [523] Peng, Y., Wan, J., Li, Y., Ren, X.: Coffe: A code efficiency benchmark for code generation. Proceedings of the ACM on Software Engineering **2**(FSE), 242–265 (2025) <https://doi.org/10.1145/3715727>
- [524] Altmayer Pizzorno, J., Berger, E.D.: Coverup: Effective high coverage test generation for python. Proceedings of the ACM on Software Engineering **2**(FSE), 2897–2919 (2025) <https://doi.org/10.1145/3729398>
- [525] Dhulipala, H., Yadavally, A., Patel, S.S., Nguyen, T.N.: Crispe: Semantic-guided execution planning and dynamic reasoning for enhancing code coverage prediction. Proceedings of the ACM on Software Engineering **2**(FSE), 2965–2986 (2025) <https://doi.org/10.1145/3729401>
- [526] Yu, Z., Zhang, Y., Wen, M., Nie, Y., Zhang, W., Yang, M.: Cxxcrafter: An llm-based agent for automated c/c++ open source software building. Proceedings of the ACM on Software Engineering **2**(FSE), 2618–2640 (2025) <https://doi.org/10.1145/3729386>

- [527] Zhou, T., Zhao, Y., Hou, X., Sun, X., Chen, K., Wang, H.: Declarui: Bridging design and development with automated declarative ui code generation. Proceedings of the ACM on Software Engineering **2**(FSE), 219–241 (2025) <https://doi.org/10.1145/3715726>
- [528] Xia, C.S., Deng, Y., Dunn, S., Zhang, L.: Demystifying llm-based software engineering agents. Proceedings of the ACM on Software Engineering **2**(FSE), 801–824 (2025) <https://doi.org/10.1145/3715754>
- [529] Kong, J., Xie, X., Liu, S.: Demystifying memorization in llm-based program repair via a general hypothesis testing framework. Proceedings of the ACM on Software Engineering **2**(FSE), 2712–2734 (2025) <https://doi.org/10.1145/3729390>
- [530] Wu, W., Cao, Y., Yi, N., Ou, R., Zheng, Z.: Detecting and reducing the factual hallucinations of large language models with metamorphic testing. Proceedings of the ACM on Software Engineering **2**(FSE), 1432–1453 (2025) <https://doi.org/10.1145/3715784>
- [531] Su, X., Liang, H., Wu, H., Niu, B., Xu, F., Zhong, S.: Disco: Towards decompiling evm bytecode to source code using large language models. Proceedings of the ACM on Software Engineering **2**(FSE), 2311–2334 (2025) <https://doi.org/10.1145/3729373>
- [532] Ibrahimzada, A.R., Ke, K., Pawagi, M., Abid, M.S., Pan, R., Sinha, S., Jabbarvand, R.: Alphatrans: A neuro-symbolic compositional approach for repository-level code translation and validation. Proceedings of the ACM on Software Engineering **2**(FSE), 2454–2476 (2025) <https://doi.org/10.1145/3729379>
- [533] Wan, Y., Wang, C., Dong, Y., Wang, W., Li, S., Huo, Y., Lyu, M.: Divide-and-conquer: Generating ui code from screenshots. Proceedings of the ACM on Software Engineering **2**(FSE), 2099–2122 (2025) <https://doi.org/10.1145/3729364>
- [534] Hossain, S.B., Taylor, R., Dwyer, M.: Doc2oraccl: Investigating the impact of documentation on llm-based test oracle generation. Proceedings of the ACM on Software Engineering **2**(FSE), 1870–1891 (2025) <https://doi.org/10.1145/3729354>
- [535] Wang, J., Qiu, Y., Limpanukorn, B., Kang, H.J., Zhang, Q., Kim, M.: Duoreduce: Bug isolation for multi-layer extensible compilation. Proceedings of the ACM on Software Engineering **2**(FSE), 647–667 (2025) <https://doi.org/10.1145/3715747>
- [536] Wang, X., Zhang, M., Meng, X., Zhang, J., Liu, Y., Hu, C.: Element-based automated dnn repair with fine-tuned masked language model. Proceedings of the ACM on Software Engineering **2**(FSE), 106–129 (2025) <https://doi.org/10.1145/3715727>

- [537] Sun, W., Chen, Y., Fang, C., Feng, Y., Xiao, Y., Guo, A., Zhang, Q., Chen, Z., Xu, B., Liu, Y.: Eliminating backdoors in neural code models for secure code understanding. Proceedings of the ACM on Software Engineering **2**(FSE), 1386–1408 (2025) <https://doi.org/10.1145/3715782>
- [538] He, Z., Huq, S.F., Malek, S.: Enhancing web accessibility: Automated detection of issues with generative ai. Proceedings of the ACM on Software Engineering **2**(FSE), 2264–2287 (2025) <https://doi.org/10.1145/3729371>
- [539] Yang, B., Al Mamun, M.A., Zhang, J.M., Uddin, G.: Hallucination detection in large language models with metamorphic relations. Proceedings of the ACM on Software Engineering **2**(FSE), 425–445 (2025) <https://doi.org/10.1145/3715735>
- [540] Khan, H.A., Jiang, Y., Umer, Q., Zhang, Y., Akram, W., Liu, H.: Has my code been stolen for model training? a naturalness based approach to code contamination detection. Proceedings of the ACM on Software Engineering **2**(FSE), 1046–1067 (2025) <https://doi.org/10.1145/3715765>
- [541] Rahe, C., Maalej, W.: How do programming students use generative ai? Proceedings of the ACM on Software Engineering **2**(FSE), 978–1000 (2025) <https://doi.org/10.1145/3715762>
- [542] Calikli, G., Alhamed, M.: Impact of request formats on effort estimation: Are llms different than humans? Proceedings of the ACM on Software Engineering **2**(FSE), 1114–1135 (2025) <https://doi.org/10.1145/3715771>
- [543] Saad, M., López, J.A.H., Chen, B., Varró, D., Sharma, T.: An adaptive language-agnostic pruning method for greener language models for code. Proceedings of the ACM on Software Engineering **2**(FSE), 1183–1204 (2025) <https://doi.org/10.1145/3715773>
- [544] Alon, Y., David, C.: Integrating large language models and reinforcement learning for non-linear reasoning. Proceedings of the ACM on Software Engineering **2**(FSE), 957–977 (2025) <https://doi.org/10.1145/3715761>
- [545] Imtiaz, S.M., Singh, A., Batole, F., Rajan, H.: Irepair: An intent-aware approach to repair data-driven errors in large language models. Proceedings of the ACM on Software Engineering **2**(FSE), 1226–1248 (2025) <https://doi.org/10.1145/3715775>
- [546] Sovrano, F., Bauer, A., Bacchelli, A.: Large language models for in-file vulnerability localization can be “lost in the end”. Proceedings of the ACM on Software Engineering **2**(FSE), 891–913 (2025) <https://doi.org/10.1145/3715758>
- [547] Zhang, J., Hu, X., Gao, S., Xia, X., Lo, D., Li, S.: Less is more: On the

importance of data quality for unit test generation. Proceedings of the ACM on Software Engineering **2**(FSE), 1293–1316 (2025) <https://doi.org/10.1145/3715778>

