

Appendix: A survey on large language models for software engineering

Quanjun ZHANG¹, Chunrong FANG^{1*}, Yang XIE¹, Yaxin ZHANG¹, Shengcheng YU¹,
Weisong SUN², Yun YANG³ & Zhenyu CHEN^{1*}

¹State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China

²College of Computing and Data Science, Nanyang Technological University, Singapore 639798, Singapore

³Department of Computing Technologies, Swinburne University of Technology, Melbourne 3122, Australia.

Abstract This appendix provides the detailed data referenced in the paper titled “Survey on Large Language Models for Software Engineering”. The data provided here serves as supplementary material to support the discussions and conclusions outlined in the main text, offering additional insights and validation. For the sake of brevity and due to space limitations, these comprehensive data tables have been excluded from the main body of the paper.

Keywords software engineering, large language model, AI and software engineering, LLM4SE

Citation Zhang Q J, Fang C R, Xie Y, et al. A survey on large language models for software engineering. *Sci China Inf Sci*, 2026, 69(4): 141102, <https://doi.org/10.1007/s11432-025-4670-0>

1 Appendix A: Detailed summarization of existing LLMs of code

In the following, we summarize some representative LLMs of code.

1.1 Encoder-only LLMs

(1) **CuBERT: First Adaption of BERT for Source Code.** CuBERT [1] is the first attempt to apply BERT to source code by replicating the training procedure of BERT on a code corpus. In particular, Kanade et al. [1] construct a massive corpus of 7.4M Python files from GitHub and pre-train CuBERT with masked language modeling and next sentence prediction as the objectives. CuBERT is fine-tuned on six downstream tasks, including five classification tasks and one program repair task, demonstrating its superior performance over LSTM and vanilla Transformer models.

(2) **CodeBERT: Code-aware BERT-based LLM.** CodeBERT [2] represents a successful adaptation of BERT from NLP to the source code domain. CodeBERT follows the BERT architecture (i.e., a multi-layer bidirectional Transformer model), but unlike BERT, which only considers natural language (NL), CodeBERT takes into account both NL and programming language (PL). Regarding input representation, CodeBERT’s input is divided into two parts: NL and PL, forming the format $[CLS], w_1, w_2, \dots, w_n, [SEP], c_1, c_2, \dots, c_m, [EOS]$, where the special marker $[CLS]$ is positioned before these two segments. CodeBERT’s output comprises contextual representations for each token and the representation of $[CLS]$. In the pre-training phase, CodeBERT employs two training objectives: masked language modeling and replaced token detection. The first objective aims to predict the original tokens that are masked, where only bimodal data (NL-PL pairs) are utilized for training. The second objective is optimized to train on both unimodal and multimodal data, implying that the generator uses both NL and PL data.

Compared with CuBERT [1], CodeBERT is more powerful due to several improvements during pre-training. First, CuBERT is pre-trained with code snippets, while CodeBERT is pre-trained with both bimodal NL-PL data and unimodal PL/NL data. Second, CuBERT is only pre-trained with Python,

* Corresponding author (email: fangchunrong@nju.edu.cn, zyichen@nju.edu.cn)

while CodeBERT is pre-trained with six programming languages. Third, CuEBRT follows the objectives of BERT, while CodeBERT is trained with a new learning objective based on replaced token detection.

(3) GraphCodeBERT: Structure-aware Pre-training for Source Code. Although CodeBERT introduces code snippets during pre-training, its training paradigm is still derived from NLP by regarding a code snippet as a sequence of tokens while overlooking the inherent structure of source code. In 2020, Guo et al. [3] introduce GraphCodeBERT, a graph-based LLM built upon the BERT architecture designed for code-related applications. GraphCodeBERT employs a representation approach rooted in data flow learning for code. It involves extracting ASTs through tree-sitter and capturing variables from the ASTs to form a sequence of variables. The relationships between extracted variables, such as data source connections, are used to construct a data flow graph. During the model's pre-training phase, GraphCodeBERT introduces two innovative training tasks alongside the inherited MLM task from CodeBERT, i.e., edge prediction and node alignment. The edge prediction task aims to learn code structural information by predicting edges within the data flow graph, while the node alignment task aims to learn which specific node in the data flow graph corresponds to which code token in the input code. Besides, to accommodate the structure of AST graphs, GraphCodeBERT employs graph-guided masked attention.

1.2 Encoder-decoder LLMs

(1) PYMT5: First Attempt of Encoder-decoder LLM. Similar to CuBERT [1] in the encoder-only LLM domain, as early as 2020, PYMT5 [4] is the first attempt to apply encoder-decoder LLMs to source code by replicating the pre-training process of T5 on a code corpus. PYMT5 is pre-trained with a similar span masking objective from T5 on 26 million Python code snippets and built on an encoder-decoder Transformer with 374 million parameters. PYMT5 is fine-tuned with two tasks, i.e., method and comment generation, demonstrating superior performance against GPT-2.

(2) T5-Learning: Adaption of T5 for Source Code. In parallel with PYMT5 [4], Mastropaolo et al. [5] propose T5-learning, to empirically investigate how the T5 model performs when pre-trained and fine-tuned to support code-related tasks. T5-learning is first pre-trained in a self-supervised way from T5 on CodeSearchNet with both natural language text and programming language code, i.e., masking tokens in code and asking the model to guess the masked tokens. T5-learning is then fine-tuned to support four downstream tasks, i.e., program repair, mutant injection, assertion generation, and code summarization. The results demonstrate that T5-learning outperforms previous baselines, showcasing the potential of T5 in code-related tasks.

(3) PLBART: BART-based LLM for Code. Unlike PYMT5 [4] only focusing on Python code generation, in 2021, Ahmad et al. [6] propose PLBART, an encoder-decoder LLM capable of performing a broad spectrum of code understanding and generation tasks. PLBART is pre-trained with the denoising objective and built on the BART architecture. During the pre-training, PLBART learns to reconstruct an original text that is corrupted using an arbitrary noise function, including three noise strategies in this work, i.e., token masking, token deletion, and token infilling. PLBART is fine-tuned for two categories of four downstream tasks (i.e., code generation, translation, summarization, and classification) across seven programming languages. The experimental results demonstrate that PLBART outperforms previous LLMs, such as CodeBERT and GraphCodeBERT, demonstrating its promise in both code understanding and generation.

(4) CodeT5: Code-aware T5-based LLM. Despite introducing source code, PLBART simply processes code snippets as natural language and ignores the code-specific characteristics. In 2021, Wang et al. [7] introduce CodeT5, a unified encoder-decoder LLM based on the T5 architecture by leveraging the code semantics from the developer-assigned identifiers. CodeT5 considers two types of input representations based on whether a code snippet has a corresponding NL description: unimodal (i.e., PL) and bimodal (i.e., PL-NL pairs) data. To encode the input data, CodeT5 concatenates PL and NL into a whole sequence X with a delimiter token $[SEP]$, i.e., $X = (w_1, \dots, w_n, [SEP], c_1, \dots, c_m, [SEP])$, where n and m denote the number of NL word tokens and PL code tokens, respectively. CodeT5 employs three identifier-aware pre-training tasks (i.e., masked span prediction, masked identifier prediction, and identifier tagging) to consider the crucial token type information and a bimodal dual generation pre-training task to learn a better NL-PL alignment between the code and its accompanying comment. CodeT5 is then fine-tuned with the CodeXGLUE benchmark to perform both code generation and understanding tasks, i.e., code summarization, code generation, code translation, code refinement, defect

detection, and clone detection. The results demonstrate that CodeT5 significantly outperforms previous LLMs in most downstream tasks, such as RoBERTa, CodeBERT, GraphCodeBERT, GPT2, CodeGPT, and PLBART. Overall, CodeT5 represents a successful adaptation of encoder-decoder LLMs from NLP to the source code domain and has been widely used in SE research.

(5) SPT-Code: Code-aware Pre-training for Source Code. However, previous LLMs simply reuse the pre-training tasks designed for NL, while failing to learn the connection between a piece of code and the associated NL for code-related tasks. In May 2022, Niu et al. [8] introduce SPT-Code, which is a sequence-to-sequence LLM designed for source code. When given a complete method, SPT-Code aims to acquire general knowledge from the method's source code, its underlying code structure, and the corresponding natural language description. The input is represented as $\{c_1, \dots, c_l, [SEP], a_1, \dots, a_m, [SEP], n_1, \dots, n_p\}$, where l represents the number of code tokens, m denotes the length of the linearized Abstract Syntax Tree (AST) sequence, and p signifies the number of tokens in the natural language description. SPT-Code introduces three specialized code-specific pre-training tasks, i.e., Code-AST Prediction (CAP), Masked Sequence to Sequence (MASS), and Method Name Generation (MNG). Each of these tasks enables SPT-Code to capture a distinct aspect of the data instance. Specifically, CAP focuses on understanding the source code by masking a random fragment of the code tokens. MNG aims to predict whether a given AST accurately represents a particular code fragment, thereby gaining insights into the syntactic structure. Finally, MNG's objective is to generate subtokens corresponding to the method name, a concise natural language description of the method. These three pre-training tasks are meticulously designed to enable SPT-Code to learn about source code, its underlying structure, and the natural language descriptions associated with it. Importantly, SPT-Code does not rely on any bilingual corpora. This knowledge is leveraged when SPT-Code is applied to downstream tasks, making use of these three informational sources.

(6) CodeRL: CodeT5-derived LLM for program synthesis. Unlike previous general-purpose LLMs, in 2022, Le et al. [9] propose CodeRL, a successor of CodeT5, for the program synthesis task based on deep reinforcement learning. CodeRL is built on top of CodeT5-large architecture with (1) an enlarged pre-training dataset, which has 10.5B tokens and is 10x larger than the CodeSearchNet corpus used in the original CodeT5; and (2) enhanced learning objectives, i.e., masked span prediction and next-token prediction. In particular, CodeRL considers program synthesis as a reinforcement learning problem and applies the actor-critic reinforcement learning method, enhancing CodeT5's performance by leveraging unit test signals during model optimization and generation.

(7) CoditT5: CodeT5-derived LLM for Code Editing. Despite achieving impressive performance in numerous code-related generation tasks, previous LLMs are not well-suited for editing tasks. In 2022, Zhang et al. [10] propose CoditT5, an encoder-decoder LLM for code-related editing tasks based on CodeT5. Initialized from the CodeT5-base model, CoditT5 is pre-trained with an edit-aware pre-training objective on the CodeSearchNet dataset, i.e., generating the edit-based output sequence given the corrupted input sequence. CoditT5 is fine-tuned on three downstream tasks, including comment updating, bug fixing, and automated code review, demonstrating superior performance against previous generation-based LLMs (e.g., PLBART and CodeT5) in tackling code editing tasks, such as program repair.

(7) AlphaCode: Competition-level Code Generation LLM. Despite demonstrating remarkable abilities in code generation, previous LLMs have shown limited success when confronted with competition-level programming problems that require problem-solving skills beyond simply translating instructions into code. In 2022, Li et al. [11] from DeepMind propose AlphaCode, an encoder-decoder LLM specifically designed to generate solutions for competitive programming problems that require deep reasoning. AlphaCode is built on top of an encoder-decoder transformer-based architecture and is pre-trained with 86.31 million files across 13 programming languages from public GitHub repositories. The encoder and decoder are pre-trained with masked language modeling and next-token prediction objectives, respectively. AlphaCode takes the problem description as input to the encoder and generates a code autoregressively from the decoder one token at a time until an end-of-code token is produced. AlphaCode is then fine-tuned with the CodeContests dataset and the results show that AlphaCode performs roughly at the level of the median competitor, i.e., achieving on average a ranking of top 54.3% in competitions with more than 5,000 participants.

(8) CodeT5+: Successor LLM of CodeT5. Although existing LLMs are adept at learning rich contextual representations applicable to a variety of code-related tasks, they often rely on a limited set of pre-training objectives. Such objectives might result in substantial performance degradation in cer-

tain downstream tasks due to the discrepancy between the pre-training and fine-tuning stages. In 2023, Wang et al. [12] present CodeT5+, a successor of CodeT5 where component modules can be flexibly combined to accommodate a wide range of downstream code tasks. CodeT5+ is pre-trained with two objectives (i.e., span denoising and causal language modeling) on unimodal code corpora and two objectives (i.e., text-code contrastive learning and text-code matching) on bimodal text-code corpora. CodeT5+ is built on top of the encoder-decoder Transformer architecture and is classified into two groups according to model size. CodeT5+ 220M and 770M are trained from scratch following T5's architecture and CodeT5+ 2B, 6B, 16B are initialized from off-the-shelf CodeGen checkpoints. The evaluation experiments are conducted on 20 code-related benchmarks under different settings, including zero-shot, fine-tuning, and instruction-tuning. The experimental results demonstrate that CodeT5+ achieves substantial performance on various code-related tasks, such as code generation and completion, math programming, and text-to-code retrieval tasks.

(9) JuPyT5: PyMT5-derived LLM for Jupyter Notebook. Unlike existing LLMs generating code from descriptions, in 2022, Chandel et al. [13] propose JuPyT5, an encoder-decoder LLM designed as a data science assistant for the Jupyter Notebook. JuPyT5 is built on the BART architecture and initialized from a pre-trained PyMT5 checkpoint with the same training hyperparameters. JuPyT5 is then pre-trained with a cell-infilling objective on a Data Science Problems (DSP) dataset, which is constructed from almost all publicly available Jupyter Notebook GitHub repositories. DSP consists of 1119 problems curated from 306 pedagogical notebooks with 92 dataset dependencies, natural language and Markdown problem descriptions, and assert-based unit tests. These problems are designed to assess university students' mastery of various Python implementations in Math and Data Science. The experimental results demonstrate that JuPyT5 achieves a 77.5% success rate in solving DSP problems based on 100 sampling attempts, proving the potential of using LLMs as data science assistants.

(10) ERNIE-Code: Multilingual NL-and-PL LLM. Despite achieving impressive performance in various SE tasks, existing LLMs have been essentially connecting English texts (e.g., comments or docstrings) and multilingual code snippets (e.g., Python and Java). Such an English-centricity issue dramatically limits the application of such LLMs in practice, given that 95% of the world's population are non-native English speakers. In 2023, Chai et al. [14] from Baidu propose ERNIE-Code, which is a unified LLM designed to bridge the gap between multilingual natural languages (NLs) and multilingual programming languages (PLs). The cross-lingual NL-PL ability of ERNIE-Code is learned from two pre-training tasks, i.e., span-corruption language modeling and Pivot-based translation language modeling. The former learns intra-modal patterns from PL or NL only, while the latter learns cross-modal alignment from many NLs and PLs. ERNIE-Code is built on the T5 encoder-decoder architecture and trained on PL corpus (i.e., CodeSearchNe with six PLs), monolingual NL corpus (i.e., CC100 with monolingual NLs), and parallel NL corpus (i.e., OPUS with 105 bilingual pairs). ERNIE-Code is capable of understanding and generating code and text in 116 different NLs and 6 PLs, and outperforms previous LLMs such as PLBART and CodeT5 in various code tasks, such as code-to-text, text-to-code, code-to-code, and text-to-text generation. Importantly, ERNIE-Code demonstrates superior performance in zero-shot prompting for multilingual code summarization and text-to-text translation.

1.3 Decoder-only LLMs

(1) GPT-C: First Attempt LLM for Code Generation. As early as 2020, Svyatkovskiy et al. [15] from Microsoft propose GPT-C, a variant of the GPT-2 trained from scratch on a large unsupervised multilingual source code dataset. GPT-C is a multi-layer generative transformer model trained to predict sequences of code tokens of arbitrary types, generating up to entire lines of syntactically correct code. The pre-training dataset contains 1.2 billion lines of code in Python, C#, JavaScript, and TypeScript. The experimental results demonstrate that GPT-C achieves an average edit similarity of 86.7% on code completion tasks for Python programming language. Importantly, GPT-C is implemented as a cloud-based web service, offering real-time code completion suggestions in Visual Studio Code IDE and Azure Notebook environments.

(2) CodeGPT: A Variant of GPT-2 for Source Code. In 2021, similar to GPT-C, Lu et al. [16] from Microsoft propose CodeGPT, a decoder-only Transformer-based LLM, following the model architecture and training objectives of GPT-2. As one of the baseline LLMs in CodeXGLUE, CodeGPT is designed to support code completion and text-to-code generation tasks. CodeGPT undergoes pre-training on the CodeSearchNet dataset, particularly focusing on the Python and Java corpora. There exist two

versions of CodeGPT, i.e., the original CodeGPT, which is pre-trained from scratch with randomly initialized parameters; and CodeGPT-adapted, which is re-trained from the checkpoint of GPT-2 on the code corpus. The experimental results show that in code completion tasks on the PY150 and Github Java Corpus datasets, CodeGPT achieves a performance of 70.65%, while its enhanced version, CodeGPT-adapted, reaches 71.28%. In the text-to-code generation task on the CONCODE dataset, CodeGPT attains a CodeBLEU performance of 32.71%, and CodeGPT-adapted achieves 35.98%.

(3) Codex: A Descendant of GPT-3 for Code Tasks. Inspired by the considerable success of LLMs (such as GPT-3) in NLP and the abundance of publicly available code, Chen et al. [17] from OpenAI propose Codex, a descendant of GPT-3 model fine-tuned with publicly available code corpus from GitHub. Codex is primarily trained for the task of generating independent Python functions from docstrings. The HumanEval benchmark is constructed to evaluate the functional correctness of generated code with 164 handwritten programming problems, each accompanied by a function signature, docstring, body, and several unit tests. The experimental results demonstrate that Codex exhibits remarkable performance, with its model solving more problems on the HumanEval dataset than GPT-3 and GPT-J, achieving a success rate of 28.8%. Furthermore, Codex can solve 70.2% of the questions using a repeated sampling strategy, with 100 samples per question. This suggests that generating multiple samples from the model and selecting the optimal solution is a highly effective approach for challenging prompts. Importantly, Codex and descendants are deployed in GitHub Copilot, indicating the power of LLMs in transforming the landscape of code-related tasks.

(4) PolyCoder: Open-sourced LLM Comparable to Codex. Despite the impressive success of LLMs of code, some powerful LLMs (such as Codex) are not publicly available, preventing the research community from studying and improving such LLMs. In 2022, Xu et al. [18] propose PolyCoder, a decoder-only LLM based on GPT-2 architecture. PolyCoder is trained with 249GB of code from 12 programming languages and contains three sizes, 160M, 400M, and 2.7B parameters. The results demonstrate that despite Codex's primary focus on Python, it still performs well on other programming languages, even outperforming GPT-J and GPT-NeoX. However, for the C programming language, PolyCoder outperforms all LLMs including Codex. Importantly, unlike Codex, three PolyCoder models of different sizes are made available for the research community.

(5) CodeGen: LLM for Program Synthesis. To investigate program synthesis with LLMs, in 2023, Nijkamp et al. [19] from Salesforce introduce CodeGen, which employs a self-regressive Transformer architecture and is trained sequentially on natural language and programming language datasets (THEPILE, BIGQUERY, and BIGPYTHON). It is designed for multi-round program synthesis. CodeGen undergoes evaluation for both single-round and multi-round program synthesis. In single-round evaluation, the synthetic benchmark HumanEval is utilized, and it is observed that CodeGen performance improves with data sizes. Experimental results demonstrate that the performance of the CodeGen NL model either surpasses or is comparable to that of GPT-NEO and GPT-J. CodeGen-Multi demonstrates a significant performance advantage over GPT-NEO, GPT-J, and CodeGen-NL. Furthermore, CodeGen-Mono, fine-tuned on a pure Python dataset, exhibits remarkable enhancements in program synthesis.

(6) InCoder: LLM for Code Infilling and Synthesis. Existing LLMs generate code in a left-to-right manner, which may be unsuitable to many many ubiquitous code editing tasks, such as bug fixing. In 2022, Fried et al. [20] from Facebook propose InCoder, a decoder-only LLM designed for program synthesis and editing. InCoder is pre-trained by a causal masking objective, i.e., learning through the random replacement of code segments with sentinel tokens, moving them to the end of the sequence. InCoder's training data consists solely of open-licensed code (Apache 2.0, MIT, BSD-2, and BSD-3 licenses) from online sources such as GitHub, GitLab, and StackOverflow. It primarily focuses on Python and JavaScript but encompasses a total of 28 languages, amounting to approximately 200GB of data in total. There are two versions of the publicly released pre-trained models: one with 6.7 billion parameters and another with 1.3 billion parameters. The experimental results demonstrate that InCoder is able to infill arbitrary regions of code under a zero-shot setting for several tasks, such as type inference and comment generation. InCoder achieves performance roughly equivalent to CodeGen-Multi on the HumanEval benchmark.

(7) PyCodeGPT: LLM for Library-oriented Code Generation. Previous state-of-the-art LLMs are not publicly available, hindering the progress of related research topics and applications. Similar to PolyCoder [18], in 2022, Zan et al. [21] propose PyCodeGPT, a publicly available LLM particular designed for Python. PyCodeGPT is derived from GPT-Neo 125M with a vocabulary size of 32K and incorporates

a novel byte-level BPE tokenizer tailored for Python source code. The training dataset consists of 13 million Python files with 96GB crawled from GitHub. The experimental results demonstrate that PyCodeGPT achieves a pass@1 score of 8.33% and pass@10 of 13.53% on the HumanEval benchmark, surpassing other LLMs with similar parameters, such as AlphaCode, CodeClippy, and CodeParrot.

(8) SantaCoder. Regarding the removal of personally identifiable information, the BigCode community [22] propose SantaCoder, a decoder-only LLM with 1.1 billion parameters. SantaCoder’s architecture is based on GPT-2 with multi-query attention and Fill-in-the-Middle objective. Its training dataset consists of Python, Java, and JavaScript files from The Stack v1.1. The dataset has undergone several preprocessing steps, including partial data removal, near-duplication removal, de-identification of personally identifiable information, and filtering based on line length and the percentage of alphanumeric characters. Files containing test samples from benchmarks such as HumanEval, APPS, MBPP, and MultiPL-E have also been excluded. The experimental results on the MultiPL-E benchmark demonstrate that SantaCoder outperforms InCoder-6.7B and CodeGen-2.7B in code generation and filling tasks.

(9) StarCoder. Committed to developing responsible LLMs, Li et al. [23] from Hugging Face present StarCoder and StarCoderBase, which are LLMs for code. StarCoder is trained on Stack v1.2, and to ensure the secure release of open-source LLMs, it has improved the personally identifiable information editing pipeline and introduced innovative attribution tracking tools. StarCoder undergoes evaluations on HumanEval and MBPP, the experimental results show that StarCoder outperforms PaLM, LaMDA, LLaMA, CodeGen-16B-Mono, and OpenAI’s code-cushman-001 (12B) on HumanEval.

(10) PanGu-Coder: LLM for Text-to-code Generation. To address the specific task of text-to-code generation, adapt to more specific language domains, and handle signals beyond natural language, In 2022, Christopoulou et al. [24] from Huawei propose PanGu-Coder, a decoder-only LLM for text-to-code generation, i.e., generating stand-alone Python functions from docstrings and evaluating the correctness of code examples through unit tests. PanGu-Coder is built on top of the PanGu-Alpha architecture, a uni-directional decoder-only transformer with an extra query layer stacked on top. PanGu-Coder is trained with two objectives, i.e., a causal language modeling on raw programming language data, and a combination of causal language modeling and masked language modeling for the downstream task of text-to-code generation. The results under a zero-shot manner show that PanGu-Coder outperforms industry LLMs such as Codex and AlphaCode on the HumanEval and MBPP datasets.

(11) PanGu-Coder2: LLM with Reinforcement Learning. Inspired by the success of Reinforcement Learning from Human Feedback in LLMs, in 2023, Shen et al. [25] propose PanGu-Coder2, a successor of PanGu-Coder with more powerful code generation capability. PanGu-Coder2 is trained with a new training paradigm, i.e., Rank Responses to align Test&Teacher Feedback and built on top of the advanced StarCoder 15B model. The experimental results demonstrate that PanGu-Coder2 is able to outperform previous LLMs, such as StarCoder, CodeT5+, and AlphaCode on HumanEval, CodeEval, and LeetCode benchmarks.

