Unifying the Perspectives of NLP and Software Engineering: A Survey on Language Models for Code

Anonymous authors
Paper under double-blind review

Abstract

In this work we systematically review the recent advancements in software engineering with language models, covering 70+ models, 40+ evaluation tasks, 180+ datasets, and 900 related works. Unlike previous works, we integrate software engineering (SE) with natural language processing (NLP) by discussing the perspectives of both sides: SE applies language models for development automation, while NLP adopts SE tasks for language model evaluation. We break down code processing models into general language models represented by the GPT family and specialized models that are specifically pretrained on code, often with tailored objectives. We discuss the relations and differences between these models, and highlight the historical transition of code modeling from statistical models and RNNs to pretrained Transformers and LLMs, which is exactly the same course that had been taken by NLP. We also go beyond programming and review LLMs' application in other software engineering activities including requirement engineering, testing, deployment, and operations in an endeavor to provide a global view of NLP in SE, and identify key challenges and potential future directions in this domain.

1 Introduction

Language modeling has advanced remarkably in recent years with the advent of pretrained Transformers (Vaswani et al., 2017) such as BERT (Devlin et al., 2019) and GPT (Radford et al., 2018). As large language models (LLMs) scaled to hundreds of billions of parameters and started to display early signs of artificial general intelligence (Brown et al., 2020; Chowdhery et al., 2023; OpenAI, 2023), their applications have also transcended text processing. Pioneered by Codex (Chen et al., 2021b), LLMs have achieved impressive results in code processing, giving rise to commercial products such as GitHub Copilot[1] and open-source multi-billion code models such as StarCoder (Li et al., 2023h) and Code LLaMA (Rozière et al., 2023).

The application of pretrained Transformers in software engineering, however, can be traced back to dates before decoder-only autoregressive models became dominant (Feng et al., 2020; Liu et al., 2020), and also goes beyond code processing (Arora et al., 2023; Feng et al., 2024; Ma et al., 2024). However, this domain is yet to witness a comprehensive review. In an attempt to bridge the gap between natural language processing (NLP) community and software engineering (SE) community on the topic of language model applications, we undertake a panoramic survey of language models for SE in this work, covering 70+ models, 40+ downstream tasks, 180+ datasets, and 900 related works. Putting the emphasis on code language models, we analyze their development, discuss their differences from general language models, and highlight the integration of code-specific features such as abstract syntax trees or data flows, as well as the latest techniques adapted from NLP.

Related to our work, we are aware of several surveys on similar topics, with three works concurrent to us (Hou et al., 2023; Zheng et al., 2023b; She et al., 2023). These works, however, focus either on NLP side (Zan et al., 2023; Xu & Zhu, 2022) or SE side (Niu et al., 2023; Hou et al., 2023; Zheng et al., 2023b; She et al., 2023), and do not cover models, tasks, and challenges from the other side. For example, Zan et al. (2023) focus on LLMs for text-to-code generation, while giving little discussion of other applications in software

[1] `https://github.com/features/copilot`

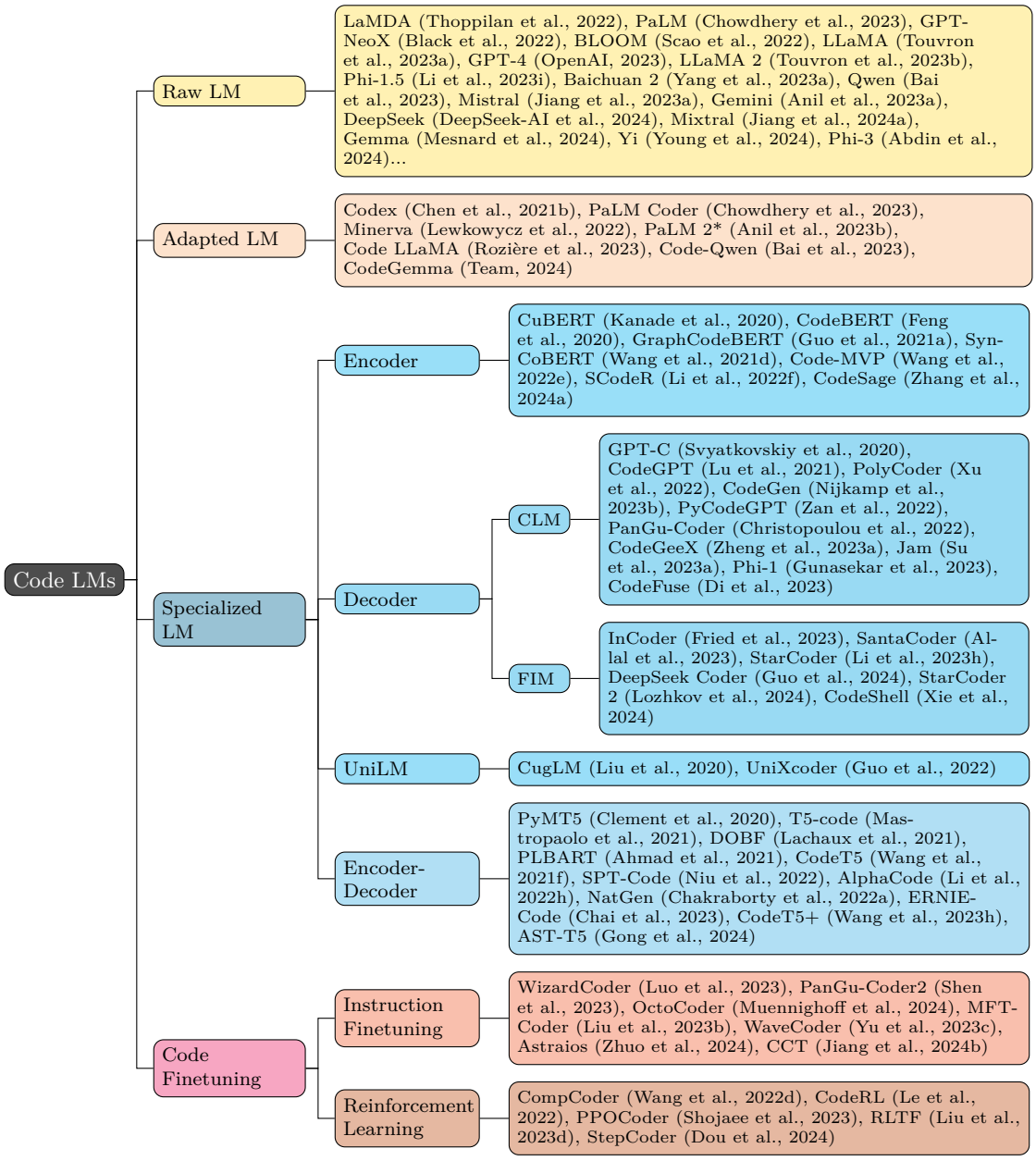

Figure 1: Our taxonomy of pretrained language models for code.

engineering community. Hou et al. (2023) and She et al. (2023), in contrast, comprehensively review works from SE venues such as ASE and ICSE, but cite only a handful of works from deep learning and NLP venues such as ACL, EMNLP, NeurIPS, and ICLR.

Thus, building on these works, we endeavor to unite the perspectives from both communities, and highlight the relationship between NLP and SE: SE applies language models to various tasks for automation, while NLP adopts tasks from SE to evaluate language models. We observe that advanced topics from language modeling have been recently introduced into code processing, including instruction tuning (Honovich et al., 2023; Xu et al., 2024; Luo et al., 2023), infilling objectives (Tay et al., 2023b; Li et al., 2023h; Rozière et al., 2023), recontemplation of scaling laws (Hoffmann et al., 2022; Gunasekar et al., 2023; Li et al., 2023i), architectural improvements (Shazeer, 2019; Su et al., 2024; Dao et al., 2022), and autonomous agents (Qian et al., 2023; Hong et al., 2023), while in return SE requirements, represented by programming (Chen et al.,

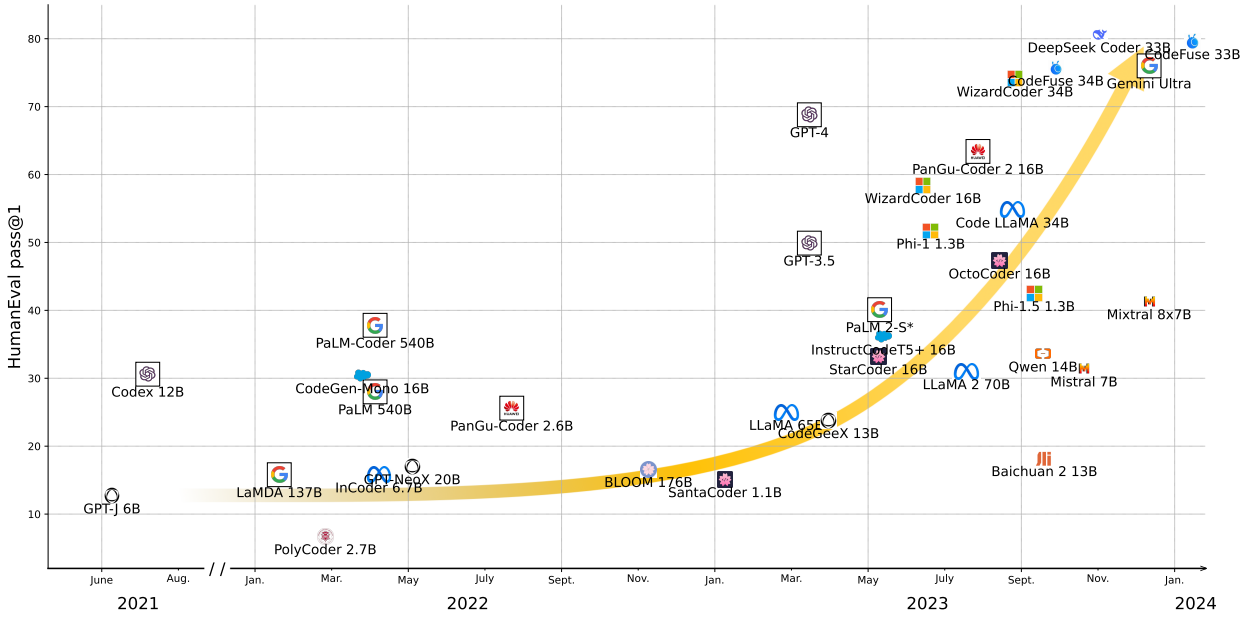

Figure 2: The timeline of code language models' progress on HumanEval.

2021b), are providing real-world testbeds for these technologies and driving the development of LLMs forward into production and deployment. We believe a systematic review of these advancements would benefit both communities. Furthermore, unlike the existing reviews that focus on programming-related tasks, we are also the first to explicitly go beyond programming and provide a global view of LLM applications in the full life cycle of software development, covering distinct stages such as requirement engineering, deployment, and operations.

The rest of this work is organized following the taxonomy presented in Figure 1. In Section 2 we first contextualize the downstream tasks in software engineering, highlighting the focus on programming related tasks. Then, in Section 3 we provide the preliminaries of language modeling and Transformer models, and in Section 4 we discuss the plethora of LLMs that have demonstrated coding ability. In Section 5 we review the specialized and often smaller models by their architecture, with special attention on the recent application of infilling objectives, instruction tuning, reinforcement learning, and engineering improvements. Then, in Section 6, we discuss unique features of code that are not available to natural languages but have been utilized to aid code processing. In Section 7, we review the most recent integration between LLMs and software development, before finally concluding this work in Section 8 and highlighting the current challenges in code processing.

2 Downstream Tasks in Software Engineering

Over the past decade, the SE community has found applications of language models in various downstream tasks. CodeXGLUE (Lu et al., 2021) consolidates most of the code-related tasks into a single benchmark, while HumanEval Chen et al. (2021b) brought NL-to-code synthesis into the spotlight in the NLP community, which has since become a standard task for evaluating LLMs (Figure 2). However, other tasks, especially those not directly related to coding, have remained understudied.

In this section, we first briefly introduce each of the traditional SE tasks and the application of pretrained language models in them in 2.1, and provide a comprehensive list of related works for each task. Then, we review the evaluation metrics in 2.2 and investigate program synthesis in more detail in 2.3. Lastly, we also discuss the latest trend of repository-level coding in 2.4. In Appendix A and B respectively, we list benchmarks for each downstream task and language models' performance on them.

2.1 Downstream Tasks in SE

The custom in software engineering is to categorize downstream tasks according to their input/output modality, such as NL-to-PL (also referred to as text-to-code) tasks and PL-to-NL (i.e. code-to-text) tasks (Lu et al., 2021). However, as we are dedicated to the full life cycle of software development, we adopt a different taxonomy in this work, and classify downstream tasks according to the stages in software development: requirement engineering (2.1.1), development (2.1.2), testing and analysis (2.1.3), deployment and operations (2.1.4), and lastly several novel tasks recently proposed for evaluating LLMs (2.1.5). We note that this taxonomy is interleaved with the understanding-generation dichotomy in NLP, since each category may contain both understanding and generation tasks, as discussed in 2.1.6.

2.1.1 Requirement Engineering

Requirement engineering refers to the specification, analysis, and validation of software requirements, software modeling, and UI/UX design. Related works are listed in Figure 3.

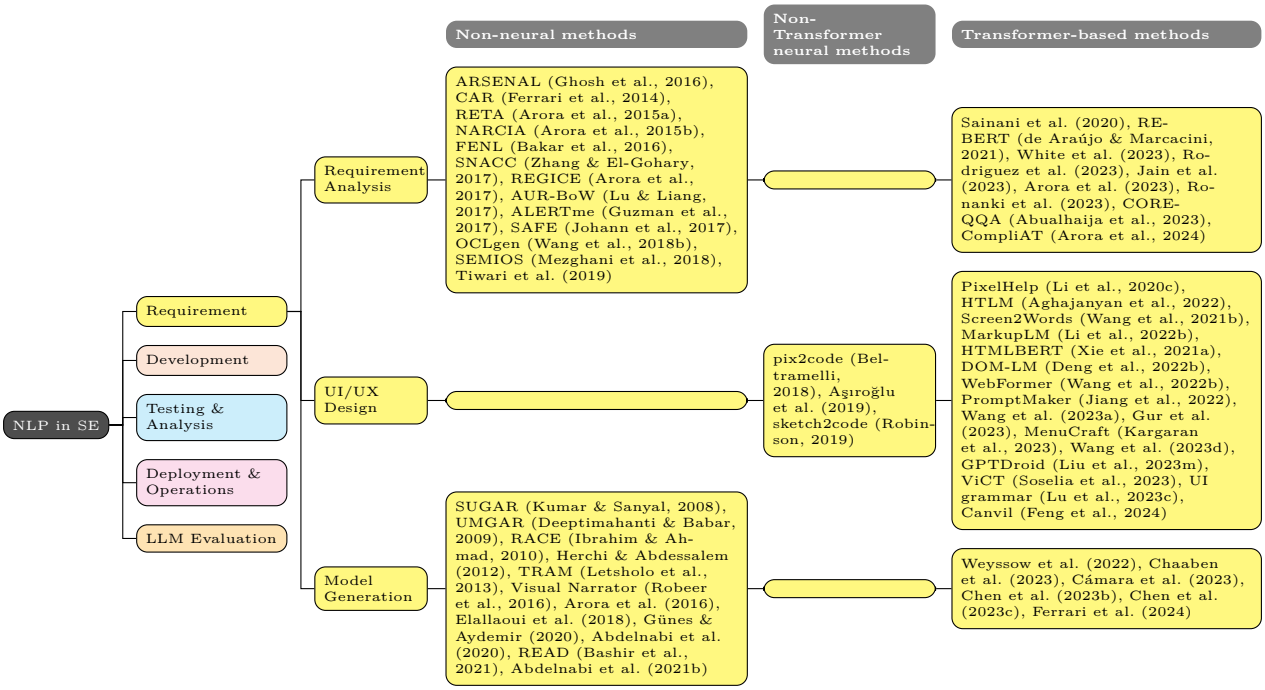

Figure 3: NLP applications in the *requirement engineering* stage of SE.

- *Requirement analysis* refers to the process of extracting, summarizing, classifying, and validating software requirements. Most early works in this field rely on statistical NLP technologies such as POS (Part-of-Speech) tagging and dependency parsing to extract actions from requirements, while more recent works utilize LLMs to directly classify, summarize, or extract requirement clauses. Zhao et al. (2020) provide a fine-grained survey on requirement analysis.

- *UI/UX Design*, short for User Interface and User Experience design, is a fundamental step in software development. While many works on UI design utilize computer vision technologies for layout design (Cheng et al., 2023; Kulkarni et al., 2023; Feng et al., 2023b), we only focus on the intersection of this task and NLP technologies, such as using language models to generate markup languages.

- *Model generation* aims to generate software models in modeling languages such as UML (Unified Modeling Language). Abdelnabi et al. (2021a) and Ahmed et al. (2022) provide comprehensive reviews on this task before the rise of LLMs.

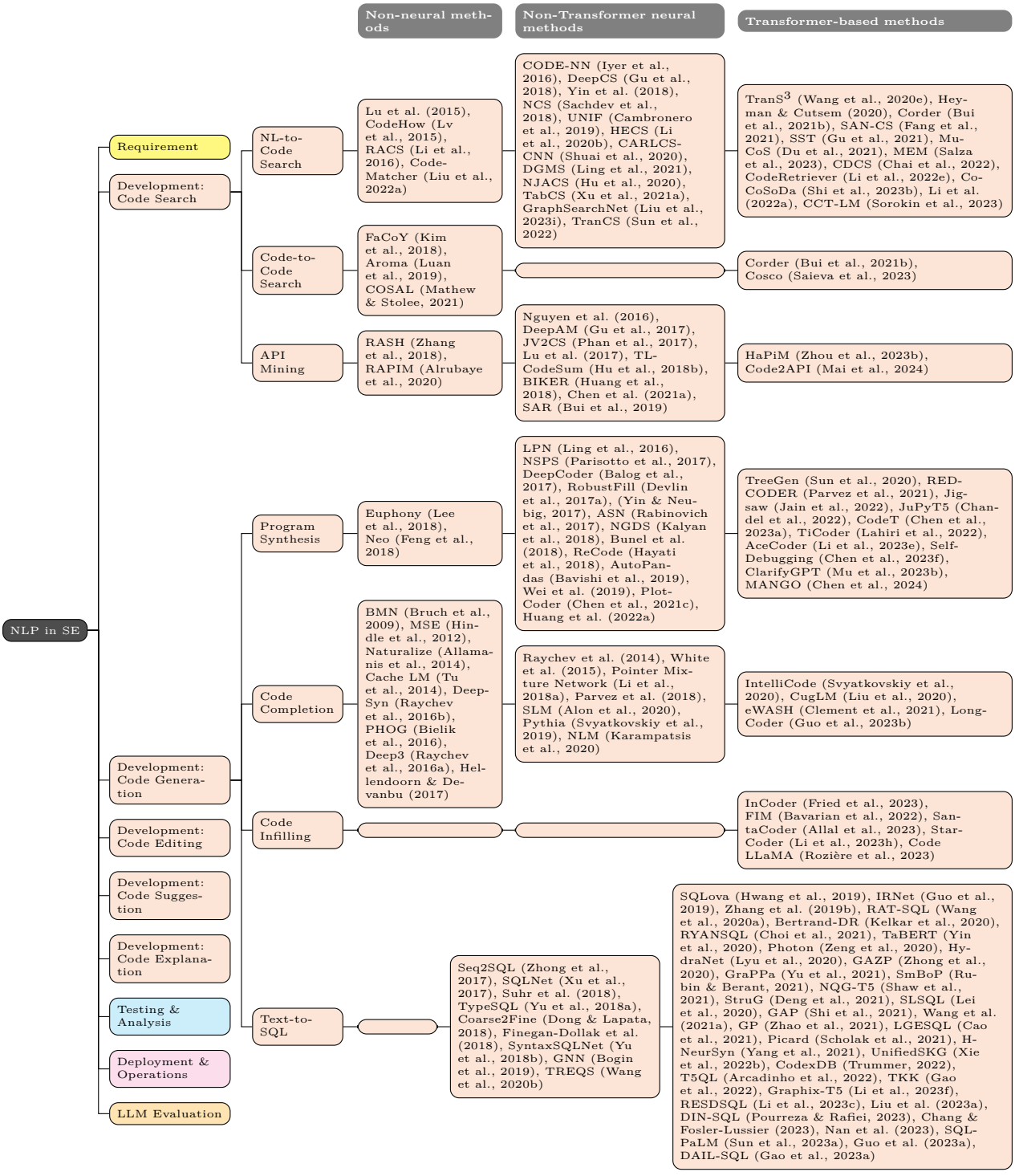

Figure 4: NLP applications in the *development* stage of SE.

2.1.2 Development

The development phase is the stage where most programming activities take place, and also the stage where LLM applications are most abundant. We further categorize this stage into different activities such as code

search, code generation, code editing, code suggestion, and code explanation. Related works are listed in Figure 4 and 5.

Code Search

- *NL-to-code search*, also referred to as text-to-code search, aims to retrieve relevant code given natural language queries, or to mine parallel text-code pairs from an unannotated corpus. This task is usually performed by computing a similarity metric between the embedding of query and candidate code, and the contextual embeddings produced by bidirectional language models - such as BERT - has proven to be extremely helpful. Grazia & Pradel (2023) and Xie et al. (2023a) provide comprehensive reviews on this topic.

- *Code-to-code search* is a similar task where the input is an existing code snippet, often in a different programming language from the target. Code-to-code search can be reformulated as finding clones of the query in the pool of targets, and is thus equivalent to clone detection to some extent.

- *API mining* refers to the process of finding similar APIs in different libraries, potentially in different programming languages. API mining is traditionally tackled by computing similarity metrics between source and target APIs using information retrieval models, but as generative models become ever more capable, it is also worth exploring to directly generate the target API as a sequence-to-sequence task. Another closely related task is *idiom mining* (Allamanis & Sutton, 2014), where the objective is to discover commonly used code patterns, which exposes the potential need for new APIs (Sivaraman et al., 2022).