- [548] Kim, M., Sinha, S., Orso, A.: Llamaresttest: Effective rest api testing with small language models. Proceedings of the ACM on Software Engineering **2**(FSE), 465–488 (2025) <https://doi.org/10.1145/3715737>
- [549] Akram, W., Jiang, Y., Zhang, Y., Khan, H.A., Liu, H.: Llm-based method name suggestion with automatically generated context-rich prompts. Proceedings of the ACM on Software Engineering **2**(FSE), 779–800 (2025) <https://doi.org/10.1145/3715753>
- [550] Wang, C., Liu, T., Zhao, Y., Yang, M., Wang, H.: Llmdroid: Enhancing automated mobile app gui testing coverage with large language model guidance. Proceedings of the ACM on Software Engineering **2**(FSE), 1001–1022 (2025) <https://doi.org/10.1145/3715763>
- [551] Ren, S., He, L., Tu, T., Wu, D., Liu, J., Ren, K., Chen, C.: Lookahead: Preventing defi attacks via unveiling adversarial contracts. Proceedings of the ACM on Software Engineering **2**(FSE), 1847–1869 (2025) <https://doi.org/10.1145/3729353>
- [552] Ran, D., Cao, Y., Guo, Y., Li, Y., Wu, M., Chen, S., Yang, W., Xie, T.: Medusa: A framework for collaborative development of foundation models with automated parameter ownership assignment. Proceedings of the ACM on Software Engineering **2**(FSE), 2594–2617 (2025) <https://doi.org/10.1145/3729385>
- [553] Wu, S., Wang, R., Cao, Y., Chen, B., Zhou, Z., Huang, Y., Zhao, J., Peng, X.: Mystique: Automated vulnerability patch porting with semantic and syntactic-enhanced llm. Proceedings of the ACM on Software Engineering **2**(FSE), 130–152 (2025) <https://doi.org/10.1145/3715718>
- [554] Lu, W., Tian, Y., Zhong, X., Ma, H., Xu, Z., Cheung, S.-C., Sun, C.: An empirical study of bugs in data visualization libraries. Proceedings of the ACM on Software Engineering **2**(FSE), 2075–2098 (2025) <https://doi.org/10.1145/3729363>
- [555] Huang, J., Jiang, Z., Chen, Z., Lyu, M.: No more labelled examples? an unsupervised log parser with llms. Proceedings of the ACM on Software Engineering **2**(FSE), 2406–2429 (2025) <https://doi.org/10.1145/3729377>
- [556] Yang, X., Wang, S., Zhou, J., Zhu, W.: One-for-all does not work! enhancing vulnerability detection by mixture-of-experts (moe). Proceedings of the ACM on Software Engineering **2**(FSE), 446–464 (2025) <https://doi.org/10.1145/3715736>

- [557] Liang, J.T., Lin, M., Rao, N., Myers, B.A.: Prompts are programs too! understanding how developers build software containing prompts. Proceedings of the ACM on Software Engineering **2**(FSE), 1591–1614 (2025) <https://doi.org/10.1145/3729342>
- [558] Leesatapornwongsa, T., Faisal, F., Nath, S.: Reprocopilot: Llm-driven failure reproduction with dynamic refinement. Proceedings of the ACM on Software Engineering **2**(FSE), 2920–2943 (2025) <https://doi.org/10.1145/3729399>
- [559] Tan, X., Long, X., Zhu, Y., Shi, L., Lian, X., Zhang, L.: Revolutionizing newcomers’ onboarding process in oss communities: The future ai mentor. Proceedings of the ACM on Software Engineering **2**(FSE), 1091–1113 (2025) <https://doi.org/10.1145/3715767>
- [560] Chen, Y., Ye, Y., Li, Z., Ma, Y., Gao, C.: Smaller but better: Self-paced knowledge distillation for lightweight yet effective lcms. Proceedings of the ACM on Software Engineering **2**(FSE), 3057–3080 (2025) <https://doi.org/10.1145/3729405>
- [561] Daneshyan, F., He, R., Wu, J., Zhou, M.: Smartnote: An llm-powered, personalised release note generator that just works. Proceedings of the ACM on Software Engineering **2**(FSE), 1663–1686 (2025) <https://doi.org/10.1145/3729345>
- [562] Chen, Z., Li, X., Zhang, J.M., Sun, W., Xiao, Y., Li, T., Lou, Y., Liu, Y.: Software fairness dilemma: Is bias mitigation a zero-sum game? Proceedings of the ACM on Software Engineering **2**(FSE), 1780–1801 (2025) <https://doi.org/10.1145/3729350>
- [563] Chen, M., Liu, Z., Chen, C., Wang, J., Wu, B., Hu, J., Wang, Q.: Standing on the shoulders of giants: Bug-aware automated gui testing via retrieval augmentation. Proceedings of the ACM on Software Engineering **2**(FSE), 825–846 (2025) <https://doi.org/10.1145/3715755>
- [564] Du, X., Wen, M., Wang, H., Wei, Z., Jin, H.: Statement-level adversarial attack on vulnerability detection models via out-of-distribution features. Proceedings of the ACM on Software Engineering **2**(FSE), 3009–3032 (2025) <https://doi.org/10.1145/3729403>
- [565] Wu, W., Hu, H., Fan, Z., Qiao, Y., Huang, Y., Li, Y., Zheng, Z., Lyu, M.: An empirical study of code clones from commercial ai code generators. Proceedings of the ACM on Software Engineering **2**(FSE), 2874–2896 (2025) <https://doi.org/10.1145/3729397>
- [566] Gao, A., Zhang, Z., Wang, S., Huang, L., Wei, S., Ng, V.: Teaching ai the ‘why’ and ‘how’ of software vulnerability fixes. Proceedings of the ACM on Software Engineering **2**(FSE), 2006–2029 (2025) <https://doi.org/10.1145/3729360>