(12) PaLM-Coder: A variant of PaLM for Source Code. To investigate the captivity of PaLM for source code, in 2022, Chowdhery et al. [26] from Google propose PaLM-Coder, a variant of PaLM by code-specific fine-tuning. The based model PaLM is pre-trained with a high-quality corpus of 780 billion tokens, including 196GB of source code from open-source repositories on GitHub. PaLM-Coder is further derived from PaLM with a two-stage fine-tuning process, i.e., (1) an initial fine-tuning over 6.5 billion tokens, consisting of a blend with 60% Python code, 30% multi-language code and 10% natural language; and (2) an extended fine-tuning with 1.9 billion Python code tokens. The experimental results show that PaLM-Coder is able to achieve 88.4% pass@100 on HumanEval and 80.8% pass@80 on MBPP. Besides, PaLM-Coder demonstrates impressive performance on the DeepFix code repair task with a compile rate of 82.1%, opening up opportunities for fixing complex errors that arise during software development.

(13) CodeGeeX: LLM for Multilingual Code Generation. Despite demonstrating impressive performance, previous LLMs (such as Codex) mainly focus on code generation and are closed-source. In 2023, Zheng et al. [27] introduce CodeGeeX, a multilingual decoder-only open-sourced LLM with 13 billion parameters for both code generation and translation tasks. CodeGeeX is implemented with the Huawei MindSpore framework and pre-trained on 850 billion tokens from 23 programming languages, including C++, Java, JavaScript, and Go. Besides, on top of the well-known HumanEval benchmark, a multilingual code generation benchmark HumanEval-X is constructed to evaluate CodeGeeX by hand-writing the solutions in C++, Java, JavaScript, and Go. The experimental results demonstrate that CodeGeeX performs exceptionally well in code generation and translation tasks on the HumanEval-X benchmark. Importantly, CodeGeeX has been integrated with Visual Studio Code, JetBrains, and Cloud

Studio. It generates 4.7 billion tokens weekly for tens of thousands of active developers, enhancing the coding efficiency of 83.4% of its users. Besides, CodeGeeX is open-sourced, as well as its code, model weights, API, and HumanEval-X, facilitating the understanding and advances in the community.

(14) CodeGen2: A Successor of CodeGen. Considering the high computational cost of training LLMs, in 2023, Nijkamp et al. [28] propose CodeGen2, an successor of CodeGen, aimed at addressing the challenge of more efficiently training LLMs for program synthesis and understanding tasks. CodeGen2 provide a training framework along with open-source CodeGen2 models in four variations, including 1B, 3.7B, 7B, and 16B parameters in size. CodeGen2 is trained on the BigPython dataset and evaluated on the Stack dataset to assess its learning performance in HumanEval and HumanEval-Infill tasks. The experimental results demonstrate that CodeGen2 performs well across various model sizes and program synthesis and understanding tasks, outperforming InCoder in the evaluation on HumanEval.

(15) Code Llama: Llama-based LLM for Source Code. On top of the powerful Llama 2 model in NLP, in 2023, Roziere et al. [29] from Meta AI propose Code Llama, a series of LLMs specialized in handling code-related tasks. Code Llama exhibits capabilities such as infilling, support for large input contexts, and zero-shot instruction-following abilities. The dataset of Code Llama primarily comprises an extensive collection of programming language content, with a special emphasis on Python language, trained through a code-heavy dataset containing 500B tokens and an additional Python-intensive data mix of 100B tokens. It comprises multiple versions, covering Code Llama, Code Llama - Python, and Code Llama - Instruct with 7B, 13B and 34B parameters each. Code Llama undergoes evaluation on HumanEval and MBPP, the experimental results indicate that Code Llama outperforms LLama and Llama 2 in terms of performance.

2 Appendix B: Detailed summarization of existing LLM-based SE studies

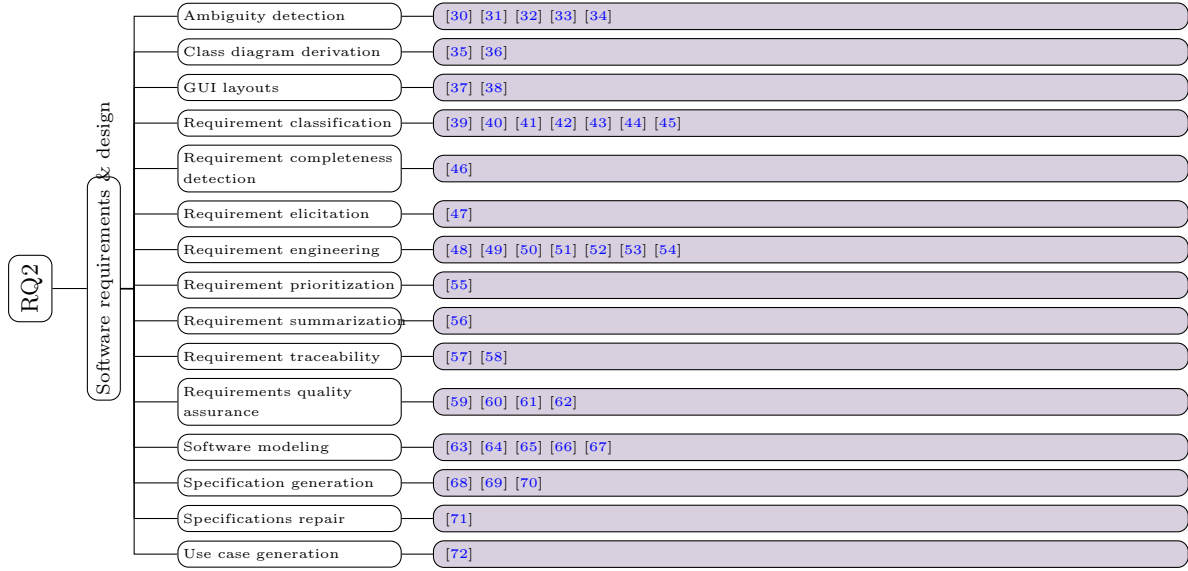
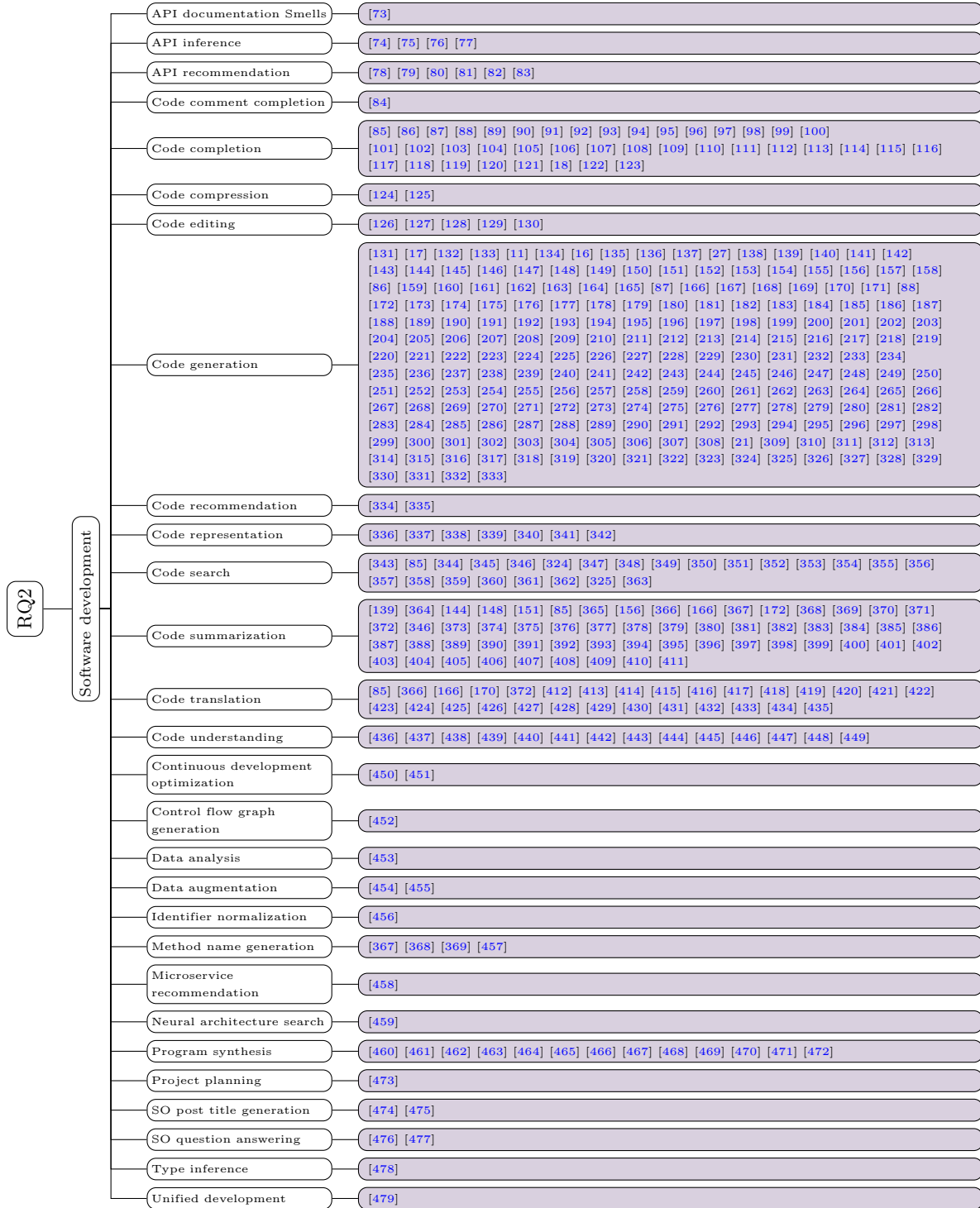


Figure 1 Taxonomy of the application of LLMs in different domains within software engineering

**Figure 1** Taxonomy of the application of LLMs in different domains within software engineering (Continue)

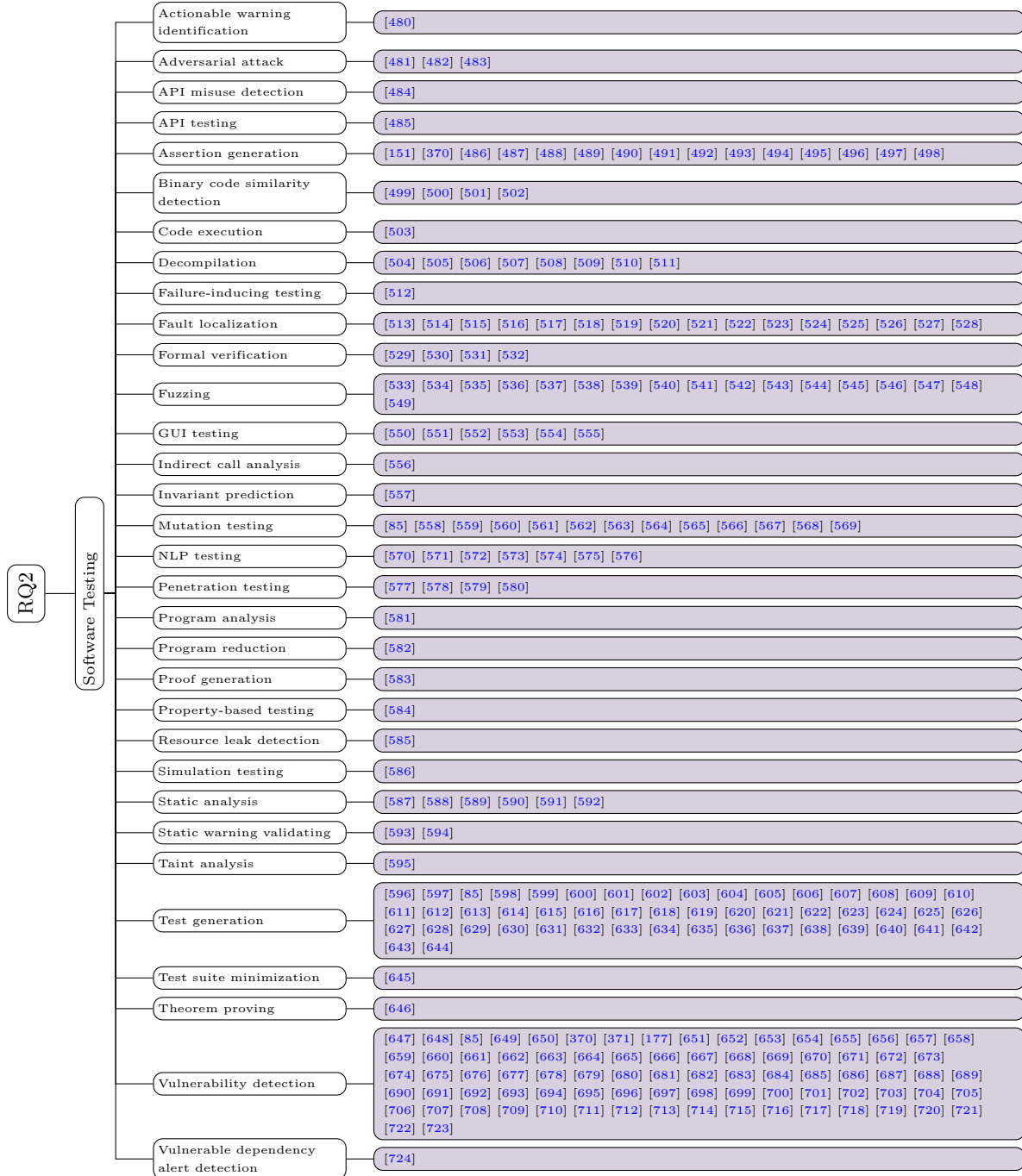
**Figure 1** Taxonomy of the application of LLMs in different domains within software engineering (Continue)



Figure 1 Taxonomy of the application of LLMs in different domains within software engineering (Continue)

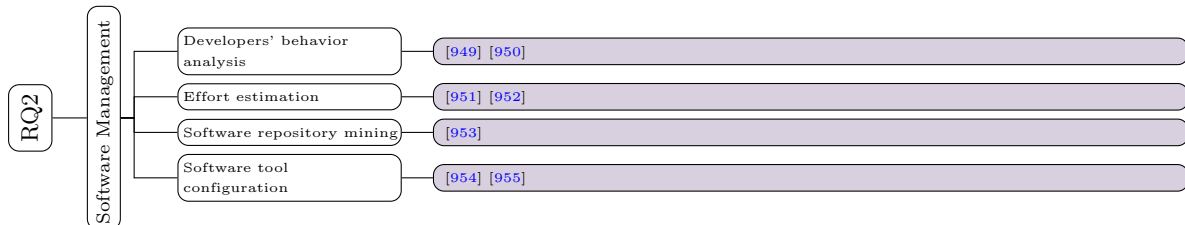


Figure 1 Taxonomy of the application of LLMs in different domains within software engineering (Continue)

References

- 1 Kanade A, Maniatis P, Balakrishnan G, et al. Learning and evaluating contextual embedding of source code. In *International conference on machine learning*. PMLR, 2020. 5110–5121
- 2 Feng Z, Guo D, Tang D, et al. Codebert: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. 2020. 1536–1547
- 3 Guo D, Ren S, Lu S, et al. Graphcodebert: Pre-training code representations with data flow. arXiv preprint arXiv:200908366, 2020
- 4 Clement C B, Drain D, Timcheck J, et al. Pymt5: multi-mode translation of natural language and python code with transformers. arXiv preprint arXiv:201003150, 2020
- 5 Mastropaolo A, Scalabrino S, Cooper N, et al. 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)*. IEEE, 2021. 336–347
- 6 Ahmad W U, Chakraborty S, Ray B, et al. Unified pre-training for program understanding and generation. arXiv preprint arXiv:210306333, 2021
- 7 Wang Y, Wang W, Joty S, et al. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP'21)*. 2021. 8696–8708
- 8 Niu C, Li C, Ng V, et al. Spt-code: sequence-to-sequence pre-training for learning source code representations. In *Proceedings of the 44th International Conference on Software Engineering*. 2022. 2006–2018
- 9 Le H, Wang Y, Gotmare A D, et al. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems*, 2022. 35:21314–21328
- 10 Zhang J, Panthaplackel S, Nie P, et al. Coditt5: Pretraining for source code and natural language editing. In *37th IEEE/ACM International Conference on Automated Software Engineering*. 2022. 1–12
- 11 Li Y, Choi D, Chung J, et al. Competition-level code generation with alphacode. *Science*, 2022. 378(6624):1092–1097
- 12 Wang Y, Le H, Gotmare A D, et al. Codet5+: Open code large language models for code understanding and generation. arXiv preprint arXiv:230507922, 2023
- 13 Chandel S, Clement C B, Serrato G, et al. Training and evaluating a jupyter notebook data science assistant. arXiv preprint arXiv:220112901, 2022
- 14 Chai Y, Wang S, Pang C, et al. Ernie-code: Beyond english-centric cross-lingual pretraining for programming languages. arXiv preprint arXiv:221206742, 2022
- 15 Syatkovskiy A, Deng S K, Fu S, et al. Intellicode compose: Code generation using transformer. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2020. 1433–1443
- 16 Lu S, Guo D, Ren S, et al. Codexglue: A machine learning benchmark dataset for code understanding and generation. In J Vanschoren and S Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*. 2021. 1–14
- 17 Chen M, Tworek J, Jun H, et al. Evaluating large language models trained on code. arXiv preprint arXiv:210703374, 2021
- 18 Xu F F, Alon U, Neubig G, et al. A systematic evaluation of large language models of code. In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*. 2022. 1–10
- 19 Nijkamp E, Pang B, Hayashi H, et al. Codegen: An open large language model for code with multi-turn program synthesis. arXiv preprint arXiv:220313474, 2022
- 20 Fried D, Aghajanyan A, Lin J, et al. Incoder: A generative model for code infilling and synthesis. arXiv preprint arXiv:220405999, 2022
- 21 Zan D, Chen B, Yang D, et al. Cert: Continual pre-training on sketches for library-oriented code generation. arXiv preprint arXiv:220606888, 2022
- 22 Allal L B, Li R, Kocetkov D, et al. Santacoder: don't reach for the stars! arXiv preprint arXiv:230103988, 2023
- 23 Li R, Allal L B, Zi Y, et al. Starcoder: may the source be with you! arXiv preprint arXiv:230506161, 2023
- 24 Christopoulou F, Lampouras G, Gritta M, et al. Pangu-coder: Program synthesis with function-level language modeling. arXiv preprint arXiv:220711280, 2022
- 25 Shen B, Zhang J, Chen T, et al. Pangu-coder2: Boosting large language models for code with ranking feedback. arXiv preprint arXiv:230714936, 2023
- 26 Chowdhery A, Narang S, Devlin J, et al. Palm: Scaling language modeling with pathways. arXiv preprint arXiv:220402311, 2022
- 27 Zheng Q, Xia X, Zou X, et al. Codegeex: A pre-trained model for code generation with multilingual evaluations on humaneval-x. arXiv preprint arXiv:230317568, 2023
- 28 Nijkamp E, Hayashi H, Xiong C, et al. Codegen2: Lessons for training llms on programming and natural languages. arXiv preprint arXiv:230502309, 2023
- 29 Roziere B, Gehring J, Gloeckle F, et al. Code llama: Open foundation models for code. arXiv preprint arXiv:230812950, 2023
- 30 Ezzini S, Abualhaija S, Arora C, et al. Automated handling of anaphoric ambiguity in requirements: a multi-solution study. In *Proceedings of the 44th International Conference on Software Engineering*. 2022. 187–199
- 31 Gärtner A E and Göhlich D. Automated requirement contradiction detection through formal logic and llms. *Automated Software Engineering*, 2024. 31(2):49
- 32 Moharil A and 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*. 2022. 51–58
- 33 Moharil A and Sharma A. Tabasco: A transformer based contextualization toolkit. *Science of Computer Programming*, 2023. 230:102994
- 34 Sridhara G, Mazumdar S, et al. Chatgpt: A study on its utility for ubiquitous software engineering tasks. arXiv preprint arXiv:230516837, 2023
- 35 Li Y, Keung J, Ma X, et al. Llm-based class diagram derivation from user stories with chain-of-thought promptings. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2024. 45–50
- 36 Sanyal R, Ghoshal B, et al. A hybrid approach to extract conceptual diagram from software requirements. *Science of Computer Programming*, 2024:103186
- 37 Kolthoff K, Bartelt C, and Ponzetto S P. Data-driven prototyping via natural-language-based gui retrieval. *Automated software engineering*, 2023. 30(1):13
- 38 Brie P, Burny N, Sluÿters A, et al. Evaluating a large language model on searching for gui layouts. *Proceedings of the ACM on Human-Computer Interaction*, 2023. 7(EICS):1–37
- 39 El-Hajjaji A, Fafin N, and Salinesi C. Which ai technique is better to classify requirements? an experiment with svm, lstm, and chatgpt. arXiv preprint arXiv:231111547, 2023
- 40 Han L, Zhou Q, and Li T. Improving requirements classification models based on explainable requirements concerns. In *2023 IEEE 31st International Requirements Engineering Conference Workshops (REW)*. IEEE, 2023. 95–101

- 41 Hey T, Keim J, Koziolok A, et al. Norbert: Transfer learning for requirements classification. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*. IEEE, 2020. 169–179
- 42 Khan M A, Khan M S, Khan I, et al. Non functional requirements identification and classification using transfer learning model. IEEE Access, 2023
- 43 Luo X, Xue Y, Xing Z, et al. Prcbert: Prompt learning for requirement classification using bert-based pretrained language models. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 2022. 1–13
- 44 Rahman K, Ghani A, Alzahrani A, et al. Pre-trained model-based nfr classification: Overcoming limited data challenges. IEEE Access, 2023
- 45 Subahi A F. Bert-based approach for greening software requirements engineering through non-functional requirements. IEEE Access, 2023
- 46 Luitel D, Hassani S, and Sabetzadeh M. Improving requirements completeness: Automated assistance through large language models. *Requirements Engineering*, 2024. 29(1):73–95
- 47 Ren S, Nakagawa H, and Tsuchiya T. Combining prompts with examples to enhance llm-based requirement elicitation. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2024. 1376–1381
- 48 Arora C, Grundy J, and Abdelrazek M. Advancing requirements engineering through generative ai: Assessing the role of llms. In *Generative AI for Effective Software Development*. Springer, 2024. 129–148
- 49 Fazelnia M, Koscinski V, Herzog S, et al. Lessons from the use of natural language inference (nli) in requirements engineering tasks. arXiv preprint arXiv:240505135, 2024
- 50 Hassani S. Enhancing legal compliance and regulation analysis with large language models. arXiv preprint arXiv:240417522, 2024
- 51 Jin D, Jin Z, Chen X, et al. Mare: Multi-agents collaboration framework for requirements engineering. arXiv preprint arXiv:240503256, 2024
- 52 Pilone A, Meirelles P, Kon F, et al. Multilingual crowd-based requirements engineering using large language models. arXiv preprint arXiv:240806505, 2024
- 53 Ronanki K, Cabrero-Daniel B, Horkoff J, et al. Requirements engineering using generative ai: Prompts and prompting patterns. In *Generative AI for Effective Software Development*. Springer, 2024. 109–127
- 54 Vogelsang A. From specifications to prompts: On the future of generative llms in requirements engineering. arXiv preprint arXiv:240809127, 2024
- 55 Sami M A, Rasheed Z, Waseem M, et al. Prioritizing software requirements using large language models. arXiv preprint arXiv:240501564, 2024
- 56 Jain C, Anish P R, Singh A, et al. A transformer-based approach for abstractive summarization of requirements from obligations in software engineering contracts. In *2023 IEEE 31st International Requirements Engineering Conference (RE)*. IEEE, 2023. 169–179
- 57 Guo J L, Steghöfer J P, Vogelsang A, et al. Natural language processing for requirements traceability. arXiv preprint arXiv:240510845, 2024
- 58 Lin J, Liu Y, Zeng Q, et al. Traceability transformed: Generating more accurate links with pre-trained bert models. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021. 324–335
- 59 Lubos S, Felfernig A, Tran T N T, et al. Leveraging llms for the quality assurance of software requirements. In *2024 IEEE 32nd International Requirements Engineering Conference (RE)*. IEEE, 2024. 389–397
- 60 Poudel A, Lin J, and Cleland-Huang J. Leveraging transformer-based language models to automate requirements satisfaction assessment. arXiv preprint arXiv:231204463, 2023
- 61 Preda A R, Mayr-Dorn C, Mashkoo A, et al. Supporting high-level to low-level requirements coverage reviewing with large language models. In *Proceedings of the 21st International Conference on Mining Software Repositories*. 2024. 242–253
- 62 Ronanki K, Cabrero-Daniel B, and Berger C. Chatgpt as a tool for user story quality evaluation: Trustworthy out of the box? In *International Conference on Agile Software Development*. Springer, 2022. 173–181
- 63 Chaaben M B, Burgueño L, and Sahraoui H. Towards using few-shot prompt learning for automating model completion. In *2023 IEEE/ACM 45th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 2023. 7–12
- 64 Ferrari A, Abualhaija S, and Arora C. Model generation from requirements with llms: an exploratory study. arXiv preprint arXiv:240406371, 2024
- 65 Tinnes C, Welter A, and Apel S. Leveraging large language models for software model completion: Results from industrial and public datasets. arXiv preprint arXiv:240617651, 2024
- 66 Wang B, Wang C, Liang P, et al. How llms aid in uml modeling: An exploratory study with novice analysts. arXiv preprint arXiv:240417739, 2024
- 67 Wei J, Chen X, Xiao H, et al. Natural language processing-based requirements modeling: A case study on problem frames. In *2023 30th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2023. 191–200
- 68 Ma L, Liu S, Li Y, et al. Specgen: Automated generation of formal program specifications via large language models. arXiv preprint arXiv:240108807, 2024
- 69 Mandal S, Chethan A, Janfaza V, et al. Large language models based automatic synthesis of software specifications. arXiv preprint arXiv:230409181, 2023
- 70 Xie D, Yoo B, Jiang N, et al. Impact of large language models on generating software specifications. arXiv preprint arXiv:230603324, 2023
- 71 Hasan M R, Li J, Ahmed I, et al. Automated repair of declarative software specifications in the era of large language models. arXiv preprint arXiv:231012425, 2023
- 72 Zhang S, Wang J, Dong G, et al. Experimenting a new programming practice with llms. arXiv preprint arXiv:240101062, 2024
- 73 Khan J Y, Khondaker M T I, Uddin G, et al. Automatic detection of five api documentation smells: Practitioners' perspectives. In *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2021. 318–329
- 74 Huang Q, Wu Y, Xing Z, et al. Adaptive intellect unleashed: The feasibility of knowledge transfer in large language models. arXiv preprint arXiv:230804788, 2023
- 75 Patil S G, Zhang T, Wang X, et al. Gorilla: Large language model connected with massive apis. arXiv preprint arXiv:230515334, 2023
- 76 Wang S, Jean S, Sengupta S, et al. Measuring and mitigating constraint violations of in-context learning for utterance-to-api semantic parsing. arXiv preprint arXiv:230515338, 2023
- 77 Zhuo T Y, Du X, Xing Z, et al. Pop quiz! do pre-trained code models possess knowledge of correct api names? arXiv preprint arXiv:230907804, 2023
- 78 Chen Y, Gao C, Zhu M, et al. Apigen: Generative api method recommendation. arXiv preprint arXiv:240115843, 2024
- 79 Huang Q, Wan Z, Xing Z, et al. Let's chat to find the apis: Connecting human, llm and knowledge graph through ai chain. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023. 471–483