Code Generation

- *Program synthesis* aims to generate code (usually a function or a method) given a natural language description. This task can be viewed as an updated version of NL-to-code retrieval using generative models instead of retrieval models. Statistical machine translation (SMT) and neural machine translation (NMT) models have been widely adopted for this task, often with enhanced decoders that leverage the unique grammatical rules of programming languages (Yin & Neubig, 2017; Rabinovich et al., 2017). Pretrained language models based on Transformer architecture, however, changed the game by directly generating the source code in the autoregressive language modeling style, even without task-specific finetuning (Chen et al., 2021b). We discuss this task in more detail in Section 2.3.

- *Code completion* aims to complete a piece of code given its prefix, and remains to date one of the most popular applications of code language models in IDEs. This is essentially language modeling applied to code (which we dub "code modeling"), and related technologies have been progressively introduced: n-gram, RNN, and Transformer. However, due to the structured nature of programming languages, many early works found grammar-aided statistical models to perform better (Bielik et al., 2016; Hellendoorn & Devanbu, 2017), and neural models only became dominant after 2018.

- *Code infilling* is another recently proposed task, after fill-in-the-middle pretraining (Bavarian et al., 2022) became popular. It is a generalization of code completion, where not only the left context but also the right context is given.

- *Text-to-SQL* is a special case of program synthesis, where the model is tasked to generate SQL commands from natural language queries. It has been a topic of special interest in the NLP community (as can be seen from Figure 4), and one hypothesis for this is that SQL's declarative nature and restricted grammar make it easier to optimize compared with imperative programming languages such as C and Python (Marcus et al., 2019; 2020). We refer to Kumar et al. (2022); Deng et al. (2022a); Qin et al. (2022a); Katsogiannis-Meimarakis & Koutrika (2023) for surveys on this topic.

Code Editing

- *Code translation* aims to translate a piece of code (usually a function or method) into another programming language. The relation between code translation and cross-lingual code search is similar to the one between program synthesis and text-to-code retrieval, and SMT/MNT models have also been widely applied to this task. One of the important applications of code translation is migrating old projects written in obsolete

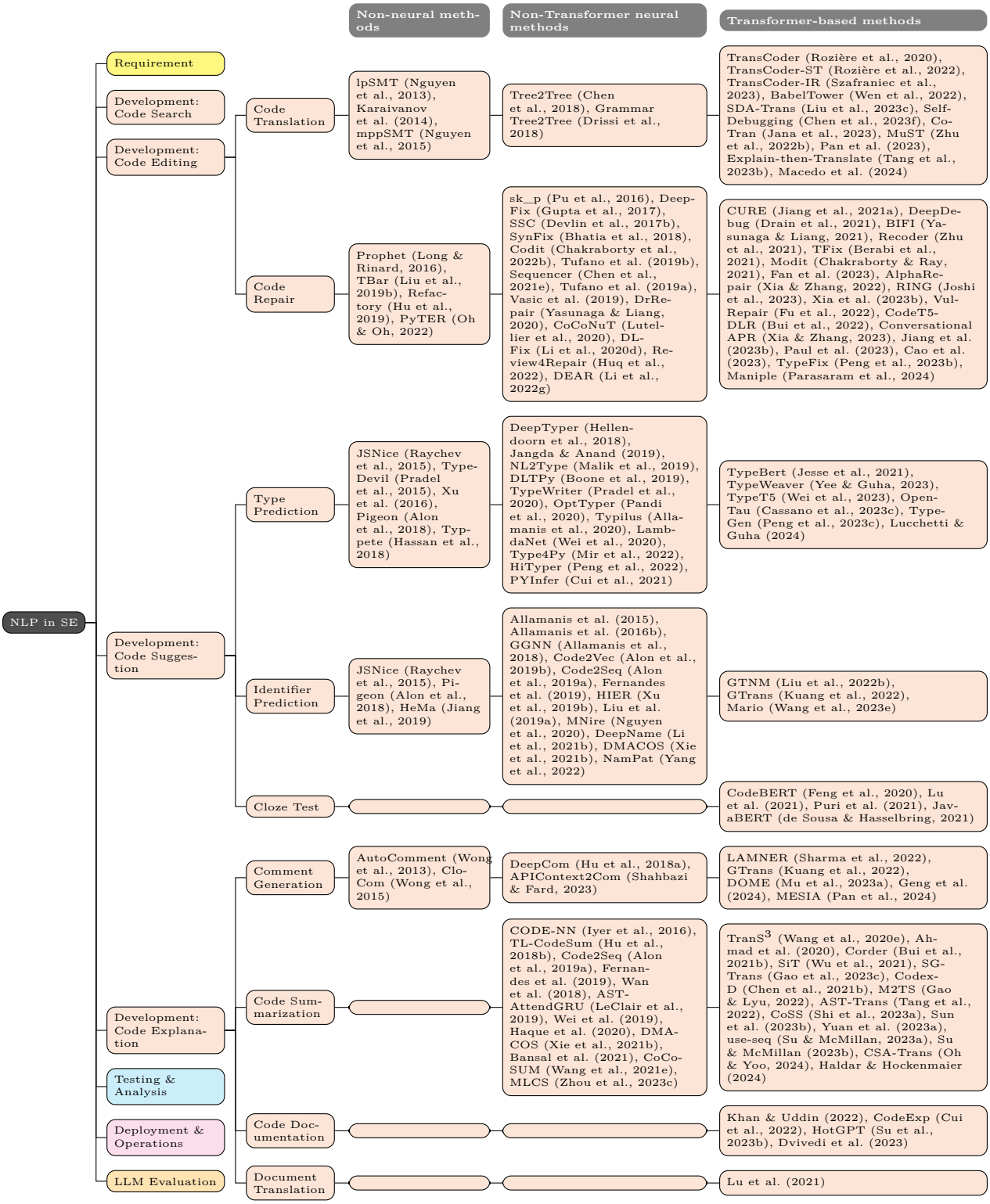

Figure 5: NLP applications in the *development* stage of SE.

languages. However, we are yet to witness such applications at scale in the LLM era, as the context window

of even the most powerful language models are quite limited in the face of such projects. Malyala et al. (2023) provide a short survey on this task from the SE perspective.

- *Program repair*, also known as bug fix, aims to fix a piece of buggy code. Like code translation, it is a traditional sequence-to-sequence generation task, and surveys are abundant on this topic (Gazzola et al., 2018; Monperrus, 2018; Zhong et al., 2022; Zhang et al., 2023d; Huang et al., 2023a).

Code Suggestion

- *Type prediction* aims to predict the type of variables in dynamic programming languages such as Python and JavaScript. It has been used as a pretraining objective for code language models (Wang et al., 2022e), where it is often simplified as a binary tagging task to predict which tokens in the code are identifiers (Wang et al., 2021d;f).

- *Identifier prediction* is the task of predicting identifier names in the code. As these names are deemed to contain important semantic information, this task has been utilized for code summarization (Allamanis et al., 2016b), as well as pretraining code models (Wang et al., 2021f; Niu et al., 2022). A special case of identifier prediction is *method name prediction*.

- *Cloze test* is a recently proposed task for code understanding, after the rise of BERT-style pretraining. Due to the unique semantics of programming languages, several keywords are often selected for this test, such as `min` and `max` (Feng et al., 2020).

Code Explanation

- *Code summarization*, *comment generation*, and *code documentation* all aim to generate explanations for code to facilitate understanding of the code, but with slightly different emphasis. Code summarization generates a natural language summary of the given code, while comment generation generates comments. These two terms are often used interchangeably in the literature, and their intended audience is usually developers. Code documentation, on the other hand, is more structured, includes more detailed information about the code's usage and functions, and is usually intended for a broader audience. Zhang et al. (2022) provide a survey on code summarization.

- *Document translation* is the automatic translation of code-related documents. Since models, datasets, and prompting strategies for machine translation are abundant in NLP (Vaswani et al., 2017; Goyal et al., 2022; He et al., 2023b), we do not go into detail about this task.

2.1.3 Testing and Analysis

The third stage in software development is about testing and analyzing the correctness of programs (as well as other properties, such as security). Related works are listed in Figure 6.

Testing

- *Unit test generation* aims to generate unit tests for a given program. Prior to the rise of Codex and other code LLMs, almost all works in this area employed non-neural methods (see Figure 6). In the age of LLMs, however, this task is ever more important, as research has shown that the current unit tests for evaluating LLMs' program synthesis capability may be insufficient (Liu et al., 2023f). Wang et al. (2023b) provide a comprehensive survey on software testing with LLMs.

- *Assertion generation* is a subtask of unit testing. Given a program and a partial unit test, this task aims to generate assertions (also known as *oracles* in software engineering) within the unit test. This task has generally went unnoticed by the NLP community, as the program synthesis task used for evaluating LLMs often concern standalone, competition-style methods, for which the simple assertion of the equality between program output and expected answer suffices.

- *Mutant generation* aims to generate mutants of a given program for the purpose of mutation testing, and relates closely to unit test generation and assertion generation. A mutant that is not detected by a given set of unit tests and assertions indicates that either additional test cases or better assertions are required (Fraser

& Arcuri, 2011). Recently, masking out tokens in the source code and sampling them from the output of a masked language model has become a common method for this task. Ojdanic et al. (2021; 2023) give empirical comparisons between different mutation methods.

- *Fuzzing* is another software testing task, where the objective is to generate a large set of inputs covering as many corner cases as possible. While many recent works on fuzzing target deep learning libraries, few have utilized language models to conduct this process (see Figure 6).

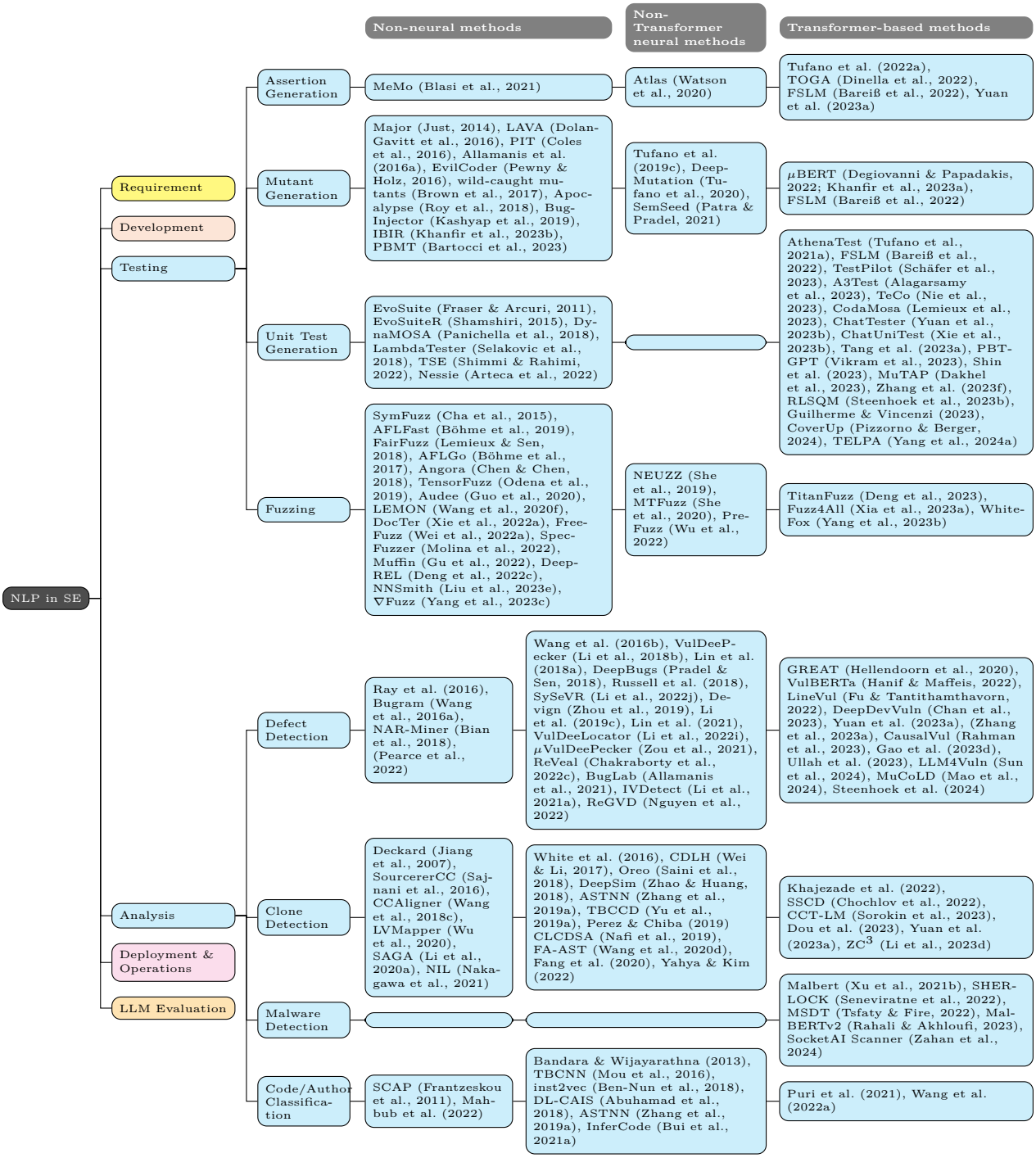

Figure 6: NLP applications in the *testing & analysis* stage of SE.

Analysis

- *Defect detection*, or *vulnerability detection*, predicts whether the input code is buggy or not, and is a standard single-sentence classification task. Nong et al. (2023); Steenhoek et al. (2023a); Bi et al. (2023); Harzevili et al. (2023); Zhou et al. (2024) provide surveys on this task.

- *Malware detection* is a similar task to defect detection. However, malware differs from other types of vulnerable code in that the bugs therein are malicious - i.e. they are intentionally injected. We refer to Sahin & Bahtiyar (2020) and Gopinath & Sethuraman (2023) for reviews on non-Transformer based methods for this task.

- *Clone detection* predicts whether or not two pieces of code are clones of each other. In software engineering there exist four types of code clones, and the most challenging type to identify is semantic clones, i.e. syntactically dissimilar code that have the same functionality. As this task can be viewed as a two-sentence classification task, BERT-style language models have been widely applied to it. Svajlenko & Roy (2020) and Zhang & Sakurai (2021) provide comprehensive reviews on non-deep-learning based methods for this task.

- *Code classification* aims to predict the functionality of a piece of code within a predefined set of labels. A very similar task is *author identification*, which predicts the author of the input code. Both tasks are standard single-sentence classification tasks, and traditional machine learning methods have been widely adopted in them (Kalgutkar et al., 2019), while pretrained language models have seen almost no application.

2.1.4 Deployment and Operations

After software passes the testing stage, it is deployed into production, and often continuously maintained or updated. For this stage, we summarize the specific tasks and their related works in Figure 7.

Deployment

Compiler optimization is the task of optimizing compiled code to increase its efficiency. Many early works applied reinforcement learning and other machine learning technologies to this task by optimizing task scheduling or pass ordering (Wang & O'Boyle, 2018; Leather & Cummins, 2020), and it is only recently that researchers started to view it as a sequence-to-sequence generation task with the help of powerful LLMs (Cummins et al., 2023).

- *Decompilation* is the reverse process of compilation, and an important topic in reverse engineering. In this task a model takes a compiled program - represented in assembly code or binary machine code - and aims to output the original source program in high-level languages such as C.

- *Obfuscation* refers to the process of renaming identifiers (e.g. variables, methods, and classes), for example to generic names like `var_1, var_2` or `x, y`. It is an important technique in virus detection, intellectual property protection, and code size reduction (Collberg & Thomborson, 2002; Murad et al., 2010; Vasilescu et al., 2017). *Deobfuscation* refers to the reverse process, where meaningful identifier names are recovered from obfuscated programs. Obfuscation can be easily achieved statically, but deobfuscation has been a subject of more interest in recent years. It plays a significant role in decompiling, and has also been adopted as a pretraining objective for code language models (Lachaux et al., 2021; Ding et al., 2022a; Liu et al., 2022d).

- *Log analysis* aims to analyze the system logs produced by software products, for example parsing logs into structured templates or finding anomalies from raw logs. Zhu et al. (2019) provide a survey on traditional methods for this task up to 2018, and Chen et al. (2021d) give an empirical comparison between neural network based methods. Zhang et al. (2023e) also cover more recent methods for log parsing, while Landauer et al. (2022) survey methods for anomaly detection in logs.

Operations

- *Code review* aims to automate the process of peer code review, and includes many subtasks, such as review necessity prediction, review comment generation, code refinement, and review decision prediction.

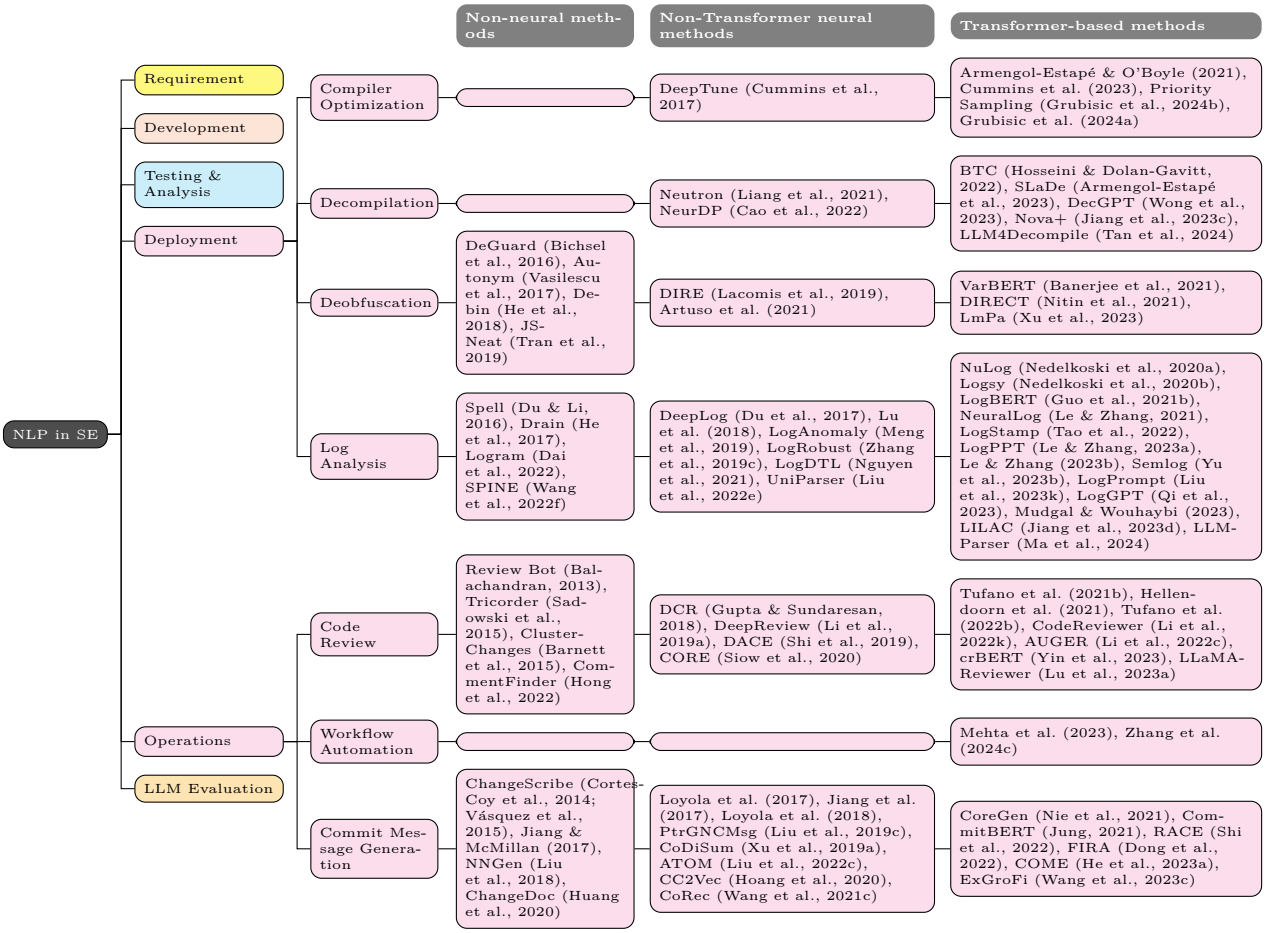

Figure 7: NLP applications in the *deployment & operations* stage of SE.

- *Commit message generation* aims to automatically generate commit messages for code changes. This task takes the code before and after change as input, and output the description for the change. This can be viewed as the dual task of program repair, as many code changes and their accompanying commit messages concern bug fixing. Tao et al. (2021) provide a survey on methods and datasets for this task up to 2021.

- *Workflow Automation* is the automation of workflows in Git. Due to the complexity involved in this task, it was only recently brought to attention after the advent of powerful LLMs.

2.1.5 LLM Evaluation

Apart from the previously covered stages in SE, we observe that programming-related evaluation of LLMs has been gaining momentum in both the NLP and the SE community since the release of Codex. Thus, we also list several novel tasks that appeared recently for this purpose. We discuss this topic in more detail in Section 7.

- *Code reasoning* requires a model to reason about code or algorithms, and answer related questions which are written in multiple-choice format or free-form QA format, which may range from conceptual understanding to numerical calculation and complexity analysis. It often comes as a subset of composite evaluation benchmarks such as MMLU (Hendrycks et al., 2021b).