- [567] Moumoula, M.B., Kaboré, A.K., Klein, J., Bissyandé, T.F.: The struggles of llms in cross-lingual code clone detection. Proceedings of the ACM on Software Engineering **2**(FSE), 1023–1045 (2025) <https://doi.org/10.1145/3715764>
- [568] Paramitha, R., Feng, Y., Massacci, F.: Today’s cat is tomorrow’s dog: Accounting for time-based changes in the labels of ml vulnerability detection approaches. Proceedings of the ACM on Software Engineering **2**(FSE), 335–357 (2025) <https://doi.org/10.1145/3715731>
- [569] Wang, H., Xing, Z., Sun, C., Wang, Z., Tan, S.H.: Towards diverse program transformations for program simplification. Proceedings of the ACM on Software Engineering **2**(FSE), 312–334 (2025) <https://doi.org/10.1145/3715730>
- [570] Hasan, M.R., Hasan, M.R., Bagheri, H.: Unlocking optimal orm database designs: Accelerated tradeoff analysis with transformers. Proceedings of the ACM on Software Engineering **2**(FSE), 1639–1662 (2025) <https://doi.org/10.1145/3729344>
- [571] Wang, Z., Zhou, Z., Song, J., Huang, Y., Shu, Z., Ma, L.: Vlatest: Testing and evaluating vision-language-action models for robotic manipulation. Proceedings of the ACM on Software Engineering **2**(FSE), 1615–1638 (2025) <https://doi.org/10.1145/3729343>
- [572] Qin, M., Zhang, Y., Stol, K.-J., Liu, H.: Who will stop contributing to oss projects? predicting company turnover based on initial behavior. Proceedings of the ACM on Software Engineering **2**(FSE), 2782–2805 (2025) <https://doi.org/10.1145/3729393>
- [573] Liang, K., Liu, Z., Liu, C., Wan, Z., Lo, D., Yang, X.: Zero-shot cross-domain code search without fine-tuning. Proceedings of the ACM on Software Engineering **2**(FSE), 1937–1959 (2025) <https://doi.org/10.1145/3729357>
- [574] Su, Y., Xing, Z., Wang, C., Chen, C., Xu, S.X., Lu, Q., Zhu, L.: Automated soap opera testing directed by llms and scenario knowledge: Feasibility, challenges, and road ahead. Proceedings of the ACM on Software Engineering **2**(FSE), 757–778 (2025) <https://doi.org/10.1145/3715752>
- [575] Nguyen Tung, L., Cho, S., Du, X., Neelofar, N., Terragni, V., Ruberto, S., Aleti, A.: Automated trustworthiness oracle generation for machine learning text classifiers. Proceedings of the ACM on Software Engineering **2**(FSE), 2382–2405 (2025) <https://doi.org/10.1145/3729376>
- [576] Gao, Y., Hu, X., Yang, X., Xia, X.: Automated unit test refactoring. Proceedings of the ACM on Software Engineering **2**(FSE), 713–733 (2025) <https://doi.org/10.1145/3715750>

- [577] Steenhoek, B., Sivaraman, S., Saldivar, R., Mohylevskyy, Y., Zilouchian Moghaddam, R., Le, W.: Closing the Gap: A User Study on the Real-world Usefulness of AI-powered Vulnerability Detection & Repair in the IDE. figshare (2025). <https://doi.org/10.6084/M9.FIGSHARE.26367139> . [https://figshare.com/articles/dataset/Closing\\_the\\_Gap\\_A\\_User\\_Study\\_on\\_the\\_Real-world\\_Usefulness\\_of\\_AI-powered\\_Vulnerability\\_Detection\\_Repair\\_in\\_the\\_IDE/26367139](https://figshare.com/articles/dataset/Closing_the_Gap_A_User_Study_on_the_Real-world_Usefulness_of_AI-powered_Vulnerability_Detection_Repair_in_the_IDE/26367139)
- [578] Pourasad, A.E., Maalej, W.: Does genai make usability testing obsolete? In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 437–449. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00138> . <http://dx.doi.org/10.1109/ICSE55347.2025.00138>
- [579] Wang, C., Liu, J., Peng, X., Liu, Y., Lou, Y.: Boosting static resource leak detection via llm-based resource-oriented intention inference. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2905–2917. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00131> . <http://dx.doi.org/10.1109/ICSE55347.2025.00131>
- [580] Cheng, X., Sang, F., Zhai, Y., Zhang, X., Kim, T.: Rug: Turbo llm for rust unit test generation. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2983–2995. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00097> . <http://dx.doi.org/10.1109/ICSE55347.2025.00097>
- [581] Deligiannis, P., Lal, A., Mehrotra, N., Poddar, R., Rastogi, A.: Rustassistant: Using llms to fix compilation errors in rust code. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 3097–3109. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00022> . <http://dx.doi.org/10.1109/ICSE55347.2025.00022>
- [582] Zhang, K., Wang, S., Han, J., Zhu, X., Li, X., Wang, S., Wen, S.: Your fix is my exploit: Enabling comprehensive dl library api fuzzing with large language models. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 3110–3122. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00041> . <http://dx.doi.org/10.1109/ICSE55347.2025.00041>
- [583] Xiao, Y., Liu, A., Zhang, X., Zhang, T., Li, T., Liang, S., Liu, X., Liu, Y., Tao, D.: Bdefects4nn: A backdoor defect database for controlled localization studies in neural networks. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 3123–3135. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00069> . <http://dx.doi.org/10.1109/ICSE55347.2025.00069>
- [584] Alian, P., Nashid, N., Shahbandeh, M., Shabani, T., Mesbah, A.: Feature-driven end-to-end test generation. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 450–462. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00141> . <http://dx.doi.org/10.1109/ICSE55347.2025.00141>

- [585] Wu, Y., Wang, Y., Li, Y., Tao, W., Yu, S., Yang, H., Jiang, W., Li, J.: An empirical study on commit message generation using llms via in-context learning. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 553–565. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00091> . <http://dx.doi.org/10.1109/ICSE55347.2025.00091>
- [586] Zhang, T., Yu, Y., Mao, X., Wang, S., Yang, K., Lu, Y., Zhang, Z., Zhao, Y.: Instruct or interact? exploring and eliciting llms' capability in code snippet adaptation through prompt engineering. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 566–577. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00104> . <http://dx.doi.org/10.1109/ICSE55347.2025.00104>
- [587] Gao, S., Gao, C., Gu, W., Lyu, M.R.: Search-based llms for code optimization. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 578–590. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00021> . <http://dx.doi.org/10.1109/ICSE55347.2025.00021>
- [588] Zhang, Y., Xie, Y., Lit, S., Liu, K., Wang, C., Jia, Z., Huang, X., Song, J., Luo, C., Zheng, Z., Xu, R., Liu, Y., Zheng, S., Liao, X.: Unseen horizons: Unveiling the real capability of llm code generation beyond the familiar. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 604–615. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00082> . <http://dx.doi.org/10.1109/ICSE55347.2025.00082>
- [589] Mao, Z., Wang, J., Sun, J., Qin, S., Xiong, J.: Llm-aided automatic modeling for security protocol verification. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 642–654. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00197> . <http://dx.doi.org/10.1109/ICSE55347.2025.00197>
- [590] Zhou, S., Li, T., Wang, K., Huang, Y., Shi, L., Liu, Y., Wang, H.: Understanding the effectiveness of coverage criteria for large language models: A special angle from jailbreak attacks. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 730–742. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00209> . <http://dx.doi.org/10.1109/ICSE55347.2025.00209>
- [591] Yan, Y., Duong, V., Shao, H., Poshyvanyk, D.: Towards more trustworthy deep code models by enabling out-of-distribution detection. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 769–781. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00177> . <http://dx.doi.org/10.1109/ICSE55347.2025.00177>
- [592] Sakkas, G., Sahu, P., Ong, K., Jhala, R.: Neurosymbolic modular refinement type inference. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 846–858. IEEE, ??? (2025). <https://doi.org/10.1109/>