- 80 Li Z, Li C, Tang Z, et al. Ptm-apirec: Leveraging pre-trained models of source code in api recommendation. *ACM Transactions on Software Engineering and Methodology*, 2024. 33(3):1–30
- 81 Wei M, Harzevili N S, Huang Y, et al. Clear: contrastive learning for api recommendation. In *Proceedings of the 44th International Conference on Software Engineering*. 2022. 376–387
- 82 Wu D, Feng Y, Zhang H, et al. Automatic recognizing relevant fragments of apis using api references. *Automated Software Engineering*, 2024. 31(1):3
- 83 Zhang K, Zhang H, Li G, et al. Toolcoder: Teach code generation models to use api search tools. *arXiv preprint arXiv:230504032*, 2023
- 84 Mastropaolo A, Aghajani E, Pascarella L, et al. An empirical study on code comment completion. In *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2021. 159–170
- 85 Niu C, Li C, Ng V, et al. An empirical comparison of pre-trained models of source code. *arXiv preprint arXiv:230204026*, 2023
- 86 Zhou X, Kim K, Xu B, et al. The devil is in the tails: How long-tailed code distributions impact large language models. *arXiv preprint arXiv:230903567*, 2023
- 87 Jesse K, Ahmed T, Devanbu P T, et al. Large language models and simple, stupid bugs. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. IEEE, 2023. 563–575
- 88 Oh S, Lee K, Park S, et al. Poisoned chatgpt finds work for idle hands: Exploring developers' coding practices with insecure suggestions from poisoned ai models. *arXiv preprint arXiv:231206227*, 2023
- 89 Schuster R, Song C, Tromer E, et al. You autocomple me: Poisoning vulnerabilities in neural code completion. In *30th USENIX Security Symposium (USENIX Security 21)*. 2021. 1559–1575
- 90 Shi E, Wang Y, Zhang H, et al. Towards efficient fine-tuning of pre-trained code models: An experimental study and beyond. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2023. 39–51
- 91 Cheng W, Wu Y, and Hu W. Dataflow-guided retrieval augmentation for repository-level code completion. *arXiv preprint arXiv:240519782*, 2024
- 92 Ciniselli M, Cooper N, Pascarella L, et al. An empirical study on the usage of bert models for code completion. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. IEEE, 2021. 108–119
- 93 Ciniselli M, Cooper N, Pascarella L, et al. An empirical study on the usage of transformer models for code completion. *IEEE Transactions on Software Engineering*, 2021. 48(12):4818–4837
- 94 Deng K, Liu J, Zhu H, et al. R2c2-coder: Enhancing and benchmarking real-world repository-level code completion abilities of code large language models. *arXiv preprint arXiv:240601359*, 2024
- 95 Ding H, Kumar V, Tian Y, et al. A static evaluation of code completion by large language models. *arXiv preprint arXiv:230603203*, 2023
- 96 Ding Y, Wang Z, Ahmad W, et al. Crosscodeeval: A diverse and multilingual benchmark for cross-file code completion. *Advances in Neural Information Processing Systems*, 2024. 36
- 97 Dinh T, Zhao J, Tan S, et al. Large language models of code fail at completing code with potential bugs. *Advances in Neural Information Processing Systems*, 2024. 36
- 98 Döderlein J B, Acher M, Khelladi D E, et al. Piloting copilot and codex: Hot temperature, cold prompts, or black magic? *arXiv preprint arXiv:221014699*, 2022
- 99 Eghbali A and Pradel M. De-hallucinator: Iterative grounding for llm-based code completion. *arXiv preprint arXiv:240101701*, 2024
- 100 Gong L, Wang S, Elhoushi M, et al. Evaluation of llms on syntax-aware code fill-in-the-middle tasks. *arXiv preprint arXiv:240304814*, 2024
- 101 Izadi M, Gismondi R, and 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*. 2022. 401–412
- 102 Kabir A, Wang S, Tian Y, et al. Zs4c: Zero-shot synthesis of compilable code for incomplete code snippets using chatgpt. *arXiv preprint arXiv:240114279*, 2024
- 103 Khan J Y and 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)*. IEEE, 2022. 765–776
- 104 Li J, Huang R, Li W, et al. Toward less hidden cost of code completion with acceptance and ranking models. In *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2021. 195–205
- 105 Li Y, Peng Y, Huo Y, et al. Enhancing llm-based coding tools through native integration of ide-derived static context. In *2024 IEEE/ACM International Workshop on Large Language Models for Code (LLM4Code)*. IEEE, 2024. 70–74
- 106 Liu T, Xu C, and McAuley J. Repobench: Benchmarking repository-level code auto-completion systems. *arXiv preprint arXiv:230603091*, 2023
- 107 Liu W, Yu A, Zan D, et al. Graphcoder: Enhancing repository-level code completion via code context graph-based retrieval and language model. *arXiv preprint arXiv:240607003*, 2024
- 108 Liu J, Chen Y, Liu M, et al. Stall+: Boosting llm-based repository-level code completion with static analysis. *arXiv preprint arXiv:240610018*, 2024
- 109 Li Z, Wang C, Liu Z, et al. Cctest: Testing and repairing code completion systems. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023. 1238–1250
- 110 Nashid N, Shabani T, Alian P, et al. Contextual api completion for unseen repositories using llms. *arXiv preprint arXiv:240504600*, 2024
- 111 Nie P, Banerjee R, Li J J, et al. Learning deep semantics for test completion. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023. 2111–2123
- 112 Ochs M, Narasimhan K, and 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)*. IEEE, 2023. 834–844
- 113 Phan H N, Phan H N, Nguyen T N, et al. Repohyper: Better context retrieval is all you need for repository-level code completion. *arXiv preprint arXiv:240306095*, 2024
- 114 Prenner J A and Robbes R. Making the most of small software engineering datasets with modern machine learning. *IEEE Transactions on Software Engineering*, 2021. 48(12):5050–5067
- 115 Pudari R and Ernst N A. From copilot to pilot: Towards ai supported software development. *arXiv preprint arXiv:230304142*, 2023
- 116 Sun Z, Du X, Song F, et al. When neural code completion models size up the situation: Attaining cheaper and faster completion through dynamic model inference. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 2024. 1–12
- 117 Tan H, Luo Q, Jiang L, et al. Prompt-based code completion via multi-retrieval augmented generation. *arXiv preprint arXiv:240507530*, 2024
- 118 Tang Z, Ge J, Liu S, et al. Domain adaptive code completion via language models and decoupled domain databases. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023. 421–433
- 119 van Dam T, Izadi M, and van Deursen A. Enriching source code with contextual data for code completion models: An empirical study. *arXiv preprint arXiv:230412269*, 2023

- 120 Wang Y, Wang Y, Guo D, et al. RlCoder: Reinforcement learning for repository-level code completion. arXiv preprint arXiv:240719487, 2024
- 121 Wu D, Ahmad W U, Zhang D, et al. Repoformer: Selective retrieval for repository-level code completion. arXiv preprint arXiv:240310059, 2024
- 122 Zhang L, Li Y, Li J, et al. Hierarchical context pruning: Optimizing real-world code completion with repository-level pretrained code llms. arXiv preprint arXiv:240618294, 2024
- 123 Zhang M, Yuan B, Li H, et al. Llm-cloud complete: Leveraging cloud computing for efficient large language model-based code completion. *Journal of Artificial Intelligence General science (JAIGS)* ISSN: 3006-4023, 2024. 5(1):295–326
- 124 Gilbert H, Sandborn M, Schmidt D C, et al. Semantic compression with large language models. arXiv preprint arXiv:230412512, 2023
- 125 Von der Mosel J, Trautsch A, and Herbold S. On the validity of pre-trained transformers for natural language processing in the software engineering domain. *IEEE Transactions on Software Engineering*, 2022. 49(4):1487–1507
- 126 Dilhara M, Bellur A, Bryksin T, et al. Unprecedented code change automation: The fusion of llms and transformation by example. *Proceedings of the ACM on Software Engineering*, 2024. 1(FSE):631–653
- 127 Gupta P, Khare A, Bajpai Y, et al. Grace: Generation using associated code edits. arXiv preprint arXiv:230514129, 2023
- 128 Li J, Li G, Li Z, et al. Codeeditor: Learning to edit source code with pre-trained models. *ACM Transactions on Software Engineering and Methodology*, 2023. 32(6):1–22
- 129 Liu C, Cetin P, Patodia Y, et al. Automated code editing with search-generate-modify. In *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*. 2024. 398–399
- 130 Moon S, Song Y, Chae H, et al. Coffee: Boost your code llms by fixing bugs with feedback. arXiv preprint arXiv:231107215, 2023
- 131 Cassano F, Gouwar J, Nguyen D, et al. Multi-pl-e: a scalable and polyglot approach to benchmarking neural code generation. *IEEE Transactions on Software Engineering*, 2023. 49(7):3675–3691
- 132 Du X, Liu M, Wang K, et al. Evaluating large language models in class-level code generation. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 2024. 982–994
- 133 Hendrycks D, Basart S, Kadavath S, et al. Measuring coding challenge competence with apps. arXiv preprint arXiv:210509938, 2021
- 134 Liu J, Xia C S, Wang Y, et al. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. arXiv preprint arXiv:230501210, 2023
- 135 Manh D N, Hai N L, Dau A T, et al. The vault: A comprehensive multilingual dataset for advancing code understanding and generation. arXiv preprint arXiv:230506156, 2023
- 136 Niu C, Li C, Ng V, et al. Crosscodebench: Benchmarking cross-task generalization of source code models. arXiv preprint arXiv:230204030, 2023
- 137 Yu H, Shen B, Ran D, et al. Codereval: A benchmark of pragmatic code generation with generative pre-trained models. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 2024. 1–12
- 138 Zhuo T Y, Vu M C, Chim J, et al. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. arXiv preprint arXiv:240615877, 2024
- 139 Wei X, Gonugondla S K, Wang S, et al. Towards greener yet powerful code generation via quantization: An empirical study. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2023. 224–236
- 140 Geng C, Yihan Z, Pientka B, et al. Can chatgpt pass an introductory level functional language programming course? arXiv preprint arXiv:230502230, 2023
- 141 Nguyen P T, Di Rocco J, Di Sipio C, et al. Is this snippet written by chatgpt? an empirical study with a codebert-based classifier. arXiv preprint arXiv:230709381, 2023
- 142 Sandoval G, Pearce H, Nys T, et al. Lost at c: A user study on the security implications of large language model code assistants. In *32nd USENIX Security Symposium (USENIX Security 23)*. 2023. 2205–2222
- 143 Xue Y, Chen H, Bai G R, et al. Does chatgpt help with introductory programming? an experiment of students using chatgpt in cs1. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training*. 2024. 331–341
- 144 Tian H, Lu W, Li T O, et al. Is chatgpt the ultimate programming assistant—how far is it? arXiv preprint arXiv:230411938, 2023
- 145 Cowan B, Watanobe Y, and Shirafuji A. Enhancing programming learning with llms: Prompt engineering and flipped interaction. In *Proceedings of the 2023 4th Asia Service Sciences and Software Engineering Conference*. 2023. 10–16
- 146 Liang J T, Badea C, Bird C, et al. Can gpt-4 replicate empirical software engineering research? *Proceedings of the ACM on Software Engineering*, 2024. 1(FSE):1330–1353
- 147 Fakhoury S, Naik A, Sakkas G, et al. Llm-based test-driven interactive code generation: User study and empirical evaluation. arXiv preprint arXiv:240410100, 2024
- 148 Gao S, Wen X C, Gao C, et al. What makes good in-context demonstrations for code intelligence tasks with llms? In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023. 761–773
- 149 Kuhail M A, Mathew S S, Khalil A, et al. “will i be replaced?” assessing chatgpt’s effect on software development and programmer perceptions of ai tools. *Science of Computer Programming*, 2024. 235:103111
- 150 Liang W and Xiao G. An exploratory evaluation of large language models using empirical software engineering tasks. In *Proceedings of the 15th Asia-Pacific Symposium on Internetwork*. 2024. 31–40
- 151 Mastropaolo A, Cooper N, Palacio D N, et al. Using transfer learning for code-related tasks. *IEEE Transactions on Software Engineering*, 2022. 49(4):1580–1598
- 152 Piya S and Sullivan A. Llm4ddd: Best practices for test driven development using large language models. arXiv preprint arXiv:231204687, 2023
- 153 Rasnayaka S, Wang G, Shariffdeen R, et al. An empirical study on usage and perceptions of llms in a software engineering project. arXiv preprint arXiv:240116186, 2024
- 154 Sakib F A, Khan S H, and Karim A. Extending the frontier of chatgpt: Code generation and debugging. arXiv preprint arXiv:230708260, 2023
- 155 Tanzil M H, Khan J Y, and Uddin G. Chatgpt incorrectness detection in software reviews. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 2024. 1–12
- 156 Tufano R, Mastropaolo A, Pepe F, et al. Unveiling chatgpt’s usage in open source projects: A mining-based study. In *2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)*. IEEE, 2024. 571–583
- 157 Wang W, Ning H, Qian S, et al. Characterizing developers’ behaviors in llm-supported software development. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2024. 1168–1177
- 158 Wang W, Ning H, Zhang G, et al. Rocks coding, not development: A human-centric, experimental evaluation of llm-supported se tasks. *Proceedings of the ACM on Software Engineering*, 2024. 1(FSE):699–721
- 159 Billah M M, Roy P R, Codabux Z, et al. Are large language models a threat to programming platforms? an exploratory study. In *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*.

2024. 292–301
- 160 Zeng Z, Tan H, Zhang H, et al. 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*. 2022. 39–51
- 161 Al-Kaswan A, Izadi M, and Van Deursen A. Traces of memorisation in large language models for code. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 2024. 1–12
- 162 Anwar U, Saparov A, Rando J, et al. Foundational challenges in assuring alignment and safety of large language models. arXiv preprint arXiv:240409932, 2024
- 163 Cheng Y, Chen J, Huang Q, et al. Prompt sapper: a llm-empowered production tool for building ai chains. *ACM Transactions on Software Engineering and Methodology*, 2024. 33(5):1–24
- 164 Feng L, Yen R, You Y, et al. Coprompt: Supporting prompt sharing and referring in collaborative natural language programming. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 2024. 1–21
- 165 Idialu O J, Mathews N S, Maipradit R, et al. Whodunit: Classifying code as human authored or gpt-4 generated-a case study on codechef problems. In *Proceedings of the 21st International Conference on Mining Software Repositories*. 2024. 394–406
- 166 Li Z, Wang C, Ma P, et al. On extracting specialized code abilities from large language models: A feasibility study. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE)*. IEEE, 2024. 893–905
- 167 Lin Z, Cui J, Liao X, et al. Malla: Demystifying real-world large language model integrated malicious services. arXiv preprint arXiv:240103315, 2024
- 168 Liu Z, Tang Y, Luo X, et al. No need to lift a finger anymore? assessing the quality of code generation by chatgpt. *IEEE Transactions on Software Engineering*, 2024
- 169 Liu Y, Le-Cong T, Widyasari R, et al. Refining chatgpt-generated code: Characterizing and mitigating code quality issues. *ACM Transactions on Software Engineering and Methodology*, 2024. 33(5):1–26
- 170 Li Y, Liu S, Chen K, et al. Multi-target backdoor attacks for code pre-trained models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9–14, 2023*. Association for Computational Linguistics, 2023. 7236–7254
- 171 Nguyen P T, Di Rocco J, Di Sipio C, et al. Gptsniffer: A codebert-based classifier to detect source code written by chatgpt. *Journal of Systems and Software*, 2024. 214:112059
- 172 Song D, Xie X, Song J, et al. Luna: A model-based universal analysis framework for large language models. *IEEE Transactions on Software Engineering*, 2024
- 173 Xu Z and Sheng V S. Detecting ai-generated code assignments using perplexity of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38. 2024. 23155–23162
- 174 Li D, Yan M, Zhang Y, et al. Cosec: On-the-fly security hardening of code llms via supervised co-decoding. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2024. 1428–1439
- 175 Weyssow M, Zhou X, Kim K, et al. Exploring parameter-efficient fine-tuning techniques for code generation with large language models. arXiv preprint arXiv:230810462, 2023
- 176 Weyssow M, Zhou X, Kim K, et al. On the usage of continual learning for out-of-distribution generalization in pre-trained language models of code. arXiv preprint arXiv:230504106, 2023
- 177 Zhuo T Y, Zebaze A, Suppattarachai N, et al. Astraios: Parameter-efficient instruction tuning code large language models. arXiv preprint arXiv:240100788, 2024
- 178 Allamanis M, Panthapackel S, and Yin P. Unsupervised evaluation of code llms with round-trip correctness. arXiv preprint arXiv:240208699, 2024
- 179 Alshahwan N, Harman M, Harper I, et al. Assured llm-based software engineering. arXiv preprint arXiv:240204380, 2024
- 180 Antal G, Vozár R, and Ferenc R. Assessing gpt-4-vision’s capabilities in uml-based code generation. arXiv preprint arXiv:240414370, 2024
- 181 Arora C, Sayeed A I, Licorish S, et al. Optimizing large language model hyperparameters for code generation. arXiv preprint arXiv:240810577, 2024
- 182 Assogba Y and Ren D. Evaluating long range dependency handling in code generation models using multi-step key retrieval. arXiv preprint arXiv:240721049, 2024
- 183 Bairi R, Sonwane A, Kanade A, et al. Codeplan: Repository-level coding using llms and planning. *Proceedings of the ACM on Software Engineering*, 2024. 1(FSE):675–698
- 184 Bareiß P, Souza B, d’Amorim M, et al. Code generation tools (almost) for free? a study of few-shot, pre-trained language models on code. arXiv preprint arXiv:220601335, 2022
- 185 Buscemi A. A comparative study of code generation using chatgpt 3.5 across 10 programming languages. arXiv preprint arXiv:230804477, 2023
- 186 Busch D, Bainczyk A, and Steffen B. Towards llm-based system migration in language-driven engineering. In *International Conference on Engineering of Computer-Based Systems*. Springer, 2023. 191–200
- 187 Chen M, Zhang H, Wan C, et al. On the effectiveness of large language models in domain-specific code generation. arXiv preprint arXiv:231201639, 2023
- 188 Chen A, Scheurer J, Korbak T, et al. Improving code generation by training with natural language feedback. arXiv preprint arXiv:230316749, 2023
- 189 Chen X, Lin M, Schärli N, et al. Teaching large language models to self-debug. arXiv preprint arXiv:230405128, 2023
- 190 Chen J, Hu X, Li Z, et al. Code search is all you need? improving code suggestions with code search. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 2024. 1–13
- 191 Chen Y, Liu Y, Meng F, et al. Comments as natural logic pivots: Improve code generation via comment perspective. arXiv preprint arXiv:240407549, 2024
- 192 Chen M, Tian H, Liu Z, et al. Jumpcoder: Go beyond autoregressive coder via online modification. arXiv preprint arXiv:240107870, 2024
- 193 Chen B, Zhu M, Dolan-Gavitt B, et al. Model cascading for code: Reducing inference costs with model cascading for llm based code generation. arXiv preprint arXiv:240515842, 2024
- 194 Coignon T, Quinton C, and Rouvoy R. A performance study of llm-generated code on leetcode. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*. 2024. 79–89
- 195 Dai Z, Yao C, Han W, et al. Mpcoder: Multi-user personalized code generator with explicit and implicit style representation learning. arXiv preprint arXiv:240617255, 2024
- 196 Dibia V, Fourney A, Bansal G, et al. Aligning offline metrics and human judgments of value of ai-pair programmers. arXiv preprint arXiv:221016494, 2022
- 197 Dong G, Yuan H, Lu K, et al. How abilities in large language models are affected by supervised fine-tuning data composition. arXiv preprint arXiv:231005492, 2023
- 198 Dong Y, Ding J, Jiang X, et al. Codescore: Evaluating code generation by learning code execution. arXiv preprint arXiv:230109043, 2023
- 199 Dong Y, Jiang X, Jin Z, et al. Self-collaboration code generation via chatgpt. *ACM Trans Softw Eng Methodol*, 2024. ISSN 1049-331X. doi:10.1145/3672459. Just Accepted