- *Coding for reasoning* is a special case of reasoning where LLMs solve complex reasoning problems by generating code that will be executed by external interpreters. This task abstracts the reasoning process from numerical calculations, and is thus of special interest in evaluating LLMs. More recently research has

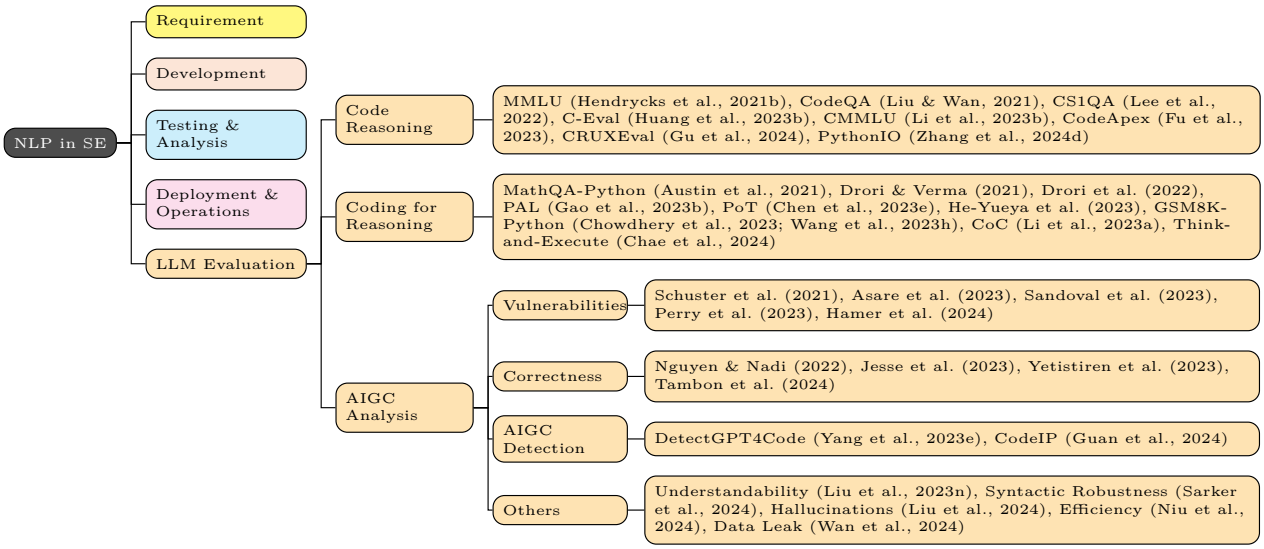

Figure 8: Programming-related evaluation of LLMs.

also shown that asking the LLM itself to simulate the execution of generated code can also help its reasoning process.

- *AIGC analysis* aims to analyze AI-generated code[2] from various aspects, such as correctness, vulnerabilities, and detection of AI-generated code.

2.1.6 NLP Point-of-View

Unlike software engineering, evaluation tasks in NLP are generally categorized into understanding and generation. The former, represented by GLUE (Wang et al., 2018a) and SuperGLUE (Wang et al., 2019), emphasizes the comprehension of input text, and is typically formalized as classification, regression, sequence tagging, or span extraction. The later, on the other hand, involves autoregressive generation of text, such as machine translation and summarization.

Among the previously listed tasks, code synthesis, code translation, code repair, deobfuscation, unit test generation, assertion generation, mutant generation, code summarization, code review, identifier prediction, and commit message geneartion are sequence-to-sequence generation tasks. Formally, each instance of these tasks has a source sequence $\mathbf{x}$ (e.g. a piece of source code) and a target sequence $\mathbf{y}$ (e.g. its corresponding summarization), and the language model is tasked to maximize the conditional probability given by (5), where θ can be either a decoder-only model or an encoder-decoder model. In the former case, $\mathbf{x}$ and $\mathbf{y}$ are concatenated. In the later case, $\mathbf{x}$ is processed by the encoder and $\mathbf{y}$ is processed by the decoder.

Code completion and code infilling are also generation tasks, but differ from sequence-to-sequence tasks where the input and output are related by different sequences. In these two tasks, the target is a continuation or infill of the input. They correlate closely to the language modeling objectives given in Equation (3) and (5). Similarly, cloze test takes the same form as Equation (4) but is usually considered an understanding task, as its output is usually a single token and does not involve autoregressive generation.

Defect detection, malware detection, clone detection, code classification, and author identification are sequence classification tasks. In these tasks, a set of labels $\mathcal{Y}$ is defined over the input, and each instance is assigned a label $y \in \mathcal{Y}$ (e.g. for defect detection $\mathcal{Y} = \{0, 1\}$, while for author identification a possible $\mathcal{Y}$ is {Alice, Bob, John, others}). The model is then tasked to maximize

$$p_\theta(y|\mathbf{x}). \tag{1}$$

[2]In the AI community "AIGC" usually refers to AI-Generated *Content*. In this work we overload this expression with AI-Generated *Code*.

Type prediction is a token classification task, also known as tagging. In this task, each token x_i is assigned a label $y_i \in \mathcal{Y}$, with an example $\mathcal{Y}$ being {int, float, string, bool, non-identifier, other}. The model's objective is to maximize

$$\prod_{i=1}^{n} p_\theta(y_i|\mathbf{x}). \tag{2}$$

The last two tasks - code retrieval and code search - also belong to understanding tasks. In these tasks, each source sequence $\mathbf{x}$ is paired with a positive target sequence $\mathbf{y}$ and a set of negative targets $\bar{\mathbf{y}} \in \{\mathbf{y}_1, \cdots, \mathbf{y}_k\}$. The model's task is to find a similarity metric s such that $s(\mathbf{x}, \mathbf{y})$ is larger than $s(\mathbf{x}, \bar{\mathbf{y}})$.

2.1.7 Code LLMs for Low-Resource, Low-Level, and Domain-Specific Languages

In NLP, human languages are categorized into high-, middle-, and low-resource languages based on the amount of available data in each language. High-resource languages such as English are extensively studied, while low-resource languages such as Swahili and Yoruba often rely on transfer learning from other languages to improve performance due to data scarcity (Conneau et al., 2020; Xue et al., 2021; Zhang et al., 2023g).

This phenomenon also exists in code modeling, as most works listed in Figure 3-8 target the most popular programming languages such as C, Java, and Python. However, a major difference between NLP and SE is that in NLP, low-resource languages are often spoken by few people or even endangered, while in SE low-resource languages are often designed for specific domains and purposes, and thus have an active user community. Verilog, a hardware description language, is one such example, for which code modeling research is quite abundant (Thakur et al., 2023a;b; Lu et al., 2023b; Liu et al., 2023g; Tsai et al., 2023; Thorat et al., 2023; Liu et al., 2023h; Nadimi & Zheng, 2024).

From a different perspective, most of these popular languages are imperative languages, while few works have studied NLP and LLM applications in declarative or functional languages, except those concerning SQL and one recent work on Haskell (van Dam et al., 2024). However, research has shown that declarative languages are more aligned for optimization since they describe the "what" and not "how" in programming (Marcus et al., 2019; 2020), thus they are worth more attention in the future research.

Similar to NLP, many works in code modeling have studied the transfer of model capability across languages. Chen et al. (2022), for example, investigate the performance of multilingual language models on Ruby (which is one of the six languages in the popular CodeSearchNet pertaining corpus (Husain et al., 2019), but relatively low-resourced compared with the other five), finding that multilingual models have lower performance-to-time ratio compared with monolingual ones. Cassano et al. (2023a), on the other hand, propose MutiPL-T, an approach for translating high-resource training data into low-resource languages for fine-tuning. However, such research is yet scarce compared with the NLP community, but with the recent advent of colossal, multilingual datasets such as The Stack v2 (Lozhkov et al., 2024) we expect to see more of it in the future.

2.2 Evaluation Metrics

Of the tasks mentioned in Section 2.1, the understanding tasks are similar in form to natural language understanding tasks (Wang et al., 2018a; 2019) and evaluated likewise by metrics such as accuracy, F1 and Mean Reciprocal Rank (MRR), while short generation tasks such as identifier prediction is also evaluated by accuracy of exact matches. Code-to-text tasks are evaluated with common metrics for text generation such as BLEU (Papineni et al., 2002).

Evaluation of tasks involving code generation, on the other hand, is more complicated. Most early works evaluate syntactical correctness, i.e. the percentage of generations that can be successfully parsed. Chen et al. (2018) argue against such metrics and suggest reference match instead, which is the percentage of generations that are exactly the same as the references. Ren et al. (2020) propose CodeBLUE, a variant of BLEU that takes code syntax and semantics into account by evaluating the overlap of abstract syntax tree (AST) and data flow.

As code generation models became more capable over the years, however, these metrics based on content-overlap have been found to be inadequate (Rozière et al., 2020; Hendrycks et al., 2021a; Austin et al., 2021), since functionally equivalent snippets of code can differ dramatically in their lexical forms. Consequently, researchers have turned their attention to functional correctness. One popular example of such metrics is pass@k, proposed by Kulal et al. (2019) and refined by Chen et al. (2021b), which is an unbiased estimator of the model's chance in passing all unit tests of a program with any of k generated samples. This metric can be generalized to pass$n$@k (Li et al., 2022h), which limits the number of model submissions to n but allows filtering by unit tests given in the input from k samples.

2.3 Program Synthesis

While dozens of evaluation tasks exist in software engineering, they have generally stayed out of the focus of the NLP community until very recently. The only exception is program synthesis, which has become a standard evaluation task for LLMs since the advent of HumanEval in 2021. Looking back at this task, we identify four changes in program synthesis over the years: shift of coding paradigms (from example-based to intention-based), generalization in languages (from domain-specific languages to general-purpose languages), simplification of model architectures (from grammar-guided decoders to general-purpose language models), and application of execution-based feedback.

Many of the early methods for program synthesis are example-based (Menon et al., 2013), which means they induce programs from input-output examples, often in domain-specific languages (DSLs) such as FlashFill (Devlin et al., 2017a) and Karel[3] (Bunel et al., 2018), as these languages are usually simple in syntax and structure.

As code generation models became more capable over the years, researchers started to pay attention to program synthesis in general-purpose programming languages as well. Hearthstone (Ling et al., 2016) and CONCODE (Iyer et al., 2018) are two of the early datasets, representing Python and Java respectively. Each example in Hearthstone is the description of a card in the game and its corresponding class implementation, while examples in CONCODE are simply Java methods paired with their natural-language documentation crawled from public GitHub repositories. Synthesizing programs from their corresponding natural language descriptions has since then become a standard practice in program synthesis, and has led to some of the most widely used benchmarks, such as HumanEval (Chen et al., 2021b), which has even been translated into multiple languages (Cassano et al., 2023b; Zheng et al., 2023a; Muennighoff et al., 2024). Some recent benchmarks use general-purpose languages but focus on specific domains, such as data science (Bavishi et al., 2019; Lai et al., 2023) or Jupyter notebooks (Agashe et al., 2019), while several math reasoning benchmarks have also been converted to programming tasks, including MathQA-Python (Amini et al., 2019; Austin et al., 2021) and GSM8K-Python (Cobbe et al., 2021; Chowdhery et al., 2023; Wang et al., 2023h).

Many early works argue that simply treating program synthesis as a text generation task does not utilize the underlying syntax of programming languages, and thus often use syntax-enhanced decoders to inject the target syntax as prior knowledge (Yin & Neubig, 2017). LLMs, however, have demonstrated that pretrained language models are capable of generating syntactically correct programs without loss of generality. Under this setting, researchers start to *execute* the generated programs and provide feedback to the generation model to inject the prior knowledge of code instead. This has recently led to the popularity of *interactive coding*, which we discuss in more detail in Section 7.1.

2.4 Repository-Level Coding

Most tasks discussed in Section 2.1 are limited to a single file or even a single function, as cross-file code modeling poses challenges that are beyond the capability of most existing language models. Recently, however, position interpolation techniques (Chen et al., 2023d; Rozière et al., 2023; Peng et al., 2023a) have extended the context window of LLMs to hundreds of thousands of tokens, making it possible to contextualize coding activities within entire repositories. Several works have studied code completion (Shrivastava et al.,

[3]FlashFill is used in Microsoft Excel for string transformation. Karel is a simple programming language for educational purpose.

2023b; Ding et al., 2022b; Zhang et al., 2023b; Shrivastava et al., 2023a; Phan et al., 2024; Wu et al., 2024b) and generation (Liao et al., 2023; Zan et al., 2024) leveraging repository-level context, and corresponding benchmarks have been proposed Liu et al. (2023j); Ding et al. (2023); Li et al. (2024a). Bairi et al. (2023) investigate the more challenging tasks of repository-level API migration and temporal editing, while Jimenez et al. (2023) introduce a related benchmark, SWE-bench.

3 Language Modeling Preliminaries

As code is ultimately a subset of natural languages, language models have been extensively used to tackle the tasks listed in Section 2. Before diving into the language models themselves, we first briefly review the preliminaries of Transformer-based language modeling in this section following the common choices of training objectives, and also some implementation designs.

3.1 Causal Language Modeling

Unidirectional language models (also known as causal language models[4]) factor the probability of a sentence into the product of each token's conditional probability with the chain rule. A piece of input text $\mathbf{x} = [x_1, x_2, \cdots, x_n]$ consisting of n tokens is modeled as

$$P(\mathbf{x}) = \prod_{i=1}^{n} p_\theta(x_i|\mathbf{x}_{1:i-1}), \tag{3}$$

where $\mathbf{x}_{1:i-1}$ is a shorthand for tokens before x_i in the input, and θ is the parameters of the model. With Transformer decoders such as GPT (Radford et al., 2018; 2019; Brown et al., 2020) and LLaMA (Touvron et al., 2023a;b), the conditional probability in (3) is modeled by adding an attention mask to the attention matrix of each Transformer block, ensuring that x_i can only attend to previous tokens. During training, the cross entropy loss on all tokens in the input is calculated in parallel, while at inference time each new token is generated autoregressively. For further details about the Transformer architecture we refer to Vaswani et al. (2017).

3.2 Masked Language Modeling

Unlike causal language models, bidirectional language models are trained to acquire a better contextual representation of text rather than generating text autoregressively. In the vanilla Transformer, the encoder part is allowed to attend to a token's left as well as right context for this purpose. BERT (Devlin et al., 2019) takes one step further and pretrains only a Transformer encoder. A set $\mathcal{M}$ of randomly chosen tokens in the input are replaced by a special token `[MASK]` to obtain a noisy input $\hat{\mathbf{x}}$, for example $[\texttt{[CLS]}, x_1, \texttt{[MASK]}, x_3, \texttt{[MASK]}, x_5, \texttt{[EOS]}]$[5], and the model is trained to recover the original tokens by maximizing

$$\prod_{m \in \mathcal{M}} p_\theta(m|\hat{\mathbf{x}}). \tag{4}$$

While this objective requires the model to have a deep understanding of the input text to reconstruct it, it suffers from low training efficiency, since only a small set of tokens (usually 15%) are masked (and thus "trained on"). To address this issue, Clark et al. (2020) propose ELECTRA, which is trained to discriminate whether or not each token in the input has been replaced by a BERT-like model instead, thereby computing loss on all input tokens.

[4]The training objective of such language models is Causal Language Modeling (CLM), but also referred to as Next Token Prediction.

[5]Both `[CLS]` and `[EOS]` are artificial tokens added to the input text. `[CLS]` is added at the beginning and its representation is used for sentence classification, while `[EOS]` indicates end of sentence. The original BERT also uses another special token `[SEP]`, which is not in common use in LLMs, and we refer to Devlin et al. (2019) for details.

3.3 Denoising Objectives

GPT-style causal LM and BERT-style bidirectional LM each has its own strengths and weaknesses. While GPT can be used for autoregressive generation, it lacks a bidirectional representation of input text, and is thus unsuitable for sequence-to-sequence (seq2seq) generation tasks such as translation and summarization. BERT, on the other hand, can produce bidirectional representations, but is pretrained only for mask filling, not generation.

The vanilla Transformer encoder-decoder architecture combines the respective advantages of GPT and BERT. T5 (Raffel et al., 2020) is such a model pretrained with *span corruption*, which can be regarded as a variant of MLM. During pretraining, spans of text in the input are replaced with sentinel tokens, which play the same role as `[MASK]` in BERT. The noisy input is first processed by the encoder with bidirectional attention, and the masked spans are then generated autoregressively by the decoder. Formally, if k spans are sampled for corruption in input $\mathbf{x}$, the noisy input $\hat{\mathbf{x}}$ is then constructed by replacing each span with a special token `<extra_id_i>`, for $i = 1, 2, \cdots, k$, and the target $\mathbf{y}$ is constructed by concatenating all spans prepended with corresponding sentinels: [`<extra_id_1>`, span_1, $\cdots$, `<extra_id_k>`, span_k]. The model is then trained with a standard seq2seq objective, by maximizing

$$p_\theta(\mathbf{y}|\hat{\mathbf{x}}) = \prod_{i=1}^{n_y} p_\theta(y_i|\hat{\mathbf{x}}, \mathbf{y}_{1:i-1}). \tag{5}$$

Lester et al. (2021) show that models pretrained with such objectives can be adapted for autoregressive language modeling with extra pretraining using the prefix language modeling objective, i.e. splitting the text into two parts, processing the first part with encoder and generating the second part with decoder.

Tay et al. (2023b) argue that span corruption is also closely related to CLM, since one can mask out the whole input text as a single span and train the decoder to generate it autoregressively. Inspired by this relation, they propose UL2, which is the combination of many span corruption objectives that differ in corruption rate and span length. Applying it to both encoder-decoder models and decoder-only models, they find that encoder-decoder models perform better under the same computation budget constraint. Other researches have also found that such encoder-decoder models generally perform better than causal decoder-only models (Wang et al., 2022c; Soltan et al., 2022).

3.4 Auxiliary Objectives

Language modeling objectives, such as previously discussed CLM and MLM, mainly train the model to capture token-level information and are ineffective at modeling document structures. Thus, auxiliary objectives are often added to help the models learn such global information. BERT is pretrained with next sentence prediction (NSP) along with MLM, which is formulated as a binary classification task to predict whether two segments in the input are neighboring in the original corpus. Lan et al. (2020) propose a more challenging sentence-order prediction (SOP) task, where the negative samples are constructed by swapping the order of two neighboring sentences instead of sampling a random sentence from other documents.

Relatedly, Raffel et al. (2020) mix supervised downstream samples such as GLUE (Wang et al., 2018a) into T5's pretraining dataset to conduct multi-task pretraining. However, it is worth noting that since they unify all tasks into a text-to-text format, the training objective is the same for their self-supervised pretraining and supervised downstream tasks, i.e. conditional generation of the target sequence.

3.5 Implementation Design

While most researches on pretraining language models have focused on designing novel training objectives, low-level implementation of the Transformer architecture itself is also being continuously improved over the years in pursuit of stability, performance, and efficiency, as shown in Figure 9.

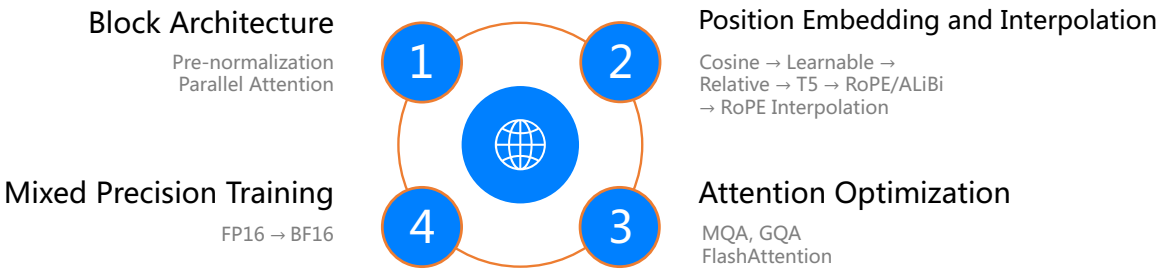

Figure 9: Major implementation changes in LLM over the past few years.

The original Transformer block proposed by Vaswani et al. (2017) is formulated as

$$h = \text{LN}(\text{Attention}(x) + x), \tag{6}$$
$$y = \text{LN}(\text{FFN}(h) + h), \tag{7}$$

where x is the layer's input; y is the layer's output; "Attention" is the self-attention sublayer; "FFN" is the feed-forward sublayer, and "LN" is layer normalization (Ba et al., 2016).

GPT-2 (Radford et al., 2019) moves layer normalization to the input of each Transformer sub-block to stabilize training:

$$h = \text{Attention}(\text{LN}(x)) + x, \tag{8}$$
$$y = \text{FFN}(\text{LN}(h)) + h, \tag{9}$$

and such pre-norm has since become a standard practice in Transformer decoders.

GPT-J (Wang & Komatsuzaki, 2021) modifies the Transformer block to compute FFN sub-layer and self-attention sub-layer in parallel to increase computation throughput:

$$y = x + \text{FFN}(\text{LN}(x)) + \text{Attention}(\text{LN}(x)), \tag{10}$$

and Chowdhery et al. (2023) observes limited performance degradation when applying this design to larger models.

As self-attention alone cannot capture the position information of each input token, the vanilla Transformer adds a non-learnable vector that's dependent on the absolute position in the input sequence to the embedding of each token, which is known as cosine position embedding. Later works such as GPT and BERT use learnable embeddings instead, while T5 adopts relative position embedding (Shaw et al., 2018), where the information of relative position between query and key is injected in the computation of self-attention instead. A more recent scheme RoPE (Su et al., 2024) multiplies the keys and queries by a position-dependent rotation matrix, and is later shown to enable position interpolation for processing of longer sequences (Chen et al., 2023d; Rozière et al., 2023; Peng et al., 2023a). Alternatively, Press et al. (2022) propose ALiBi, which directly attenuates the attention scores according to the relative position between key and query. RoPE is popularized by its application in PaLM (Chowdhery et al., 2023) and GPT-NeoX (Black et al., 2022), while ALiBi is adopted by BLOOM (Scao et al., 2022). We refer to Zhao et al. (2023) for a survey on position embedding and interpolation in Transformers.

Apart from position embeddings, another issue in Transformer that has long troubled researchers is the fact that the complexity of self-attention scales quadratically with the input sequence length. Some works such as Reformer (Kitaev et al., 2020), Linformer (Wang et al., 2020c), Performer (Choromanski et al., 2021) and cosFormer (Qin et al., 2022b) use approximate attention to reduce this complexity, but they mostly come at the cost of degraded performance (Tay et al., 2023a; Lin et al., 2022). Other works tackle this issue from an engineering point-of-view. MQA (Shazeer, 2019) shares the same set of keys and values across all attention

heads to optimize memory-to-arithmetic ratio and significantly improves inference speed at small costs of model performance. Its variant Grouped-Query Attention (GQA, Ainslie et al., 2023) takes a middle-ground approach by dividing attention heads into groups and sharing the same set of keys/values within each group. Orthogonally, Dao et al. (2022) introduce FlashAttention, which is an exact but improved implementation of self-attention that optimizes IO operations on the accelerating device via tiling to improve memory efficiency.