[icse55347.2025.00090](http://dx.doi.org/10.1109/ICSE55347.2025.00090) . <http://dx.doi.org/10.1109/ICSE55347.2025.00090>

- [593] Ma, L., Liu, S., Li, Y., Xie, X., Bu, L.: Specgen: Automated generation of formal program specifications via large language models. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 16–28. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00129> . <http://dx.doi.org/10.1109/ICSE55347.2025.00129>
- [594] Suh, H., Tafreshipour, M., Li, J., Bhattacharjee, A., Ahmed, I.: An empirical study on automatically detecting ai-generated source code: How far are we? In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 859–871. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00064> . <http://dx.doi.org/10.1109/ICSE55347.2025.00064>
- [595] Patel, S., Yadavally, A., Dhulipala, H., Nguyen, T.N.: Planning a large language model for static detection of runtime errors in code snippets. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 872–884. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00102> . <http://dx.doi.org/10.1109/ICSE55347.2025.00102>
- [596] Wang, C., Huang, K., Zhang, J., Feng, Y., Zhang, L., Liu, Y., Peng, X.: Llms meet library evolution: Evaluating deprecated api usage in llm-based code completion. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 885–897. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00245> . <http://dx.doi.org/10.1109/ICSE55347.2025.00245>
- [597] Ouyang, S., Zhang, J.M., Sun, Z., Penuela, A.M.: Knowledge-enhanced program repair for data science code. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 898–910. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00246> . <http://dx.doi.org/10.1109/ICSE55347.2025.00246>
- [598] Ma, Z., Kim, D.J., Chen, T.-H.P.: Librelog: Accurate and efficient unsupervised log parsing using open-source large language models. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 924–936. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00103> . <http://dx.doi.org/10.1109/ICSE55347.2025.00103>
- [599] Li, X., Wang, S., Li, S., Ma, J., Yu, J., Liu, X., Wang, J., Ji, B., Zhang, W.: Model editing for llms4code: How far are we? In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 937–949. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00049> . <http://dx.doi.org/10.1109/ICSE55347.2025.00049>
- [600] Tinnes, C., Welter, A., Apel, S.: Software model evolution with large language models: Experiments on simulated, public, and industrial datasets. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE),

- pp. 950–962. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00112> . <http://dx.doi.org/10.1109/ICSE55347.2025.00112>
- [601] Ruan, H., Zhang, Y., Roychoudhury, A.: Specrover: Code intent extraction via llms. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 963–974. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00080> . <http://dx.doi.org/10.1109/ICSE55347.2025.00080>
- [602] Le, V.-H., Xiao, Y., Zhang, H.: Unleashing the true potential of semantic-based log parsing with pre-trained language models. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 975–987. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00174> . <http://dx.doi.org/10.1109/ICSE55347.2025.00174>
- [603] Panichella, A.: Metamorphic-based many-objective distillation of llms for code-related tasks. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1001–1013. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00230> . <http://dx.doi.org/10.1109/ICSE55347.2025.00230>
- [604] Zhou, S., Wang, J., Ye, H., Zhou, H., Goues, C.L., Luo, X.: Lwdiff: an llm-assisted differential testing framework for webassembly runtimes. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 153–164. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00233> . <http://dx.doi.org/10.1109/ICSE55347.2025.00233>
- [605] Ke, K.: Niodebugger: A novel approach to repair non-idempotent-outcome tests with llm-based agent. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1014–1025. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00226> . <http://dx.doi.org/10.1109/ICSE55347.2025.00226>
- [606] Nan, Z., Guo, Z., Liu, K., Xia, X.: Test intention guided llm-based unit test generation. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1026–1038. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00243> . <http://dx.doi.org/10.1109/ICSE55347.2025.00243>
- [607] Yin, X., Ni, C., Xu, X., Yang, X.: What you see is what you get: Attention-based self-guided automatic unit test generation. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1039–1051. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00105> . <http://dx.doi.org/10.1109/ICSE55347.2025.00105>
- [608] Janecek, M., Ezzati-Jivan, N., Hamou-Lhadj, A.: Execution trace reconstruction using diffusion-based generative models. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1077–1088. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00063> . <http://dx.doi.org/10.1109/ICSE55347.2025.00063>

- [609] Yadavally, A., Rong, X., Nguyen, P., Nguyen, T.N.: Large language models for safe minimization. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1114–1126. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00203> . <http://dx.doi.org/10.1109/ICSE55347.2025.00203>
- [610] Guo, Q., Xie, X., Liu, S., Hu, M., Li, X., Bu, L.: Intention is all you need: Refining your code from your intention. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1127–1139. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00191> . <http://dx.doi.org/10.1109/ICSE55347.2025.00191>
- [611] Wang, Y., Wang, Y., Guo, D., Chen, J., Zhang, R., Ma, Y., Zheng, Z.: Rlcoder: Reinforcement learning for repository-level code completion. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1140–1152. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00014> . <http://dx.doi.org/10.1109/ICSE55347.2025.00014>
- [612] Macedo, M., Tian, Y., Nie, P., Cogo, F.R., Adams, B.: Intertrans: Leveraging transitive intermediate translations to enhance llm-based code translation. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1153–1164. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00236> . <http://dx.doi.org/10.1109/ICSE55347.2025.00236>
- [613] Sens, Y., Knopp, H., Peldszus, S., Berger, T.: A large-scale study of model integration in ml-enabled software systems. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1165–1177. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00185> . <http://dx.doi.org/10.1109/ICSE55347.2025.00185>
- [614] Shao, Y., Huang, Y., Shen, J., Ma, L., Su, T., Wan, C.: Are llms correctly integrated into software systems? In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1178–1190. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00204> . <http://dx.doi.org/10.1109/ICSE55347.2025.00204>
- [615] Huq, S.F., Tafreshipour, M., Kalcevich, K., Malek, S.: Automated generation of accessibility test reports from recorded user transcripts. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 204–216. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00043> . <http://dx.doi.org/10.1109/ICSE55347.2025.00043>
- [616] Batole, F., OBrien, D., Nguyen, T.N., Dyer, R., Rajan, H.: An llm-based agent-oriented approach for automated code design issue localization. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1320–1332. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00100> . <http://dx.doi.org/10.1109/ICSE55347.2025.00100>