- 200 Du K, Rui R, Chai H, et al. Codegrag: Extracting composed syntax graphs for retrieval augmented cross-lingual code generation. arXiv preprint arXiv:240502355, 2024
- 201 Fakhoury S, Chakraborty S, Musuvathi M, et al. Towards generating functionally correct code edits from natural language issue descriptions. arXiv preprint arXiv:230403816, 2023
- 202 Fan Z, Ruan H, Mehtaev S, et al. Oracle-guided program selection from large language models. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2024. 628–640
- 203 Feng Y, Vanam S, Cherukupally M, et al. Investigating code generation performance of chatgpt with crowdsourcing social data. In *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2023. 876–885
- 204 Fu M and Tantithamthavorn C. Gpt2sp: A transformer-based agile story point estimation approach. *IEEE Transactions on Software Engineering*, 2022. 49(2):611–625
- 205 Ghosh Paul D, Zhu H, and Bayley I. Benchmarks and metrics for evaluations of code generation: A critical review. arXiv e-prints, 2024:arXiv-2406
- 206 Gong L, Zhang J, Wei M, et al. 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*, 2023. 32(3):1–57
- 207 Gu Q. Llm-based code generation method for golang compiler testing. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2023. 2201–2203
- 208 Guo L, Wang Y, Shi E, et al. When to stop? towards efficient code generation in llms with excess token prevention. arXiv preprint arXiv:240720042, 2024
- 209 Han H, Kim J, Yoo J, et al. Archcode: Incorporating software requirements in code generation with large language models. arXiv preprint arXiv:240800994, 2024
- 210 Hassid M, Remez T, Gehring J, et al. The larger the better? improved llm code-generation via budget reallocation. arXiv preprint arXiv:240400725, 2024
- 211 Hong S, Zheng X, Chen J, et al. Metagpt: Meta programming for multi-agent collaborative framework. arXiv preprint arXiv:230800352, 2023
- 212 Hu X, Kuang K, Sun J, et al. Leveraging print debugging to improve code generation in large language models. arXiv preprint arXiv:240105319, 2024
- 213 Huang D, Bu Q, Zhang J M, et al. Agentcoder: Multi-agent-based code generation with iterative testing and optimisation. arXiv preprint arXiv:231213010, 2023
- 214 Huang D, Nan Z, Hu X, et al. Anpl: Compiling natural programs with interactive decomposition. CoRR, 2023
- 215 Huang D, Bu Q, Zhang J, et al. Bias assessment and mitigation in llm-based code generation. arXiv preprint arXiv:230914345, 2023
- 216 Huang D, Bu Q, and Cui H. Codecot and beyond: Learning to program and test like a developer. arXiv preprint arXiv:230808784, 2023
- 217 Huang B, Lu S, Chen W, et al. Enhancing large language models in coding through multi-perspective self-consistency. arXiv preprint arXiv:230917272, 2023
- 218 Huang T, Sun Z, Jin Z, et al. Knowledge-aware code generation with large language models. In *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension*. 2024. 52–63
- 219 Jain A, Adiole C, Chaudhuri S, et al. Coarse-tuning models of code with reinforcement learning feedback. arXiv preprint arXiv:230518341, 2023
- 220 Jain N, Zhang T, Chiang W L, et al. Llm-assisted code cleaning for training accurate code generators. arXiv preprint arXiv:231114904, 2023
- 221 Ji Z, Ma P, Li Z, et al. Benchmarking and explaining large language model-based code generation: A causality-centric approach. arXiv preprint arXiv:231006680, 2023
- 222 Jiang X, Dong Y, Wang L, et al. Self-planning code generation with large language models. *ACM Transactions on Software Engineering and Methodology*, 2024. 33(7):1–30
- 223 Jiang S, Wang Y, and Wang Y. Selfevolve: A code evolution framework via large language models. arXiv preprint arXiv:230602907, 2023
- 224 Jimenez C E, Yang J, Wettig A, et al. Swe-bench: Can language models resolve real-world github issues? arXiv preprint arXiv:231006770, 2023
- 225 Jin K, Wang C Y, Pham H V, et al. Can chatgpt support developers? an empirical evaluation of large language models for code generation. In *2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)*. IEEE, 2024. 167–171
- 226 Jones E and Steinhardt J. Capturing failures of large language models via human cognitive biases. *Advances in Neural Information Processing Systems*, 2022. 35:11785–11799
- 227 Kang D, Seo K J, and Kim T. Revisiting the impact of pursuing modularity for code generation. arXiv preprint arXiv:240711406, 2024
- 228 Kazemitabaar M, Hou X, Henley A, et al. How novices use llm-based code generators to solve cs1 coding tasks in a self-paced learning environment. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research*. 2023. 1–12
- 229 Khan M F A, Ramsdell M, Falor E, et al. Assessing the promise and pitfalls of chatgpt for automated code generation. arXiv preprint arXiv:231102640, 2023
- 230 Khan M A M, Bari M S, Do X L, et al. xcodeeval: A large scale multilingual multitask benchmark for code understanding, generation, translation and retrieval. arXiv preprint arXiv:230303004, 2023
- 231 Kou B, Chen S, Wang Z, et al. Do large language models pay similar attention like human programmers when generating code? *Proceedings of the ACM on Software Engineering*, 2024. 1(FSE):2261–2284
- 232 Koziolok H, Grüner S, Hark R, et al. Llm-based and retrieval-augmented control code generation. In *Proc. 1st Int. Workshop on Large Language Models for Coffee (LLM4Code) at ICSE*, volume 2024. 2024
- 233 Lahiri S K, Fakhoury S, Naik A, et al. Interactive code generation via test-driven user-intent formalization. arXiv preprint arXiv:220805950, 2022
- 234 Lai Y, Li C, Wang Y, et al. Ds-1000: A natural and reliable benchmark for data science code generation. In *International Conference on Machine Learning*. PMLR, 2023. 18319–18345
- 235 Laskar M T R, Bari M S, Rahman M, et al. A systematic study and comprehensive evaluation of chatgpt on benchmark datasets. arXiv preprint arXiv:230518486, 2023
- 236 Le H, Chen H, Saha A, et al. Codechain: Towards modular code generation through chain of self-revisions with representative sub-modules. arXiv preprint arXiv:231008992, 2023
- 237 Li P, Sun T, Tang Q, et al. Codeie: Large code generation models are better few-shot information extractors. arXiv preprint arXiv:230505711, 2023
- 238 Li K, Hong S, Fu C, et al. Discriminating human-authored from chatgpt-generated code via discernable feature analysis. In *2023 IEEE 34th International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2023. 120–127
- 239 Li J, Li G, Li Y, et al. Enabling programming thinking in large language models toward code generation. arXiv preprint

- arXiv:230506599, 2023
- 240 Li J, Chen P, and Jia J. Motcoder: Elevating large language models with modular of thought for challenging programming tasks. arXiv preprint arXiv:231215960, 2023
 - 241 Li Y, Shi J, and Zhang Z. A novel approach for rapid development based on chatgpt and prompt engineering. arXiv preprint arXiv:231213115, 2023
 - 242 Li J, Li Y, Li G, et al. Skcoder: A sketch-based approach for automatic code generation. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023. 2124–2135
 - 243 Li J, Li G, Li Y, et al. Structured chain-of-thought prompting for code generation. *ACM Transactions on Software Engineering and Methodology*, 2023
 - 244 Li X Y, Xue J T, Xie Z, et al. Think outside the code: Brainstorming boosts large language models in code generation. arXiv preprint arXiv:230510679, 2023
 - 245 Li J, Zhao Y, Li Y, et al. Acecoder: An effective prompting technique specialized in code generation. *ACM Transactions on Software Engineering and Methodology*, 2024
 - 246 Li M, Mishra A, and Mujumdar U. Bridging the language gap: Enhancing multilingual prompt-based code generation in llms via zero-shot cross-lingual transfer. arXiv preprint arXiv:240809701, 2024
 - 247 Li J, Li G, Zhao Y, et al. Deval: Evaluating code generation in practical software projects. arXiv preprint arXiv:240106401, 2024
 - 248 Li J and Mooney R. Distilling algorithmic reasoning from llms via explaining solution programs. arXiv preprint arXiv:240408148, 2024
 - 249 Li W D and Ellis K. Is programming by example solved by llms? arXiv preprint arXiv:240608316, 2024
 - 250 Lin F, Kim D J, et al. When llm-based code generation meets the software development process. arXiv preprint arXiv:240315852, 2024
 - 251 Liu J, Tang X, Li L, et al. Which is a better programming assistant? a comparative study between chatgpt and stack overflow. arXiv preprint arXiv:230813851, 2023
 - 252 Liu M, Yang T, Lou Y, et al. Codegen4libs: A two-stage approach for library-oriented code generation. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023. 434–445
 - 253 Liu C, Bao X, Zhang H, et al. Guiding chatgpt for better code generation: An empirical study. In *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2024. 102–113
 - 254 Liu C, Bao X, Zhang H, et al. Improving chatgpt prompt for code generation. arXiv preprint arXiv:230508360, 2023
 - 255 Lyu Z C, Li X Y, Xie Z, et al. Top pass: Improve code generation by pass@ k-maximized code ranking. arXiv preprint arXiv:240805715, 2024
 - 256 Ma Q, Wu T, and Koedinger K. Is ai the better programming partner? human-human pair programming vs. human-ai pair programming. arXiv preprint arXiv:230605153, 2023
 - 257 Ma Z, An S, Xie B, et al. Compositional api recommendation for library-oriented code generation. In *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension*. 2024. 87–98
 - 258 Malkadi A, Tayeb A, and Haiduc S. Improving code extraction from coding screencasts using a code-aware encoder-decoder model. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023. 1492–1504
 - 259 Mastropaolo A, Pascarella L, Guglielmi E, et al. On the robustness of code generation techniques: An empirical study on github copilot. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023. 2149–2160
 - 260 Murali V, Maddila C, Ahmad I, et al. Codecompose: A large-scale industrial deployment of ai-assisted code authoring. arXiv preprint arXiv:230512050, 2023
 - 261 Mu F, Shi L, Wang S, et al. Clarifygpt: Empowering llm-based code generation with intention clarification. arXiv preprint arXiv:231010996, 2023
 - 262 Nascimento N, Alencar P, and Cowan D. Comparing software developers with chatgpt: An empirical investigation. arXiv preprint arXiv:230511837, 2023
 - 263 Ni A, Yin P, Zhao Y, et al. L2ceval: Evaluating language-to-code generation capabilities of large language models. arXiv preprint arXiv:230917446, 2023
 - 264 Nichols D, Davis J H, Xie Z, et al. Can large language models write parallel code? arXiv preprint arXiv:240112554, 2024
 - 265 Nichols D, Polasam P, Menon H, et al. Performance-aligned llms for generating fast code. arXiv preprint arXiv:240418864, 2024
 - 266 Ni A, Iyer S, Radev D, et al. Lever: Learning to verify language-to-code generation with execution. In *International Conference on Machine Learning*. PMLR, 2023. 26106–26128
 - 267 Okuda K and Amarasinghe S. Askit: Unified programming interface for programming with large language models. In *2024 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. IEEE, 2024. 41–54
 - 268 Olausson T X, Inala J P, Wang C, et al. Demystifying gpt self-repair for code generation. arXiv preprint arXiv:230609896, 2023
 - 269 Ouyang S, Zhang J M, Harman M, et al. Llm is like a box of chocolates: the non-determinism of chatgpt in code generation. arXiv preprint arXiv:230802828, 2023
 - 270 Patel A, Reddy S, Bahdanau D, et al. Evaluating in-context learning of libraries for code generation. arXiv preprint arXiv:231109635, 2023
 - 271 Pegolotti T, Frantar E, Alistarh D, et al. Qigen: Generating efficient kernels for quantized inference on large language models. arXiv preprint arXiv:230703738, 2023
 - 272 Rao N, Tsay J, Kate K, et al. Ai for low-code for ai. In *Proceedings of the 29th International Conference on Intelligent User Interfaces*. 2024. 837–852
 - 273 Ren X, Ye X, Zhao D, et al. From misuse to mastery: Enhancing code generation with knowledge-driven ai chaining. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023. 976–987
 - 274 Riddell M, Ni A, and Cohan A. Quantifying contamination in evaluating code generation capabilities of language models. arXiv preprint arXiv:240304811, 2024
 - 275 Ridnik T, Kredo D, and Friedman I. Code generation with alphacodium: From prompt engineering to flow engineering. arXiv preprint arXiv:240108500, 2024
 - 276 Sarker L, Downing M, Desai A, et al. Syntactic robustness for llm-based code generation. arXiv preprint arXiv:240401535, 2024
 - 277 Shapkin A, Litvinov D, and Bryksin T. Entity-augmented code generation. arXiv preprint arXiv:231208976, 2023
 - 278 Siddiq M L, Casey B, and Santos J. A lightweight framework for high-quality code generation. arXiv preprint arXiv:230708220, 2023
 - 279 Siddiq M L, Dristi S, Saha J, et al. Quality assessment of prompts used in code generation. arXiv preprint arXiv:240410155, 2024
 - 280 Singh A K, Yang Y, Tirumala K, et al. Brevity is the soul of wit: Pruning long files for code generation. arXiv preprint arXiv:240700434, 2024

- 281 Su H, Kasai J, Wu C H, et al. Selective annotation makes language models better few-shot learners. arXiv preprint arXiv:220901975, 2022
- 282 Sun C, Sheng Y, Padon O, et al. Clover: Clo sed-loop ver ifiable code generation. In *International Symposium on AI Verification*. Springer, 2024. 134–155
- 283 Taherkhani H, Sepindband M, Pham H V, et al. Epic: Cost-effective search-based prompt engineering of llms for code generation. arXiv preprint arXiv:240811198, 2024
- 284 Tambon F, Nikanjam A, Khomh F, et al. Assessing programming task difficulty for efficient evaluation of large language models. arXiv preprint arXiv:240721227, 2024
- 285 Tan C W, Guo S, Wong M F, et al. Copilot for xcode: Exploring ai-assisted programming by prompting cloud-based large language models. arXiv preprint arXiv:230714349, 2023
- 286 Tarassow A. The potential of llms for coding with low-resource and domain-specific programming languages. arXiv preprint arXiv:230713018, 2023
- 287 Thakur S, Ahmad B, Pearce H, et al. Verigen: A large language model for verilog code generation. *ACM Transactions on Design Automation of Electronic Systems*, 2024. 29(3):1–31
- 288 Tian Z and Chen J. Test-case-driven programming understanding in large language models for better code generation. arXiv preprint arXiv:230916120, 2023
- 289 Tian Y and Zhang T. Selective prompt anchoring for code generation. arXiv preprint arXiv:240809121, 2024
- 290 To H, Nguyen M, and Bui N. Functional overlap reranking for neural code generation. In *Findings of the Association for Computational Linguistics ACL 2024*. 2024. 3686–3704
- 291 Tsai Y D, Liu M, and Ren H. Code less, align more: Efficient llm fine-tuning for code generation with data pruning. arXiv preprint arXiv:240705040, 2024
- 292 Ugare S, Suresh T, Kang H, et al. Improving llm code generation with grammar augmentation. arXiv preprint arXiv:240301632, 2024
- 293 Wang S, Li Z, Qian H, et al. Recode: Robustness evaluation of code generation models. arXiv preprint arXiv:221210264, 2022
- 294 Wang Z, Li J, Li G, et al. Chatcoder: Chat-based refine requirement improves llms' code generation. arXiv preprint arXiv:231100272, 2023
- 295 Wang X, Peng H, Jabbarvand R, et al. Leti: Learning to generate from textual interactions. arXiv preprint arXiv:230510314, 2023
- 296 Wang D, Zhao J, Pei H, et al. Fine-tuning language models for joint rewriting and completion of code with potential bugs. In *Findings of the Association for Computational Linguistics ACL 2024*. 2024. 15854–15868
- 297 Wang S, Ding L, Shen L, et al. Oop: Object-oriented programming evaluation benchmark for large language models. arXiv preprint arXiv:240106628, 2024
- 298 Wang C, Zhang J, Feng Y, et al. Teaching code llms to use autocompletion tools in repository-level code generation. *ACM Transactions on Software Engineering and Methodology*, 2025. 34(7):1–27
- 299 Wu F, Liu X, and Xiao C. Deceptprompt: Exploiting llm-driven code generation via adversarial natural language instructions. arXiv preprint arXiv:231204730, 2023
- 300 Wu J, Schoop E, Leung A, et al. Uicoder: Finetuning large language models to generate user interface code through automated feedback. arXiv preprint arXiv:240607739, 2024
- 301 Wu X, Cheri re N, Zhang C, et al. Rustgen: An augmentation approach for generating compilable rust code with large language models. <https://openreview.net/forum?id=y9A0vJ5vuM>, 2023. OpenReview submission
- 302 Xu J, Liu Z, Suryanarayanan N A V, et al. Large language models synergize with automated machine learning. arXiv preprint arXiv:240503727, 2024
- 303 Yan D, Gao Z, and 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)*. IEEE, 2023. 1887–1898
- 304 Yang K, Mao X, Wang S, et al. Enhancing code intelligence tasks with chatgpt. arXiv preprint arXiv:231215202, 2023
- 305 Yang G, Zhou Y, Chen X, et al. Exploitgen: Template-augmented exploit code generation based on codebert. *Journal of Systems and Software*, 2023. 197:111577
- 306 Yang G, Zhou Y, Chen X, et al. A syntax-guided multi-task learning approach for turducken-style code generation. *Empirical Software Engineering*, 2023. 28(6):141
- 307 Yen R, Zhu J, Suh S, et al. Coladder: Supporting programmers with hierarchical code generation in multi-level abstraction. arXiv preprint arXiv:231008699, 2023
- 308 Yeti tiren B,  zsoy I, Ayerdem M, et al. Evaluating the code quality of ai-assisted code generation tools: An empirical study on github copilot, amazon codewhisperer, and chatgpt. arXiv preprint arXiv:230410778, 2023
- 309 Zan D, Chen B, Lin Z, et al. When language model meets private library. arXiv preprint arXiv:221017236, 2022
- 310 Zan D, Chen B, Gong Y, et al. Private-library-oriented code generation with large language models. arXiv preprint arXiv:230715370, 2023
- 311 Zelikman E, Lorch E, Mackey L, et al. Self-taught optimizer (stop): Recursively self-improving code generation. arXiv preprint arXiv:231002304, 2023
- 312 Zhang T, Yu T, Hashimoto T, et al. Coder reviewer reranking for code generation. In *International Conference on Machine Learning*. PMLR, 2023. 41832–41846
- 313 Zhang S, Chen Z, Shen Y, et al. Planning with large language models for code generation. arXiv preprint arXiv:230305510, 2023
- 314 Zhang K, Li J, Li G, et al. Codeagent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. arXiv preprint arXiv:240107339, 2024
- 315 Zhang W, Fu T, Yuan T, et al. A lightweight framework for adaptive retrieval in code completion with critique model. arXiv preprint arXiv:240610263, 2024
- 316 Zhang K, Li Z, Li J, et al. Self-edit: Fault-aware code editor for code generation. arXiv preprint arXiv:230504087, 2023
- 317 Zheng W, Sharan S, Jaiswal A K, et al. Outline, then details: Syntactically guided coarse-to-fine code generation. In *International Conference on Machine Learning*. PMLR, 2023. 42403–42419
- 318 Zheng L, Yuan J, Zhang Z, et al. Self-infilling code generation. In *Forty-first International Conference on Machine Learning*. 2023
- 319 Zhong L and Wang Z. Can chatgpt replace stackoverflow? a study on robustness and reliability of large language model code generation. arXiv preprint arXiv:230810335, 2023
- 320 Zhong L and Wang Z. Can llm replace stack overflow? a study on robustness and reliability of large language model code generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38. 2024. 21841–21849
- 321 Zhou S, Alon U, Agarwal S, et al. Codebertscore: Evaluating code generation with pretrained models of code. arXiv preprint arXiv:230205527, 2023
- 322 Zhu Y, Li J, Li G, et al. Hot or cold? adaptive temperature sampling for code generation with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38. 2024. 437–445
- 323 Zhu-Tian C, Xiong Z, Yao X, et al. Sketch then generate: Providing incremental user feedback and guiding llm code

- generation through language-oriented code sketches. arXiv preprint arXiv:240503998, 2024
- 324 Wang S, Lin B, Sun Z, et al. Two birds with one stone: Boosting code generation and code search via a generative adversarial network. *Proceedings of the ACM on Programming Languages*, 2023. 7(OOPSLA2):486–515
- 325 Wang S, Geng M, Lin B, et al. 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*. 2023. 375–387
- 326 Xue T, Li X, Azim T, et al. Multi-programming language ensemble for code generation in large language model. arXiv preprint arXiv:240904114, 2024
- 327 Wang E, Cassano F, Wu C, et al. Planning in natural language improves llm search for code generation. arXiv preprint arXiv:240903733, 2024
- 328 Zhang Q, Fang C, Shang Y, et al. No man is an island: Towards fully automatic programming by code search, code generation and program repair. arXiv preprint arXiv:240903267, 2024
- 329 Zhang W, Leon M, Xu R, et al. Benchmarking llm code generation for audio programming with visual dataflow languages. arXiv preprint arXiv:240900856, 2024
- 330 Zhu Q, Liang Q, Sun Z, et al. Grammart5: Grammar-integrated pretrained encoder-decoder neural model for code. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 2024. 1–13
- 331 Quoc T T, Minh D H, Thanh T Q, et al. An empirical study on self-correcting large language models for data science code generation. arXiv preprint arXiv:240815658, 2024
- 332 Luo Z, Xu C, Zhao P, et al. Wizardcoder: Empowering code large language models with evol-instruct. arXiv preprint arXiv:230608568, 2023
- 333 Liu J, Zhu Y, Xiao K, et al. Rlrf: Reinforcement learning from unit test feedback. arXiv preprint arXiv:230704349, 2023
- 334 Rahmani S, Naghshzan A, and Guerrouj L. Improving code example recommendations on informal documentation using bert and query-aware lsh: A comparative study. arXiv preprint arXiv:230503017, 2023
- 335 Zong X, Zheng S, Zou H, et al. Graphpyrec: A novel graph-based approach for fine-grained python code recommendation. *Science of Computer Programming*, 2024:103166
- 336 Xian Z, Cui C, Huang R, et al. zslmlcode: An effective approach for functional code embedding via llm with zero-shot learning. arXiv preprint arXiv:240914644, 2024
- 337 Agarwal M, Shen Y, Wang B, et al. Structured code representations enable data-efficient adaptation of code language models. arXiv preprint arXiv:240110716, 2024
- 338 Cui L, Yin J, Cui J, et al. Api2vec++: Boosting api sequence representation for malware detection and classification. *IEEE Transactions on Software Engineering*, 2024
- 339 He J, Zhou X, Xu B, et al. Representation learning for stack overflow posts: How far are we? *ACM Transactions on Software Engineering and Methodology*, 2024. 33(3):1–24
- 340 Lin Y, Wan C, Bai S, et al. Vargan: Adversarial learning of variable semantic representations. *IEEE Transactions on Software Engineering*, 2024
- 341 Liu S, Wu B, Xie X, et al. Contrabert: Enhancing code pre-trained models via contrastive learning. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 2023. 2476–2487
- 342 Saberi I and Fard F H. Model-agnostic syntactical information for pre-trained programming language models. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. IEEE, 2023. 183–193
- 343 Li X, Dong K, Lee Y Q, et al. Coir: A comprehensive benchmark for code information retrieval models. arXiv preprint arXiv:240702883, 2024
- 344 Sun W, Chen Y, Tao G, et al. Backdoorling neural code search. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2023. 9692–9708
- 345 Wan Y, Zhang S, Zhang H, et al. You see what i want you to see: poisoning vulnerabilities in neural code search. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2022. 1233–1245
- 346 Wang D, Chen B, Li S, et al. One adapter for all programming languages? adapter tuning for code search and summarization. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023. 5–16
- 347 Chi K, Li C, Ge J, et al. An empirical study on code search pre-trained models: Academic progresses vs. industry requirements. In *Proceedings of the 15th Asia-Pacific Symposium on Internetwork*. 2024. 41–50
- 348 Fan G, Chen S, Gao C, et al. Rapid: Zero-shot domain adaptation for code search with pre-trained models. *ACM Transactions on Software Engineering and Methodology*, 2024. 33(5):1–35
- 349 Jain S, Dora A, Sam K S, et al. Llm agents improve semantic code search. arXiv preprint arXiv:240811058, 2024
- 350 Li D, Shen Y, Jin R, et al. Generation-augmented query expansion for code retrieval. arXiv preprint arXiv:221210692, 2022
- 351 Li J, Liu F, Li J, et al. Mcodesearcher: Multi-view contrastive learning for code search. In *Proceedings of the 14th Asia-Pacific Symposium on Internetwork*. 2023. 270–280
- 352 Li Y, Zhang T, Luo X, et al. Do pre-trained language models indeed understand software engineering tasks? *IEEE Transactions on Software Engineering*, 2023
- 353 Li H, Zhou X, and Shen Z. Rewriting the code: A simple method for large language model augmented code search. arXiv preprint arXiv:240104514, 2024
- 354 Li X, Gong Y, Shen Y, et al. Coderetriever: A large scale contrastive pre-training method for code search. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 2022. 2898–2910
- 355 Mao Y, Wan C, Jiang Y, et al. 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*. 2023. 363–374
- 356 Saieva A, Chakraborty S, and Kaiser G. On contrastive learning of semantic similarity forcode to code search. arXiv preprint arXiv:230503843, 2023
- 357 Salza P, Schwizer C, Gu J, et al. On the effectiveness of transfer learning for code search. *IEEE Transactions on Software Engineering*, 2022
- 358 Shi Z, Xiong Y, Zhang X, et al. Cross-modal contrastive learning for code search. In *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2022. 94–105
- 359 Shi E, Wang Y, Gu W, et al. Cocosoda: Effective contrastive learning for code search. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023. 2198–2210
- 360 Shi Z, Xiong Y, Zhang Y, et al. Improving code search with multi-modal momentum contrastive learning. In *2023 IEEE/ACM 31st International Conference on Program Comprehension (ICPC)*. IEEE, 2023. 280–291
- 361 Varkey A, Jiang S, and Huang W. Codecse: A simple multilingual model for code and comment sentence embeddings. arXiv preprint arXiv:240706360, 2024
- 362 Wang Y, Guo L, Shi E, et al. You augment me: Exploring chatgpt-based data augmentation for semantic code search. In *2023 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2023. 14–25
- 363 Sorokin N, Abulkhanov D, Nikolenko S, et al. Cct-code: Cross-consistency training for multilingual clone detection and code search. arXiv preprint arXiv:230511626, 2023
- 364 Su C Y and McMillan C. Distilled gpt for source code summarization. *Automated Software Engineering*, 2024. 31(1):22
- 365 Saberi I, Fard F, and Chen F. Utilization of pre-trained language models for adapter-based knowledge transfer in software