Another important technique for improving LLMs' training efficiency is mixed precision training (Micikevicius et al., 2018), which stores model weights and activations in lower precision to save memory consumption. Early works use FP16 format for this low precision, while BF16 has now become more popular (Chowdhery et al., 2023; Scao et al., 2022), as it can represent the same range of floating point numbers as FP32, thus preventing numerical overflowing and underflowing during training.

4 General Language Models for Code

Since language models scaled to hundreds of billions of parameters (Brown et al., 2020; Chowdhery et al., 2023), many of them have demonstrated non-trivial coding capability, even if they are not specifically designed or trained for code. Pioneered by Codex, researchers have also found continual pretraining on code to significantly benefit language models' performance on code[6].

4.1 Off-the-Shelf Language Models

Large language models are often pretrained on trillions of tokens following the scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022), and such an amount of text data is often a diverse composite with a non-negligible part of code. The Pile (Gao et al., 2021), for example, includes 95GB of code crawled from GitHub out of its 800GB raw dataset, while the multilingual pretraining dataset ROOTS (Laurençon et al., 2022) also contains 163GB of code spanning 13 programming languages in its 1.6TB compound. As two of the largest open-source pretraining datasets, they have supported many language models with coding ability. GPT-J (Wang & Komatsuzaki, 2021), for example, is reported by Chen et al. (2021b) to demonstrate non-trivial performance on HumanEval, while Scao et al. (2022) report similar results for GPT-NeoX (Black et al., 2022) and BLOOM. LLaMA (Touvron et al., 2023a), whose pretraining dataset includes 328GB code from GitHub, achieves 23.7 pass@1 performance on HumanEval, and its successor LLaMA 2 (Touvron et al., 2023b), achieves an even higher score of 29.9.

Closed-source models, on the other hand, perform generally better. LaMDA (Thoppilan et al., 2022) and PaLM (Chowdhery et al., 2023), whose pre-

Table 1: Pass@k performance of raw language models (top) and language models with extra training on code (bottom) on HumanEval (0-shot) and MBPP (3-shot), ordered chronologically. For Phi-1.5 we consider Phi-1.5-web version, and for Code LLaMA we consider its Python version. [1] Chen et al. (2021b); [2] Chowdhery et al. (2023); [3] Austin et al. (2021); [4] Scao et al. (2022); [5] Touvron et al. (2023a); [6] OpenAI (2023); [7] Bubeck et al. (2023); [8] Touvron et al. (2023b); [9] Li et al. (2023i); [10] Yang et al. (2023a); [11] Bai et al. (2023); [12] Jiang et al. (2023a); [13] Anil et al. (2023a); [14] DeepSeek-AI et al. (2024); [15] Jiang et al. (2024a); [16] Dai et al. (2024a); [17] Anil et al. (2023b); [18] Rozière et al. (2023).

	Size	HumanEval (0)		MBPP (3)	
		k=1	k=100	k=1	k=80
GPT-J[1]	6B	11.6	27.7		
LaMDA[23]	137B	14.0	47.3	14.8	62.4
PaLM[2]	540B	26.2	76.2	36.8	75.0
GPT-NeoX[4]	20B	15.4	41.2		
BLOOM[4]	176B	15.5	55.5		
LLaMA[5]	65B	23.7	79.3	37.7	76.8
GPT-4		67.0[6]/82[7]			
LLaMA 2[8]	70B	29.9	89.0	45.0	81.5
Phi-1.5[9]	1.3B	41.4		43.5	
Baichuan 2[10]	13B	17.1		30.2	
Qwen[11]	14B	32.3		40.8	
Mistral[12]	7B	30.5		47.5	
Gemini[13]	Ultra	74.7			
DeepSeek[14]	67B	42.7		57.4	
Mixtral[15]	8x7B	40.2		60.7	
DeepSeekMoE[16]	16B	26.8		39.2	
Codex[1]	12B	28.8	72.3		
PaLM-Coder[2]	540B	36.0	88.4	47.0	80.8
PaLM 2*[17]	S	37.6	88.4	50.0	86.8
Code LLaMA[18]	34B	53.7	94.7	56.2	
Code-Qwen[11]	14B	45.1		51.4	

[6]While some works refer to this process as "finetuning on code", it is still self-supervised in nature. Thus we choose to adopt the term "extra/additional/continual pretraining" in this work to avoid confusion with supervised in-task finetuning or instruction finetuning.

training dataset contains 12.5% and 5% code respectively, achieve 14.0 and 26.2 pass@1 performance on HumanEval, while GPT-4 (OpenAI, 2023) set a staggering record of 67.0 (and an early version is reported by Bubeck et al. (2023) to be 82) that until recently has remained higher than any specialized models pretrained or instruction-finetuned for code.

More recently, the general trend has been to train smaller models with larger datasets, following the revised scaling law (Hoffmann et al., 2022). Baichuan 2 (Yang et al., 2023a), for example, is a 13B model trained on 2.6T tokens, while Qwen (Bai et al., 2023) is a 14B model trained on 3T tokens. They achieve 17.1 and 32.3 pass@1 on HumanEval, respectively. Li et al. (2023i), however, demonstrate that models as small as 1.3B can acquire coding capability that's comparable to much larger models while also maintaining a reasonable performance on general text processing and even manifesting some emergent abilities (Wei et al., 2022c) such as chain-of-though reasoning (Wei et al., 2022d). Their model, Phi-1.5, is trained on 21B tokens of textbook data generated by ChatGPT, and 100B tokens of filtered web data from Stack Overflow and Refined Web (Penedo et al., 2023), and attains 41.4 pass@1 performance on HumanEval.

Another rising trend is mixture-of-expert models (Lepikhin et al., 2021; Fedus et al., 2022; Du et al., 2022). Notably, Jiang et al. (2024a) recently introduce Mixtral 8x7B, where only 13B parameters are activated for each token during inference, but achieving 40.2 on HumanEval and surpassing much larger dense models such as LLaMA 2. Similarly, Dai et al. (2024a) present DeepSeekMoE, where only 2.8B parameters are activated in a 16B model, scoring 26.8 on HumanEval.

The exact performance of these models are presented in Table 1.

4.2 Language Models with Additional Pretraining on Code

Along with the seminal benchmark HumanEval, Chen et al. (2021b) kick-started the age of LLM for code with Codex, which are GPT-3 checkpoints pretrained on 100B additional code tokens and one of the earliest multi-billion models for code. Following their work, other researchers have also specialized their LLMs on code with additional pretraining. Chowdhery et al. (2023) train PaLM on 7.8B additional code tokens to obtain PaLM-Coder, setting new state-of-the-art on HumanEval and MBPP (Table 1) that are only broken later by its successor PaLM 2-S*, the smallest version of PaLM 2 (Anil et al., 2023b) further trained on an undisclosed amount of code. Similarly, Lewkowycz et al. (2022) train PaLM on 38.5B tokens of arXiv papers and mathematical content, while Rozière et al. (2023) train LLaMA 2 (Touvron et al., 2023b) on more than 500B code tokens to acquire Code LLaMA, whose performance on HumanEval surpasses all previous LMs except GPT-4 (Table 1). Liu et al. (2023b) further train Code LLaMA with multi-task finetuning (MFT) to introduce CodeFuse-CodeLLaMA, achieving 74.4 pass@1 on HumanEval and surpassing even the performance of GPT-4 published in OpenAI (2023).

While almost all of these models are Transformer decoders pretrained with CLM, several architectural modifications have been introduced along the way, as we noted in Section 3.5. All these models use pre-norm, and GPT-J introduces parallel attention, which is later adopted by PaLM, GPT-NeoX, and Phi-1.5. PaLM introduces MQA and RoPE into LLMs, and RoPE is now employed by most language models, including GPT-NeoX, two generations of LLaMA, Qwen, and the 7B version of Baichuan 2. BLOOM and the 13B version of Baichuan 2, however, use ALiBi for position embeddings, while LLaMA 2 and Code LLaMA adopt GQA instead of MHA or MQA. In Section 5, we show that specialized models pretrained exclusively on code have also followed these advancements closely.

5 Specialized Language Models for Code

As pretrained Transformers such as GPT and BERT achieved remarkable success in natural language processing, such model architectures, learning paradigms, and training objectives were soon adopted by the software engineering community to produce specialized models for code understanding and generation. In this section, we first review common datasets used for pretraining code language models (Section 5.1), and then dive into the complex family of code LMs by their model architecture: encoder-only models (Section 5.2), encoder-decoder models (Section 5.3), decoder-only models (Section 5.4), UniLM (Section 5.5),

and diffusion models (Section 5.6). Lastly, in Section 5.7 we also illustrate the current trend of applying more recent techniques in NLP, such as instruction tuning (Wei et al., 2022b; Sanh et al., 2022; Chung et al., 2022) and reinforcement learning (Ouyang et al., 2022) to code processing. An overview of these pretrained models are provided in Table 3.

5.1 Training Dataset for Code

While text data for pretraining language models are often crawled from the web and must undergo meticulous and often aggressive preprocessing (Raffel et al., 2020), code data come naturally as whole documents from public GitHub repositories. Even better, they come with readily available quality indicators such as the count of stars or forks (although Allal et al. (2023) suggest that star count correlates poorly with downstream performance). As a result, many large-scale code pretraining datasets have been introduced, including CodeSearchNet (Husain et al., 2019), CodeParrot (Tunstall et al., 2022), and the Stack (Kocetkov et al., 2023), totaling 20GB, 50GB and 3TB of code documents respectively (Table 2).

Table 2: Statistics of several pretraining datasets for code models: size in bytes, number of files, and number of programming languages. In CodeSearchNet each file is a function. For Pile and ROOTS we only consider their code composite. [a] Husain et al. (2019); [b] Gao et al. (2021); [c] Biderman et al. (2022); [d] `https://huggingface.co/datasets/codeparrot/github-code`; [e] Kocetkov et al. (2023); [f] Laurençon et al. (2022); [g] Lozhkov et al. (2024).

Dataset	Size (GB)	Files (M)	# PL
CodeSearchNet[a]	20	6.5	6
The Pile[bc]	95	19	-
CodeParrot[d]	1K	115	30
The Stack[e]	3136	317	30
ROOTS[f]	163	15	13
The Stack v2[g]	32K	3K	619

While these datasets are meant for training code models, it should be noted that code is ultimately a special form of natural language, as the vocabulary of most programming languages is a small subset of English. Besides, high-quality code is often interleaved with natural language comments or documentations, which also enables models to acquire certain knowledge of general text representation. In fact, of the 6.5M functions in CodeSearchNet, 2.3M are paired with natural language documentation, allowing models to train explicitly on such bimodal data.

Compared with natural language, another byproduct of scraping code from GitHub is commit histories, which consist of code before commit, code after commit, and a short message describing the commit, which can loosely serve as an instruction for language models. Muennighoff et al. (2024) utilize this feature and construct a 2GB dataset CommitPackFT containing 742K samples of instruction data for code, obviating the need of extensive human labor that's required to construct natural language instructions (Sanh et al., 2022; Wang et al., 2022g).

Apart from bimodal training and instruction finetuning, another recent trend in constructing code dataset is synthesizing data with powerful models such as ChatGPT. While this method is originally proposed for generating instruction data in natural language (Wang et al., 2023g; Honovich et al., 2023), Gunasekar et al. (2023) take one step further and synthesize 1B tokens of Python textbooks and coding exercises to pretrain a 1.3B model, achieving state-of-the-art results on HumanEval that's comparable to much larger models trained on significantly larger datasets.

Table 3: An overview of pretrained code language models' architecture and training details: their base architecture, model size, vocabulary, context length, position embedding, training precision, attention type (MHA, MQA, or GQA), layer normalization type (post-norm or pre-norm), usage of FlashAttention, training initialization, objectives, dataset size (either in disk size, measured by GB/TB, or in number of tokens, measured by B/T), tokens seen during training, supported number of programming languages, and institute. We note that the number of training tokens does not count the training tokens of the model used for initialization, if any. The training objectives are: MLM (Masked Language Modeling), NSP (Next Sentence Prediction), RTD (Replaced Token Detection), IP (Identifier Prediction), CL (Contrastive Learning), SC (Span Corruption), DAE (Denoising Auto-Encoding), Text↔Code (text-to-code generation and code-to-text generation), MT (Machine Translation). Missing information (such as AlphaCode's position embedding type) is left as blank.

Date	Model	Arch.	Size	Vocab	Context	PE	Precision	Atten. Type	Parallel Atten.	Pre-Norm	Flash Atten.	Init. from	Objectives	Dataset	Training	PL	Inst.
2019-12	CuBERT	BERT	350M	50K	1024	absolute	fp32	MHA				-	MLM + NSP	9.3B	93B	1	Google
2020-02	CodeBERT	RoBERTa	125M	50K	512	absolute	fp32	MHA				RoBERTa	MLM + RTD	20GB	105B	6	Microsoft
2020-09	GraphCode-BERT	RoBERTa	125M	50K	640	absolute	fp32	MHA				CodeBERT	MLM + Edge Prediction + Node Alignment	20GB	131B	6	Microsoft
2021-08	SynCoBERT	RoBERTa	125M	50K	512	absolute	fp32	MHA				CodeBERT	MLM + IP + AST Edge Prediction + CL	20GB	7B	6	Huawei
2021-10	DISCO	BERT	100M	20K	512	absolute	fp32	MHA				-	MLM + Node Type MLM + CL	1.8GB		2	Columbia & IBM
2022-05	Code-MVP	RoBERTa	125M	50K	512	absolute	fp32	MHA				GraphCodeBERT	MLM + Type Inference + CL	2GB	39B	1	Huawei
2022-10	SCodeR	RoBERTa	125M	51K	1024	absolute	fp32	MHA				UniXcoder	CL	20GB		6	Microsoft
2020-05	GPT-C	GPT-2	374M	60K	1024	absolute		MHA		✓		-	CLM	11B	270B	4	Microsoft
2021-02	CodeGPT	GPT-2	124M	50K	1024	absolute	fp32	MHA		✓		GPT-2	CLM	2GB		1	Microsoft
2022-02	PolyCoder	GPT-NeoX	160M-2.7B	50K	2048	RoPE	fp32	MHA		✓		-	CLM	254GB	39B	12	CMU
2022-03	CodeGen-Multi(Mono)	GPT-J	350M-16.1B	50K	2048	RoPE	fp16	MHA	✓	✓		-	CLM	1.6TB(1.8TB)/ 506B(577B)	1T(1.2T)	6(1)	Salesforce
2022-04	InCoder	GPT-3	6.7B	50K	2048	Cosine	fp32	MHA		✓		-	Causal Masking	204GB	52B	28	Meta
2022-06	PyCodeGPT	GPT-Neo	110M	32K	1024	absolute	fp32	MHA		✓		-	CLM	96GB	100B	1	Microsoft
2022-07	PanGu-Coder	PanGu-α	317M-2.6B	42K	1024	absolute		MHA		✓		-	CLM	147GB	230B	1	Huawei
2023-01	SantaCoder	GPT-2	1.1B	49K	2048	absolute	fp32	MQA		✓		-	FIM	268GB	236B	3	BigCode
2023-03	CodeGeeX	PanGu-α	13B	52K	2048	absolute	fp16	MHA		✓		-	CLM	158B	850B	23	Tsinghua
2023-05	StarCoder	GPT-2	15.5B	49K	8192	absolute	fp32	MQA		✓	✓	-	FIM	815GB	1T	86	BigCode
2023-05	Jam	GPT-2	350M	50K	256	absolute	bf16	MHA		✓		-	CLM	36GB/20B	9B	1	U. Notre Dame
2023-06	Phi-1	GPT-J	1.3B	51K	2048	RoPE	fp16	MHA	✓	✓	✓	-	CLM	7B	53B	1	Microsoft
2023-10	CodeFuse	GPT-J	350M-13B	101K	4096	RoPE	fp16	MHA	✓	✓	✓	-	CLM	1.6TB / 1T		40+	Ant Group
2023-10	CodeShell	GPT-2	7B	70K	8192	RoPE	bf16	GQA		✓		-	FIM	100B	500B		Peking U.
2020-10	PyMT5	GPT-2	374M	50K	1024+1024	absolute	fp16	MHA		✓		-	SC	27GB		1	Microsoft
2021-02	Mastropaolo et al.	T5	60M	32k	512+512	T5	bf16	MHA		✓		-	SC	1GB		1	USI
2021-02	DOBF		250M	50K	512+512	absolute	fp16	MHA				-	MLM + Deobfuscation	45GB		2	Meta
2021-03	PLBART	BART	140M	50K	1024+1024	absolute	fp32	MHA				-	DAE	655GB / 71B	210B	2	UCLA & Columbia
2021-09	CodeT5	T5	60M-220M	32K	512+256	T5	fp16	MHA		✓		-	SC + IP + Masked IP + Text↔Code	∼25GB		8	Salesforce
2022-01	SPT-Code	BART	262M	80K	512+512	absolute	fp16	MHA				-	NSP + SC + Method Name Prediction	20GB		6	Nanjing U.
2022-02	AlphaCode		300M-41B	8K	1536+768		bf16	MQA				-	MLM + CLM	715GB	967B	13	DeepMind
2022-06	NatGen	T5	220M	32K	512+256	T5	fp16	MHA		✓		CodeT5	Naturalization	∼26GB	14B	8	Columbia & UCD
2022-12	ERNIE-Code	mT5	580M	250K	1024+1024	T5	bf16	MHA		✓		mT5	SC + Text↔Code + MT		197B	6	Baidu
2023-05	CodeT5+	T5/GPT-3	220M-16B	50K	2048+2048	absolute	fp16	MHA	✓	✓		CodeGen-mono	SC + CLM + CL + Text↔Code	52B		9	Salesforce
2020-12	CugLM	BERT	51M	50K	128	absolute	fp32	MHA				-	MLM + NSP + CLM	8M	1.2B	2	Peking U.
2022-03	UniXcoder	RoBERTa	125M	51K	1024	absolute	fp32	MHA				-	MLM + CLM + SC + CL + Code2Text	20GB+	839B	6	Microsoft

5.2 Encoders

Pretrained Transformer encoders such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019d), and ELECTRA (Clark et al., 2020) have attained impressive results on natural language understanding tasks, and these methods were soon introduced into code processing after their advent. Kanade et al. (2020) replicate the training procedure of BERT on a code corpus to produce CuBERT, showcasing its superior performance over LSTM (Hochreiter & Schmidhuber, 1997) and non-pretrained Transformers. Feng et al. (2020), on the other hand, train CodeBERT with MLM and ELECTRA's RTD on CodeSearchNet. They also utilize the explicit text-code pairs in CodeSearchNet, and use them respectively as the first and second segment in BERT's input. When using CodeBERT to initialize the encoder part of a vanilla Transformer for sequence-to-sequence generation tasks such as code summarization, they observe a moderate performance gain over non-pretrained baselines.

Apart from these standard training objectives, many auxiliary objectives specifically designed for code have also been introduced. GraphCodeBERT (Guo et al., 2021a) and SynCoBERT (Wang et al., 2021d) both extract graphs from the source code (data flow graph and abstract syntax tree, respectively) and train the models to predict the typological relations between the nodes, while SynCoBERT and Code-MVP (Wang et al., 2022e) also add type inference to their pretraining stage in the form of tagging. Another common objective is contrastive learning: SynCoBERT and Code-MVP contrast between different views of the input (such as code, comment, AST, and transformed code), while DISCO (Ding et al., 2022a) constructs positive sample pairs by semantic-preserving transformations such as obfuscation, and negative pairs by injecting artificial bugs.

5.3 Encoder-Decoders

In NLP, pretrained Transformer encoder-decoders such as T5 (Raffel et al., 2020) and BART (Lewis et al., 2020) have also left a notable mark in the past few years' advancement in language modeling. T5, for example, unifies all textual tasks into a sequence to sequence format and sets new records on GLUE (Wang et al., 2018a) and SuperGLUE (Wang et al., 2019). Compared with encoder-only models, encoder-decoders are naturally more powerful as they can be used for conditional text generation, while their encoder part can always be taken out to perform tasks that require an encoder-only architecture, such as regression (Tay et al., 2023b).

Inspired by these advantages of encoder-decoder architecture, many such models have been proposed for code processing. PyMT5 (Clement et al., 2020) and Mastropaolo et al. (2021) replicate the pretraining and multi-task finetuning process of T5 on code corpus, while Ahmad et al. (2021) introduce PLBART, a BART pretrained on 655GB combined data of Java, Pyhton, and natural language. Lachaux et al. (2021) argue that MLM could be too easy a task for programming languages as identifier names often occur multiple times in a single context window, and propose a deobfuscation pretraining objective, where the model is trained to convert obfuscated code back to its original form. Related to this method, we note that meaningful variable names have also been found to have a positive impact on the code generation process of large language models (Chen et al., 2023e).

Building on these early works, Wang et al. (2021f) propose CodeT5, which is pretrained alternatively with 1) T5's original span corruption; 2) identifier tagging (where each token in the code input is tagged as either identifier or non-identifier); 3) masked identifier prediction (a special form of span corruption where all identifiers are masked); and 4) text-to-code & code-to-text generation. Its successor, CodeT5+ (Wang et al., 2023h), take inspiration from UL2 (Tay et al., 2023b) and introduce causal language modeling (CLM) into pretraining, along with additional contrastive objectives based on text-code matching.

AlphaCode (Li et al., 2022h) is also trained with multiple objectives, where the encoder is trained with MLM and the decoder is trained with CLM, with architecture modifications such as shallow-encoder & deep-decoder, multi-query attention (Shazeer, 2019), and being much larger than CodeT5 (up to 41B parameters). NatGen (Chakraborty et al., 2022a), on the other hand, is pretrained with a "naturalization" objective similar to deobfuscation: semantically equivalent but unnatural code is generated by predefined operations such as loop transformation, dead code injection, and variable renaming, and the model is pretrained to translate

these unnatural code back to its original form. We note that some of these models are built on previous works. For example, NatGen is initialized with CodeT5, while the largest version of CodeT5+ is initialized from a decoder-only model, CodeGen (Nijkamp et al., 2023b).