- [617] Li, Y., Wu, Y., Liu, J., Jiang, Z., Chen, Z., Yu, G., Lyu, M.R.: Coca: Generative root cause analysis for distributed systems with code knowledge. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1346–1358. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00234> . <http://dx.doi.org/10.1109/ICSE55347.2025.00234>
- [618] Di, Y., Zhang, T.: Enhancing code generation via bidirectional comment-level mutual grounding. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1359–1371. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00165> . <http://dx.doi.org/10.1109/ICSE55347.2025.00165>
- [619] Zheng, D., Wang, Y., Shi, E., Zhang, R., Ma, Y., Zhang, H., Zheng, Z.: Humanevo: An evolution-aware benchmark for more realistic evaluation of repository-level code generation. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1372–1384. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00228> . <http://dx.doi.org/10.1109/ICSE55347.2025.00228>
- [620] Schwedt, S., Ströder, T.: From bugs to benefits: Improving user stories by leveraging crowd knowledge with cruise-ac. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1385–1395. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00217> . <http://dx.doi.org/10.1109/ICSE55347.2025.00217>
- [621] Fuchß, D., Hey, T., Keim, J., Liu, H., Ewald, N., Thirolf, T., Kozolek, A.: Lissa: Toward generic traceability link recovery through retrieval-augmented generation. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1396–1408. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00186> . <http://dx.doi.org/10.1109/ICSE55347.2025.00186>
- [622] Kim, M., Stennett, T., Sinha, S., Orso, A.: A multi-agent approach for rest api testing with semantic graphs and llm-driven inputs. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1409–1421. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00179> . <http://dx.doi.org/10.1109/ICSE55347.2025.00179>
- [623] Gao, H., Yang, Y., Sun, M., Wu, J., Zhou, Y., Xu, B.: Clozemaster: Fuzzing rust compiler by harnessing llms for infilling masked real programs. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1422–1435. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00175> . <http://dx.doi.org/10.1109/ICSE55347.2025.00175>
- [624] Li, J., Dong, Z., Wang, C., You, H., Zhang, C., Liu, Y., Peng, X.: Llm based input space partitioning testing for library apis. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1436–1448. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00153> . <http://dx.doi.org/10.1109/ICSE55347.2025.00153>

- [625] Deljouyi, A., Koohestani, R., Izadi, M., Zaidman, A.: Leveraging large language models for enhancing the understandability of generated unit tests. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1449–1461. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00032> . <http://dx.doi.org/10.1109/ICSE55347.2025.00032>
- [626] Lee, C.-T., Neeser, A., Xu, S., Katyan, J., Cross, P., Pathakota, S., Norman, M., Simeone, J., Chandrasekaran, J., Ramakrishnan, N.: Can an llm find its way around a spreadsheet? In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 294–306. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00101> . <http://dx.doi.org/10.1109/ICSE55347.2025.00101>
- [627] Zhang, J., Liu, Y., Nie, P., Li, J.J., Gligoric, M.: exlong: Generating exceptional behavior tests with large language models. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1462–1474. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00176> . <http://dx.doi.org/10.1109/ICSE55347.2025.00176>
- [628] Hossain, S.B., Dwyer, M.B.: Togll: Correct and strong test oracle generation with llms. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1475–1487. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00098> . <http://dx.doi.org/10.1109/ICSE55347.2025.00098>
- [629] Toma, T.R., Grewal, B., Bezemer, C.-P.: Answering user questions about machine learning models through standardized model cards. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1488–1500. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00066> . <http://dx.doi.org/10.1109/ICSE55347.2025.00066>
- [630] Tian, Z., Chen, J., Zhang, X.: Fixing large language models’ specification misunderstanding for better code generation. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1514–1526. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00108> . <http://dx.doi.org/10.1109/ICSE55347.2025.00108>
- [631] Lin, F., Kim, D.J., Chen, T.-H.: Soen-101: Code generation by emulating software process models using large language model agents. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1527–1539. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00140> . <http://dx.doi.org/10.1109/ICSE55347.2025.00140>
- [632] Nahar, N., Zhang, H., Lewis, G., Zhou, S., Kästner, C.: The product beyond the model - an empirical study of repositories of open-source ml products. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1540–1552. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00006> . <http://dx.doi.org/10.1109/ICSE55347.2025.00006>

- [633] Li, Z., Wu, J., Ling, X., Luo, T., Rui, Z., Wu, Y.: The seeds of the future sprout from history: Fuzzing for unveiling vulnerabilities in prospective deep-learning libraries. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1616–1627. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00132> . <http://dx.doi.org/10.1109/ICSE55347.2025.00132>
- [634] Shi, Y., Zhang, H., Wan, C., Gu, X.: Between lines of code: Unraveling the distinct patterns of machine and human programmers. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1628–1639. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00005> . <http://dx.doi.org/10.1109/ICSE55347.2025.00005>
- [635] Tufano, R., Martin-Lopez, A., Tayeb, A., Dabić, O., Haiduc, S., Bavota, G.: Deep learning-based code reviews: A paradigm shift or a double-edged sword? In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1640–1652. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00060> . <http://dx.doi.org/10.1109/ICSE55347.2025.00060>
- [636] Gomes, L.F., Hellendoorn, V.J., Aldrich, J., Abreu, R.: An exploratory study of ml sketches and visual code assistants. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1653–1664. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00124> . <http://dx.doi.org/10.1109/ICSE55347.2025.00124>
- [637] Jiang, X., Dong, Y., Tao, Y., Liu, H., Jin, Z., Li, G.: Rocode: Integrating backtracking mechanism and program analysis in large language models for code generation. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 334–346. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00133> . <http://dx.doi.org/10.1109/ICSE55347.2025.00133>
- [638] Xu, W., Gao, K., He, H., Zhou, M.: Licoeval: Evaluating llms on license compliance in code generation. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1665–1677. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00052> . <http://dx.doi.org/10.1109/ICSE55347.2025.00052>
- [639] Sabouri, S., Eibl, P., Zhou, X., Ziyadi, M., Medvidovic, N., Lindemann, L., Chattopadhyay, S.: Trust dynamics in ai-assisted development: Definitions, factors, and implications. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1678–1690. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00199> . <http://dx.doi.org/10.1109/ICSE55347.2025.00199>
- [640] Lian, X., Chen, Y., Cheng, R., Huang, J., Thakkar, P., Zhang, M., Xu, T.: Large language models as configuration validators. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1704–1716. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00017> . <http://dx.doi.org/10.1109/ICSE55347.2025.00017>