- engineering. *Empirical Software Engineering*, 2024. 29(4):94
- 366 Jha A and Reddy C K. Codeattack: Code-based adversarial attacks for pre-trained programming language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 2023. 14892–14900
- 367 Liu D and Zhang S. Alanca: Active learning guided adversarial attacks for code comprehension on diverse pre-trained and large language models. In *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2024. 602–613
- 368 Wang Y, Wang K, and Wang L. An explanation method for models of code. *Proceedings of the ACM on Programming Languages*, 2023. 7(OOPSLA2):801–827
- 369 Yang Z, Xu B, Zhang J M, et al. Stealthy backdoor attack for code models. *IEEE Transactions on Software Engineering*, 2024
- 370 Gao S, Mao W, Gao C, et al. 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*. 2024. 1–13
- 371 Gao S, Zhang H, Gao C, et al. Keeping pace with ever-increasing data: Towards continual learning of code intelligence models. *arXiv preprint arXiv:230203482*, 2023
- 372 Liu J, Sha C, and Peng X. An empirical study of parameter-efficient fine-tuning methods for pre-trained code models. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023. 397–408
- 373 Ahmed T, Pai K S, Devanbu P, et al. Automatic semantic augmentation of language model prompts (for code summarization). In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 2024. 2720–2732
- 374 Al-Kaswan A, Ahmed T, Izadi M, et al. Extending source code pre-trained language models to summarise decompiled binaries. In *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2023. 260–271
- 375 Arakelyan S, Das R, Mao Y, et al. Exploring distributional shifts in large language models for code analysis. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 2023. 16298–16314
- 376 Chen F, Fard F H, Lo D, et al. 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*. 2022. 401–412
- 377 Drivedi S S, Vijay V, Pujari S L R, et al. A comparative analysis of large language models for code documentation generation. *arXiv preprint arXiv:231210349*, 2023
- 378 Fan G, Chen S, Wu H, et al. Dialog summarization for software collaborative platform via tuning pre-trained models. *Journal of Systems and Software*, 2023. 204:111763
- 379 Fang C, Sun W, Chen Y, et al. Esale: Enhancing code-summary alignment learning for source code summarization. *IEEE Transactions on Software Engineering*, 2024. 50(8):2077–2095
- 380 Gao S, Wen X C, Gao C, et al. Constructing effective in-context demonstration for code intelligence tasks: An empirical study. *arXiv preprint arXiv:230407575*, 2023:17
- 381 Geng M, Wang S, Dong D, et al. Large language models are few-shot summarizers: Multi-intent comment generation via in-context learning. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 2024. 1–13
- 382 Gu J, Salza P, and Gall H C. Assemble foundation models for automatic code summarization. In *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2022. 935–946
- 383 Haldar R and Hockenmaier J. Analyzing the performance of large language models on code summarization. *arXiv preprint arXiv:240408018*, 2024
- 384 Jin X, Larson B, Yang W, et al. Binary code summarization: Benchmarking chatgpt/gpt-4 and other large language models. *arXiv preprint arXiv:231209601*, 2023
- 385 Jin X and Lin Z. Simllm: Calculating semantic similarity in code summaries using a large language model-based approach. *Proceedings of the ACM on Software Engineering*, 2024. 1(FSE):1376–1399
- 386 Kang S, Milliken L, and Yoo S. Identifying inaccurate descriptions in llm-generated code comments via test execution. *arXiv preprint arXiv:240614836*, 2024
- 387 Kumar J and Chimalakonda S. Code summarization without direct access to code-towards exploring federated llms for software engineering. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*. 2024. 100–109
- 388 Li L, Liang B, Chen L, et al. Cross-modal retrieval-enhanced code summarization based on joint learning for retrieval and generation. *Information and Software Technology*, 2024. 175:107527
- 389 Li J, Zhang Y, Karas Z, et al. Do machines and humans focus on similar code? exploring explainability of large language models in code summarization. In *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension*. 2024. 47–51
- 390 Lu H, Peng H, Nan G, et al. Malsight: Exploring malicious source code and benign pseudocode for iterative binary malware summarization. *arXiv preprint arXiv:240618379*, 2024
- 391 Oh S and Yoo S. Csa-trans: Code structure aware transformer for ast. *arXiv preprint arXiv:240405767*, 2024
- 392 Pordanesh S and Tan B. Exploring the efficacy of large language models (gpt-4) in binary reverse engineering. *arXiv preprint arXiv:240606637*, 2024
- 393 Poudel B, Cook A, Traore S, et al. Documint: Docstring generation for python using small language models. *arXiv preprint arXiv:240510243*, 2024
- 394 Rukmono S A, Ochoa L, and Chaudron M R. Achieving high-level software component summarization via hierarchical chain-of-thought prompting and static code analysis. In *2023 IEEE International Conference on Data and Software Engineering (ICoDSE)*. IEEE, 2023. 7–12
- 395 Saberi I, Fard F, and Chen F. Multilingual adapter-based knowledge aggregation on code summarization for low-resource languages. *arXiv preprint arXiv:230707854*, 2023
- 396 Sadik A R, Ceravola A, Joublin F, et al. Analysis of chatgpt on source code. *arXiv preprint arXiv:230600597*, 2023
- 397 Shen Y, Ju X, Chen X, et al. Bash comment generation via data augmentation and semantic-aware codebert. *Automated Software Engineering*, 2024. 31(1):30
- 398 Shi E, Zhang F, Wang Y, et al. Sotana: The open-source software development assistant. *arXiv preprint arXiv:230813416*, 2023
- 399 Shi K, Altınbüken D, Anand S, et al. Natural language outlines for code: Literate programming in the llm era. *arXiv preprint arXiv:240804820*, 2024
- 400 Su C Y and McMillan C. Semantic similarity loss for neural source code summarization. *Journal of Software: Evolution and Process*, 2023:e2706
- 401 Su C Y, Bansal A, Huang Y, et al. Context-aware code summary generation. *arXiv preprint arXiv:240809006*, 2024
- 402 Sun W, Fang C, You Y, et al. Automatic code summarization via chatgpt: How far are we? *arXiv preprint arXiv:230512865*, 2023
- 403 Sun W, Fang C, You Y, et al. A prompt learning framework for source code summarization. *arXiv preprint arXiv:231216066*, 2023

- 404 Sun W, Miao Y, Li Y, et al. Source code summarization in the era of large language models. arXiv preprint arXiv:240707959, 2024
- 405 Szalontai B, Szalay G, Márton T, et al. Large language models for code summarization, 2024
- 406 Virk Y, Devanbu P, and Ahmed T. Enhancing trust in llm-generated code summaries with calibrated confidence scores. arXiv preprint arXiv:240419318, 2024
- 407 Wang C, Lou Y, Liu J, et al. Generating variable explanations via zero-shot prompt learning. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023. 748–760
- 408 Wang Y, Li X, Nguyen T, et al. Natural is the best: Model-agnostic code simplification for pre-trained large language models. arXiv preprint arXiv:240511196, 2024
- 409 Wang Y, Huang Y, Guo D, et al. Sparsecoder: Identifier-aware sparse transformer for file-level code summarization. arXiv preprint arXiv:240114727, 2024
- 410 Zhao J, Chen X, Yang G, et al. Automatic smart contract comment generation via large language models and in-context learning. *Information and Software Technology*, 2024. 168:107405
- 411 Shin J, Tang C, Mohati T, et al. Prompt engineering or fine tuning: An empirical assessment of large language models in automated software engineering tasks. arXiv preprint arXiv:231010508, 2023
- 412 Baltaji R, Pujar S, Mandel L, et al. Learning transfers over several programming languages. arXiv preprint arXiv:231016937, 2023
- 413 Bhattarai M, Santos J E, Jones S, et al. Enhancing code translation in language models with few-shot learning via retrieval-augmented generation. arXiv preprint arXiv:240719619, 2024
- 414 Bui N D, Le H, Wang Y, et al. Codetf: One-stop transformer library for state-of-the-art code llm. arXiv preprint arXiv:230600029, 2023
- 415 Dearing M T, Tao Y, Wu X, et al. Lassi: An llm-based automated self-correcting pipeline for translating parallel scientific codes. arXiv preprint arXiv:240701638, 2024
- 416 Eniser H F, Zhang H, David C, et al. Towards translating real-world code with llms: A study of translating to rust. arXiv preprint arXiv:240511514, 2024
- 417 Huang Y, Qi M, Yao Y, et al. Program translation via code distillation. arXiv preprint arXiv:231011476, 2023
- 418 Jana P, Jha P, Ju H, et al. Cotran: An llm-based code translator using reinforcement learning with feedback from compiler and symbolic execution. arXiv preprint arXiv:230606755, 2023
- 419 Li X, Yuan S, Gu X, et al. Few-shot code translation via task-adapted prompt learning. *Journal of Systems and Software*, 2024. 212:112002
- 420 Macedo M, Tian Y, Cogo F, et al. Exploring the impact of the output format on the evaluation of large language models for code translation. In *Proceedings of the 2024 IEEE/ACM First International Conference on AI Foundation Models and Software Engineering*. 2024. 57–68
- 421 Nitin V and Ray B. Spectra: Enhancing the code translation ability of language models by generating multi-modal specifications. arXiv preprint arXiv:240518574, 2024
- 422 Pan J, Sadé A, Kim J, et al. Stelocoder: a decoder-only llm for multi-language to python code translation. arXiv preprint arXiv:231015539, 2023
- 423 Pan R, Ibrahimzada A R, Krishna R, et al. Lost in translation: A study of bugs introduced by large language models while translating code. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE)*. IEEE, 2024. 995–1007
- 424 Qi M, Huang Y, Wang M, et al. Sut: Active defects probing for transcompiler models. arXiv preprint arXiv:231014209, 2023
- 425 Tang Z, Agarwal M, Shypula A, et al. Explain-then-translate: an analysis on improving program translation with self-generated explanations. arXiv preprint arXiv:231107070, 2023
- 426 Wang B, Li R, Li M, et al. Transmap: Pinpointing mistakes in neural code translation. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2023. 999–1011
- 427 Xue M, Andrzejak A, and Leuther M. An interpretable error correction method for enhancing code-to-code translation. In *The Twelfth International Conference on Learning Representations*. 2024. 1–18
- 428 Yan W, Tian Y, Li Y, et al. Codetransocean: A comprehensive multilingual benchmark for code translation. arXiv preprint arXiv:231004951, 2023
- 429 Yang G, Zhou Y, Zhang X, et al. Assessing and improving syntactic adversarial robustness of pre-trained models for code translation. arXiv preprint arXiv:231018587, 2023
- 430 Yang Z, Liu F, Yu Z, et al. Exploring and unleashing the power of large language models in automated code translation. arXiv preprint arXiv:240414646, 2024
- 431 Yang A Z, Takashima Y, Paulsen B, et al. Vert: Verified equivalent rust transpilation with few-shot learning. arXiv preprint arXiv:240418852, 2024
- 432 Yin X, Ni C, Nguyen T N, et al. Rectifier: Code translation with corrector via llms. arXiv preprint arXiv:240707472, 2024
- 433 Zhu M, Suresh K, and Reddy C K. Multilingual code snippets training for program translation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36. 2022. 11783–11790
- 434 Jiao M, Yu T, Li X, et al. On the evaluation of neural code translation: Taxonomy and benchmark. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023. 1529–1541
- 435 Zhou B, Wang X, Xu S, et al. Hybrid api migration: A marriage of small api mapping models and large language models. In *Proceedings of the 14th Asia-Pacific Symposium on Internetworking*. 2023. 12–21
- 436 Wan Y, Zhao W, Zhang H, et al. What do they capture? a structural analysis of pre-trained language models for source code. In *Proceedings of the 44th International Conference on Software Engineering*. 2022. 2377–2388
- 437 Artuso F, Mormando M, Di Luna G A, et al. Binbert: Binary code understanding with a fine-tunable and execution-aware transformer. *IEEE Transactions on Dependable and Secure Computing*, 2024
- 438 Ding Y, Peng J, Min M J, et al. Semcoder: Training code language models with comprehensive semantics. arXiv preprint arXiv:240601006, 2024
- 439 Khakhar A, Mell S, and Bastani O. Pac prediction sets for large language models of code. In *International Conference on Machine Learning*. PMLR, 2023. 16237–16249
- 440 Ma W, Liu S, Wang W, et al. The scope of chatgpt in software engineering: A thorough investigation. arXiv preprint arXiv:230512138, 2023
- 441 Paranjape B, Lundberg S, Singh S, et al. Art: Automatic multi-step reasoning and tool-use for large language models. arXiv preprint arXiv:230309014, 2023
- 442 Pei H, Zhao J, Lausen L, et al. Better context makes better code language models: A case study on function call argument completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37. 2023. 5230–5238
- 443 Shen D, Chen X, Wang C, et al. Benchmarking language models for code syntax understanding. arXiv preprint arXiv:221014473, 2022
- 444 Shi J, Jiang S, Xu B, et al. Shellgpt: Generative pre-trained transformer model for shell language understanding. In *2023*

- IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2023. 671–682
- 445 Utpala S, Gu A, and Chen P Y. Language agnostic code embeddings. arXiv preprint arXiv:231016803, 2023
- 446 Zhao J, Rong Y, Guo Y, et al. Understanding programs by exploiting (fuzzing) test cases. arXiv preprint arXiv:230513592, 2023
- 447 Abdelaziz I, Dolby J, McCusker J, et al. Can machines read coding manuals yet?—a benchmark for building better language models for code understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36. 2022. 4415–4423
- 448 Nam D, Macvean A, Hellendoorn V, et al. In-side generation-based information support with a large language model. arXiv preprint arXiv:230708177, 2023
- 449 Nam D, Macvean A, Hellendoorn V, et al. Using an llm to help with code understanding. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 2024. 1–13
- 450 Baral T, Rahman S, Chanumolu B N, et al. Optimizing continuous development by detecting and preventing unnecessary content generation. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023. 901–913
- 451 Zhang Y, Wu Y, Chen T, et al. How do developers talk about github actions? evidence from online software development community. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE)*. IEEE, 2024. 492–504
- 452 Huang Q, Zou Z, Xing Z, et al. Ai chain on large language model for unsupervised control flow graph generation for statically-typed partial code. arXiv preprint arXiv:230600757, 2023
- 453 Cheng L, Li X, and Bing L. Is gpt-4 a good data analyst? arXiv preprint arXiv:230515038, 2023
- 454 Abdellatif A, Badran K, Costa D E, et al. A transformer-based approach for augmenting software engineering chatbots datasets. arXiv preprint arXiv:240711955, 2024
- 455 Banday B H, Islam T Z, and Marathe A. Perfgen: A synthesis and evaluation framework for performance data using generative ai. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2024. 188–197
- 456 Zhang J, Liu S, Gong L, et al. Beqain: An effective and efficient identifier normalization approach with bert and the question answering system. *IEEE Transactions on Software Engineering*, 2022. 49(4):2597–2620
- 457 Zhu J, Li L, Yang L, et al. Automating method naming with context-aware prompt-tuning. In *2023 IEEE/ACM 31st International Conference on Program Comprehension (ICPC)*. IEEE, 2023. 203–214
- 458 Alsayed A S, Dam H K, and Nguyen C. Microrec: Leveraging large language models for microservice recommendation. In *2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)*. IEEE, 2024. 419–430
- 459 Nasir M U, Earle S, Togelius J, et al. Llmatic: Neural architecture search via large language models and quality diversity optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 2024. 1110–1118
- 460 Austin J, Odena A, Nye M, et al. Program synthesis with large language models. arXiv preprint arXiv:210807732, 2021
- 461 Barke S, Gonzalez E A, Kasibatla S R, et al. Hysynth: Context-free llm approximation for guiding program synthesis. arXiv preprint arXiv:240515880, 2024
- 462 Gandhi A, Nguyen T Q, Jiao H, et al. Natural language commanding via program synthesis. arXiv preprint arXiv:230603460, 2023
- 463 Hajali P and Budvytis I. Function-constrained program synthesis. arXiv preprint arXiv:231115500, 2023
- 464 Jain N, Vaidyanath S, Iyer A, et al. Jigsaw: Large language models meet program synthesis. In *Proceedings of the 44th International Conference on Software Engineering*. 2022. 1219–1231
- 465 Kuznia K, Mishra S, Parmar M, et al. Less is more: Summary of long instructions is better for program synthesis. arXiv preprint arXiv:220308597, 2022
- 466 Li Y, Parsert J, and Polgreen E. Guiding enumerative program synthesis with large language models. In *International Conference on Computer Aided Verification*. Springer, 2024. 280–301
- 467 Liventsev V, Grishina A, Härmä A, et al. Fully autonomous programming with large language models. arXiv preprint arXiv:230410423, 2023
- 468 Shirafuji A, Watanobe Y, Ito T, et al. Exploring the robustness of large language models for solving programming problems. arXiv preprint arXiv:230614583, 2023
- 469 Singla A. Evaluating chatgpt and gpt-4 for visual programming. In *Proceedings of the 2023 ACM Conference on International Computing Education Research-Volume 2*. 2023. 14–15
- 470 Tao N, Ventresque A, Nallur V, et al. Enhancing program synthesis with large language models using many-objective grammar-guided genetic programming. *Algorithms*, 2024. 17(7):287
- 471 Vella Zarb D, Parks G, and Kipourous T. Synergistic utilization of llms for program synthesis. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 2024. 539–542
- 472 Ye J, Li C, Kong L, et al. Generating data for symbolic language with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 2023. 8418–8443
- 473 Schroder M. Autoscrum: Automating project planning using large language models. arXiv preprint arXiv:230603197, 2023
- 474 Le D A, Bui A M, Nguyen P T, et al. Good things come in three: Generating so post titles with pre-trained models, self improvement and post ranking. arXiv preprint arXiv:240615633, 2024
- 475 Yang S, Chen X, Liu K, et al. Automatic bi-modal question title generation for stack overflow with prompt learning. *Empirical Software Engineering*, 2024. 29(3):63
- 476 Firouzi E and Ghafari M. Time to separate from stackoverflow and match with chatgpt for encryption. *Journal of Systems and Software*, 2024:112135
- 477 Kabir S, Udo-Imeh D N, Kou B, et al. Is stack overflow obsolete? an empirical study of the characteristics of chatgpt answers to stack overflow questions. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 2024. 1–17
- 478 Jesse K, Devanbu P T, and Sawant A. Learning to predict user-defined types. *IEEE Transactions on Software Engineering*, 2022. 49(4):1508–1522
- 479 Qian C, Liu W, Liu H, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2024. 15174–15186
- 480 Ge X, Fang C, Zhang Q, et al. Pre-trained model-based actionable warning identification: A feasibility study. arXiv preprint arXiv:240302716, 2024
- 481 Yan S, Wang S, Duan Y, et al. An llm-assisted easy-to-trigger backdoor attack on code completion models: Injecting disguised vulnerabilities against strong detection. arXiv preprint arXiv:240606822, 2024
- 482 Zhang H, Lu S, Li Z, et al. Codebert-attack: Adversarial attack against source code deep learning models via pre-trained model. *Journal of Software: Evolution and Process*, 2024. 36(3):e2571
- 483 Li J, Yang Y, Wu Z, et al. Chatgpt as an attack tool: Stealthy textual backdoor attack via blackbox generative model trigger. arXiv preprint arXiv:230414475, 2023
- 484 Xia Y, Xie Z, Liu P, et al. Exploring automatic cryptographic api misuse detection in the era of llms. arXiv preprint arXiv:240716576, 2024
- 485 Le T, Tran T, Cao D, et al. Kat: Dependency-aware automated api testing with large language models. In *2024 IEEE*

- Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2024. 82–92
- 486 Dinella E, Ryan G, Mytkowicz T, et al. Toga: A neural method for test oracle generation. In *Proceedings of the 44th International Conference on Software Engineering*. 2022. 2130–2141
- 487 Endres M, Fakhoury S, Chakraborty S, et al. Can large language models transform natural language intent into formal method postconditions? *Proceedings of the ACM on Software Engineering*, 2024. 1(FSE):1889–1912
- 488 He F, Zhai J, and Pan M. Beyond code generation: Assessing code llm maturity with postconditions. *arXiv preprint arXiv:240714118*, 2024
- 489 He Y, Huang J, Yu H, et al. An empirical study on focal methods in deep-learning-based approaches for assertion generation. *Proceedings of the ACM on Software Engineering*, 2024. 1(FSE):1750–1771
- 490 Hossain S B and Dwyer M. Togll: Correct and strong test oracle generation with llms. *arXiv preprint arXiv:240503786*, 2024
- 491 Mali B, Maddala K, Reddy S, et al. Chiraag: Chatgpt informed rapid and automated assertion generation. *arXiv preprint arXiv:240200093*, 2024
- 492 Nashid N, Sintaha M, and Mesbah A. Retrieval-based prompt selection for code-related few-shot learning. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023. 2450–2462
- 493 Pulavarthi V, Nandal D, Dan S, et al. Assertionbench: A benchmark to evaluate large-language models for assertion generation. *arXiv preprint arXiv:240618627*, 2024
- 494 Tufano M, Drain D, Svyatkovskiy A, et al. Generating accurate assert statements for unit test cases using pretrained transformers. In *Proceedings of the 3rd ACM/IEEE International Conference on Automation of Software Test*. 2022. 54–64
- 495 Zhang Q, Sun W, Fang C, et al. Exploring automated assertion generation via large language models. *ACM Transactions on Software Engineering and Methodology*, 2025. 34(3):1–25
- 496 Wang H, Hu H, Chen C, et al. Chat-like asserts prediction with the support of large language model. *arXiv preprint arXiv:240721429*, 2024
- 497 Hossain S B, Filieri A, Dwyer M B, et al. Neural-based test oracle generation: A large-scale evaluation and lessons learned. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2023. 120–132
- 498 Liu Z, Liu K, Xia X, et al. Towards more realistic evaluation for neural test oracle generation. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2023. 589–600
- 499 Ahn S, Ahn S, Koo H, et al. Practical binary code similarity detection with bert-based transferable similarity learning. In *Proceedings of the 38th Annual Computer Security Applications Conference*. 2022. 361–374
- 500 Feng Y, Li H, Cao Y, et al. Crabs-former: Cross-architecture binary code similarity detection based on transformer. In *Proceedings of the 15th Asia-Pacific Symposium on Internetwork*. 2024. 11–20
- 501 Wang H, Qu W, Katz G, et al. Jtrans: Jump-aware transformer for binary code similarity detection. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2022. 1–13
- 502 Yu Z, Cao R, Tang Q, et al. Order matters: Semantic-aware neural networks for binary code similarity detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34. 2020. 1145–1152
- 503 Xue Z, Gao Z, Wang S, et al. Selfpico: Self-guided partial code execution with llms. *arXiv preprint arXiv:240716974*, 2024
- 504 Armengol-Estapé J, Woodruff J, Cummins C, et al. Slade: A portable small language model decompiler for optimized assembly. In *2024 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. IEEE, 2024. 67–80
- 505 Hu P, Liang R, and Chen K. Degpt: Optimizing decompiler output with llm. In *Proceedings 2024 Network and Distributed System Security Symposium*. 2024. 1–16
- 506 Jiang N, Wang C, Liu K, et al. Nova⁺: Generative language models for binaries. *arXiv preprint arXiv:231113721*, 2023
- 507 Shang X, Cheng S, Chen G, et al. How far have we gone in stripped binary code understanding using large language models. *arXiv preprint arXiv:240409836*, 2024
- 508 She X, Zhao Y, and Wang H. Wadec: Decompile webassembly using large language model. *arXiv preprint arXiv:240611346*, 2024
- 509 Wong W K, Wang H, Li Z, et al. Refining decompiled c code with large language models. *arXiv preprint arXiv:231006530*, 2023
- 510 Xu X, Zhang Z, Feng S, et al. Lmpa: Improving decompilation by synergy of large language model and program analysis. *arXiv preprint arXiv:230602546*, 2023
- 511 Tan H, Luo Q, Li J, et al. Llm4decompile: Decompiling binary code with large language models. *arXiv preprint arXiv:240305286*, 2024
- 512 Li T O, Zong W, Wang Y, et al. Finding failure-inducing test cases with chatgpt. *arXiv preprint arXiv:230411686*, 2023
- 513 Bin Murtaza S, McCoy A, Ren Z, et al. Llm fault localisation within evolutionary computation based automated program repair. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 2024. 1824–1829
- 514 Chandramohan M, Nguyen D Q, Krishnan P, et al. Supporting cross-language cross-project bug localization using pre-trained language models. *arXiv preprint arXiv:240702732*, 2024
- 515 Ciborowska A and Damevski K. Too few bug reports? exploring data augmentation for improved changeset-based bug localization. *arXiv preprint arXiv:230516430*, 2023
- 516 Ciborowska A and Damevski K. Fast changeset-based bug localization with bert. In *Proceedings of the 44th International Conference on Software Engineering (ICSE'22)*. 2022. 946–957
- 517 Du Y and 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*. 2023. 579–591
- 518 Ji S, Lee S, Lee C, et al. Impact of large language models of code on fault localization. *arXiv preprint arXiv:240809657*, 2024
- 519 Kang S, An G, and Yoo S. A quantitative and qualitative evaluation of llm-based explainable fault localization. *Proceedings of the ACM on Software Engineering*, 2024. 1(FSE):1424–1446
- 520 Mohsen A M, Hassan H, Wassif K, et al. Enhancing bug localization using phase-based approach. *IEEE Access*, 2023
- 521 Qin Y, Wang S, Lou Y, et al. Agentff: Scaling llm-based fault localization to project-level context. *arXiv preprint arXiv:240316362*, 2024
- 522 Shan S, Huo Y, Su Y, et al. Face it yourselves: An llm-based two-stage strategy to localize configuration errors via logs. *arXiv preprint arXiv:240400640*, 2024
- 523 Widyasari R, Ang J W, Nguyen T G, et al. Demystifying faulty code: Step-by-step reasoning for explainable fault localization. In *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2024. 568–579
- 524 Wu Y, Li Z, Zhang J M, et al. Large language models in fault localisation. *arXiv preprint arXiv:230815276*, 2023
- 525 Yan J, Huang J, Fang C, et al. Better debugging: Combining static analysis and llms for explainable crashing fault localization. *arXiv preprint arXiv:240812070*, 2024