Apart from these general pretraining objectives, several works have also trained Transformer encoder-decoders with a focus on code translation, which is a natural application of Transformer models in code as the Transformer architecture was originally proposed by Vaswani et al. (2017) for machine translation (MT). However, unlike natural languages, where parallel corpus across two or more human languages exist in abundance, there is little parallel data for code. To tackle this issue, Rozière et al. (2020) propose Transcoder, which first pretrains an encoder with XLM (Conneau & Lample, 2019), and then initializes a vanilla Transformer with this encoder and continue to pretrain it with Denoising Auto-Encoding (DAE, Lewis et al., 2020) and back translation (Sennrich et al., 2016), while its follow-up work (Szafraniec et al., 2023) also utilize language-independent intermediate representations to enhance this process, which we discuss in more detail in Section 6.

Apart from training data and objectives, these models mostly keep to the original architectures proposed by the NLP community, as shown in Table 3. Models based on BART, for example, use post-normalization and learnable absolute position embeddings, while those based on T5 use its simplified relative position embeddings and pre-normalization.

Table 4: Pass@1 performance of pretrained code models (top), instruction finetuned code models (middle), in comparison with some of the best general language models (bottom), with models in each category ordered chronologically. The sources of these figures can be found in Section 5.3, Section 5.4, and Table 1.

Model	Size	HumanEval	MBPP
PolyCoder	2.7B	5.6	-
CodeGen-Mono	16.1B	29.3	35.3
InCoder	6.7B	15.2	19.4
PyCodeGPT	110M	8.3	-
Pangu-Coder	2.6B	23.8	23.0
SantaCoder	1.1B	14.0	35.0
CodeGeeX	13B	22.9	24.4
StarCoder	15.5B	33.6	52.7
CodeT5+	16B	30.9	-
Phi-1	1.3B	50.6	55.5
CodeFuse	13B	24.8	-
DeepSeek Coder	33B	56.1	66.0
InstructCodeT5+	16B	35.0	-
WizardCoder	15.5B	57.3	51.8
Pangu-Coder 2	15.5B	61.6	-
OctoCoder	15.5B	46.2	-
CodeFuse	34B	74.4	-
DeepSeek Coder-Instruct	33B	79.3	70.0
GPT-4	-	67.0/82	-
PaLM 2*	S	37.6	50.0
Code LLaMA	34B	53.7	56.2
Phi-1.5	1.3B	41.4	43.5

5.4 Decoders

After the monumental debut of GPT-3 (Brown et al., 2020) and the discovery of in-context learning, decoder-only Transformer models have become dominant in language modeling (Rae et al., 2021; Hoffmann et al., 2022; Chowdhery et al., 2023; Scao et al., 2022; Touvron et al., 2023a;b, *inter alia*). Many models similarly pretrained with CLM have also emerged in code processing, such as GPT-C (Svyatkovskiy et al., 2020), CodeGPT (Lu et al., 2021), PolyCoder (Xu et al., 2022), CodeGen (Nijkamp et al., 2023b), PyCodeGPT (Zan et al., 2022), Pangu-Coder (Christopoulou et al., 2022), CodeGeeX (Zheng et al., 2023a), Jam (Su et al., 2023a), Phi-1 (Gunasekar et al., 2023), and CodeFuse (Di et al., 2023). Of these models, several alternative training objectives have been experimented with, such as MLM and Masked CLM[7] in Pangu-Coder, but are found to underperform compared with CLM-only training. Zan et al. (2022) also propose continual training on sketches, where the model learns to first generate a sketch of a program and then the actual code. Notably, Gunasekar et al. (2023) present Phi-1, a 1.3B small model trained on a dataset of only 7B tokens consisting of 6B tokens from StackOverflow and 1B synthetic data generated by ChatGPT but achieving 50.6 pass@1 on HumanEval and 55.5 pass@1 on MBPP, comparable to much larger (both in model size and training data size) models such as Code LLaMA or PaLM 2.

[7]In their paper, MLM is conducted by replacing tokens in the input with `<mask>` and predicting it from only the left context, while Masked CLM is performed by adding a `<mask>` in the input and predicting the next token from it. Both tasks do not change the attention mask patterns of the model.

Although Christopoulou et al. (2022) report denoising objectives to underperform in decoder-only models, there have been other works that successfully combine denoising or multi-task pretraining with decoder architecture. Incoder (Fried et al., 2023), SantaCoder (Allal et al., 2023), StarCoder (Li et al., 2023h), DeepSeek Coder (Guo et al., 2024), and CodeShell (Xie et al., 2024) are trained with fill-in-the-middle (FIM) objective, also referred to as causal masking by Fried et al. (2023), which is essentially span corruption (Raffel et al., 2020) adopted to decoder-only architecture. One of the visible advantages of these infilling objectives is that they inject the models with the ability to fill in blanks in the middle of input code at inference time, while CLM allows only for autoregressive generation. As Table 4 shows, however, these objectives also lead to higher performance on downstream tasks when compared with CLM-only models such as CodeGen, although the exact benefits of infilling training remain controversial (Nijkamp et al., 2023a).

Observing Table 3, it is clear that decoder-only models for code have generally followed the practices in NLP more closely, when compared with other model architectures. All these models use pre-normalization, while MQA, RoPE, and parallel attention have also been adopted by several models. Notably, the three most recent models - StarCoder, Phi-1, and CodeFuse - also employ FlashAttention to improve model throughput.

5.5 UniLMs

Following UniLM (Dong et al., 2019) in NLP, several works in code processing have also pretrained this fourth family of Transformer models on code. CugLM (Liu et al., 2020) is trained with both CLM and MLM + NSP via alternating attention masks, while UniXcoder is trained with CLM, MLM, Span Corruption (in Prefix LM style) along with auxiliary objectives including contrastive learning and text-code mutual generation. Both two models, however, are relatively small in size, and whether or not this architecture is suitable for code processing is yet to be explored at scale.

5.6 Diffusion Models

Currently the Transformer architecture dominates text generation, but several works (Li et al., 2022d; Lin et al., 2023) have also adopted Diffusion Models (Ho et al., 2020) from computer vision for text generation. Recently CodeFusion (Singh et al., 2023) also introduces diffusion models into code modeling, and demonstrates that a 75M diffusion model can outperform StarCoder, CodeT5+, and GPT-3 on three code synthesis datasets.

5.7 Instruction Finetuning and Reinforcement Learning for Code

In natural language processing, training models on a diverse set of tasks with instruction prefix, known as instruction finetuning, has been shown to unlock the ability of cross-task generalization (Ouyang et al., 2022; Chung et al., 2022; Iyer et al., 2022). At first, these instruction data samples are manually compiled or crowd-sourced (Wei et al., 2022b; Sanh et al., 2022), but later researches find LLM-generated instructions to be sufficient (Wang et al., 2023g; Honovich et al., 2023).

Following these works in natural language, researchers from the code community have applied instruction tuning to their models as well. Wang et al. (2023h) finetune CodeT5+ with 20K instruction data generated by InstructGPT (Ouyang et al., 2022) to obtain InstructCodeT5+. WizardCoder (Luo et al., 2023) follows the methods of WizardLM (Xu et al., 2024) to evolve 20K code Alpaca (Taori et al., 2023) samples into a 78K dataset and uses it to finetune StarCoder. Pangu-Coder 2 (Shen et al., 2023) also uses WizardLM's Evol-Instruct to generate 68K instruction samples from 20K code Alpaca, but also introduces reinforcement learning via Rank Responses to align Test & Teacher Feedback (RRTF). OctoCoder (Muennighoff et al., 2024), on the other hand, takes a different path and uses Git commit histories as instruction data to finetune StarCoder and CodeGeeX2. More recently, CodeFuse (Liu et al., 2023b) also employs multitask-finetuning and explicitly introduces multiple downstream tasks into their instruction data. The performance of these instruction finetuned code models can also be found in Table 4.

In NLP, another technology closely related to instruction finetuning is reinforcement learning from human feedback (RLHF), which has played a significant role in aligning LLMs with human values (Ouyang et al., 2022; Bai et al., 2022). The merit of reinforcement learning is that it can incorporate non-differentiable

reward signals into training, such as BLEU (Bahdanau et al., 2017) and human preference (Christiano et al., 2017), but the human feedback required in aligning LLMs often involves extensive labor on annotation. In comparison, applying reinforcement learning to code models has a natural advantage, as compilers can be used for automatically generating feedback for code samples produced by language models.

CodeRL (Le et al., 2022) is one such model, which defines four levels of rewards for each generated program (viz. compile error, runtime error, unit test failure, pass) as well as fine-grained token-level reward estimated by a critic model. The actor model, which is an extension of CodeT5, is then trained with REINFORCE algorithm (Williams, 1992). Similarly, CompCoder (Wang et al., 2022d) and PPOCoder (Shojaee et al., 2023) train CodeGPT and CodeT5 respectively with proximal policy optimization (Schulman et al., 2017), while RLTF (Liu et al., 2023d) proposes fine-grained feedback based on the error information and location provided by the compiler, as well as adaptive feedback that takes the ratio of passed test cases into account.

6 Code Features for Language Models

A major difference between programming languages and natural languages is that the former is artificially defined to be precise and unambiguous, and need to be compiled (or interpreted) without error before execution. This allows for a much larger flexibility in designing pretraining objectives on code, beside lexical manipulations such as CLM, MLM, and Span Corruption. A similar trend can be observed in the last years before neural networks were introduced into mainstream NLP literature (Sutskever et al., 2014; Bahdanau et al., 2015), when researchers in the MT community utilized alternative views of text such as syntactic features to improve the performance of SMT systems (Galley et al., 2006; Chiang, 2007). These features, however, are not universally applicable or even agreed upon, and often result in highly complicated systems (for example, the size of English part-of-speech tagging's label set may range from dozens to hundreds).

Programming languages, however, fare much better in these aspects. Each mainstream programming language, such as C, Python, and Java, comes with readily available compiler toolkits that allow for easy and accurate extraction of semantic information such as Abstract Syntax Tree (AST), language-independent Intermediate Representation (IR), and auxiliary information such as type of each token and control/data flow graph (CFG/DFG). Thus, in the context of Transformer-based language modeling for code, many works have incorporated these features into their training procedure.

6.1 Abstract Syntax Tree and Intermediate Representation

AST is one of the most common intermediate results of the compiling process, where a program is parsed into a tree of operations and their operands. Before the popularization of Transformer in the code processing community, there had been works such as InferCode (Bui et al., 2021a) that processes these representations with special network architectures like Tree-Based CNN and conducts self-supervised pretraining by predicting subtrees.

TreeBERT (Jiang et al., 2021b) is one of the first attempts to take AST into the Transformer-based pretraining-finetuning framework. It's a Transformer encoder-decoder pretrained with Tree MLM and Node Order Prediction, where the encoder takes a set of constituent paths in the AST as input (with each token being a path, which is the concatenation of its nodes' representations) while the decoder takes the code as input. Tree MLM is then performed by masking certain nodes in a path representation and its corresponding code tokens in the decoder input, while Node Order Prediction is accomplished by swapping nodes in a path and predicting it with a `[CLS]` token similar to BERT.

The method used by TreeBERT, however, is complicated and does not scale well. Later works mostly opt to first process AST into a text sequence and treat it like a normal part of the input. Wang et al. (2021d), for example, process AST with depth-first traversal and concatenate it with code and comment, and then train SynCoBERT (which, unlike TreeBERT, is actually a BERT-like encoder-only model) with four objectives: 1) MLM; 2) identifier tagging; 3) AST edge prediction (predicting whether there exists an edge between two AST nodes from the dot product of these nodes' representations); and 4) contrastive learning over i) code and AST pairs, as well as ii) text and code-AST pairs. Similarly, SPT-Code (Niu et al., 2022), a Transformer encoder-decoder, takes the concatenation of code, sequentialized AST, and text as input, and

is pretrained with 1) span corruption; 2) code-AST prediction (NSP with one segment being code and one segment being AST); and 3) method name generation, a special form of span corruption where a method name is masked. Different from other works, however, they do not take the docstrings as the text segment in their input, but instead concatenate all method names appearing in the code as a succinct natural language description. Likewise, UniXcoder (Guo et al., 2022) takes flattened AST instead of source code as its input during training.

In the compiling pipeline, AST is usually followed by language-independent intermediate representations, such as LLVM IR (Lattner & Adve, 2004). Such features' independence from specific programming languages makes them suitable candidates for translation pivots, as is English in machine translation of low-resource natural languages (Leng et al., 2019). Szafraniec et al. (2023) take advantage of this characteristic and extend Transcoder (Rozière et al., 2020) with translation language modeling (Conneau & Lample, 2019) over code and IR, as well as IR generation from code. They also investigate other objectives such as IR decompilation (i.e. generating code from IR) and IR pivot (i.e. directly generating code in one language from the IR of another language), both showing promising results.

6.2 Control Flow and Data Flow

While AST and IR have proved to be useful information in certain tasks such as code translation, they are static by nature, just like the source code, and may fail to capture semantic properties of code that are only revealed at runtime (Wang & Su, 2020). Such semantics, however, are contained in dynamic features such as control flow and data flow. Similar to AST, specialized networks were used to process such information before the rise of pretrained Transformers, such as Message Passing Neural Network used by ProGraML (Cummins et al., 2021). Unlike AST, however, even after pretrained Transformers became dominant few works have looked in this direction.

GraphCodeBERT (Guo et al., 2021a) is one of such works, which creates special tokens and position embeddings for variables in the flow graph, and concatenates the variable sequence after text and source code to construct model input, with tailored attention masks on the code and variable segments: tokens from code segment and variable segment can attend to each other if and only if the variable is identified from the code token, and for tokens within the variable segment, v_i is allowed to attend to v_j if there is a direct edge from v_j to v_i in the dataflow. The model is then pretrained with MLM in combination with edge prediction and node alignment, both of which are accomplished by binary classification from the dot product of two tokens' representations (one from code segment and one from variable segment for node alignment, and both from variable segment for edge prediction).

6.3 Type

Apart from AST, IR, and data flow, type information has also been used to aid language models in processing code. CugLM (Liu et al., 2020), for example, uses type information during finetuning to aid in the prediction of tokens for unidirectional MLM (i.e. MLM with unidirectional attention mask): the type of a masked token is first predicted from the final Transformer layer's representation, and then the token itself is predicted based on both the hidden representation and predicted type. In contrast, both CodeT5 (Wang et al., 2021f) and SynCoBERT (Wang et al., 2021d) include identifier tagging in their pretraining objectives, which can be viewed as coarse-grained type prediction.

6.4 Program Transformation

As we have shown in Section 5.3, function-preserving program transformations have proven to be important techniques in pretraining code language models. Obfuscation is one instance of program transformation, and others include loop transformation (for-while), condition transformation (if-switch), dead code injection (e.g. `if True: pass`), and statement swapping. DOBF (Lachaux et al., 2021) and NatGen (Chakraborty et al., 2022a) are two code language models pretrained to recover the original program from such transformations, while Wang et al. (2022a) also apply program transformations during the finetuning stage of language models to make them more robust to transformed test samples.

Notably, Wang et al. (2022e) integrate many of the aforementioned features into Code-MVP: source code, docstrings, AST, CFG, and transformed source code via identifier renaming, loop exchange, and dead code insertion. The model, initialized from GraphCodeBERT, is then trained with MLM, fine-grained type prediction, and contrastive learning across different views, such as text vs. code, code vs. AST, and code vs. CFG.

7 LLMs in Software Development

As language models set new records on software engineering benchmarks, software engineering technologies are also expanding the boundaries of language models in return, and have subsequently led them into real-world development cycles.

7.1 LLMs Extended with Coding Tools

Research in the NLP community has shown that LLMs can learn to use external tools such as calculators, MT systems, and search engines (Thoppilan et al., 2022; Schick et al., 2023). As such, *interpreter* has been used to augment LLMs in complex reasoning tasks. PAL (Gao et al., 2023b) and PoT (Chen et al., 2023e) both extend Codex with Python interpreters for numerical calculations, while ViperGPT (Surís et al., 2023) extends it further by calling vision APIs to extract information from visual input and answer related questions.

However, LLMs do not always produce code that can be interpreted and executed, and there has also been a line of works that explore the possibility of emulating the interpreter with LLMs themselves. Nye et al. (2021) trains LLMs to emulate the execution of programs by outputting program states at each step. More recently, Li et al. (2023a) propose Chain-of-Code, where a real interpreter executes the generated code until an error occurs, whereupon the LLM takes over to simulate execution. Chae et al. (2024) similarly propose Think-and-Execute framework, where an LLM generates pseudo code and then simulates its execution to solve reasoning tasks.

Apart from alleviating the burden of numerical calculation in abstract reasoning tasks, interpreter (together with unit tests) also provides feedback on the process of code generation itself, allowing for interactive generation and refinement of code. Such works include Self-Edit (Zhang et al., 2023c), LeTI (Wang et al., 2023f), OpenCodeInterpreter (Zheng et al., 2024), ProCoder (Bi et al., 2024), Cycle (Ding et al., 2024), and SOAP (Huang et al., 2024), which run model-generated code against unit tests to provide feedback for further refinement. Alternatively, CodeT (Chen et al., 2023a), TiCoder (Bareiß et al., 2022), and CONLINE (He et al., 2024) also utilize the LLM itself to generate unit tests, while Self-Refine (Madaan et al., 2023) sends the generated code to an LLM instead of an interpreter for feedback. Zhou et al. (2023a) show that OpenAI's interpreter plugin[8] allows GPT-4 to self-debug, while InterCode (Yang et al., 2023d) provides a benchmark for evaluating interactive coding. In Section 5.7 we have also shown that the execution results on unit tests serve as natural supervision signals for reinforcement learning on code.

A topic closely related to tool using in LLM research is *planning* as intelligent agents, which has been shown to enhance LLMs' capability both theoretically and empirically (Feng et al., 2023a). Ruan et al. (2023) find that LLMs can plan to solve complex tasks using external SQL generators and Python generators, while CodePlan (Bairi et al., 2023) demonstrates they can perform repository-level coding via adaptive planning.

Another stream of works use LLMs to create multi-agent systems for code generation, such as self-collaboration (Dong et al., 2023), ChatDev (Qian et al., 2023), MetaGPT (Hong et al., 2023), LCG (Lin et al., 2024), MAGIS (Tao et al., 2024), and SoA (Ishibashi & Nishimura, 2024). In these frameworks, multiple LLMs are prompted to play distinct roles such as programmer, reviewer, and manager. These roles interact with each other, breakdown code generation into different phases (e.g. designing, coding, testing, and documenting), and collaborate to complete complex tasks.

[8] `https://openai.com/blog/chatgpt-plugins#code-interpreter`

7.2 LLMs Integrated into Software Development

With the increase in LLMs' interactive coding capability, researchers have also started to integrate them into each and every process of software development.

Auto code completion is one of the earliest applications of language models in software development, as they require only the ability to predict the next token. Even before language models scaled to billions of parameters, there had been integration of completion systems such as Pythia (Svyatkovskiy et al., 2019) and IntelliCode (Svyatkovskiy et al., 2020) into popular IDEs.

Recently, however, the application of code language models have transcended simple code completion. GitHub Copilot is arguably one of the most popular AI code assistants, with diverse features including code generation, vulnerability detection, and license management[9], while CodeFuse (Di et al., 2023) also integrates code generation, code translation, code commenting, and testcase generation into a single IDE extension. As code language models become larger, however, their client-side deployment and real-time performance also raise new challenges.

As LLMs continue to advance, building applications on top of them is also evolving into a consequential task itself. Many open-source frameworks for such applications have been released, including LangChain[10], AutoGPT[11], and WorkGPT[12]. These frameworks provide abstractions over language models for developers, and are actively revolutionizing the entire process of software development even as this survey is being finalized.

7.3 Analysis of LLM-Generated Code

As AI code assistants become prevalent, many recent works have also focused on examining AI-generated code from different aspects, including correctness (Nguyen & Nadi, 2022; Yetistiren et al., 2023), bugs (Jesse et al., 2023; Tambon et al., 2024), vulnerabilities (Asare et al., 2023; Sandoval et al., 2023; Perry et al., 2023; Hamer et al., 2024), syntactic robustness (Sarker et al., 2024), efficiency (Niu et al., 2024), and hallucinations (Liu et al., 2024). Asare et al. (2023) and Sandoval et al. (2023)'s studies show that AI code assistants do not introduce extra security risks compared with human programmers, and Hamer et al. (2024) also find that ChatGPT's responses to security-related questions contain less vulnerabilities than answers from StackOverflow. Contrarily, Perry et al. (2023) find that users write significantly less secure code when assisted by AI. In terms of code complexity, Nguyen & Nadi (2022) find GitHub Copilot to generate low-complexity code with no statistically significant differences between languages, whereas Liu et al. (2023n) find ChatGPT to generate the most complex code in C and the least complex code in Python. Overall, AI code assistants are still in their nascent stage and constantly evolving, and their implications on software development are yet to be investigated systematically.

8 Conclusion and Challenges

In this work, we systematically reviewed the history of pretrained Transformer language models in code processing and other software engineering tasks. The advancement in code modeling generally follows the history course of NLP, evolving from SMT models, to NMT models, and then to finetuning pretrained Transformers and lastly to few-shot application of LLMs and even autonomous agents in real-world production. Unlike natural languages, the nature of code makes it easy to extract auxiliary information from alternative views, and to utilize interpreter and unit tests for automatic feedback.

With these in mind, we identify several challenges in the current development of code modeling.

- **More comprehensive and challenging benchmarks**. The widely used HumanEval benchmark plays a key role in the evolution of Code LLMs. However, it is relatively small and its scoreboard has been manipulated to near perfect, and the community is eager for a new standard to evaluate LLMs. HumanEval and

[9] https://github.com/features/copilot
[10] https://www.langchain.com/
[11] https://github.com/Significant-Gravitas/AutoGPT
[12] https://github.com/team-openpm/workgpt

other similar benchmarks focus on generating standalone Python functions from well-structured docstrings, which do not reflect real-world user behaviors. In real-world scenarios, software requirements are seldom condensed into a single docstring, and LLMs must learn to communicate effectively for clarifications when uncertain about the requirements (Wu & Fard, 2024). Also, standalone functions are hardly used in production, where software has complex dependencies within a repository. Thus, next-generation benchmarks should also take such cross-file context into account, and recent benchmarks such as SWE-bench (Jimenez et al., 2023) represent promising efforts in this direction. Contrary to the models' perfect scores on HumanEval, these repository-level benchmarks expose LLMs' limits in real-world applications, with the most recent SWE-agent resolving only 12.5% real-world GitHub issues in SWE-bench (Yang et al., 2024b).