- [641] Mohammed, N., Lal, A., Rastogi, A., Sharma, R., Roy, S.: Llm assistance for memory safety. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1717–1728. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00023> . <http://dx.doi.org/10.1109/ICSE55347.2025.00023>
- [642] Ding, Y., Fu, Y., Ibrahim, O., Sitawarin, C., Chen, X., Alomair, B., Wagner, D., Ray, B., Chen, Y.: Vulnerability detection with code language models: How far are we? In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1729–1741. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00038> . <http://dx.doi.org/10.1109/ICSE55347.2025.00038>
- [643] Ma, W., Wu, D., Sun, Y., Wang, T., Liu, S., Zhang, J., Xue, Y., Liu, Y.: Combining fine-tuning and llm-based agents for intuitive smart contract auditing with justifications. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1742–1754. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00027> . <http://dx.doi.org/10.1109/ICSE55347.2025.00027>
- [644] Chakraborty, S., Ebner, G., Bhat, S., Fakhoury, S., Fatima, S., Lahiri, S., Swamy, N.: Towards neural synthesis for smt-assisted proof-oriented programming. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1755–1767. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00002> . <http://dx.doi.org/10.1109/ICSE55347.2025.00002>
- [645] Pedro, R., Coimbra, M.E., Castro, D., Carreira, P., Santos, N.: Prompt-to-sql injections in llm-integrated web applications: Risks and defenses. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1768–1780. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00007> . <http://dx.doi.org/10.1109/ICSE55347.2025.00007>
- [646] Shabani, T., Nashid, N., Alian, P., Mesbah, A.: Dockerfile flakiness: Characterization and repair. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1793–1805. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00238> . <http://dx.doi.org/10.1109/ICSE55347.2025.00238>
- [647] Rong, G., Yu, Y., Liu, S., Tan, X., Zhang, T., Shen, H., Hu, J.: Code comment inconsistency detection and rectification using a large language model. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1832–1843. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00035> . <http://dx.doi.org/10.1109/ICSE55347.2025.00035>
- [648] Thompson, K., Saavedra, N., Carrott, P., Fisher, K., Sanchez-Stern, A., Brun, Y., Ferreira, J.F., Lerner, S., First, E.: Rango: Adaptive retrieval-augmented proving for automated software verification. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 347–359. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00161> . <http://dx.doi.org/10.1109/ICSE55347.2025.00161>

- [649] Imani, A., Ahmed, I., Moshirpour, M.: Context conquers parameters: Outperforming proprietary llm in commit message generation. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1844–1856. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00048> . <http://dx.doi.org/10.1109/ICSE55347.2025.00048>
- [650] Chen, G., Xie, X., Tang, D., Xin, Q., Liu, W.: Hedgecode: A multi-task hedging contrastive learning framework for code search. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1857–1868. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00008> . <http://dx.doi.org/10.1109/ICSE55347.2025.00008>
- [651] Chen, J., Pan, Z., Hu, X., Li, Z., Li, G., Xia, X.: Reasoning runtime behavior of a program with llm: How far are we? In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1869–1881. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00012> . <http://dx.doi.org/10.1109/ICSE55347.2025.00012>
- [652] Sun, W., Miao, Y., Li, Y., Zhang, H., Fang, C., Liu, Y., Deng, G., Liu, Y., Chen, Z.: Source code summarization in the era of large language models. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1882–1894. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00034> . <http://dx.doi.org/10.1109/ICSE55347.2025.00034>
- [653] Huang, K., Zhang, J., Meng, X., Liu, Y.: Template-guided program repair in the era of large language models. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1895–1907. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00030> . <http://dx.doi.org/10.1109/ICSE55347.2025.00030>
- [654] Sun, Y., Poskitt, C.M., Wang, K., Sun, J.: Fixdrive: Automatically repairing autonomous vehicle driving behaviour for \$0.08 per violation. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 1921–1933. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00216> . <http://dx.doi.org/10.1109/ICSE55347.2025.00216>
- [655] Erhabor, D., Udayashankar, S., Nagappan, M., Al-Kiswany, S.: Measuring the runtime performance of c++ code written by humans using github copilot. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2062–2074. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00059> . <http://dx.doi.org/10.1109/ICSE55347.2025.00059>
- [656] Yang, S., Lin, X., Chen, J., Zhong, Q., Xiao, L., Huang, R., Wang, Y., Zheng, Z.: Hyperion: Unveiling dapp inconsistencies using llm and dataflow-guided symbolic execution. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2125–2137. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00015> . <http://dx.doi.org/10.1109/ICSE55347.2025.00015>

- [657] Bouzenia, I., Devanbu, P., Pradel, M.: Repairagent: An autonomous, llm-based agent for program repair. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2188–2200. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00157> . <http://dx.doi.org/10.1109/ICSE55347.2025.00157>
- [658] Wang, S., Yu, Y., Feldt, R., Parthasarathy, D.: Automating a complete software test process using llms: An automotive case study. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 373–384. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00211> . <http://dx.doi.org/10.1109/ICSE55347.2025.00211>
- [659] Shen, Q., Tian, Y., Ma, H., Chen, J., Huang, L., Fu, R., Cheung, S.-C., Wang, Z.: A tale of two dl cities: When library tests meet compiler. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2201–2212. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00025> . <http://dx.doi.org/10.1109/ICSE55347.2025.00025>
- [660] Hundal, R.S., Xiao, Y., Cao, X., Dong, J.S., Rigger, M.: On the mistaken assumption of interchangeable deep reinforcement learning implementations. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2225–2237. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00222> . <http://dx.doi.org/10.1109/ICSE55347.2025.00222>
- [661] Thomas, D.-G., Biagiola, M., Humbatova, N., Wardat, M., Jahangirova, G., Rajan, H., Tonella, P.:  $\mu$ PRL: A mutation testing pipeline for deep reinforcement learning based on real faults. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2238–2250 (2025). <https://doi.org/10.1109/ICSE55347.2025.00036>
- [662] Zeng, Q., Zhang, Y., Qiu, Z., Liu, H.: A first look at conventional commits classification. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2277–2289. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00011> . <http://dx.doi.org/10.1109/ICSE55347.2025.00011>
- [663] Tamanna, S.B., Uddin, G., Wang, S., Xia, L., Zhang, L.: Chatgpt inaccuracy mitigation during technical report understanding: Are we there yet? In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2290–2302. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00145> . <http://dx.doi.org/10.1109/ICSE55347.2025.00145>
- [664] Yuan, M., Chen, J., Xing, Z., Quigley, A., Luo, Y., Luo, T., Mohammadi, G., Lu, Q., Zhu, L.: Designrepair: Dual-stream design guideline-aware frontend repair with large language models. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2483–2494. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00109> . <http://dx.doi.org/10.1109/ICSE55347.2025.00109>