- 526 Yang A Z, Le Goues C, Martins R, et al. Large language models for test-free fault localization. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE)*. IEEE, 2024. 165–176
- 527 Zhu Z, Wang Y, and Li Y. Trobo: A novel deep transfer model for enhancing cross-project bug localization. In *Knowledge Science, Engineering and Management: 14th International Conference, KSEM 2021, Tokyo, Japan, August 14–16, 2021, Proceedings, Part I*. Springer, 2021. 529–541
- 528 Wu Y, Li Z, Zhang J M, et al. Condefects: A complementary dataset to address the data leakage concern for llm-based fault localization and program repair. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*. 2024. 642–646
- 529 Charalambous Y, Tihanyi N, Jain R, et al. A new era in software security: Towards self-healing software via large language models and formal verification. arXiv preprint arXiv:230514752, 2023
- 530 Liu Y, Xue Y, Wu D, et al. Propertygpt: Llm-driven formal verification of smart contracts through retrieval-augmented property generation. arXiv preprint arXiv:240502580, 2024
- 531 Tihanyi N, Bisztray T, Jain R, et al. 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*. 2023. 33–43
- 532 Wen C, Cao J, Su J, et al. Enchanting program specification synthesis by large language models using static analysis and program verification. In *International Conference on Computer Aided Verification*. Springer, 2024. 302–328
- 533 Yang L, Yang J, Wei C, et al. Fuzzcoder: Byte-level fuzzing test via large language model. arXiv preprint arXiv:240901944, 2024
- 534 Dakham A, Even-Mendoza K, Langdon W B, et al. Searchgem5: Towards reliable gem5 with search based software testing and large language models. In *International Symposium on Search Based Software Engineering*. Springer, 2023. 160–166
- 535 Deng Y, Xia C S, Peng H, et al. 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*. 2023. 423–435
- 536 Wang D, Zhou G, Chen L, et al. Prophetfuzz: Fully automated prediction and fuzzing of high-risk option combinations with only documentation via large language model. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 2024. 735–749
- 537 Deng Y, Xia C S, Yang C, et al. Large language models are edge-case generators: Crafting unusual programs for fuzzing deep learning libraries. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE)*. IEEE, 2024. 841–853
- 538 Eom J, Jeong S, and Kwon T. Fuzzing javascript interpreters with coverage-guided reinforcement learning for llm-based mutation. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2024. 1656–1668
- 539 Hu J, Zhang Q, and Yin H. Augmenting greybox fuzzing with generative ai. arXiv preprint arXiv:230606782, 2023
- 540 Meng R, Mirchev M, Böhme M, et al. Large language model guided protocol fuzzing. In *Proceedings of the 31st Annual Network and Distributed System Security Symposium*. NDSS, 2024. 1–17
- 541 Asmita, Oliynyk Y, Scott M, et al. Fuzzing BusyBox: Leveraging LLM and crash reuse for embedded bug unearthing. In *33rd USENIX Security Symposium (USENIX Security 24)*. USENIX Association, Philadelphia, PA. ISBN 978-1-939133-44-1, 2024. 883–900
- 542 Ackerman J and Cybenko G. Large language models for fuzzing parsers (registered report). In *Proceedings of the 2nd International Fuzzing Workshop*. 2023. 31–38
- 543 Shou C, Liu J, Lu D, et al. Llm4fuzz: Guided fuzzing of smart contracts with large language models. arXiv preprint arXiv:240111108, 2024
- 544 Wang J, Yu L, and Luo X. Llmif: Augmented large language model for fuzzing iot devices. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2024. 196–196
- 545 Xia C S, Paltenghi M, Le Tian J, et al. Fuzz4all: Universal fuzzing with large language models. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE)*. IEEE, 2024. 1547–1559
- 546 Yang C, Zhao Z, and Zhang L. Kernelgpt: Enhanced kernel fuzzing via large language models. arXiv preprint arXiv:240100563, 2023
- 547 Yang C, Deng Y, Lu R, et al. White-box compiler fuzzing empowered by large language models. arXiv preprint arXiv:231015991, 2023
- 548 Zhang H, Rong Y, He Y, et al. Llamafuzz: Large language model enhanced greybox fuzzing. arXiv preprint arXiv:240607714, 2024
- 549 Zhang C, Zheng Y, Bai M, et al. 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*. 2024. 1223–1235
- 550 Liu Z, Li C, Chen C, et al. Vision-driven automated mobile gui testing via multimodal large language model. arXiv preprint arXiv:240703037, 2024
- 551 Liu Z, Chen C, Wang J, et al. Fill in the blank: Context-aware automated text input generation for mobile gui testing. arXiv preprint arXiv:221204732, 2022
- 552 Liu Z, Chen C, Wang J, et al. Make llm a testing expert: Bringing human-like interaction to mobile gui testing via functionality-aware decisions. arXiv preprint arXiv:231015780, 2023
- 553 Yoon J, Feldt R, and Yoo S. Autonomous large language model agents enabling intent-driven mobile gui testing. arXiv preprint arXiv:231108649, 2023
- 554 Yoon J, Feldt R, and Yoo S. Intent-driven mobile gui testing with autonomous large language model agents. In *2024 IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2024. 129–139
- 555 Ran D, Wang H, Song Z, et al. Guardian: A runtime framework for llm-based ui exploration. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2024. 958–970
- 556 Cheng B, Zhang C, Wang K, et al. Semantic-enhanced indirect call analysis with large language models. arXiv preprint arXiv:240804344, 2024
- 557 Pei K, Bieber D, Shi K, et al. Can large language models reason about program invariants? In *International Conference on Machine Learning*. PMLR, 2023. 27496–27520
- 558 Degiovanni R and Papadakis M. μ bert: Mutation testing using pre-trained language models. In *2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2022. 160–169
- 559 Garg A, Degiovanni R, Papadakis M, et al. On the coupling between vulnerabilities and llm-generated mutants: A study on vul4j dataset. In *2024 IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2024. 305–316
- 560 Hassan M, Ahmadi-Pour S, Qayyum K, et al. Llm-guided formal verification coupled with mutation testing. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2024. 1–2
- 561 Ibrahimzada A R, Chen Y, Rong R, et al. Automated bug generation in the era of large language models. arXiv preprint arXiv:231002407, 2023
- 562 Jain K, Alon U, Groce A, et al. Contextual predictive mutation testing. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2023. 250–261

- 563 Khanfir A, Degiovanni R, Papadakis M, et al. Efficient mutation testing via pre-trained language models. arXiv preprint arXiv:230103543, 2023
- 564 Li Z and Shin D. Mutation-based consistency testing for evaluating the code understanding capability of llms. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*. 2024. 150–159
- 565 Nong Y, Ou Y, Pradel M, et al. Vulgen: Realistic vulnerability generation via pattern mining and deep learning. In *Proceedings of the 45th International Conference on Software Engineering*. 2023. 2527–2539
- 566 Richter C and Wehrheim H. Learning realistic mutations: Bug creation for neural bug detectors. In *2022 IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2022. 162–173
- 567 Tian Z, Shu H, Wang D, et al. Large language models for equivalent mutant detection: How far are we? arXiv preprint arXiv:240801760, 2024
- 568 Tip F, Bell J, and Schäfer M. Llmorpheus: Mutation testing using large language models. arXiv preprint arXiv:240409952, 2024
- 569 Wang B, Chen M, Lin Y, et al. An exploratory study on using large language models for mutation testing. arXiv preprint arXiv:240609843, 2024
- 570 Gupta S, He P, Meister C, et al. Machine translation testing via pathological invariance. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2020. 863–875
- 571 He P, Meister C, and Su Z. Structure-invariant testing for machine translation. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 2020. 961–973
- 572 Liu Z, Feng Y, and Chen Z. Dialtest: automated testing for recurrent-neural-network-driven dialogue systems. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2021. 115–126
- 573 Liu Z, Feng Y, Yin Y, et al. Qatest: A uniform fuzzing framework for question answering systems. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 2022. 1–12
- 574 Sun Z, Zhang J M, Xiong Y, et al. Improving machine translation systems via isotopic replacement. In *Proceedings of the 44th International Conference on Software Engineering*. 2022. 1181–1192
- 575 Wang W, Huang J t, Wu W, et al. Mttm: Metamorphic testing for textual content moderation software. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023. 2387–2399
- 576 Yu B, Hu Y, Mang Q, et al. Automated testing and improvement of named entity recognition systems. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2023*. Association for Computing Machinery, New York, NY, USA, 2023. 883–894
- 577 Deng G, Liu Y, Mayoral-Vilches V, et al. Pentestgpt: An llm-empowered automatic penetration testing tool. arXiv preprint arXiv:230806782, 2023
- 578 Happe A and Cito J. Getting pwn'd by ai: Penetration testing with large language models. arXiv preprint arXiv:230800121, 2023
- 579 Pratama D, Suryanto N, Adiputra A A, et al. Cipher: Cybersecurity intelligent penetration-testing helper for ethical researcher. arXiv preprint arXiv:240811650, 2024
- 580 Wu L, Zhong X, Liu J, et al. Ptgroup: An automated penetration testing framework using llms and multiple prompt chains. In *International Conference on Intelligent Computing*. Springer, 2024. 220–232
- 581 Su J, Deng L, Wen C, et al. cfstra: Enhancing configurable program analysis through llm-driven strategy selection based on code features. In *International Symposium on Theoretical Aspects of Software Engineering*. Springer, 2024. 374–391
- 582 Zhang M, Tian Y, Xu Z, et al. Lpr: Large language models-aided program reduction. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2024. 261–273
- 583 Zhang L, Lu S, and Duan N. Selene: Pioneering automated proof in software verification. arXiv preprint arXiv:240107663, 2024
- 584 Vikram V, Lemieux C, and Padhye R. Can large language models write good property-based tests? arXiv preprint arXiv:230704346, 2023
- 585 Wang C, Liu J, Peng X, et al. Boosting static resource leak detection via llm-based resource-oriented intention inference. arXiv preprint arXiv:231104448, 2023
- 586 Lu Y, Tian Y, Bi Y, et al. Diavio: Llm-empowered diagnosis of safety violations in ads simulation testing. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2024. 376–388
- 587 Chapman P J, Rubio-González C, and Thakur A V. Interleaving static analysis and llm prompting. In *Proceedings of the 13th ACM SIGPLAN International Workshop on the State Of the Art in Program Analysis*. 2024. 9–17
- 588 Fang C, Miao N, Srivastav S, et al. Large language models for code analysis: Do {LLMs} really do their job? In *33rd USENIX Security Symposium (USENIX Security 24)*. 2024. 829–846
- 589 Hao Y, Chen W, Zhou Z, et al. E&v: Prompting large language models to perform static analysis by pseudo-code execution and verification. arXiv preprint arXiv:231208477, 2023
- 590 Li H, Hao Y, Zhai Y, et al. Assisting static analysis with large language models: A chatgpt experiment. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2023. 2107–2111
- 591 Li H, Hao Y, Zhai Y, et al. Enhancing static analysis for practical bug detection: An llm-integrated approach. *Proceedings of the ACM on Programming Languages*, 2024. 8(OOPSLA1):474–499
- 592 Mohajer M M, Aleithan R, Harzevili N S, et al. Skipanalyzer: An embodied agent for code analysis with large language models. arXiv preprint arXiv:231018532, 2023
- 593 Wen C, Cai Y, Zhang B, et al. Automatically inspecting thousands of static bug warnings with large language model: How far are we? *ACM Transactions on Knowledge Discovery from Data*, 2024. 18(7):1–34
- 594 Pujar S, Zheng Y, Buratti L, et al. Analyzing source code vulnerabilities in the d2a dataset with ml ensembles and c-bert. *Empirical Software Engineering*, 2024. 29(2):48
- 595 Liu P, Sun C, Zheng Y, et al. Harnessing the power of llm to support binary taint analysis. arXiv preprint arXiv:231008275, 2023
- 596 He Y, Huang J, Rong Y, et al. Unitsyn: A large-scale dataset capable of enhancing the prowess of large language models for program testing. arXiv preprint arXiv:240203396, 2024
- 597 Jalil S, Rafi S, LaToza T D, et al. Chatgpt and software testing education: Promises & perils. In *2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2023. 4130–4137
- 598 Arora C, Herda T, and Homm V. Generating test scenarios from nl requirements using retrieval-augmented llms: An industrial study. arXiv preprint arXiv:240412772, 2024
- 599 Azaria A, Azoulay R, and Reches S. Chatgpt is a remarkable tool—for experts. *Data Intelligence*, 2024. 6(1):240–296
- 600 Cui C, Li T, Wang J, et al. Large language models for mobile gui text input generation: An empirical study. arXiv preprint arXiv:240408948, 2024
- 601 Dakhel A M, Nikanjam A, Majdinasab V, et al. Effective test generation using pre-trained large language models and mutation testing. arXiv preprint arXiv:230816557, 2023
- 602 Deljouyi A, Koohestani R, Izadi M, et al. Leveraging large language models for enhancing the understandability of generated

- unit tests. arXiv preprint arXiv:240811710, 2024
- 603 Etemadi K, Mohammadi B, Su Z, et al. Mokav: Execution-driven differential testing with llms. arXiv preprint arXiv:240610375, 2024
- 604 Gu S, Fang C, Zhang Q, et al. Testart: Improving llm-based unit test via co-evolution of automated generation and repair iteration. arXiv preprint arXiv:240803095, 2024
- 605 Guilherme V and Vincenzi A. An initial investigation of chatgpt unit test generation capability. In *Proceedings of the 8th Brazilian Symposium on Systematic and Automated Software Testing*. 2023. 15–24
- 606 He Y, Wang J, Rong Y, et al. Exploring fuzzing as data augmentation for neural test generation. arXiv preprint arXiv:240608665, 2024
- 607 Karanjai R, Hussain A, Rabin M R I, et al. Harnessing the power of llms: Automating unit test generation for high-performance computing. arXiv preprint arXiv:240705202, 2024
- 608 Karmarkar H, Agrawal S, Chauhan A, et al. Navigating confidentiality in test automation: A case study in llm driven test data generation. In *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2024. 337–348
- 609 Kirinuki H and Tanno H. Chatgpt and human synergy in black-box testing: A comparative analysis. arXiv preprint arXiv:240113924, 2024
- 610 Lemieux C, Inala J P, Lahiri S K, et al. Codamosa: Escaping coverage plateaus in test generation with pre-trained large language models. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023. 919–931
- 611 Li M, Li D, Liu J, et al. Dllens: Testing deep learning libraries via llm-aided synthesis. arXiv preprint arXiv:240607944, 2024
- 612 Li K and Yuan Y. Large language models as test case generators: Performance evaluation and enhancement. arXiv preprint arXiv:240413340, 2024
- 613 Li T, Cui C, Ma L, et al. Leveraging large language models for automated web-form-test generation: An empirical study. arXiv preprint arXiv:240509965, 2024
- 614 Liu K, Liu Y, Chen Z, et al. Llm-powered test case generation for detecting tricky bugs. arXiv preprint arXiv:240410304, 2024
- 615 Lops A, Narducci F, Ragone A, et al. A system for automated unit test generation using large language models and assessment of generated test suites. arXiv preprint arXiv:240807846, 2024
- 616 Mündler N, Müller M N, He J, et al. Code agents are state of the art software testers. arXiv preprint arXiv:240612952, 2024
- 617 Nabeel M, Nimara D D, and Zanouda T. Test code generation for telecom software systems using two-stage generative model. arXiv preprint arXiv:240409249, 2024
- 618 Ni C, Wang X, Chen L, et al. Casmodatest: A cascaded and model-agnostic self-directed framework for unit test generation. arXiv preprint arXiv:240615743, 2024
- 619 Ouédraogo W C, Kaboré K, Tian H, et al. Large-scale, independent and comprehensive study of the power of llms for test case generation. arXiv preprint arXiv:240700225, 2024
- 620 Pizzorno J A and Berger E D. Coverup: Coverage-guided llm-based test generation. arXiv preprint arXiv:240316218, 2024
- 621 Plein L, Ouédraogo W C, Klein J, et al. Automatic generation of test cases based on bug reports: a feasibility study with large language models. In *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*. 2024. 360–361
- 622 Rao N, Jain K, Alon U, et al. Cat-lm training language models on aligned code and tests. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023. 409–420
- 623 Ryan G, Jain S, Shang M, et al. Code-aware prompting: A study of coverage guided test generation in regression setting using llm. arXiv preprint arXiv:240200097, 2024
- 624 Schäfer M, Nadi S, Eghbali A, et al. An empirical evaluation of using large language models for automated unit test generation. *IEEE Transactions on Software Engineering*, 2023
- 625 Shin J, Hashtroudi S, Hemmati H, et al. Domain adaptation for deep unit test case generation. arXiv e-prints, 2023:arXiv:2308.2308
- 626 Siddiq M L, Santos J, Tanvir R H, et al. Exploring the effectiveness of large language models in generating unit tests. arXiv preprint arXiv:230500418, 2023
- 627 Steenhoek B, Tufano M, Sundaresan N, et al. Reinforcement learning from automatic feedback for high-quality unit test generation. arXiv preprint arXiv:231002368, 2023
- 628 Tang Y, Liu Z, Zhou Z, et al. Chatgpt vs sbst: A comparative assessment of unit test suite generation. *IEEE Transactions on Software Engineering*, 2024
- 629 Tufano M, Drain D, Svyatkovskiy A, et al. Unit test case generation with transformers and focal context. arXiv preprint arXiv:200905617, 2020
- 630 Wang Z, Liu K, Li G, et al. Hits: High-coverage llm-based unit test generation via method slicing. arXiv preprint arXiv:240811324, 2024
- 631 Xiao D, Guo Y, Li Y, et al. Optimizing search-based unit test generation with large language models: An empirical study. In *Proceedings of the 15th Asia-Pacific Symposium on Internetworking*. 2024. 71–80
- 632 Chen Y, Hu Z, Zhi C, et al. Chatunitest: A framework for llm-based test generation. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*. 2024. 572–576
- 633 Xiong W, Guo Y, and Chen H. The program testing ability of large language models for code. arXiv preprint arXiv:231005727, 2023
- 634 Yang L, Yang C, Gao S, et al. An empirical study of unit test generation with large language models. arXiv preprint arXiv:240618181, 2024
- 635 Yang C, Chen J, Lin B, et al. Enhancing llm-based test generation for hard-to-cover branches via program analysis. arXiv preprint arXiv:240404966, 2024
- 636 Yuan Z, Lou Y, Liu M, et al. No more manual tests? evaluating and improving chatgpt for unit test generation. arXiv preprint arXiv:230504207, 2023
- 637 Yuan Z, Liu M, Ding S, et al. Evaluating and improving chatgpt for unit test generation. *Proceedings of the ACM on Software Engineering*, 2024. 1(FSE):1703–1726
- 638 Zhang K, Wang D, Xia J, et al. Algo: Synthesizing algorithmic programs with generated oracle verifiers. *Advances in Neural Information Processing Systems*, 2023. 36:54769–54784
- 639 Zhang Y, Song W, Ji Z, et al. How well does llm generate security tests? arXiv preprint arXiv:231000710, 2023
- 640 Zhou Z, Tang Y, Lin Y, et al. An llm-based readability measurement for unit tests' context-aware inputs. arXiv preprint arXiv:240721369, 2024
- 641 Alagarsamy S, Tantithamthavorn C, and Aleti A. A3test: Assertion-augmented automated test case generation. arXiv preprint arXiv:230210352, 2023
- 642 Xue Z, Li L, Tian S, et al. Llm4fin: Fully automating llm-powered test case generation for fintech software acceptance