- **Evaluation and application of LLMs beyond traditional code generation**. As we have pointed out in Section 2, current evaluation of LLMs' coding capability in the NLP community is focused on code generation, while overlooking other activities in software engineering, such as software modeling and testing. From an application point-of-view, the most widespread application of LLMs in software engineering is IDE plugins, which provide developers with code suggestions, while other stages in software development that we mentioned in Section 2 are also largely overlooked. However, it should be also noted that LLMs trained on code are highly specialized to reasoning-heavy tasks such as programming and math, and non-coding tasks in SE - such as requirement elicitation, product management, and marketing - are probably better suited for general-domain models. Beyond the previously mentioned text-based evaluation and applications, the recent advancement of multimodel LLMs also created new opportunities, especially for UI design and other activities that involve visual input, leading to novel tasks such as visually grounded code generation (Li et al., 2024b; Wu et al., 2024a), webpage reverse engineering (Si et al., 2024; Laurençon et al., 2024), and layout design (Feng et al., 2023b).

- **Acquisition of high-quality data**. With Gunasekar et al. (2023) achieving SOTA performance with a 1.3B model trained on textbook data, we believe the selection of training data and utilization of synthetic data will be ever more prominent in the near future, for both self-supervised pretraining and supervised finetuning.

- **Integration of code features into language models**. As we noted in Section 6.2, CFG and DFG are yet to be employed at scale in code language modeling. The few works that do employ data flow make changes to the models' attention masks, which severely limits their cross-task generalization and scaling ability. We believe the seamless integration of such features into textual input is worth researching in the future.

- **Beyond the imperative programming paradigm**. In Section 2.1.7 we have shown that most of the current research on code LLM focus on popular imperative programming languages such as C and Python, while neglecting declarative and functional languages except SQL. However, the paradigm of declarative and functional languages makes them more aligned to natural languages, and thus is worth more research attention in the future.

- **Alternative model architectures and training objectives**. In Table 3, we have shown that many code language models are pretrained with auxiliary objectives specific to code, but these models all belong to the encoder-only or encoder-decoder family, while decoder-only models are yet to be augmented with alternative objectives. Also, as pioneered by Singh et al. (2023), we believe diffusion models will find its ground in code modeling in the future.

- **Safety and ethics issues related to code LLMs**. As language models grow in might, they also raise safety concerns including but not limited to data contamination, toxic or biased generation, personal information leak, and hallucinations. In software development, these models should be deployed with extra caution, as their generated code may contain security risks leading to catastrophic results. Pretraining data is also becoming a sensitive topic of ethics, and Kocetkov et al. (2023) take a meaningful step towards this issue by allowing developers to remove their code from the Stack. As synthetic training data becomes widespread, researchers should also proceed with caution about such practice, as the consequence of training AI models with AI generated data is yet to be investigated at scale.

With the presentation of this survey, we hope to provide a global view of language models' application in software engineering and connect the research from the two communities. We believe the current surge of LLMs will be ultimately transformed into real-world applications, and lead humanity into a brighter future.

References

Esra A. Abdelnabi, Abdelsalam M. Maatuk, Tawfig M. Abdelaziz, and Salwa M. Elakeili. Generating uml class diagram using nlp techniques and heuristic rules. In *2020 20th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pp. 277–282, 2020. doi: 10.1109/STA50679.2020.9329301.

Esra A. Abdelnabi, Abdelsalam M. Maatuk, and Mohammed Hagal. Generating uml class diagram from natural language requirements: A survey of approaches and techniques. In *2021 IEEE 1st International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering MI-STA*, pp. 288–293, 2021a. doi: 10.1109/MI-STA52233.2021.9464433.

Esra A. Abdelnabi, Abdelsalam M. Maatuk, and Mohammed Hagal. Generating uml class diagram from natural language requirements: A survey of approaches and techniques. In *2021 IEEE 1st International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering MI-STA*, pp. 288–293, 2021b. doi: 10.1109/MI-STA52233.2021.9464433.

Marah I Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat S. Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Caio César Teodoro Mendes, Weizhu Chen, Vishrav Chaudhary, Parul Chopra, Allie Del Giorno, Gustavo de Rosa, Matthew Dixon, Ronen Eldan, Dan Iter, Amit Garg, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Jamie Huynh, Mojan Javaheripi, Xin Jin, Piero Kauffmann, Nikos Karampatziakis, Dongwoo Kim, Mahoud Khademi, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Chen Liang, Weishung Liu, Eric Lin, Zeqi Lin, Piyush Madan, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Xia Song, Masahiro Tanaka, Xin Wang, Rachel Ward, Guanhua Wang, Philipp Witte, Michael Wyatt, Can Xu, Jiahang Xu, Sonali Yadav, Fan Yang, Ziyi Yang, Donghan Yu, Chengruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. Phi-3 technical report: A highly capable language model locally on your phone. *CoRR*, abs/2404.14219, 2024. doi: 10.48550/ARXIV.2404.14219. URL `https://doi.org/10.48550/arXiv.2404.14219`.

Sallam Abualhaija, Marcello Ceci, and Lionel C. Briand. Legal requirements analysis. *CoRR*, abs/2311.13871, 2023. doi: 10.48550/ARXIV.2311.13871. URL `https://doi.org/10.48550/arXiv.2311.13871`.

Mohammed Abuhamad, Tamer AbuHmed, Aziz Mohaisen, and DaeHun Nyang. Large-scale and language-oblivious code authorship identification. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang (eds.), *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pp. 101–114. ACM, 2018. doi: 10.1145/3243734.3243738. URL `https://doi.org/10.1145/3243734.3243738`.

Rajas Agashe, Srinivasan Iyer, and Luke Zettlemoyer. Juice: A large scale distantly supervised dataset for open domain context-based code generation. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 5435–5445. Association for Computational Linguistics, 2019. doi: 10.18653/V1/D19-1546. URL `https://doi.org/10.18653/v1/D19-1546`.

Armen Aghajanyan, Dmytro Okhonko, Mike Lewis, Mandar Joshi, Hu Xu, Gargi Ghosh, and Luke Zettlemoyer. HTLM: hyper-text pre-training and prompting of language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL `https://openreview.net/forum?id=P-pPW1nxf1r`.

Lakshya A Agrawal, Aditya Kanade, Navin Goyal, Shuvendu K. Lahiri, and Sriram K. Rajamani. Guiding language models of code with global context using monitors. *CoRR*, abs/2306.10763, 2023. doi: 10.48550/ARXIV.2306.10763. URL https://doi.org/10.48550/arXiv.2306.10763.

Wasi Uddin Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. A transformer-based approach for source code summarization. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 4998–5007. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.449. URL https://doi.org/10.18653/v1/2020.acl-main.449.

Wasi Uddin Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. Unified pre-training for program understanding and generation. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pp. 2655–2668. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.211. URL https://doi.org/10.18653/v1/2021.naacl-main.211.

Wasi Uddin Ahmad, Md Golam Rahman Tushar, Saikat Chakraborty, and Kai-Wei Chang. AVATAR: A parallel corpus for java-python program translation. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 2268–2281. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-ACL.143. URL https://doi.org/10.18653/v1/2023.findings-acl.143.

Sharif Ahmed, Arif Ahmed, and Nasir U. Eisty. Automatic transformation of natural to unified modeling language: A systematic review. In Juyeon Jo, Yeong-Tae Song, Lin Deng, and Junghwan John Rhee (eds.), *20th IEEE/ACIS International Conference on Software Engineering Research, Management and Applications, SERA 2022, Las Vegas, NV, USA, May 25-27, 2022*, pp. 112–119. IEEE, 2022. doi: 10.1109/SERA54885.2022.9806783. URL https://doi.org/10.1109/SERA54885.2022.9806783.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: training generalized multi-query transformer models from multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 4895–4901. Association for Computational Linguistics, 2023. URL https://aclanthology.org/2023.emnlp-main.298.

Saranya Alagarsamy, Chakkrit Tantithamthavorn, and Aldeida Aleti. A3test: Assertion-augmented automated test case generation. *CoRR*, abs/2302.10352, 2023. doi: 10.48550/ARXIV.2302.10352. URL https://doi.org/10.48550/arXiv.2302.10352.

Loubna Ben Allal, Raymond Li, Denis Kocetkov, Chenghao Mou, Christopher Akiki, Carlos Muñoz Ferrandis, Niklas Muennighoff, Mayank Mishra, Alex Gu, Manan Dey, Logesh Kumar Umapathi, Carolyn Jane Anderson, Yangtian Zi, Joel Lamy-Poirier, Hailey Schoelkopf, Sergey Troshin, Dmitry Abulkhanov, Manuel Romero, Michael Lappert, Francesco De Toni, Bernardo García del Río, Qian Liu, Shamik Bose, Urvashi Bhattacharyya, Terry Yue Zhuo, Ian Yu, Paulo Villegas, Marco Zocca, Sourab Mangrulkar, David Lansky, Huu Nguyen, Danish Contractor, Luis Villa, Jia Li, Dzmitry Bahdanau, Yacine Jernite, Sean Hughes, Daniel Fried, Arjun Guha, Harm de Vries, and Leandro von Werra. Santacoder: don't reach for the stars! *CoRR*, abs/2301.03988, 2023. doi: 10.48550/arXiv.2301.03988. URL https://doi.org/10.48550/arXiv.2301.03988.

Miltiadis Allamanis and Charles Sutton. Mining source code repositories at massive scale using language modeling. In Thomas Zimmermann, Massimiliano Di Penta, and Sunghun Kim (eds.), *Proceedings of the 10th Working Conference on Mining Software Repositories, MSR '13, San Francisco, CA, USA, May 18-19, 2013*, pp. 207–216. IEEE Computer Society, 2013. doi: 10.1109/MSR.2013.6624029. URL https://doi.org/10.1109/MSR.2013.6624029.

Miltiadis Allamanis and Charles Sutton. Mining idioms from source code. In Shing-Chi Cheung, Alessandro Orso, and Margaret-Anne D. Storey (eds.), *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, (FSE-22), Hong Kong, China, November 16 - 22, 2014*, pp. 472–483. ACM, 2014. doi: 10.1145/2635868.2635901. URL https://doi.org/10.1145/2635868.2635901.

Miltiadis Allamanis, Earl T. Barr, Christian Bird, and Charles Sutton. Learning natural coding conventions. In Shing-Chi Cheung, Alessandro Orso, and Margaret-Anne D. Storey (eds.), *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, (FSE-22), Hong Kong, China, November 16 - 22, 2014*, pp. 281–293. ACM, 2014. doi: 10.1145/2635868.2635883. URL https://doi.org/10.1145/2635868.2635883.

Miltiadis Allamanis, Earl T. Barr, Christian Bird, and Charles Sutton. Suggesting accurate method and class names. In Elisabetta Di Nitto, Mark Harman, and Patrick Heymans (eds.), *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, Bergamo, Italy, August 30 - September 4, 2015*, pp. 38–49. ACM, 2015. doi: 10.1145/2786805.2786849. URL https://doi.org/10.1145/2786805.2786849.

Miltiadis Allamanis, Earl T. Barr, René Just, and Charles Sutton. Tailored mutants fit bugs better. *CoRR*, abs/1611.02516, 2016a. URL http://arxiv.org/abs/1611.02516.

Miltiadis Allamanis, Hao Peng, and Charles Sutton. A convolutional attention network for extreme summarization of source code. In Maria-Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 2091–2100. JMLR.org, 2016b. URL http://proceedings.mlr.press/v48/allamanis16.html.

Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. Learning to represent programs with graphs. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=BJOFETxR-.

Miltiadis Allamanis, Earl T. Barr, Soline Ducousso, and Zheng Gao. Typilus: neural type hints. In Alastair F. Donaldson and Emina Torlak (eds.), *Proceedings of the 41st ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2020, London, UK, June 15-20, 2020*, pp. 91–105. ACM, 2020. doi: 10.1145/3385412.3385997. URL https://doi.org/10.1145/3385412.3385997.

Miltiadis Allamanis, Henry Jackson-Flux, and Marc Brockschmidt. Self-supervised bug detection and repair. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 27865–27876, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/ea96efc03b9a050d895110db8c4af057-Abstract.html.

Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. A general path-based representation for predicting program properties. In Jeffrey S. Foster and Dan Grossman (eds.), *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2018, Philadelphia, PA, USA, June 18-22, 2018*, pp. 404–419. ACM, 2018. doi: 10.1145/3192366.3192412. URL https://doi.org/10.1145/3192366.3192412.

Uri Alon, Shaked Brody, Omer Levy, and Eran Yahav. code2seq: Generating sequences from structured representations of code. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019a. URL https://openreview.net/forum?id=H1gKYo09tX.

Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. code2vec: learning distributed representations of code. *Proc. ACM Program. Lang.*, 3(POPL):40:1–40:29, 2019b. doi: 10.1145/3290353. URL https://doi.org/10.1145/3290353.

Uri Alon, Roy Sadaka, Omer Levy, and Eran Yahav. Structural language models of code. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 245–256. PMLR, 2020. URL http://proceedings.mlr.press/v119/alon20a.html.

Hussein Alrubaye, Mohamed Wiem Mkaouer, Igor Khokhlov, Leon Reznik, Ali Ouni, and Jason Mcgoff. Learning to recommend third-party library migration opportunities at the API level. *Appl. Soft Comput.*, 90:106140, 2020. doi: 10.1016/J.ASOC.2020.106140. URL https://doi.org/10.1016/j.asoc.2020.106140.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 2357–2367. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1245. URL https://doi.org/10.18653/v1/n19-1245.

Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy P. Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul Ronald Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, and et al. Gemini: A family of highly capable multimodal models. *CoRR*, abs/2312.11805, 2023a. doi: 10.48550/ARXIV.2312.11805. URL https://doi.org/10.48550/arXiv.2312.11805.

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernández Ábrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan A. Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vladimir Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, and et al. Palm 2 technical report. *CoRR*, abs/2305.10403, 2023b. doi: 10.48550/arXiv.2305.10403. URL https://doi.org/10.48550/arXiv.2305.10403.

Samuel Arcadinho, David Aparício, Hugo Veiga, and António Alegria. T5QL: taming language models for SQL generation. *CoRR*, abs/2209.10254, 2022. doi: 10.48550/ARXIV.2209.10254. URL https://doi.org/10.48550/arXiv.2209.10254.

Jordi Armengol-Estapé and Michael F. P. O'Boyle. Learning C to x86 translation: An experiment in neural compilation. *CoRR*, abs/2108.07639, 2021. URL https://arxiv.org/abs/2108.07639.

Jordi Armengol-Estapé, Jackson Woodruff, Chris Cummins, and Michael F. P. O'Boyle. Slade: A portable small language model decompiler for optimized assembler. *CoRR*, abs/2305.12520, 2023. doi: 10.48550/ARXIV.2305.12520. URL https://doi.org/10.48550/arXiv.2305.12520.

Chetan Arora, Mehrdad Sabetzadeh, Lionel C. Briand, and Frank Zimmer. Automated checking of conformance to requirements templates using natural language processing. *IEEE Trans. Software Eng.*, 41(10): 944–968, 2015a. doi: 10.1109/TSE.2015.2428709. URL https://doi.org/10.1109/TSE.2015.2428709.

Chetan Arora, Mehrdad Sabetzadeh, Arda Goknil, Lionel C. Briand, and Frank Zimmer. Change impact analysis for natural language requirements: An NLP approach. In Didar Zowghi, Vincenzo Gervasi, and

Daniel Amyot (eds.), *23rd IEEE International Requirements Engineering Conference, RE 2015, Ottawa, ON, Canada, August 24-28, 2015*, pp. 6–15. IEEE Computer Society, 2015b. doi: 10.1109/RE.2015.7320403. URL https://doi.org/10.1109/RE.2015.7320403.

Chetan Arora, Mehrdad Sabetzadeh, Lionel C. Briand, and Frank Zimmer. Extracting domain models from natural-language requirements: approach and industrial evaluation. In Benoit Baudry and Benoît Combemale (eds.), *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems, Saint-Malo, France, October 2-7, 2016*, pp. 250–260. ACM, 2016. URL http://dl.acm.org/citation.cfm?id=2976769.

Chetan Arora, Mehrdad Sabetzadeh, Lionel C. Briand, and Frank Zimmer. Automated extraction and clustering of requirements glossary terms. *IEEE Trans. Software Eng.*, 43(10):918–945, 2017. doi: 10.1109/TSE.2016.2635134. URL https://doi.org/10.1109/TSE.2016.2635134.

Chetan Arora, John Grundy, and Mohamed Abdelrazek. Advancing requirements engineering through generative AI: assessing the role of llms. *CoRR*, abs/2310.13976, 2023. doi: 10.48550/ARXIV.2310.13976. URL https://doi.org/10.48550/arXiv.2310.13976.

Chetan Arora, John Grundy, Louise Puli, and Natasha Layton. Towards standards-compliant assistive technology product specifications via llms. *CoRR*, abs/2404.03122, 2024. doi: 10.48550/ARXIV.2404.03122. URL https://doi.org/10.48550/arXiv.2404.03122.

Ellen Arteca, Sebastian Harner, Michael Pradel, and Frank Tip. Nessie: Automatically testing javascript apis with asynchronous callbacks. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pp. 1494–1505. ACM, 2022. doi: 10.1145/3510003.3510106. URL https://doi.org/10.1145/3510003.3510106.

Fiorella Artuso, Giuseppe Antonio Di Luna, Luca Massarelli, and Leonardo Querzoni. In nomine function: Naming functions in stripped binaries with neural networks, 2021.

Owura Asare, Meiyappan Nagappan, and N. Asokan. Is github's copilot as bad as humans at introducing vulnerabilities in code? *Empir. Softw. Eng.*, 28(6):129, 2023. doi: 10.1007/S10664-023-10380-1. URL https://doi.org/10.1007/s10664-023-10380-1.

Ben Athiwaratkun, Sanjay Krishna Gouda, Zijian Wang, Xiaopeng Li, Yuchen Tian, Ming Tan, Wasi Uddin Ahmad, Shiqi Wang, Qing Sun, Mingyue Shang, Sujan Kumar Gonugondla, Hantian Ding, Varun Kumar, Nathan Fulton, Arash Farahani, Siddhartha Jain, Robert Giaquinto, Haifeng Qian, Murali Krishna Ramanathan, and Ramesh Nallapati. Multi-lingual evaluation of code generation models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=Bo7eeXm6An8.

Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *CoRR*, abs/2108.07732, 2021. URL https://arxiv.org/abs/2108.07732.

Batuhan Aşıroğlu, Büşta Rümeysa Mete, Eyyüp Yıldız, Yağız Nalçakan, Alper Sezen, Mustafa Dağtekin, and Tolga Ensari. Automatic html code generation from mock-up images using machine learning techniques. In *2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT)*, pp. 1–4, 2019. doi: 10.1109/EBBT.2019.8741736.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016. URL http://arxiv.org/abs/1607.06450.

Hannah McLean Babe, Sydney Nguyen, Yangtian Zi, Arjun Guha, Molly Q. Feldman, and Carolyn Jane Anderson. Studenteval: A benchmark of student-written prompts for large language models of code. *CoRR*, abs/2306.04556, 2023. doi: 10.48550/ARXIV.2306.04556. URL https://doi.org/10.48550/arXiv.2306.04556.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJDaqqveg.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *CoRR*, abs/2309.16609, 2023. doi: 10.48550/arXiv.2309.16609. URL https://doi.org/10.48550/arXiv.2309.16609.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Chris Olah, Benjamin Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback. *CoRR*, abs/2204.05862, 2022. doi: 10.48550/ARXIV.2204.05862. URL https://doi.org/10.48550/arXiv.2204.05862.

Ramakrishna Bairi, Atharv Sonwane, Aditya Kanade, Vageesh D. C, Arun Iyer, Suresh Parthasarathy, Sriram K. Rajamani, Balasubramanyan Ashok, and Shashank Shet. Codeplan: Repository-level coding using llms and planning. *CoRR*, abs/2309.12499, 2023. doi: 10.48550/ARXIV.2309.12499. URL https://doi.org/10.48550/arXiv.2309.12499.

Noor Hasrina Bakar, Zarinah Mohd Kasirun, Norsaremah Salleh, and Hamid Abdullah Jalab. Extracting features from online software reviews to aid requirements reuse. *Appl. Soft Comput.*, 49:1297–1315, 2016. doi: 10.1016/J.ASOC.2016.07.048. URL https://doi.org/10.1016/j.asoc.2016.07.048.

Vipin Balachandran. Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation. In David Notkin, Betty H. C. Cheng, and Klaus Pohl (eds.), *35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013*, pp. 931–940. IEEE Computer Society, 2013. doi: 10.1109/ICSE.2013.6606642. URL https://doi.org/10.1109/ICSE.2013.6606642.

Matej Balog, Alexander L. Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. Deepcoder: Learning to write programs. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=ByldLrqlx.

Upul Bandara and Gamini Wijayarathna. Deep neural networks for source code author identification. In Minho Lee, Akira Hirose, Zeng-Guang Hou, and Rhee Man Kil (eds.), *Neural Information Processing - 20th International Conference, ICONIP 2013, Daegu, Korea, November 3-7, 2013. Proceedings, Part II*, volume 8227 of *Lecture Notes in Computer Science*, pp. 368–375. Springer, 2013. doi: 10.1007/978-3-642-42042-9_46. URL https://doi.org/10.1007/978-3-642-42042-9_46.

Pratyay Banerjee, Kuntal Kumar Pal, Fish Wang, and Chitta Baral. Variable name recovery in decompiled binary code using constrained masked language modeling. *CoRR*, abs/2103.12801, 2021. URL https://arxiv.org/abs/2103.12801.