- [665] Liang, H., Huang, Y., Chen, T.: The same only different: On information modality for configuration performance analysis. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2522–2534. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00212> . <http://dx.doi.org/10.1109/ICSE55347.2025.00212>
- [666] Xu, J., Fu, Y., Tan, S.H., He, P.: Aligning the objective of llm-based program repair. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2548–2560. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00169> . <http://dx.doi.org/10.1109/ICSE55347.2025.00169>
- [667] Yang, A.Z.H., Kolak, S., Hellendoorn, V., Martins, R., Goues, C.L.: Revisiting unnaturalness for automated program repair in the era of large language models. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2561–2573. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00089> . <http://dx.doi.org/10.1109/ICSE55347.2025.00089>
- [668] Parasaram, N., Yan, H., Yang, B., Flahy, Z., Qudsi, A., Ziaber, D., Barr, E.T., Mechtaev, S.: The fact selection problem in llm-based program repair. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2574–2586. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00162> . <http://dx.doi.org/10.1109/ICSE55347.2025.00162>
- [669] Duvvuru, V.S.A., Zhang, B., Vierhauser, M., Agrawal, A.: Llm-agents driven automated simulation testing and analysis of small uncrewed aerial systems. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 385–397. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00223> . <http://dx.doi.org/10.1109/ICSE55347.2025.00223>
- [670] Wang, Z., Zhou, Z., Song, D., Huang, Y., Chen, S., Ma, L., Zhang, T.: Towards understanding the characteristics of code generation errors made by large language models. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2587–2599. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00180> . <http://dx.doi.org/10.1109/ICSE55347.2025.00180>
- [671] Zahan, N., Burckhardt, P., Lysenko, M., Aboukhadijeh, F., Williams, L.: Leveraging large language models to detect npm malicious packages. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2625–2637. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00146> . <http://dx.doi.org/10.1109/ICSE55347.2025.00146>
- [672] Fratantonio, Y., Invernizzi, L., Farah, L., Thomas, K., Zhang, M., Albertini, A., Galilee, F., Metitieri, G., Cretin, J., Petit-Bianco, A., Tao, D., Bursztein, E.: Magika: Ai-powered content-type detection. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2638–2649. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00158> . <http://dx.doi.org/10.1109/ICSE55347.2025.00158>

- [673] Souza, B., Pradel, M.: Treefix: Enabling execution with a tree of prefixes. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2676–2688. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00215> . <http://dx.doi.org/10.1109/ICSE55347.2025.00215>
- [674] Dong, C., Jiang, Y., Zhang, Y., Zhang, Y., Liu, H.: Chatgpt-based test generation for refactoring engines enhanced by feature analysis on examples. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2714–2725. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00210> . <http://dx.doi.org/10.1109/ICSE55347.2025.00210>
- [675] You, H., Wang, Z., Lin, B., Chen, J.: Navigating the testing of evolving deep learning systems: An exploratory interview study. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2726–2738. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00106> . <http://dx.doi.org/10.1109/ICSE55347.2025.00106>
- [676] Zhang, Y., Chen, S., Xie, X., Liu, Z., Fan, L.: Scenario-driven and context-aware automated accessibility testing for android apps. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2777–2789. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00093> . <http://dx.doi.org/10.1109/ICSE55347.2025.00093>
- [677] Yong, H., Li, Z., Pan, M., Zhang, T., Zhao, J., Li, X.: Gvi: Guided vulnerability imagination for boosting deep vulnerability detectors. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2867–2879. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00214> . <http://dx.doi.org/10.1109/ICSE55347.2025.00214>
- [678] Nie, Y., Wang, C., Wang, K., Xu, G., Xu, G., Wang, H.: Decoding secret memorization in code llms through token-level characterization. In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 2880–2892. IEEE, ??? (2025). <https://doi.org/10.1109/icse55347.2025.00229> . <http://dx.doi.org/10.1109/ICSE55347.2025.00229>
- [679] Hassan, Z., Treude, C., Norrish, M., Williams, G., Potanin, A.: Characterising reproducibility debt in scientific software: A systematic literature review. Journal of Systems and Software **222**, 112327 (2025) <https://doi.org/10.1016/j.jss.2024.112327>
- [680] Sallou, J., Durieux, T., Panichella, A.: Breaking the silence: the threats of using llms in software engineering. In: Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results. Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3639476.3639764> . <https://doi.org/10.1145/3639476.3639764>
- [681] ACM Publications Board Task Force on Data, Software, and Reproducibility:

- Artifact Review and Badging Version 1.1. <https://www.acm.org/publications/policies/artifact-review-badging> (2020)
- [682] National Information Standards Organization (NISO): Reproducibility Badging and Definitions. <https://www.niso.org/standards-committees/reproducibility-badging>. Accessed: 2025-10-20 (2020)
- [683] Institute of Electrical and Electronics Engineers (IEEE): Overview of IEEE Xplore: About Content. <https://ieeexplore.ieee.org/Xplorehelp/overview-of-ieee-xplore/about-content>. Accessed: 2025-11-19 (2025)
- [684] Rothermel, G., Brun, Y.: Why Publish in ACM TOSEM in the Twenties? <https://sigsoft.medium.com/why-publish-in-acm-tosem-in-the-twenties-8e7326f264ab>. Accessed: 2025-10-20 (2022). <https://sigsoft.medium.com/why-publish-in-acm-tosem-in-the-twenties-8e7326f264ab>
- [685] EMSE Editorial Board: EMSE Open Science Initiative. <https://github.com/emsejournal/openscience>. Accessed: 2025-10-20 (2023)
- [686] JSS Editorial Board: Journal of Systems and Software: Guide for Authors – Open Science. Elsevier. Accessed: 2025-11-19
- [687] IST Editorial Board: Information and Software Technology: Guide for Authors. Elsevier. Accessed: 2025-11-19
- [688] ICSME 2020 Artifact Evaluation Committee: ICSME 2020 Artifacts and ROSE (Recognizing and Rewarding Open Science in SE). <https://github.com/se-conf/icsme20-artifacts-and-rose>. Accessed: 2025-10-20 (2020)
- [689] ICSA 2022 Organizing Committee: ICSA 2022 Artifact Evaluation Track. <https://icsa-conferences.org/2022/conference-tracks/artifact-evaluation-track/>. Accessed: 2025-10-20 (2022)
- [690] SANER 2023 Organizing Committee: Reproducibility Studies and Negative Results Track – SANER 2023. <https://saner2023.must.edu.mo/negativerestrack>. Accessed: 2025-10-20 (2023)
- [691] NeurIPS 2021 Program Chairs: Introducing the NeurIPS 2021 Paper Checklist. <https://blog.neurips.cc/2021/03/26/introducing-the-neurips-2021-paper-checklist/>. Accessed: 2025-10-20 (2021)
- [692] ICML 2020 Program Committee: Style & Author Instructions for ICML 2020. <https://icml.cc/Conferences/2020/StyleAuthorInstructions>. Accessed: 2025-10-20 (2020)
- [693] ICLR 2026 Organizing Committee: Author Guide for ICLR 2026. <https://iclr.cc/>