- testing. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2024. 1643–1655
- 643 Zhang Z, Liu X, Lin Y, et al. Llm-based unit test generation via property retrieval. arXiv preprint arXiv:241013542, 2024
- 644 Zhang Q, Shang Y, Fang C, et al. Testbench: Evaluating class-level test case generation capability of large language models. arXiv preprint arXiv:240917561, 2024
- 645 Pan R, Ghaleb T A, and Briand L. Ltm: Scalable and black-box similarity-based test suite minimization based on language models. arXiv preprint arXiv:230401397, 2023
- 646 Liu X, Liu H, Yi X, et al. Llm-enhanced theorem proving with term explanation and tactic parameter repair. In *Proceedings of the 15th Asia-Pacific Symposium on Internetwork*. 2024. 21–30
- 647 Shi J, Yang Z, Xu B, et al. Compressing pre-trained models of code into 3 mb. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 2022. 1–12
- 648 Fu M, Tantithamthavorn C K, Nguyen V, et al. Chatgpt for vulnerability detection, classification, and repair: How far are we? In *2023 30th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2023. 632–636
- 649 Li J, Li Z, Zhang H, et al. Poison attack and poison detection on deep source code processing models. *ACM Transactions on Software Engineering and Methodology*, 2023
- 650 Yang Z, Shi J, He J, et al. Natural attack for pre-trained models of code. In *Proceedings of the 44th International Conference on Software Engineering*. 2022. 1482–1493
- 651 Ahmad B, Tan B, Karri R, et al. Flag: Finding line anomalies (in code) with generative ai. arXiv preprint arXiv:230612643, 2023
- 652 Akuthota V, Kasula R, Sumona S T, et al. Vulnerability detection and monitoring using llm. In *2023 IEEE 9th International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE)*. IEEE, 2023. 309–314
- 653 Alqarni M and Azim A. Low level source code vulnerability detection using advanced bert language model. In *Proceedings of the Canadian Conference on Artificial Intelligence*. 2022. 1–11
- 654 Ashiwal V, Finster S, and Dawoud A. Llm-based vulnerability sourcing from unstructured data. In *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2024. 634–641
- 655 Chan A, Kharkar A, Moghaddam R Z, et al. Transformer-based vulnerability detection in code at edittime: Zero-shot, few-shot, or fine-tuning? arXiv preprint arXiv:230601754, 2023
- 656 Chen Y, Ding Z, Alowain L, et al. 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*. 2023. 654–668
- 657 Chen Y, Gao C, Yang Z, et al. Bridge and hint: Extending pre-trained language models for long-range code. arXiv preprint arXiv:240511233, 2024
- 658 Cheng Y, Shar L K, Zhang T, et al. Llm-enhanced static analysis for precise identification of vulnerable oss versions. arXiv preprint arXiv:240807321, 2024
- 659 Chukkol A H A, Luo S, Sharif K, et al. Vulcatch: Enhancing binary vulnerability detection through codet5 decompilation and kan advanced feature extraction. arXiv preprint arXiv:240807181, 2024
- 660 Daneshvar S S, Nong Y, Yang X, et al. Exploring rag-based vulnerability augmentation with llms. arXiv preprint arXiv:240804125, 2024
- 661 Ding Y, Fu Y, Ibrahim O, et al. Vulnerability detection with code language models: How far are we? arXiv preprint arXiv:240318624, 2024
- 662 Do C X, Luu N T, and Nguyen P T L. Optimizing software vulnerability detection using roberta and machine learning. *Automated Software Engineering*, 2024. 31(2):40
- 663 Dozono K, Gasiba T E, and Stocco A. Large language models for secure code assessment: A multi-language empirical study. arXiv preprint arXiv:240806428, 2024
- 664 Du X, Wen M, Zhu J, et al. Generalization-enhanced code vulnerability detection via multi-task instruction fine-tuning. arXiv preprint arXiv:240603718, 2024
- 665 Du X, Zheng G, Wang K, et al. Vul-rag: Enhancing llm-based vulnerability detection via knowledge-level rag. arXiv preprint arXiv:240611147, 2024
- 666 Fu M and Tantithamthavorn C. Linevul: A transformer-based line-level vulnerability prediction. In *Proceedings of the 19th International Conference on Mining Software Repositories*. 2022. 608–620
- 667 Gao Z, Wang H, Zhou Y, et al. How far have we gone in vulnerability detection using large language models. arXiv preprint arXiv:231112420, 2023
- 668 Gomes L, da Silva Torres R, and Côrtes M L. Bert-and tf-idf-based feature extraction for long-lived bug prediction in floss: a comparative study. *Information and Software Technology*, 2023. 160:107217
- 669 Gonçalves J, Dias T, Maia E, et al. Scope: Evaluating llms for software vulnerability detection. arXiv preprint arXiv:240714372, 2024
- 670 Grishina A, Hort M, and 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*. 2023. 895–907
- 671 Hanif H and Maffei S. Vulberta: Simplified source code pre-training for vulnerability detection. In *2022 International joint conference on neural networks (IJCNN)*. IEEE, 2022. 1–8
- 672 Jiang Z, Sun W, Gu X, et al. Dfep: Data flow embedding for enhancing pre-trained model based vulnerability detection. In *Proceedings of the 15th Asia-Pacific Symposium on Internetwork*. 2024. 95–104
- 673 Ju B, Yang J, Yu T, et al. A study of using multimodal llms for non-crash functional bug detection in android apps. arXiv preprint arXiv:240719053, 2024
- 674 Kholoosi M M, Babar M A, and Croft R. A qualitative study on using chatgpt for software security: Perception vs. practicality. arXiv preprint arXiv:240800435, 2024
- 675 Koide T, Fukushi N, Nakano H, et al. Detecting phishing sites using chatgpt. arXiv preprint arXiv:230605816, 2023
- 676 Lee H, Sharma S, and Hu B. Bug in the code stack: Can llms find bugs in large python code stacks. arXiv preprint arXiv:240615325, 2024
- 677 Li Z, Dutta S, and Naik M. Llm-assisted static analysis for detecting security vulnerabilities. arXiv preprint arXiv:240517238, 2024
- 678 Liu Y, Gao L, Yang M, et al. Vuldetectbench: Evaluating the deep capability of vulnerability detection with large language models. arXiv preprint arXiv:240607595, 2024
- 679 Liu S, Cao D, Kim J, et al. {EaTVul}:{ChatGPT-based} evasion attack against software vulnerability detection. In *33rd USENIX Security Symposium (USENIX Security 24)*. 2024. 7357–7374
- 680 Lu G, Ju X, Chen X, et al. Grace: Empowering llm-based software vulnerability detection with graph structure and in-context learning. *Journal of Systems and Software*, 2024. 212:112031
- 681 Lucas K, Gheyri R, Soares E, et al. Evaluating large language models in detecting test smells. arXiv preprint arXiv:240719261, 2024

- 682 Mahyari A A. Harnessing the power of llms in source code vulnerability detection. arXiv preprint arXiv:240803489, 2024
- 683 Mao Z, Li J, Li M, et al. Multi-role consensus through llms discussions for vulnerability detection. arXiv preprint arXiv:240314274, 2024
- 684 Mao Q, Li Z, Hu X, et al. Towards effectively detecting and explaining vulnerabilities using large language models. arXiv preprint arXiv:240609701, 2024
- 685 Mathews N S, Brus Y, Aafer Y, et al. Llbzpeky: Leveraging large language models for vulnerability detection. arXiv preprint arXiv:240101269, 2024
- 686 Noever D. Can large language models find and fix vulnerable software? arXiv preprint arXiv:230810345, 2023
- 687 Nong Y, Aldeen M, Cheng L, et al. Chain-of-thought prompting of large language models for discovering and fixing software vulnerabilities. arXiv preprint arXiv:240217230, 2024
- 688 Pelofske E, Urias V, and Liebrock L M. Automated software vulnerability static code analysis using generative pre-trained transformer models. arXiv preprint arXiv:240800197, 2024
- 689 Quan V L A, Phat C T, Van Nguyen K, et al. Xgv-bert: Leveraging contextualized language model and graph neural network for efficient software vulnerability detection. arXiv preprint arXiv:230914677, 2023
- 690 Saad M, López J A H, Chen B, et al. Alpine: An adaptive language-agnostic pruning method for language models for code. arXiv preprint arXiv:240704147, 2024
- 691 Shestov A, Cheshkov A, Levichev R, et al. Finetuning large language models for vulnerability detection. arXiv preprint arXiv:240117010, 2024
- 692 Steenhoek B, Rahman M M, Roy M K, et al. A comprehensive study of the capabilities of large language models for vulnerability detection. arXiv preprint arXiv:240317218, 2024
- 693 Steenhoek B, Gao H, and Le W. Dataflow analysis-inspired deep learning for efficient vulnerability detection. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 2024. 1–13
- 694 Steenhoek B, Rahman M M, Jiles R, et al. An empirical study of deep learning models for vulnerability detection. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023. 2237–2248
- 695 Sun T, Allix K, Kim K, et al. Dexbert: Effective, task-agnostic and fine-grained representation learning of android bytecode. *IEEE Transactions on Software Engineering*, 2023
- 696 Sun Y, Wu D, Xue Y, et al. 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*. 2024. 1–13
- 697 Sun Y, Wu D, Xue Y, et al. Llm4vuln: A unified evaluation framework for decoupling and enhancing llms' vulnerability reasoning. arXiv preprint arXiv:240116185, 2024
- 698 Wu X, Cheriène N, Zhang C, et al. Using large language models to better detect and handle software vulnerabilities and cyber security threats. *ResearchGate preprint / manuscript*, 2024. DOI:10.21203/rs.3.rs-4387414/v1, posted May 21, 2024
- 699 Tamberg K and Bahsi H. Harnessing large language models for software vulnerability detection: A comprehensive benchmarking study. arXiv preprint arXiv:240515614, 2024
- 700 Tang W, Tang M, Ban M, et al. Csgvd: A deep learning approach combining sequence and graph embedding for source code vulnerability detection. *Journal of Systems and Software*, 2023. 199:111623
- 701 Tang X, Chen Z, Kim K, et al. Just-in-time security patch detection—llm at the rescue for data augmentation. arXiv preprint arXiv:231201241, 2023
- 702 Thapa C, Jang S I, Ahmed M E, et al. Transformer-based language models for software vulnerability detection. In *Proceedings of the 38th Annual Computer Security Applications Conference*. 2022. 481–496
- 703 Ullah S, Han M, Pujar S, et al. Can large language models identify and reason about security vulnerabilities? not yet. arXiv preprint arXiv:231212575, 2023
- 704 Wan S, Saxe J, Gomes C, et al. Bridging the gap: A study of ai-based vulnerability management between industry and academia. arXiv preprint arXiv:240502435, 2024
- 705 Wang Z, Li G, Li J, et al. Code structure-aware through line-level semantic learning for code vulnerability detection. arXiv preprint arXiv:240718877, 2024
- 706 Wang Z, Li G, Li J, et al. M2cvd: Multi-model collaboration for code vulnerability detection. arXiv preprint arXiv:240605940, 2024
- 707 Wen X C, Wang X, Chen Y, et al. Vuleval: Towards repository-level evaluation of software vulnerability detection. arXiv preprint arXiv:240415596, 2024
- 708 Wong M F, Guo S, Hang C N, et al. Natural language generation and understanding of big code for ai-assisted programming: A review. *Entropy*, 2023. 25(6):888
- 709 Wu H, Zhang Z, Wang S, et al. Peculiar: Smart contract vulnerability detection based on crucial data flow graph and pre-training techniques. In *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2021. 378–389
- 710 Yang Y, Zhou X, Mao R, et al. Dlap: A deep learning augmented large language model prompting framework for software vulnerability detection. arXiv preprint arXiv:240501202, 2024
- 711 Yang A Z, Tian H, Ye H, et al. Security vulnerability detection with multitask self-instructed fine-tuning of large language models. arXiv preprint arXiv:240605892, 2024
- 712 Yin X and Ni C. Multitask-based evaluation of open-source llm on software vulnerability. arXiv preprint arXiv:240402056, 2024
- 713 Yin X. Pros and cons! evaluating chatgpt on software vulnerability. arXiv preprint arXiv:240403994, 2024
- 714 Yu J, Liang P, Fu Y, et al. Security code review by llms: A deep dive into responses. arXiv preprint arXiv:240116310, 2024
- 715 Yuan B, Lu Y, Fang Y, et al. Enhancing deep learning-based vulnerability detection by building behavior graph model. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023. 2262–2274
- 716 Zhang C, Liu H, Zeng J, et al. Prompt-enhanced software vulnerability detection using chatgpt. arXiv preprint arXiv:230812697, 2023
- 717 Zhao Y, Gong L, Huang Z, et al. Coding-ptms: How to find optimal code pre-trained models for code embedding in vulnerability detection? arXiv preprint arXiv:240804863, 2024
- 718 Zhou X, Tran D M, Le-Cong T, et al. Comparison of static application security testing tools and large language models for repo-level vulnerability detection. arXiv preprint arXiv:240716235, 2024
- 719 Li Z, Ji J, Liang P, et al. An exploratory study on just-in-time multi-programming-language bug prediction. *Information and Software Technology*, 2024. 175:107524
- 720 Lu G, Ju X, Chen X, et al. Assessing the effectiveness of vulnerability detection via prompt tuning: An empirical study. In *2023 30th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2023. 415–424
- 721 Park C and Kim R Y C. Detecting common weakness enumeration through training the core building blocks of similar languages based on the codebert model. In *2023 30th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2023. 641–642
- 722 Deng X, Duan F, Xie R, et al. Improving long-tail vulnerability detection through data augmentation based on large language models. In *2024 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2024. 262–274

- 723 Guan H, Bai G, and Liu Y. Large language models can connect the dots: Exploring model optimization bugs with domain knowledge-aware prompts. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2024. 1579–1591
- 724 Sun J, Xing Z, Lu Q, et al. Silent vulnerable dependency alert prediction with vulnerability key aspect explanation. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023. 970–982
- 725 Oishwee S J, Stakhanova N, and Codabux Z. Large language model vs. stack overflow in addressing android permission related challenges. In *2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)*. IEEE, 2024. 373–383
- 726 Motger Q, Miaschi A, Dell’Orletta F, et al. T-frex: A transformer-based feature extraction method from mobile app reviews. In *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2024. 227–238
- 727 Wang Y, Wang J, Zhang H, et al. Where is your app frustrating users? In *Proceedings of the 44th International Conference on Software Engineering*. 2022. 2427–2439
- 728 Isotani H, Washizaki H, Fukazawa Y, et al. Duplicate bug report detection by using sentence embedding and fine-tuning. In *2021 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE, 2021. 535–544
- 729 Plein L and Bissyandé T F. Can llms demystify bug reports? arXiv preprint arXiv:231006310, 2023
- 730 Wu X, Li H, Yoshioka N, et al. Refining gpt-3 embeddings with a siamese structure for technical post duplicate detection. In *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2024. 114–125
- 731 Zhang T, Irsan I C, Thung F, et al. Cupid: Leveraging chatgpt for more accurate duplicate bug report detection. arXiv preprint arXiv:230810022, 2023
- 732 Helmezi R K, Cevik M, and Yildirim S. Few-shot learning for sentence pair classification and its applications in software engineering. arXiv preprint arXiv:230608058, 2023
- 733 Feng S and Chen C. Prompting is all your need: Automated android bug replay with large language models. arXiv preprint arXiv:230601987, 2023
- 734 Huang Y, Wang J, Liu Z, et al. Crashtester: Automatically reproducing mobile application crashes directly from stack trace. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 2024. 1–13
- 735 Kang S, Yoon J, Askarbekkyzy N, et al. Evaluating diverse large language models for automatic and general bug reproduction. arXiv preprint arXiv:231104532, 2023
- 736 Kang S, Yoon J, and 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)*. IEEE, 2023. 2312–2323
- 737 Dong H, Ren H, Shi J, et al. Neighborhood contrastive learning-based graph neural network for bug triaging. *Science of Computer Programming*, 2024. 235:103093
- 738 Dipongkor A K and Moran K. A comparative study of transformer-based neural text representation techniques on bug triaging. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023. 1012–1023
- 739 Lee J, Han K, and 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*. 2022. 1–11
- 740 Alam A I, Roy P R, Al-Omari F, et al. 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)*. IEEE, 2023. 1–13
- 741 Chochlov M, Ahmed G A, Patten J V, et al. Using a nearest-neighbour, bert-based approach for scalable clone detection. In *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2022. 582–591
- 742 Dou S, Shan J, Jia H, et al. Towards understanding the capability of large language models on code clone detection: a survey. arXiv preprint arXiv:230801191, 2023
- 743 Du Y, Ma T, Wu L, et al. Adaccd: Adaptive semantic contrasts discovery based cross lingual adaptation for code clone detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38. 2024. 17942–17950
- 744 Khajezade M, Wu J J, Fard F H, et al. Investigating the efficacy of large language models for code clone detection. In *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension*. 2024. 161–165
- 745 Moumoula M B, Kabore A K, Klein J, et al. Large language models for cross-language code clone detection. arXiv preprint arXiv:240804430, 2024
- 746 Sharma R, Chen F, Fard F, et al. An exploratory study on code attention in bert. In *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*. 2022. 437–448
- 747 Zhang Z and Saber T. Assessing the code clone detection capability of large language models. In *2024 4th International Conference on Code Quality (ICQ)*. IEEE, 2024. 75–83
- 748 Abid S, Cai X, and Jiang L. Interpreting codebert for semantic code clone detection. In *2023 30th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2023. 229–238
- 749 Tufano M, Chandel S, Agarwal A, et al. Predicting code coverage without execution. arXiv preprint arXiv:230713383, 2023
- 750 Weyssow M, Di Sipio C, Di Ruscio D, et al. Codell: A lifelong learning dataset to support the co-evolution of data and language models of code. In *2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)*. IEEE, 2024. 637–641
- 751 Zhang J, Nie P, Li J J, et al. Multilingual code co-evolution using large language models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2023. 695–707
- 752 White J, Hays S, Fu Q, et al. Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. In *Generative AI for Effective Software Development*. Springer, 2024. 71–108
- 753 Mohammed N, Lal A, Rastogi A, et al. Enabling memory safety of c programs using llms. arXiv preprint arXiv:240401096, 2024
- 754 Liu H, Wang Y, Wei Z, et al. 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*. 2023. 740–752
- 755 Pomian D, Bellur A, Dilhara M, et al. Next-generation refactoring: Combining llm insights and ide capabilities for extract method. In *2024 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2024. 275–287
- 756 Shirafuji A, Oda Y, Suzuki J, et al. Refactoring programs using large language models with few-shot examples. In *2023 30th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2023. 151–160
- 757 Zhang Z, Xing Z, Ren X, et al. Refactoring to pythonic idioms: A hybrid knowledge-driven approach leveraging large language models. *Proceedings of the ACM on Software Engineering*, 2024. 1(FSE):1107–1128
- 758 Ahmed T, Devanbu P, Treude C, et al. Can llms replace manual annotation of software engineering artifacts? arXiv preprint arXiv:240805534, 2024
- 759 Ben Sghaier O and Sahraoui H. Improving the learning of code review successive tasks with cross-task knowledge distillation. *Proceedings of the ACM on Software Engineering*, 2024. 1(FSE):1086–1106
- 760 Dong-Kyu L. A gpt-based code review system for programming language learning. arXiv preprint arXiv:240704722, 2024
- 761 Fan L, Liu J, Liu Z, et al. Exploring the capabilities of llms for code change related tasks. arXiv preprint arXiv:240702824,

- 2024
- 762 Ferreira I, Rafiq A, and Cheng J. Incivility detection in open source code review and issue discussions. *Journal of Systems and Software*, 2024. 209:111935
- 763 Ghadhab L, Jenhani I, Mkaouer M W, et al. Augmenting commit classification by using fine-grained source code changes and a pre-trained deep neural language model. *Information and Software Technology*, 2021. 135:106566
- 764 Guo Q, Cao J, Xie X, et al. Exploring the potential of chatgpt in automated code refinement: An empirical study. *arXiv preprint arXiv:230908221*, 2023
- 765 Kou B, Chen M, and Zhang T. Automated summarization of stack overflow posts. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023. 1853–1865
- 766 Koutchene C, Dainese N, Hellas A, et al. Evaluating language models for generating and judging programming feedback. *arXiv preprint arXiv:240704873*, 2024
- 767 Li L, Yang L, Jiang H, et al. 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*. 2022. 1009–1021
- 768 Li Z, Lu S, Guo D, et al. Automating code review activities by large-scale pre-training. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2022. 1035–1047
- 769 Lu J, Tang Z, and Liu Z. Improving code refinement for code review via input reconstruction and ensemble learning. In *2023 30th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2023. 161–170
- 770 Lu J, Yu L, Li X, et al. 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)*. IEEE, 2023. 647–658
- 771 McAleese N, Pokorny R M, Uribe J F C, et al. Llm critics help catch llm bugs. *arXiv preprint arXiv:240700215*, 2024
- 772 Pornprasit C and Tantithamthavorn C. Fine-tuning and prompt engineering for large language models-based code review automation. *Information and Software Technology*, 2024. 175:107523
- 773 Rasheed Z, Sami M A, Waseem M, et al. Ai-powered code review with llms: Early results. *arXiv preprint arXiv:240418496*, 2024
- 774 Sghaier O B and Sahraoui H. A multi-step learning approach to assist code review. In *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2023. 450–460
- 775 Tufano R, Dabić O, Mastropaolo A, et al. Code review automation: strengths and weaknesses of the state of the art. *IEEE Transactions on Software Engineering*, 2024
- 776 Tufano R, Masiero S, Mastropaolo A, et al. Using pre-trained models to boost code review automation. In *Proceedings of the 44th International Conference on Software Engineering*. 2022. 2291–2302
- 777 Vijayvergiya M, Salawa M, Budiselić I, et al. Ai-assisted assessment of coding practices in modern code review. *arXiv preprint arXiv:240513565*, 2024
- 778 Widyasari R, Zhang T, Bouraffa A, et al. Explaining explanation: An empirical study on explanation in code reviews. *arXiv preprint arXiv:231109020*, 2023
- 779 Yang C, Xu B, Khan J Y, et al. Aspect-based api review classification: How far can pre-trained transformer model go? In *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2022. 385–395
- 780 Yin Y, Zhao Y, Sun Y, et al. Automatic code review by learning the structure information of code graph. *Sensors*, 2023. 23(5):2551
- 781 Zhao Z, Xu Z, Zhu J, et al. The right prompts for the job: Repair code-review defects with large language model. *arXiv preprint arXiv:231217485*, 2023
- 782 Ma W, Yu Y, Ruan X, et al. Pre-trained model based feature envy detection. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. IEEE, 2023. 430–440
- 783 Jung T H. Commitbert: Commit message generation using pre-trained programming language model. In *Proceedings of the 1st Workshop on Natural Language Processing for Programming (NLP4Prog 2021)*. 2021. 26–33
- 784 Li J, Faragó D, Petrov C, et al. Only diff is not enough: Generating commit messages leveraging reasoning and action of large language model. *Proceedings of the ACM on Software Engineering*, 2024. 1(FSE):745–766
- 785 Lopes C V, Klotzman V I, Ma I, et al. Commit messages in the age of large language models. *arXiv preprint arXiv:240117622*, 2024
- 786 Xue P, Wu L, Yu Z, et al. Automated commit message generation with large language models: An empirical study and beyond. *arXiv preprint arXiv:240414824*, 2024
- 787 Zhang Y, Qiu Z, Stol K J, et al. Automatic commit message generation: A critical review and directions for future work. *IEEE Transactions on Software Engineering*, 2024
- 788 Lin B, Wang S, Liu Z, et al. Cct5: A code-change-oriented pre-trained model. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2023. 1509–1521
- 789 Cummins C, Seeker V, Grubisic D, et al. Large language models for compiler optimization. *arXiv preprint arXiv:230907062*, 2023
- 790 Cummins C, Seeker V, Grubisic D, et al. Meta large language model compiler: Foundation models of compiler optimization. *arXiv preprint arXiv:240702524*, 2024
- 791 Gao Z, Wang H, Wang Y, et al. Vic: Virtual compiler is all you need for assembly code search. *arXiv preprint arXiv:240806385*, 2024
- 792 Grubisic D, Seeker V, Synnaeve G, et al. Priority sampling of large language models for compilers. In *Proceedings of the 4th Workshop on Machine Learning and Systems*. 2024. 91–97
- 793 Romero Rosas M, Torres Sanchez M, and Eigenmann R. Should ai optimize your code? a comparative study of current large language models versus classical optimizing compilers. *arXiv e-prints*, 2024:arXiv-2406
- 794 Shyputa A, Madaan A, Zeng Y, et al. Learning performance-improving code edits. *arXiv preprint arXiv:230207867*, 2023
- 795 Tu H, Zhou Z, Jiang H, et al. Isolating compiler bugs by generating effective witness programs with large language models. *arXiv preprint arXiv:230700593*, 2023
- 796 Ye T, Ma T, Wu L, et al. Iterative or innovative? a problem-oriented perspective for code optimization. *arXiv preprint arXiv:240611935*, 2024
- 797 Tian R, Ye Y, Qin Y, et al. Debugbench: Evaluating debugging capability of large language models. *arXiv preprint arXiv:240104621*, 2024
- 798 Kang S, Chen B, Yoo S, et al. Explainable automated debugging via large language model-driven scientific debugging. *arXiv preprint arXiv:230402195*, 2023
- 799 Cai Y, Yadavally A, Mishra A, et al. Programming assistant for exception handling with codebert. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 2024. 1–13
- 800 Fatima S, Ghaleb T A, and Briand L. Flakify: A black-box, language model-based predictor for flaky tests. *IEEE Transactions on Software Engineering*, 2022. 49(4):1912–1927