Aakash Bansal, Sakib Haque, and Collin McMillan. Project-level encoding for neural source code summarization of subroutines. In *29th IEEE/ACM International Conference on Program Comprehension, ICPC 2021, Madrid, Spain, May 20-21, 2021*, pp. 253–264. IEEE, 2021. doi: 10.1109/ICPC52881.2021.00032. URL https://doi.org/10.1109/ICPC52881.2021.00032.

Patrick Bareiß, Beatriz Souza, Marcelo d'Amorim, and Michael Pradel. Code generation tools (almost) for free? A study of few-shot, pre-trained language models on code. *CoRR*, abs/2206.01335, 2022. doi: 10.48550/arXiv.2206.01335. URL https://doi.org/10.48550/arXiv.2206.01335.

Mike Barnett, Christian Bird, João Brunet, and Shuvendu K. Lahiri. Helping developers help themselves: Automatic decomposition of code review changesets. In Antonia Bertolino, Gerardo Canfora, and Sebastian G. Elbaum (eds.), *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 1*, pp. 134–144. IEEE Computer Society, 2015. doi: 10.1109/ICSE.2015.35. URL https://doi.org/10.1109/ICSE.2015.35.

Antonio Valerio Miceli Barone and Rico Sennrich. A parallel corpus of python functions and documentation strings for automated code documentation and code generation. In Greg Kondrak and Taro Watanabe (eds.), *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017, Volume 2: Short Papers*, pp. 314–319. Asian Federation of Natural Language Processing, 2017. URL https://aclanthology.org/I17-2053/.

Ezio Bartocci, Leonardo Mariani, Dejan Nickovic, and Drishti Yadav. Property-based mutation testing. In *IEEE Conference on Software Testing, Verification and Validation, ICST 2023, Dublin, Ireland, April 16-20, 2023*, pp. 222–233. IEEE, 2023. doi: 10.1109/ICST57152.2023.00029. URL https://doi.org/10.1109/ICST57152.2023.00029.

Nauman Bashir, Muhammad Bilal, Misbah Liaqat, Mohsen Marjani, Nadia Malik, and Mohsin Ali. Modeling class diagram using nlp in object-oriented designing. In *2021 National Computing Colleges Conference (NCCC)*, pp. 1–6, 2021. doi: 10.1109/NCCC49330.2021.9428817.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient training of language models to fill in the middle. *CoRR*, abs/2207.14255, 2022. doi: 10.48550/arXiv.2207.14255. URL https://doi.org/10.48550/arXiv.2207.14255.

Rohan Bavishi, Caroline Lemieux, Roy Fox, Koushik Sen, and Ion Stoica. Autopandas: neural-backed generators for program synthesis. *Proc. ACM Program. Lang.*, 3(OOPSLA):168:1–168:27, 2019. doi: 10.1145/3360594. URL https://doi.org/10.1145/3360594.

Tony Beltramelli. pix2code: Generating code from a graphical user interface screenshot. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS 2018, Paris, France, June 19-22, 2018*, pp. 3:1–3:6. ACM, 2018. doi: 10.1145/3220134.3220135. URL https://doi.org/10.1145/3220134.3220135.

Tal Ben-Nun, Alice Shoshana Jakobovits, and Torsten Hoefler. Neural code comprehension: A learnable representation of code semantics. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 3589–3601, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/17c3433fecc21b57000debdf7ad5c930-Abstract.html.

Berkay Berabi, Jingxuan He, Veselin Raychev, and Martin T. Vechev. Tfix: Learning to fix coding errors with a text-to-text transformer. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 780–791. PMLR, 2021. URL http://proceedings.mlr.press/v139/berabi21a.html.

Guru Prasad Bhandari, Amara Naseer, and Leon Moonen. Cvefixes: automated collection of vulnerabilities and their fixes from open-source software. In Shane McIntosh, Xin Xia, and Sousuke Amasaki (eds.),

PROMISE '21: 17th International Conference on Predictive Models and Data Analytics in Software Engineering, Athens Greece, August 19-20, 2021, pp. 30–39. ACM, 2021. doi: 10.1145/3475960.3475985. URL https://doi.org/10.1145/3475960.3475985.

Sahil Bhatia, Pushmeet Kohli, and Rishabh Singh. Neuro-symbolic program corrector for introductory programming assignments. In Michel Chaudron, Ivica Crnkovic, Marsha Chechik, and Mark Harman (eds.), *Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, pp. 60–70. ACM, 2018. doi: 10.1145/3180155.3180219. URL https://doi.org/10.1145/3180155.3180219.

Yingzhou Bi, Jiangtao Huang, Penghui Liu, and Lianmei Wang. Benchmarking software vulnerability detection techniques: A survey. *CoRR*, abs/2303.16362, 2023. doi: 10.48550/ARXIV.2303.16362. URL https://doi.org/10.48550/arXiv.2303.16362.

Zhangqian Bi, Yao Wan, Zheng Wang, Hongyu Zhang, Batu Guan, Fangxin Lu, Zili Zhang, Yulei Sui, Xuanhua Shi, and Hai Jin. Iterative refinement of project-level code context for precise code generation with compiler feedback. *CoRR*, abs/2403.16792, 2024. doi: 10.48550/ARXIV.2403.16792. URL https://doi.org/10.48550/arXiv.2403.16792.

Pan Bian, Bin Liang, Wenchang Shi, Jianjun Huang, and Yan Cai. Nar-miner: discovering negative association rules from code for bug detection. In Gary T. Leavens, Alessandro Garcia, and Corina S. Pasareanu (eds.), *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2018, Lake Buena Vista, FL, USA, November 04-09, 2018*, pp. 411–422. ACM, 2018. doi: 10.1145/3236024.3236032. URL https://doi.org/10.1145/3236024.3236032.

Benjamin Bichsel, Veselin Raychev, Petar Tsankov, and Martin T. Vechev. Statistical deobfuscation of android applications. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi (eds.), *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pp. 343–355. ACM, 2016. doi: 10.1145/2976749.2978422. URL https://doi.org/10.1145/2976749.2978422.

Stella Biderman, Kieran Bicheno, and Leo Gao. Datasheet for the pile. *CoRR*, abs/2201.07311, 2022. URL https://arxiv.org/abs/2201.07311.

Pavol Bielik, Veselin Raychev, and Martin T. Vechev. PHOG: probabilistic model for code. In Maria-Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 2933–2942. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/bielik16.html.

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. GPT-NeoX-20B: An open-source autoregressive language model. In Angela Fan, Suzana Ilic, Thomas Wolf, and Matthias Gallé (eds.), *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pp. 95–136, virtual+Dublin, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.bigscience-1.9. URL https://aclanthology.org/2022.bigscience-1.9.

Arianna Blasi, Alessandra Gorla, Michael D. Ernst, Mauro Pezzè, and Antonio Carzaniga. Memo: Automatically identifying metamorphic relations in javadoc comments for test automation. *J. Syst. Softw.*, 181: 111041, 2021. doi: 10.1016/j.jss.2021.111041. URL https://doi.org/10.1016/j.jss.2021.111041.

Ben Bogin, Jonathan Berant, and Matt Gardner. Representing schema structure with graph neural networks for text-to-sql parsing. In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 4560–4565. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1448. URL https://doi.org/10.18653/v1/p19-1448.

Marcel Böhme, Van-Thuan Pham, Manh-Dung Nguyen, and Abhik Roychoudhury. Directed greybox fuzzing. In Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (eds.), *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pp. 2329–2344. ACM, 2017. doi: 10.1145/3133956.3134020. URL https://doi.org/10.1145/3133956.3134020.

Marcel Böhme, Van-Thuan Pham, and Abhik Roychoudhury. Coverage-based greybox fuzzing as markov chain. *IEEE Trans. Software Eng.*, 45(5):489–506, 2019. doi: 10.1109/TSE.2017.2785841. URL https://doi.org/10.1109/TSE.2017.2785841.

Casper Boone, Niels de Bruin, Arjan Langerak, and Fabian Stelmach. Dltpy: Deep learning type inference of python function signatures using natural language context. *CoRR*, abs/1912.00680, 2019. URL http://arxiv.org/abs/1912.00680.

David Bingham Brown, Michael Vaughn, Ben Liblit, and Thomas W. Reps. The care and feeding of wild-caught mutants. In Eric Bodden, Wilhelm Schäfer, Arie van Deursen, and Andrea Zisman (eds.), *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, Paderborn, Germany, September 4-8, 2017*, pp. 511–522. ACM, 2017. doi: 10.1145/3106237.3106280. URL https://doi.org/10.1145/3106237.3106280.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Marcel Bruch, Martin Monperrus, and Mira Mezini. Learning from examples to improve code completion systems. In Hans van Vliet and Valérie Issarny (eds.), *Proceedings of the 7th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2009, Amsterdam, The Netherlands, August 24-28, 2009*, pp. 213–222. ACM, 2009. doi: 10.1145/1595696.1595728. URL https://doi.org/10.1145/1595696.1595728.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Túlio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with GPT-4. *CoRR*, abs/2303.12712, 2023. doi: 10.48550/arXiv.2303.12712. URL https://doi.org/10.48550/arXiv.2303.12712.

Nghi Bui, Yue Wang, and Steven C. H. Hoi. Detect-localize-repair: A unified framework for learning to debug with codet5. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 812–823. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.FINDINGS-EMNLP.57. URL https://doi.org/10.18653/v1/2022.findings-emnlp.57.

Nghi D. Q. Bui, Yijun Yu, and Lingxiao Jiang. SAR: learning cross-language API mappings with little knowledge. In Marlon Dumas, Dietmar Pfahl, Sven Apel, and Alessandra Russo (eds.), *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2019, Tallinn, Estonia, August 26-30, 2019*, pp. 796–806. ACM, 2019. doi: 10.1145/3338906.3338924. URL https://doi.org/10.1145/3338906.3338924.

Nghi D. Q. Bui, Yijun Yu, and Lingxiao Jiang. Infercode: Self-supervised learning of code representations by predicting subtrees. In *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*, pp. 1186–1197. IEEE, 2021a. doi: 10.1109/ICSE43902.2021.00109. URL https://doi.org/10.1109/ICSE43902.2021.00109.

Nghi D. Q. Bui, Yijun Yu, and Lingxiao Jiang. Self-supervised contrastive learning for code retrieval and summarization via semantic-preserving transformations. In Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (eds.), *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pp. 511–521. ACM, 2021b. doi: 10.1145/3404835.3462840. URL https://doi.org/10.1145/3404835.3462840.

Rudy Bunel, Matthew J. Hausknecht, Jacob Devlin, Rishabh Singh, and Pushmeet Kohli. Leveraging grammar and reinforcement learning for neural program synthesis. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=H1Xw62kRZ.

Javier Cámara, Javier Troya, Lola Burgueño, and Antonio Vallecillo. On the assessment of generative AI in modeling tasks: an experience report with chatgpt and UML. *Softw. Syst. Model.*, 22(3):781–793, 2023. doi: 10.1007/S10270-023-01105-5. URL https://doi.org/10.1007/s10270-023-01105-5.

José Cambronero, Hongyu Li, Seohyun Kim, Koushik Sen, and Satish Chandra. When deep learning met code search. In Marlon Dumas, Dietmar Pfahl, Sven Apel, and Alessandra Russo (eds.), *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2019, Tallinn, Estonia, August 26-30, 2019*, pp. 964–974. ACM, 2019. doi: 10.1145/3338906.3340458. URL https://doi.org/10.1145/3338906.3340458.

Jialun Cao, Meiziniu Li, Ming Wen, and Shing-Chi Cheung. A study on prompt design, advantages and limitations of chatgpt for deep learning program repair. *CoRR*, abs/2304.08191, 2023. doi: 10.48550/ARXIV.2304.08191. URL https://doi.org/10.48550/arXiv.2304.08191.

Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. LGESQL: line graph enhanced text-to-sql model with mixed local and non-local relations. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 2541–2555. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.198. URL https://doi.org/10.18653/v1/2021.acl-long.198.

Ying Cao, Ruigang Liang, Kai Chen, and Peiwei Hu. Boosting neural networks to decompile optimized binaries. In *Annual Computer Security Applications Conference, ACSAC 2022, Austin, TX, USA, December 5-9, 2022*, pp. 508–518. ACM, 2022. doi: 10.1145/3564625.3567998. URL https://doi.org/10.1145/3564625.3567998.

Federico Cassano, John Gouwar, Francesca Lucchetti, Claire Schlesinger, Carolyn Jane Anderson, Michael Greenberg, Abhinav Jangda, and Arjun Guha. Knowledge transfer from high-resource to low-resource programming languages for code llms. *CoRR*, abs/2308.09895, 2023a. doi: 10.48550/ARXIV.2308.09895. URL https://doi.org/10.48550/arXiv.2308.09895.

Federico Cassano, John Gouwar, Daniel Nguyen, Sydney Nguyen, Luna Phipps-Costin, Donald Pinckney, Ming-Ho Yee, Yangtian Zi, Carolyn Jane Anderson, Molly Q Feldman, et al. Multipl-e: a scalable and polyglot approach to benchmarking neural code generation. *IEEE Transactions on Software Engineering*, 2023b.

Federico Cassano, Ming-Ho Yee, Noah Shinn, Arjun Guha, and Steven Holtzen. Type prediction with program decomposition and fill-in-the-type training. *CoRR*, abs/2305.17145, 2023c. doi: 10.48550/ARXIV.2305.17145. URL https://doi.org/10.48550/arXiv.2305.17145.

Sang Kil Cha, Maverick Woo, and David Brumley. Program-adaptive mutational fuzzing. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pp. 725–741. IEEE Computer Society, 2015. doi: 10.1109/SP.2015.50. URL https://doi.org/10.1109/SP.2015.50.

Meriem Ben Chaaben, Lola Burgueño, and Houari A. Sahraoui. Towards using few-shot prompt learning for automating model completion. In *45th IEEE/ACM International Conference on Software Engineering:*

New Ideas and Emerging Results, NIER@ICSE, Melbourne, Australia, May 14-20, 2023, pp. 7–12. IEEE, 2023. doi: 10.1109/ICSE-NIER58687.2023.00008. URL https://doi.org/10.1109/ICSE-NIER58687.2023.00008.

Hyungjoo Chae, Yeonghyeon Kim, Seungone Kim, Kai Tzu iunn Ong, Beong woo Kwak, Moohyeon Kim, Seonghwan Kim, Taeyoon Kwon, Jiwan Chung, Youngjae Yu, and Jinyoung Yeo. Language models as compilers: Simulating pseudocode execution improves algorithmic reasoning in language models. *CoRR*, abs/2404.02575, 2024. doi: 10.48550/ARXIV.2404.02575. URL https://doi.org/10.48550/arXiv.2404.02575.

Yekun Chai, Shuohuan Wang, Chao Pang, Yu Sun, Hao Tian, and Hua Wu. Ernie-code: Beyond english-centric cross-lingual pretraining for programming languages. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 10628–10650. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-ACL.676. URL https://doi.org/10.18653/v1/2023.findings-acl.676.

Yitian Chai, Hongyu Zhang, Beijun Shen, and Xiaodong Gu. Cross-domain deep code search with meta learning. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pp. 487–498. ACM, 2022. doi: 10.1145/3510003.3510125. URL https://doi.org/10.1145/3510003.3510125.

Saikat Chakraborty and Baishakhi Ray. On multi-modal learning of editing source code. In *36th IEEE/ACM International Conference on Automated Software Engineering, ASE 2021, Melbourne, Australia, November 15-19, 2021*, pp. 443–455. IEEE, 2021. doi: 10.1109/ASE51524.2021.9678559. URL https://doi.org/10.1109/ASE51524.2021.9678559.

Saikat Chakraborty, Toufique Ahmed, Yangruibo Ding, Premkumar T. Devanbu, and Baishakhi Ray. Natgen: generative pre-training by "naturalizing" source code. In Abhik Roychoudhury, Cristian Cadar, and Miryung Kim (eds.), *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022, Singapore, Singapore, November 14-18, 2022*, pp. 18–30. ACM, 2022a. doi: 10.1145/3540250.3549162. URL https://doi.org/10.1145/3540250.3549162.

Saikat Chakraborty, Yangruibo Ding, Miltiadis Allamanis, and Baishakhi Ray. CODIT: code editing with tree-based neural models. *IEEE Trans. Software Eng.*, 48(4):1385–1399, 2022b. doi: 10.1109/TSE.2020.3020502. URL https://doi.org/10.1109/TSE.2020.3020502.

Saikat Chakraborty, Rahul Krishna, Yangruibo Ding, and Baishakhi Ray. Deep learning based vulnerability detection: Are we there yet? *IEEE Trans. Software Eng.*, 48(9):3280–3296, 2022c. doi: 10.1109/TSE.2021.3087402. URL https://doi.org/10.1109/TSE.2021.3087402.

Aaron Chan, Anant Kharkar, Roshanak Zilouchian Moghaddam, Yevhen Mohylevskyy, Alec Helyar, Eslam Kamal, Mohamed Elkamhawy, and Neel Sundaresan. Transformer-based vulnerability detection in code at edittime: Zero-shot, few-shot, or fine-tuning? *CoRR*, abs/2306.01754, 2023. doi: 10.48550/ARXIV.2306.01754. URL https://doi.org/10.48550/arXiv.2306.01754.

Shubham Chandel, Colin B. Clement, Guillermo Serrato, and Neel Sundaresan. Training and evaluating a jupyter notebook data science assistant. *CoRR*, abs/2201.12901, 2022. URL https://arxiv.org/abs/2201.12901.

Shuaichen Chang and Eric Fosler-Lussier. How to prompt llms for text-to-sql: A study in zero-shot, single-domain, and cross-domain settings. *CoRR*, abs/2305.11853, 2023. doi: 10.48550/ARXIV.2305.11853. URL https://doi.org/10.48550/arXiv.2305.11853.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. Codet: Code generation with generated tests. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023a. URL https://openreview.net/pdf?id=ktrw68Cmu9c.

Boqi Chen, Kua Chen, Shabnam Hassani, Yujing Yang, Daniel Amyot, Lysanne Lessard, Gunter Mussbacher, Mehrdad Sabetzadeh, and Dániel Varró. On the use of GPT-4 for creating goal models: An exploratory study. In Kurt Schneider, Fabiano Dalpiaz, and Jennifer Horkoff (eds.), *31st IEEE International Requirements Engineering Conference, RE 2023 - Workshops, Hannover, Germany, September 4-5, 2023*, pp. 262–271. IEEE, 2023b. doi: 10.1109/REW57809.2023.00052. URL https://doi.org/10.1109/REW57809.2023.00052.

Chunyang Chen, Zhenchang Xing, Yang Liu, and Kent Ong Long Xiong. Mining likely analogical apis across third-party libraries via large-scale unsupervised API semantics embedding. *IEEE Trans. Software Eng.*, 47(3):432–447, 2021a. doi: 10.1109/TSE.2019.2896123. URL https://doi.org/10.1109/TSE.2019.2896123.

Fuxiang Chen, Fatemeh H. Fard, David Lo, and Timofey Bryksin. On the transferability of pre-trained language models for low-resource programming languages. In Ayushi Rastogi, Rosalia Tufano, Gabriele Bavota, Venera Arnaoudova, and Sonia Haiduc (eds.), *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension, ICPC 2022, Virtual Event, May 16-17, 2022*, pp. 401–412. ACM, 2022. doi: 10.1145/3524610.3527917. URL https://doi.org/10.1145/3524610.3527917.

Kua Chen, Yujing Yang, Boqi Chen, José Antonio Hernández López, Gunter Mussbacher, and Dániel Varró. Automated domain modeling with large language models: A comparative study. In *26th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2023, Västerås, Sweden, October 1-6, 2023*, pp. 162–172. IEEE, 2023c. doi: 10.1109/MODELS58315.2023.00037. URL https://doi.org/10.1109/MODELS58315.2023.00037.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021b. URL https://arxiv.org/abs/2107.03374.

Peng Chen and Hao Chen. Angora: Efficient fuzzing by principled search. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pp. 711–725. IEEE Computer Society, 2018. doi: 10.1109/SP.2018.00046. URL https://doi.org/10.1109/SP.2018.00046.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *CoRR*, abs/2306.15595, 2023d. doi: 10.48550/ARXIV.2306.15595. URL https://doi.org/10.48550/arXiv.2306.15595.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023e. ISSN 2835-8856. URL https://openreview.net/forum?id=YfZ4ZPt8zd.

Xinyun Chen, Chang Liu, and Dawn Song. Tree-to-tree neural networks for program translation. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 2552–2562, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/d759175de8ea5b1d9a2660e45554894f-Abstract.html.

Xinyun Chen, Linyuan Gong, Alvin Cheung, and Dawn Song. Plotcoder: Hierarchical decoding for synthesizing visualization code in programmatic context. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 2169–2181. Association for Computational Linguistics, 2021c. doi: 10.18653/V1/2021.ACL-LONG.169. URL https://doi.org/10.18653/v1/2021.acl-long.169.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. *CoRR*, abs/2304.05128, 2023f. doi: 10.48550/arXiv.2304.05128. URL https://doi.org/10.48550/arXiv.2304.05128.

Yijie Chen, Yijin Liu, Fandong Meng, Yufeng Chen, Jinan Xu, and Jie Zhou. Comments as natural logic pivots: Improve code generation via comment perspective. *CoRR*, abs/2404.07549, 2024. doi: 10.48550/ARXIV.2404.07549. URL https://doi.org/10.48550/arXiv.2404.07549.

Yizheng Chen, Zhoujie Ding, Lamya Alowain, Xinyun Chen, and David A. Wagner. Diversevul: A new vulnerable source code dataset for deep learning based vulnerability detection. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses, RAID 2023, Hong Kong, China, October 16-18, 2023*, pp. 654–668. ACM, 2023g. doi: 10.1145/3607199.3607242. URL https://doi.org/10.1145/3607199.3607242.

Zhuangbin Chen, Jinyang Liu, Wenwei Gu, Yuxin Su, and Michael R. Lyu. Experience report: Deep learning-based system log analysis for anomaly detection. *CoRR*, abs/2107.05908, 2021d. URL https://arxiv.org/abs/2107.05908.