[cc/Conferences/2026/AuthorGuide](https://creativecommons.org/licenses/by/4.0/). Accessed: 2025-10-20 (2025)

- [694] Magnusson, I., Smith, N.A., Dodge, J.: Reproducibility in nlp: What have we learned from the checklist? In: Findings of the Association for Computational Linguistics: ACL 2023, pp. 12789–12811. Association for Computational Linguistics, Toronto, Canada (2023). <https://aclanthology.org/2023.findings-acl.809.pdf>
- [695] COLT 2023 Organizing Committee: COLT 2023 Call for Papers. <https://www.learningtheory.org/colt2023/>. Accessed: 2025-10-20 (2023)
- [696] KR 2025 Organizing Committee: KR 2025 Call for Papers. <https://kr.org/KR2025/>. Accessed: 2025-10-20 (2025)
- [697] KR 2024 Organizing Committee: KR 2024 Call for Papers. <https://kr.org/KR2024/>. Accessed: 2025-10-20 (2024)
- [698] arXiv Labs Team: New arXivLabs feature provides instant access to code. <https://blog.arxiv.org/2020/10/08/new-arxivlabs-feature-provides-instant-access-to-code> (2020)
- [699] Grattafiori, A., Dubey, A., Jauhri, A., al., A.P.: The Llama 3 Herd of Models (2024). <https://arxiv.org/abs/2407.21783>
- [700] Jiang, A.Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D.S., Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L.R., Lachaux, M.-A., Stock, P., Scao, T.L., Lavril, T., Wang, T., Lacroix, T., Sayed, W.E.: Mistral 7B (2023). <https://arxiv.org/abs/2310.06825>
- [701] OpenAI: Chat Completions. Accessed Mar 25, 2023 (2023). <https://platform.openai.com/docs/guides/chat>
- [702] Anthropic: Claude 3 haiku: Our fastest model yet. <https://www.anthropic.com/news/clause-3-haiku>. Accessed: 2024-03-13 (2024)
- [703] Baltes, S., Angermeir, F., Arora, C., Barón, M.M., Chen, C., Böhme, L., Calefato, F., Ernst, N., Falessi, D., Fitzgerald, B., Fucci, D., Kalinowski, M., Lambiase, S., Russo, D., Lungu, M., Prechelt, L., Ralph, P., Tonder, R., Treude, C., Wagner, S.: Guidelines for Empirical Studies in Software Engineering involving Large Language Models (2025). <https://arxiv.org/abs/2508.15503>
- [704] Baker, M.: 1,500 scientists lift the lid on reproducibility. Nature **533**(7604), 452–454 (2016) <https://doi.org/10.1038/533452a>
- [705] Krishnamurthi, S., Vitek, J.: The real software crisis: Repeatability as a core value. Communications of the ACM **58**(3), 34–36 (2015) <https://doi.org/10.1145/2723673>

- [706] González-Barahona, J.M., Robles, G.: On the reproducibility of empirical software engineering studies based on data retrieved from development repositories. *Empirical Software Engineering* **17**(1-2), 75–89 (2012) <https://doi.org/10.1007/s10664-010-9154-4>
- [707] Gonzalez-Barahona, J., Robles, G.: Revisiting the reproducibility of empirical software engineering studies based on data retrieved from development repositories. *Information and Software Technology* **164**, 107318 (2023) <https://doi.org/10.1016/j.infsof.2023.107318>
- [708] Hermann, B., Winter, S., Siegmund, J.: Community expectations for research artifacts and evaluation processes. In: Proceedings of the 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), pp. 469–480. ACM, ??? (2020). <https://doi.org/10.1145/3368089.3409767>
- [709] Timperley, C.S., Herckis, L., Le Goues, C., Hilton, M.: Understanding and improving artifact sharing in software engineering research. Springer (2021). <https://doi.org/10.1007/s10664-021-09973-5> . <http://dx.doi.org/10.1007/s10664-021-09973-5>
- [710] Liu, M., Huang, X., He, W., Xie, Y., Zhang, J.M., Jing, X., Chen, Z., Ma, Y.: Research artifacts in software engineering publications: Status and trends. Elsevier Science Inc., USA (2024). <https://doi.org/10.1016/j.jss.2024.112032> . <https://doi.org/10.1016/j.jss.2024.112032>
- [711] Abualhaija, S., Aydemir, F.B., Dalpiaz, F., Dell’Anna, D., Ferrari, A., Franch, X., Fucci, D.: Replication in requirements engineering: The nlp for re case. *ACM Transactions on Software Engineering and Methodology* **33**(6) (2024) <https://doi.org/10.1145/3658669>
- [712] Liu, C., Gao, C., Xia, X., Lo, D., Grundy, J., Yang, X.: On the reproducibility and replicability of deep learning in software engineering. *ACM Transactions on Software Engineering and Methodology* **31**(1), 1–46 (2021) <https://doi.org/10.1145/3477535>
- [713] Siddiq, M.L., Dristi, S.B., Saha, J., Santos, J.C.S.: The fault in our stars: Quality assessment of code generation benchmarks. In: 24th IEEE International Conference on Source Code Analysis and Manipulation (SCAM) (2024). <https://doi.org/10.1109/SCAM63643.2024.00028>
- [714] Wolter, M., Veeramacheneni, L., Hoyt, C.T.: More Rigorous Software Engineering Would Improve Reproducibility in Machine Learning Research (2025). <https://arxiv.org/abs/2502.00902>
- [715] Williams, D., Hort, M., Kechagia, M., Aleti, A., Petke, J., Sarro, F.: Reflecting on Empirical and Sustainability Aspects of Software Engineering Research in

the Era of Large Language Models (2025). <https://arxiv.org/abs/2510.26538>

- [716] Huang, X., Zhang, H., Zhou, X., Babar, M.A., Yang, S.: Synthesizing qualitative research in software engineering: A critical review. In: Proceedings of the 40th International Conference on Software Engineering, pp. 1207–1218 (2018)