- 801 Goel D, Husain F, Singh A, et al. X-lifecycle learning for cloud incident management using llms. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*. 2024. 417–428
- 802 Ahmed T, Ghosh S, Bansal C, et al. Recommending root-cause and mitigation steps for cloud incidents using large language models. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023. 1737–1749
- 803 Jiang Y, Zhang C, He S, et al. Xpert: Empowering incident management with query recommendations via large language models. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 2024. 1–13
- 804 Roy D, Zhang X, Bhavne R, et al. Exploring llm-based agents for root cause analysis. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*. 2024. 208–219
- 805 Chen Y, Xie H, Ma M, et al. Automatic root cause analysis via large language models for cloud incidents. In *Proceedings of the Nineteenth European Conference on Computer Systems*. 2024. 674–688
- 806 Colavito G, Lanubile F, Novielli N, et al. Impact of data quality for automatic issue classification using pre-trained language models. *Journal of Systems and Software*, 2024. 210:111838
- 807 Colavito G, Lanubile F, Novielli N, et al. Leveraging gpt-like llms to automate issue labeling. In *2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)*. IEEE, 2024. 469–480
- 808 Liu Y, Zhang H, Li Z, et al. Optimizing the utilization of large language models via schedule optimization: An exploratory study. In *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 2024. 84–95
- 809 Liu Y, Zhang H, Li Z, et al. Cpls: Optimizing the assignment of llm queries. In *2024 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2024. 151–162
- 810 Huang S, Liu Y, Qi J, et al. Gloss: Guiding large language models to answer questions from system logs. In *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2024. 91–101
- 811 Huang J, Jiang Z, Chen Z, et al. Ulog: Unsupervised log parsing with large language models through log contrastive units. arXiv preprint arXiv:240607174, 2024
- 812 Jiang Z, Liu J, Chen Z, et al. Lilac: Log parsing using llms with adaptive parsing cache. arXiv preprint arXiv:231001796, 2023
- 813 Le V H and Zhang H. Log parsing with prompt-based few-shot learning. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023. 2438–2449
- 814 Le V H and Zhang H. Log parsing: How far can chatgpt go? In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE Computer Society, 2023. 1699–1704
- 815 Li Y, Huo Y, Jiang Z, et al. Exploring the effectiveness of llms in automated logging generation: An empirical study. arXiv preprint arXiv:230705950, 2023
- 816 Liu Y, Tao S, Meng W, et al. Interpretable online log analysis using large language models with prompt strategies. In *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension*. 2024. 35–46
- 817 Liu Y, Tao S, Meng W, et al. Logprompt: Prompt engineering towards zero-shot and interpretable log analysis. In *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*. 2024. 364–365
- 818 Ma L, Yang W, Xu B, et al. Knowlog: Knowledge enhanced pre-trained language model for log understanding. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 2024. 1–13
- 819 Ma Z, Chen A R, Kim D J, et al. Llmpraser: An exploratory study on using large language models for log parsing. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 2024. 1209–1221
- 820 Mastropaolo A, Pascarella L, and Bavota G. Using deep learning to generate complete log statements. In *Proceedings of the 44th International Conference on Software Engineering*. 2022. 2279–2290
- 821 Mastropaolo A, Ferrari V, Pascarella L, et al. Log statements generation via deep learning: Widening the support provided to developers. *Journal of Systems and Software*, 2024. 210:111947
- 822 Mehrabi M, Hamou-Lhadj A, and Moosavi H. The effectiveness of compact fine-tuned llms in log parsing. In *2024 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2024. 438–448
- 823 Mudgal P and Wouhaybi R. An assessment of chatgpt on log data. In *International Conference on AI-generated Content*. Springer, 2023. 148–169
- 824 Tao S, Meng W, Cheng Y, et al. Logstamp: Automatic online log parsing based on sequence labelling. *ACM SIGMETRICS Performance Evaluation Review*, 2022. 49(4):93–98
- 825 Wu Y, Yu S, and Li Y. Log parsing with self-generated in-context learning and self-correction. arXiv preprint arXiv:240603376, 2024
- 826 Xiao Y, Le V H, and Zhang H. Stronger, faster, and cheaper log parsing with llms. arXiv preprint arXiv:240606156, 2024
- 827 Xu J, Cui Z, Zhao Y, et al. Unilog: Automatic logging via llm and in-context learning. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 2024. 1–12
- 828 Yu S, Wu Y, Li Z, et al. 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*. 2023. 425–437
- 829 Astekin M, Hort M, and Moonen L. A comparative study on large language models for log parsing. In *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 2024. 234–244
- 830 Chai X, Zhang H, Zhang J, et al. Log sequence anomaly detection based on template and parameter parsing via bert. *IEEE Transactions on Dependable and Secure Computing*, 2024
- 831 Guo H, Yuan S, and Wu X. Logbert: Log anomaly detection via bert. In *2021 international joint conference on neural networks (IJCNN)*. IEEE, 2021. 1–8
- 832 Hadadi F, Xu Q, Bianculli D, et al. Anomaly detection on unstable logs with gpt models. arXiv preprint arXiv:240607467, 2024
- 833 He S, Lei Y, Zhang Y, et al. Parameter-efficient log anomaly detection based on pre-training model and lora. In *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2023. 207–217
- 834 Huang S, Liu Y, Fung C, et al. Hitanomaly: Hierarchical transformers for anomaly detection in system log. *IEEE transactions on network and service management*, 2020. 17(4):2064–2076
- 835 Lee Y, Kim J, and Kang P. Lanobert: System log anomaly detection based on bert masked language model. *Applied Soft Computing*, 2023. 146:110689
- 836 Li X, Chen P, Jing L, et al. Swisslog: Robust anomaly detection and localization for interleaved unstructured logs. *IEEE Transactions on Dependable and Secure Computing*, 2022. 20(4):2762–2780
- 837 Liu S, Deng L, Xu H, et al. Logbd: A log anomaly detection method based on pretrained models and domain adaptation. *Applied Sciences*, 2023. 13(13):7739
- 838 Ma X, Zou H, Keung J, et al. On the influence of data resampling for deep learning-based log anomaly detection: Insights and recommendations. arXiv preprint arXiv:240503489, 2024
- 839 Qi J, Luan Z, Huang S, et al. Logencoder: Log-based contrastive representation learning for anomaly detection. *IEEE Transactions on Network and Service Management*, 2023. 20(2):1378–1391
- 840 Qi J, Huang S, Luan Z, et al. Loggpt: Exploring chatgpt for log-based anomaly detection. In *2023 IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability*

- in *Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. IEEE, 2023. 273–280
- 841 Wang W, Lu S, Luo J, et al. Deepuserlog: Deep anomaly detection on user log using semantic analysis and key-value data. In *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2023. 172–182
- 842 Xiao R, Chen H, Lu J, et al. Allinfofog: Robust diverse anomalies detection based on all log features. *IEEE Transactions on Network and Service Management*, 2022. 20(3):2529–2543
- 843 Yu Z, Wen M, Guo X, et al. Maltracker: A fine-grained npm malware tracker copiloted by llm-enhanced dataset. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2024. 1759–1771
- 844 Liu Z, Chen C, Wang J, et al. 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*. 2024. 1–12
- 845 Jin P, Zhang S, Ma M, et al. 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*. 2023. 1657–1668
- 846 Molina F, Copia J M, and Gorla A. Improving patch correctness analysis via random testing and large language models. In *2024 IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2024. 317–328
- 847 Tian H, Liu K, Kaboré A K, et al. Evaluating representation learning of code changes for predicting patch correctness in program repair. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 2020. 981–992
- 848 Tian H, Tang X, Habib A, et al. Is this change the answer to that problem? correlating descriptions of bug and code changes for evaluating patch correctness. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 2022. 1–13
- 849 Tian H, Liu K, Li Y, et al. The best of both worlds: Combining learned embeddings with engineered features for accurate prediction of correct patches. *ACM Transactions on Software Engineering and Methodology*, 2023. 32(4):1–34
- 850 Zhang Q, Fang C, Sun W, et al. Appt: Boosting automated patch correctness prediction via fine-tuning pre-trained models. *IEEE Transactions on Software Engineering*, 2024. 50(03):474–494
- 851 Zhou X, Xu B, Kim K, et al. Patchzero: Zero-shot automatic patch correctness assessment. arXiv preprint arXiv:230300202, 2023
- 852 Morales G, Pragyan K, Jahan S, et al. A large language model approach to code and privacy policy alignment. In *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2024. 79–90
- 853 Zhang Q, Zhang T, Zhai J, et al. A critical review of large language model on software engineering: An example from chatgpt and automated program repair. arXiv preprint arXiv:231008879, 2023
- 854 Hu H, Shang Y, Xu G, et al. Can gpt-o1 kill all bugs? an evaluation of gpt-family llms on quixbugs. In *2025 IEEE/ACM International Workshop on Automated Program Repair (APR)*. IEEE, 2025. 11–18
- 855 Kabir M M A, Hassan S A, Wang X, et al. An empirical study of chatgpt-3.5 on question answering and code maintenance. arXiv preprint arXiv:231002104, 2023
- 856 Yuan W, Zhang Q, He T, et al. Circle: Continual repair across programming languages. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2022. 678–690
- 857 Berabi B, He J, Raychev V, et al. Tfix: Learning to fix coding errors with a text-to-text transformer. In *International Conference on Machine Learning*. PMLR, 2021. 780–791
- 858 Bouzenia I, Devanbu P, and Pradel M. Repairagent: An autonomous, llm-based agent for program repair. arXiv preprint arXiv:240317134, 2024
- 859 Cao J, Li M, Wen M, et al. A study on prompt design, advantages and limitations of chatgpt for deep learning program repair. arXiv preprint arXiv:230408191, 2023
- 860 Charalambous Y, Manino E, and Cordeiro L C. Automated repair of ai code with large language models and formal verification. arXiv preprint arXiv:240508848, 2024
- 861 Chow Y W, Di Grazia L, and Pradel M. Pytyr: Repairing static type errors in python. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 2024. 1–13
- 862 Dehghan M, Wu J J, Fard F H, et al. Mergerepair: An exploratory study on merging task-specific adapters in code llms for automated program repair. arXiv preprint arXiv:240809568, 2024
- 863 Deligiannis P, Lal A, Mehrotra N, et al. Fixing rust compilation errors using llms. arXiv preprint arXiv:230805177, 2023
- 864 Drain D, Wu C, Svyatkovskiy A, et al. Generating bug-fixes using pretrained transformers. In *Proceedings of the 5th ACM SIGPLAN International Symposium on Machine Programming*. 2021. 1–8
- 865 Du X, Liu M, Li J, et al. Resolving crash bugs via large language models: An empirical study. arXiv preprint arXiv:231210448, 2023
- 866 Fan Z, Gao X, Roychoudhury A, et al. Automated repair of programs from large language models. arXiv preprint arXiv:220510583, 2022
- 867 First E, Rabe M N, Ringer T, et al. 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*. 2023. 1229–1241
- 868 Gharibi R, Sadreddini M H, and Fakhrahmad S M. T5apr: Empowering automated program repair across languages through checkpoint ensemble. *Journal of Systems and Software*, 2024. 214:112083
- 869 Henkel J, Silva D, Teixeira L, et al. Shipwright: A human-in-the-loop system for dockerfile repair. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021. 1148–1160
- 870 Hidvégi D, Etemadi K, Bobadilla S, et al. Cigar: Cost-efficient program repair with llms. arXiv preprint arXiv:240206598, 2024
- 871 Hossain S B, Jiang N, Zhou Q, et al. A deep dive into large language models for automated bug localization and repair. arXiv preprint arXiv:240411595, 2024
- 872 Huang Q, Zhu J, Xing Z, et al. A chain of ai-based solutions for resolving fqns and fixing syntax errors in partial code. arXiv preprint arXiv:230611981, 2023
- 873 Huang K, Meng X, Zhang J, et al. An empirical study on fine-tuning large language models of code for automated program repair. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023. 1162–1174
- 874 Islam N T and Najafirad P. Code security vulnerability repair using reinforcement learning with large language models. arXiv preprint arXiv:240107031, 2024
- 875 Jiang N, Lutellier T, and Tan L. Cure: Code-aware neural machine translation for automatic program repair. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021. 1161–1173
- 876 Jiang N, Liu K, Lutellier T, et al. Impact of code language models on automated program repair. arXiv preprint arXiv:230205020, 2023
- 877 Jin M, Shahriar S, Tufano M, et al. Inferfix: End-to-end program repair with llms. arXiv preprint arXiv:230307263, 2023
- 878 Joshi H, Sanchez J C, Gulwani S, et al. Repair is nearly generation: Multilingual program repair with llms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37. 2023. 5131–5140
- 879 Kim M, Kim Y, Jeong H, et al. An empirical study of deep transfer learning-based program repair for kotlin projects. In

- Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2022. 1441–1452
- 880 Lajkó M, Csuvik V, and Vidács L. Towards javascript program repair with generative pre-trained transformer (gpt-2). In *Proceedings of the Third International Workshop on Automated Program Repair*. 2022. 61–68
 - 881 Le-Cong T, Luong D M, Le X B D, et al. Invalidator: Automated patch correctness assessment via semantic and syntactic reasoning. *IEEE Transactions on Software Engineering*, 2023. 49(6):3411–3429
 - 882 Lee C, Xia C S, Huang J t, et al. A unified debugging approach via llm-based multi-agent synergy. arXiv preprint arXiv:240417153, 2024
 - 883 Li F, Jiang J, Sun J, et al. Hybrid automated program repair by combining large language models and program analysis. arXiv preprint arXiv:240600992, 2024
 - 884 Liu Y, Tantithamthavorn C, Liu Y, et al. On the reliability and explainability of language models for program generation. *ACM Transactions on Software Engineering and Methodology*, 2024. 33(5):1–26
 - 885 Li Y, Wang S, and Nguyen T N. Dear: A novel deep learning-based approach for automated program repair. In *Proceedings of the 44th International Conference on Software Engineering*. 2022. 511–523
 - 886 Paul R, Hossain M M, Siddiq M L, et al. Enhancing automated program repair through fine-tuning and prompt engineering. arXiv preprint arXiv:230407840, 2023
 - 887 Peng Y, Gao S, Gao C, et al. Domain knowledge matters: Improving prompts with fix templates for repairing python type errors. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE)*. IEEE, 2024. 12–24
 - 888 Ruiz F V, Grishina A, Hort M, et al. A novel approach for automatic program repair using round-trip translation with large language models. arXiv preprint arXiv:240107994, 2024
 - 889 Silva A, Fang S, and Monperrus M. Repairllama: Efficient representations and fine-tuned adapters for program repair. arXiv preprint arXiv:231215698, 2023
 - 890 Sobania D, Briesch M, Hanna C, et al. An analysis of the automatic bug fixing performance of chatgpt. arXiv preprint arXiv:230108653, 2023
 - 891 Sun Z, Zhu H, Xu B, et al. Llm as runtime error handler: A promising pathway to adaptive self-healing of software systems. arXiv preprint arXiv:240801055, 2024
 - 892 Wadhwa N, Pradhan J, Sonwane A, et al. Frustrated with code quality issues? llms can help! arXiv preprint arXiv:230912938, 2023
 - 893 Wadhwa N, Pradhan J, Sonwane A, et al. Core: Resolving code quality issues using llms. *Proceedings of the ACM on Software Engineering*, 2024. 1(FSE):789–811
 - 894 Wang Y, Guo T, Huang Z, et al. Revisiting evolutionary program repair via code language model. arXiv preprint arXiv:240810486, 2024
 - 895 Wang W, Wang Y, Joty S, et al. Rap-gen: Retrieval-augmented patch generation with codet5 for automatic program repair. arXiv preprint arXiv:230906057, 2023
 - 896 Wei Y, Xia C S, and Zhang L. Copiloting the copilots: Fusing large language models with completion engines for automated program repair. arXiv preprint arXiv:230900608, 2023
 - 897 White J, Fu Q, Hays S, et al. A prompt pattern catalog to enhance prompt engineering with chatgpt. arXiv preprint arXiv:230211382, 2023
 - 898 Widjojo P and Treude C. Addressing compiler errors: Stack overflow or large language models? arXiv preprint arXiv:230710793, 2023
 - 899 Xia C S, Wei Y, and Zhang L. Practical program repair in the era of large pre-trained language models. arXiv preprint arXiv:221014179, 2022
 - 900 Xia C S and Zhang L. Conversational automated program repair. arXiv preprint arXiv:230113246, 2023
 - 901 Xiang J, Xu X, Kong F, et al. How far can we go with practical function-level program repair? arXiv preprint arXiv:240412833, 2024
 - 902 Xia C S and Zhang L. Less training, more repairing please: revisiting automated program repair via zero-shot learning. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2022. 959–971
 - 903 Xia C S, Wei Y, and Zhang L. Automated program repair in the era of large pre-trained language models. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023. 1482–1494
 - 904 Xia C S and Zhang L. Keep the conversation going: Fixing 162 out of 337 bugs for \$0.42 each using chatgpt. arXiv preprint arXiv:230400385, 2023
 - 905 Xia C S, Ding Y, and Zhang L. Revisiting the plastic surgery hypothesis via large language models. arXiv preprint arXiv:230310494, 2023
 - 906 Xin Q, Wu H, Tang J, et al. Detecting, creating, repairing, and understanding indivisible multi-hunk bugs. *Proceedings of the ACM on Software Engineering*, 2024. 1(FSE):2747–2770
 - 907 Xin Q, Wu H, Reiss S P, et al. Towards practical and useful automated program repair for debugging. arXiv preprint arXiv:240708958, 2024
 - 908 Xu Z, Lin Y, Li Q, et al. Guiding chatgpt to fix web ui tests via explanation-consistency checking. arXiv preprint arXiv:231205778, 2023
 - 909 Xu J, Fu Y, Tan S H, et al. Aligning llms for fl-free program repair. arXiv preprint arXiv:240408877, 2024
 - 910 Yang B, Tian H, Pian W, et al. Cref: An llm-based conversational software repair framework for programming tutors. arXiv preprint arXiv:240613972, 2024
 - 911 Yang B, Tian H, Ren J, et al. Multi-objective fine-tuning for enhanced program repair with llms. arXiv preprint arXiv:240412636, 2024
 - 912 Yang A Z, Kolak S, Hellendoorn V J, et al. Revisiting unnaturalness for automated program repair in the era of large language models. arXiv preprint arXiv:240415236, 2024
 - 913 Yin X, Ni C, Wang S, et al. Thinkrepair: Self-directed automated program repair. arXiv preprint arXiv:240720898, 2024
 - 914 Zhang J, Mytkowicz T, Kaufman M, et al. 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*. 2022. 77–88
 - 915 Zhang Y, Li G, Jin Z, et al. Neural program repair with program dependence analysis and effective filter mechanism. arXiv preprint arXiv:230509315, 2023
 - 916 Zhang Y, Jin Z, Xing Y, et al. Steam: simulating the interactive behavior of programmers for automatic bug fixing. arXiv preprint arXiv:230814460, 2023
 - 917 Zhang J, Cambrono J P, Gulwani S, et al. Pydex: Repairing bugs in introductory python assignments using llms. *Proceedings of the ACM on Programming Languages*, 2024. 8(OOPSLA1):1100–1124
 - 918 Zhang Q, Fang C, Zhang T, et al. Gamma: Revisiting template-based automated program repair via mask prediction. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023. 535–547
 - 919 Zhao J, Yang D, Zhang L, et al. Enhancing llm-based automated program repair with design rationales. arXiv preprint arXiv:240812056, 2024

- 920 Zhao Y, Huang Z, Ma Y, et al. Repair: Automated program repair with process-based feedback. arXiv preprint arXiv:240811296, 2024
- 921 Xia C S and Zhang L. Automated program repair via conversation: Fixing 162 out of 337 bugs for \$0.42 each using chatgpt. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2024. 819–831
- 922 Deligiannis P, Lal A, Mehrotra N, et al. Rustassistant: Using llms to fix compilation errors in rust code. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 2024. 267–279
- 923 Acharya J and Ginde G. Graph neural network vs. large language model: A comparative analysis for bug report priority and severity prediction. In *Proceedings of the 20th International Conference on Predictive Models and Data Analytics in Software Engineering*. 2024. 2–11
- 924 Ali A, Xia Y, Umer Q, et al. Bert based severity prediction of bug reports for the maintenance of mobile applications. *Journal of Systems and Software*, 2024. 208:111898
- 925 Mashhadi E, Ahmadvand H, and Hemmati H. Method-level bug severity prediction using source code metrics and llms. In *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2023. 635–646
- 926 Biswas E, Karabulut M E, Pollock L, et al. Achieving reliable sentiment analysis in the software engineering domain using bert. In *2020 IEEE International conference on software maintenance and evolution (ICSME)*. IEEE, 2020. 162–173
- 927 Shafikuzzaman M, Islam M R, Rolli A C, et al. An empirical evaluation of the zero-shot, few-shot, and traditional fine-tuning based pretrained language models for sentiment analysis in software engineering. *IEEE Access*, 2024
- 928 Zhang T, Xu B, Thung F, et al. Sentiment analysis for software engineering: How far can pre-trained transformer models go? In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2020. 70–80
- 929 Zhang T, Irsan I C, Thung F, et al. Revisiting sentiment analysis for software engineering in the era of large language models. arXiv preprint arXiv:231011113, 2023
- 930 He J, Xu B, Yang Z, et al. Ptm4tag: sharpening tag recommendation of stack overflow posts with pre-trained models. In *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*. 2022. 1–11
- 931 Mastropaolo A, Di Penta M, and 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)*. IEEE, 2023. 585–597
- 932 Hu X, Liu Z, Xia X, et al. Identify and update test cases when production code changes: A transformer-based approach. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 2023. 1111–1122. doi: 10.1109/ASE56229.2023.00165
- 933 Liu J, Yan J, Xie Y, et al. Augmenting llms to repair obsolete test cases with static collector and neural reranker. arXiv preprint arXiv:240703625, 2024
- 934 Yaraghi A S, Holden D, Kahani N, et al. Automated test case repair using language models. arXiv preprint arXiv:240106765, 2024
- 935 Zhu J, Xiao G, Zheng Z, et al. Enhancing traceability link recovery with unlabeled data. In *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2022. 446–457
- 936 Fu M, Nguyen V, Tantithamthavorn C, et al. Vision transformer inspired automated vulnerability repair. *ACM Transactions on Software Engineering and Methodology*, 2024. 33(3):1–29
- 937 Fu M, Tantithamthavorn C, Le T, et al. Vulrepair: a t5-based automated software vulnerability repair. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2022. 935–947
- 938 Islam N T, Khoury J, Seong A, et al. Llm-powered code vulnerability repair with reinforcement learning and semantic reward. arXiv preprint arXiv:240103374, 2024
- 939 Kulsum U, Zhu H, Xu B, et al. A case study of llm for automated vulnerability repair: Assessing impact of reasoning and patch validation feedback. arXiv preprint arXiv:240515690, 2024
- 940 Pearce H, Tan B, Ahmad B, et al. Examining zero-shot vulnerability repair with large language models. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023. 2339–2356
- 941 Ságodi Z, Antal G, Bogenfürst B, et al. Reality check: Assessing gpt-4 in fixing real-world software vulnerabilities. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*. 2024. 252–261
- 942 Tol M C and Sunar B. Zeroleak: Using llms for scalable and cost effective side-channel patching. arXiv preprint arXiv:230813062, 2023
- 943 Wang R, Li Z, Wang C, et al. Navrepair: Node-type aware c/c++ code vulnerability repair. arXiv preprint arXiv:240504994, 2024
- 944 Wu Y, Jiang N, Pham H V, et al. 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. Association for Computing Machinery, 2023. 1282–1294
- 945 Wu F, Zhang Q, Bajaj A P, et al. Exploring the limits of chatgpt in software security applications. arXiv preprint arXiv:231205275, 2023
- 946 Zhang T, Irsan I C, Thung F, et al. Evaluating pre-trained language models for repairing api misuses. arXiv preprint arXiv:231016390, 2023
- 947 Zhang Q, Fang C, Yu B, et al. Pre-trained model-based automated software vulnerability repair: How far are we? *IEEE Transactions on Dependable and Secure Computing*, 2023. 21(4):2507–2525
- 948 Zhou X, Kim K, Xu B, et al. Out of sight, out of mind: Better automatic vulnerability repair by broadening input ranges and sources. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 2024. 1–13
- 949 Serafini R, Otto C, Horstmann S A, et al. Chatgpt-resistant screening instrument for identifying non-programmers. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 2024. 1–13
- 950 Imran M M, Chatterjee P, and Damevski K. Uncovering the causes of emotions in software developer communication using zero-shot llms. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 2024. 1–13
- 951 Alhamed M and 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)*. IEEE, 2022. 129–138
- 952 Li Y, Ren Z, Wang Z, et al. Fine-se: Integrating semantic features and expert features for software effort estimation. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 2024. 1–12
- 953 Abedu S, Abdellatif A, and Shihab E. Llm-based chatbots for mining software repositories: Challenges and opportunities. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*. 2024. 201–210
- 954 Felizardo K R, Lima M S, Deizepe A, et al. Chatgpt application in systematic literature reviews in software engineering: an evaluation of its accuracy to support the selection activity. In *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 2024. 25–36
- 955 Kannan J. Can llms configure software tools. arXiv preprint arXiv:231206121, 2023