Zimin Chen, Steve Kommrusch, Michele Tufano, Louis-Noël Pouchet, Denys Poshyvanyk, and Martin Monperrus. Sequencer: Sequence-to-sequence learning for end-to-end program repair. *IEEE Trans. Software Eng.*, 47(9):1943–1959, 2021e. doi: 10.1109/TSE.2019.2940179. URL https://doi.org/10.1109/TSE.2019.2940179.

Chin-Yi Cheng, Forrest Huang, Gang Li, and Yang Li. Play: Parametrically conditioned layout generation using latent diffusion. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 5449–5471. PMLR, 2023. URL https://proceedings.mlr.press/v202/cheng23b.html.

David Chiang. Hierarchical phrase-based translation. *Comput. Linguistics*, 33(2):201–228, 2007. doi: 10.1162/coli.2007.33.2.201. URL https://doi.org/10.1162/coli.2007.33.2.201.

Muslim Chochlov, Gul Aftab Ahmed, James Vincent Patten, Guoxian Lu, Wei Hou, David Gregg, and Jim Buckley. Using a nearest-neighbour, bert-based approach for scalable clone detection. In *IEEE International Conference on Software Maintenance and Evolution, ICSME 2022, Limassol, Cyprus, October 3-7, 2022*, pp. 582–591. IEEE, 2022. doi: 10.1109/ICSME55016.2022.00080. URL https://doi.org/10.1109/ICSME55016.2022.00080.

DongHyun Choi, Myeongcheol Shin, EungGyun Kim, and Dong Ryeol Shin. RYANSQL: recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. *Comput. Linguistics*, 47(2):309–332, 2021. doi: 10.1162/coli_a_00403. URL https://doi.org/10.1162/coli_a_00403.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi,

Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113, 2023. URL `http://jmlr.org/papers/v24/22-1144.html`.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 4299–4307, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html`.

Fenia Christopoulou, Gerasimos Lampouras, Milan Gritta, Guchun Zhang, Yinpeng Guo, Zhongqi Li, Qi Zhang, Meng Xiao, Bo Shen, Lin Li, Hao Yu, Li Yan, Pingyi Zhou, Xin Wang, Yuchi Ma, Ignacio Iacobacci, Yasheng Wang, Guangtai Liang, Jiansheng Wei, Xin Jiang, Qianxiang Wang, and Qun Liu. Pangu-coder: Program synthesis with function-level language modeling. *CoRR*, abs/2207.11280, 2022. doi: 10.48550/arXiv.2207.11280. URL `https://doi.org/10.48550/arXiv.2207.11280`.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Y. Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *CoRR*, abs/2210.11416, 2022. doi: 10.48550/arXiv.2210.11416. URL `https://doi.org/10.48550/arXiv.2210.11416`.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=r1xMH1BtvB`.

Colin B. Clement, Dawn Drain, Jonathan Timcheck, Alexey Svyatkovskiy, and Neel Sundaresan. Pymt5: multi-mode translation of natural language and python code with transformers. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 9052–9065. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.728. URL `https://doi.org/10.18653/v1/2020.emnlp-main.728`.

Colin B. Clement, Shuai Lu, Xiaoyu Liu, Michele Tufano, Dawn Drain, Nan Duan, Neel Sundaresan, and Alexey Svyatkovskiy. Long-range modeling of source code files with ewash: Extended window access by syntax hierarchy. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pp. 4713–4722. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EMNLP-MAIN.387. URL `https://doi.org/10.18653/v1/2021.emnlp-main.387`.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL `https://arxiv.org/abs/2110.14168`.

Henry Coles, Thomas Laurent, Christopher Henard, Mike Papadakis, and Anthony Ventresque. PIT: a practical mutation testing tool for java (demo). In Andreas Zeller and Abhik Roychoudhury (eds.), *Proceedings of the 25th International Symposium on Software Testing and Analysis, ISSTA 2016, Saarbrücken, Germany, July 18-20, 2016*, pp. 449–452. ACM, 2016. doi: 10.1145/2931037.2948707. URL https://doi.org/10.1145/2931037.2948707.

Christian S. Collberg and Clark Thomborson. Watermarking, tamper-proofing, and obfuscation - tools for software protection. *IEEE Transactions on Software Engineering*, 28(8):735–746, August 2002. ISSN 0098-5589. doi: 10.1109/TSE.2002.1027797. Funding Information: The authors are grateful for the extensive and insightful comments of two anonymous referees. This material is based upon work supported by the US National Science Foundation under Grant No. 0073483.

Alexis Conneau and Guillaume Lample. Cross-lingual language model pretraining. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 7057–7067, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/c04c19c2c2474dbf5f7ac4372c5b9af1-Abstract.html.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 8440–8451. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.747. URL https://doi.org/10.18653/v1/2020.acl-main.747.

Luis Fernando Cortes-Coy, Mario Linares Vásquez, Jairo Aponte, and Denys Poshyvanyk. On automatically generating commit messages via summarization of source code changes. In *14th IEEE International Working Conference on Source Code Analysis and Manipulation, SCAM 2014, Victoria, BC, Canada, September 28-29, 2014*, pp. 275–284. IEEE Computer Society, 2014. doi: 10.1109/SCAM.2014.14. URL https://doi.org/10.1109/SCAM.2014.14.

Viktor Csuvik and László Vidács. Fixjs: A dataset of bug-fixing javascript commits. In *19th IEEE/ACM International Conference on Mining Software Repositories, MSR 2022, Pittsburgh, PA, USA, May 23-24, 2022*, pp. 712–716. ACM, 2022. doi: 10.1145/3524842.3528480. URL https://doi.org/10.1145/3524842.3528480.

Haotian Cui, Chenglong Wang, Junjie Huang, Jeevana Priya Inala, Todd Mytkowicz, Bo Wang, Jianfeng Gao, and Nan Duan. Codeexp: Explanatory code document generation. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 2342–2354. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.FINDINGS-EMNLP.174. URL https://doi.org/10.18653/v1/2022.findings-emnlp.174.

Siwei Cui, Gang Zhao, Zeyu Dai, Luochao Wang, Ruihong Huang, and Jeff Huang. Pyinfer: Deep learning semantic type inference for python variables. *CoRR*, abs/2106.14316, 2021. URL https://arxiv.org/abs/2106.14316.

Chris Cummins, Pavlos Petoumenos, Zheng Wang, and Hugh Leather. End-to-end deep learning of optimization heuristics. In *26th International Conference on Parallel Architectures and Compilation Techniques, PACT 2017, Portland, OR, USA, September 9-13, 2017*, pp. 219–232. IEEE Computer Society, 2017. doi: 10.1109/PACT.2017.24. URL https://doi.org/10.1109/PACT.2017.24.

Chris Cummins, Zacharias V. Fisches, Tal Ben-Nun, Torsten Hoefler, Michael F P O'Boyle, and Hugh Leather. Programl: A graph-based program representation for data flow analysis and compiler optimizations. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 2244–2253. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/cummins21a.html.

Chris Cummins, Volker Seeker, Dejan Grubisic, Mostafa Elhoushi, Youwei Liang, Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Kim M. Hazelwood, Gabriel Synnaeve, and Hugh Leather. Large language models for compiler optimization. *CoRR*, abs/2309.07062, 2023. doi: 10.48550/ARXIV.2309.07062. URL https://doi.org/10.48550/arXiv.2309.07062.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William M. Fisher, Kate Hunicke-Smith, David S. Pallett, Christine Pao, Alexander I. Rudnicky, and Elizabeth Shriberg. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Human Language Technology, Proceedings of a Workshop held at Plainsboro, New Jerey, USA, March 8-11, 1994*. Morgan Kaufmann, 1994. URL https://aclanthology.org/H94-1010/.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *CoRR*, abs/2401.06066, 2024a. doi: 10.48550/ARXIV.2401.06066. URL https://doi.org/10.48550/arXiv.2401.06066.

Hetong Dai, Heng Li, Che-Shao Chen, Weiyi Shang, and Tse-Hsun Chen. Logram: Efficient log parsing using nn-gram dictionaries. *IEEE Trans. Software Eng.*, 48(3):879–892, 2022. doi: 10.1109/TSE.2020.3007554. URL https://doi.org/10.1109/TSE.2020.3007554.

Jianbo Dai, Jianqiao Lu, Yunlong Feng, Rongju Ruan, Ming Cheng, Haochen Tan, and Zhijiang Guo. Mhpp: Exploring the capabilities and limitations of language models beyond basic code generation. 2024b. URL https://doi.org/10.48550/arXiv.2405.11430.

Arghavan Moradi Dakhel, Amin Nikanjam, Vahid Majdinasab, Foutse Khomh, and Michel C. Desmarais. Effective test generation using pre-trained large language models and mutation testing. *CoRR*, abs/2308.16557, 2023. doi: 10.48550/ARXIV.2308.16557. URL https://doi.org/10.48550/arXiv.2308.16557.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract-Conference.html.

Adailton Ferreira de Araújo and Ricardo Marcondes Marcacini. RE-BERT: automatic extraction of software requirements from app reviews using BERT language model. In Chih-Cheng Hung, Jiman Hong, Alessio Bechini, and Eunjee Song (eds.), *SAC '21: The 36th ACM/SIGAPP Symposium on Applied Computing, Virtual Event, Republic of Korea, March 22-26, 2021*, pp. 1321–1327. ACM, 2021. doi: 10.1145/3412841.3442006. URL https://doi.org/10.1145/3412841.3442006.

Nelson Tavares de Sousa and Wilhelm Hasselbring. Javabert: Training a transformer-based model for the java programming language. In *36th IEEE/ACM International Conference on Automated Software Engineering, ASE 2021 - Workshops, Melbourne, Australia, November 15-19, 2021*, pp. 90–95. IEEE, 2021. doi: 10.1109/ASEW52652.2021.00028. URL https://doi.org/10.1109/ASEW52652.2021.00028.

DeepSeek-AI, Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, Guangbo Hao, Zhewen Hao, Ying He, Wenjie Hu, Panpan Huang, Erhang Li, Guowei Li, Jiashi Li, Yao Li, Y. K. Li, Wenfeng Liang, Fangyun Lin, A. X. Liu, Bo Liu, Wen Liu, Xiaodong Liu, Xin Liu, Yiyuan Liu, Haoyu Lu, Shanghao Lu, Fuli Luo, Shirong Ma, Xiaotao Nie, Tian Pei, Yishi Piao, Junjie Qiu, Hui Qu, Tongzheng Ren, Zehui Ren, Chong Ruan, Zhangli Sha, Zhihong Shao, Junxiao Song, Xuecheng Su, Jingxiang Sun, Yaofeng Sun, Minghui Tang, Bingxuan Wang, Peiyi Wang, Shiyu Wang, Yaohui Wang, Yongji Wang, Tong Wu, Y. Wu, Xin Xie, Zhenda Xie, Ziwei Xie, Yiliang Xiong, Hanwei Xu, R. X. Xu, Yanhong Xu, Dejian Yang, Yuxiang You, Shuiping Yu, Xingkai Yu, B. Zhang, Haowei Zhang, Lecong Zhang, Liyue Zhang, Mingchuan Zhang, Minghua Zhang, Wentao Zhang, Yichao Zhang, Chenggang Zhao, Yao Zhao, Shangyan Zhou, Shunfeng Zhou, Qihao Zhu, and Yuheng Zou. Deepseek llm: Scaling open-source language models with longtermism. *CoRR*, abs/2401.02954, 2024. doi: 10.48550/ARXIV.2401.02954. URL https://doi.org/10.48550/arXiv.2401.02954.

Deva Kumar Deeptimahanti and Muhammad Ali Babar. An automated tool for generating UML models from natural language requirements. In *ASE 2009, 24th IEEE/ACM International Conference on Automated Software Engineering, Auckland, New Zealand, November 16-20, 2009*, pp. 680–682. IEEE Computer Society, 2009. doi: 10.1109/ASE.2009.48. URL https://doi.org/10.1109/ASE.2009.48.

Renzo Degiovanni and Mike Papadakis. μbert: Mutation testing using pre-trained language models. In *15th IEEE International Conference on Software Testing, Verification and Validation Workshops ICST Workshops 2022, Valencia, Spain, April 4-13, 2022*, pp. 160–169. IEEE, 2022. doi: 10.1109/ICSTW55395.2022.00039. URL https://doi.org/10.1109/ICSTW55395.2022.00039.

Naihao Deng, Yulong Chen, and Yue Zhang. Recent advances in text-to-sql: A survey of what we have and what we expect. In Nicoletta Calzolari, Chu-Ren Huang, Hansaem Kim, James Pustejovsky, Leo Wanner, Key-Sun Choi, Pum-Mo Ryu, Hsin-Hsi Chen, Lucia Donatelli, Heng Ji, Sadao Kurohashi, Patrizia Paggio, Nianwen Xue, Seokhwan Kim, Younggyun Hahm, Zhong He, Tony Kyungil Lee, Enrico Santus, Francis Bond, and Seung-Hoon Na (eds.), *Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022*, pp. 2166–2187. International Committee on Computational Linguistics, 2022a. URL https://aclanthology.org/2022.coling-1.190.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. Structure-grounded pretraining for text-to-sql. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pp. 1337–1350. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.NAACL-MAIN.105. URL https://doi.org/10.18653/v1/2021.naacl-main.105.

Xiang Deng, Prashant Shiralkar, Colin Lockard, Binxuan Huang, and Huan Sun. DOM-LM: learning generalizable representations for HTML documents. *CoRR*, abs/2201.10608, 2022b. URL https://arxiv.org/abs/2201.10608.

Yinlin Deng, Chenyuan Yang, Anjiang Wei, and Lingming Zhang. Fuzzing deep-learning libraries via automated relational API inference. In Abhik Roychoudhury, Cristian Cadar, and Miryung Kim (eds.), *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022, Singapore, Singapore, November 14-18, 2022*, pp. 44–56. ACM, 2022c. doi: 10.1145/3540250.3549085. URL https://doi.org/10.1145/3540250.3549085.

Yinlin Deng, Chunqiu Steven Xia, Haoran Peng, Chenyuan Yang, and Lingming Zhang. Large language models are zero-shot fuzzers: Fuzzing deep-learning libraries via large language models. In René Just and Gordon Fraser (eds.), *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2023, Seattle, WA, USA, July 17-21, 2023*, pp. 423–435. ACM, 2023. doi: 10.1145/3597926.3598067. URL https://doi.org/10.1145/3597926.3598067.

Jacob Devlin, Jonathan Uesato, Surya Bhupatiraju, Rishabh Singh, Abdel-rahman Mohamed, and Pushmeet Kohli. Robustfill: Neural program learning under noisy I/O. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 990–998. PMLR, 2017a. URL http://proceedings.mlr.press/v70/devlin17a.html.

Jacob Devlin, Jonathan Uesato, Rishabh Singh, and Pushmeet Kohli. Semantic code repair using neuro-symbolic transformation networks. *CoRR*, abs/1710.11054, 2017b. URL http://arxiv.org/abs/1710.11054.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June*

2-7, 2019, Volume 1 (Long and Short Papers), pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Peng Di, Jianguo Li, Hang Yu, Wei Jiang, Wenting Cai, Yang Cao, Chaoyu Chen, Dajun Chen, Hongwei Chen, Liang Chen, Gang Fan, Jie Gong, Zi Gong, Wen Hu, Tingting Guo, Zhichao Lei, Ting Li, Zheng Li, Ming Liang, Cong Liao, Bingchang Liu, Jiachen Liu, Zhiwei Liu, Shaojun Lu, Min Shen, Guangpei Wang, Huan Wang, Zhi Wang, Zhaogui Xu, Jiawei Yang, Qing Ye, Gehao Zhang, Yu Zhang, Zelin Zhao, Xunjin Zheng, Hailian Zhou, Lifu Zhu, and Xianying Zhu. Codefuse-13b: A pretrained multi-lingual code large language model. *CoRR*, abs/2310.06266, 2023. doi: 10.48550/ARXIV.2310.06266. URL https://doi.org/10.48550/arXiv.2310.06266.

Elizabeth Dinella, Gabriel Ryan, Todd Mytkowicz, and Shuvendu K. Lahiri. TOGA: A neural method for test oracle generation. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pp. 2130–2141. ACM, 2022. doi: 10.1145/3510003.3510141. URL https://doi.org/10.1145/3510003.3510141.

Yangruibo Ding, Luca Buratti, Saurabh Pujar, Alessandro Morari, Baishakhi Ray, and Saikat Chakraborty. Towards learning (dis)-similarity of source code from program contrasts. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 6300–6312. Association for Computational Linguistics, 2022a. doi: 10.18653/v1/2022.acl-long.436. URL https://doi.org/10.18653/v1/2022.acl-long.436.

Yangruibo Ding, Zijian Wang, Wasi Uddin Ahmad, Murali Krishna Ramanathan, Ramesh Nallapati, Parminder Bhatia, Dan Roth, and Bing Xiang. Cocomic: Code completion by jointly modeling in-file and cross-file context. *CoRR*, abs/2212.10007, 2022b. doi: 10.48550/ARXIV.2212.10007. URL https://doi.org/10.48550/arXiv.2212.10007.

Yangruibo Ding, Zijian Wang, Wasi Uddin Ahmad, Hantian Ding, Ming Tan, Nihal Jain, Murali Krishna Ramanathan, Ramesh Nallapati, Parminder Bhatia, Dan Roth, and Bing Xiang. Crosscodeeval: A diverse and multilingual benchmark for cross-file code completion. *CoRR*, abs/2310.11248, 2023. doi: 10.48550/ARXIV.2310.11248. URL https://doi.org/10.48550/arXiv.2310.11248.

Yangruibo Ding, Marcus J. Min, Gail E. Kaiser, and Baishakhi Ray. CYCLE: learning to self-refine the code generation. *CoRR*, abs/2403.18746, 2024. doi: 10.48550/ARXIV.2403.18746. URL https://doi.org/10.48550/arXiv.2403.18746.

Brendan Dolan-Gavitt, Patrick Hulin, Engin Kirda, Tim Leek, Andrea Mambretti, William K. Robertson, Frederick Ulrich, and Ryan Whelan. LAVA: large-scale automated vulnerability addition. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pp. 110–121. IEEE Computer Society, 2016. doi: 10.1109/SP.2016.15. URL https://doi.org/10.1109/SP.2016.15.

Jinhao Dong, Yiling Lou, Qihao Zhu, Zeyu Sun, Zhilin Li, Wenjie Zhang, and Dan Hao. FIRA: fine-grained graph-based code change representation for automated commit message generation. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pp. 970–981. ACM, 2022. doi: 10.1145/3510003.3510069. URL https://doi.org/10.1145/3510003.3510069.

Li Dong and Mirella Lapata. Coarse-to-fine decoding for neural semantic parsing. In Iryna Gurevych and Yusuke Miyao (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pp. 731–742. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-1068. URL https://aclanthology.org/P18-1068/.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B.

Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 13042–13054, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/c20bb2d9a50d5ac1f713f8b34d9aac5a-Abstract.html.

Yihong Dong, Xue Jiang, Zhi Jin, and Ge Li. Self-collaboration code generation via chatgpt. *CoRR*, abs/2304.07590, 2023. doi: 10.48550/ARXIV.2304.07590. URL https://doi.org/10.48550/arXiv.2304.07590.

Shihan Dou, Junjie Shan, Haoxiang Jia, Wenhao Deng, Zhiheng Xi, Wei He, Yueming Wu, Tao Gui, Yang Liu, and Xuanjing Huang. Towards understanding the capability of large language models on code clone detection: A survey. *CoRR*, abs/2308.01191, 2023. doi: 10.48550/arXiv.2308.01191. URL https://doi.org/10.48550/arXiv.2308.01191.

Shihan Dou, Yan Liu, Haoxiang Jia, Limao Xiong, Enyu Zhou, Wei Shen, Junjie Shan, Caishuang Huang, Xiao Wang, Xiaoran Fan, Zhiheng Xi, Yuhao Zhou, Tao Ji, Rui Zheng, Qi Zhang, Xuanjing Huang, and Tao Gui. Stepcoder: Improve code generation with reinforcement learning from compiler feedback. *CoRR*, abs/2402.01391, 2024. doi: 10.48550/ARXIV.2402.01391. URL https://doi.org/10.48550/arXiv.2402.01391.

Dawn Drain, Colin B. Clement, Guillermo Serrato, and Neel Sundaresan. Deepdebug: Fixing python bugs using stack traces, backtranslation, and code skeletons. *CoRR*, abs/2105.09352, 2021. URL https://arxiv.org/abs/2105.09352.

Mehdi Drissi, Olivia Watkins, Aditya Khant, Vivaswat Ojha, Pedro Sandoval Segura, Rakia Segev, Eric Weiner, and Robert Keller. Program language translation using a grammar-driven tree-to-tree model. *CoRR*, abs/1807.01784, 2018. URL http://arxiv.org/abs/1807.01784.

Iddo Drori and Nakul Verma. Solving linear algebra by program synthesis. *CoRR*, abs/2111.08171, 2021. URL https://arxiv.org/abs/2111.08171.

Iddo Drori, Sarah Zhang, Reece Shuttleworth, Leonard Tang, Albert Lu, Elizabeth Ke, Kevin Liu, Linda Chen, Sunny Tran, Newman Cheng, Roman Wang, Nikhil Singh, Taylor L. Patti, Jayson Lynch, Avi Shporer, Nakul Verma, Eugene Wu, and Gilbert Strang. A neural network solves, explains, and generates university math problems by program synthesis and few-shot learning at human level. *Proceedings of the National Academy of Sciences (PNAS)*, 119(32), 2022.

Lun Du, Xiaozhou Shi, Yanlin Wang, Ensheng Shi, Shi Han, and Dongmei Zhang. Is a single model enough? mucos: A multi-model ensemble learning approach for semantic code search. In Gianluca Demartini, Guido Zuccon, J. Shane Culpepper, Zi Huang, and Hanghang Tong (eds.), *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, pp. 2994–2998. ACM, 2021. doi: 10.1145/3459637.3482127. URL https://doi.org/10.1145/3459637.3482127.

Min Du and Feifei Li. Spell: Streaming parsing of system event logs. In Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu (eds.), *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, pp. 859–864. IEEE Computer Society, 2016. doi: 10.1109/ICDM.2016.0103. URL https://doi.org/10.1109/ICDM.2016.0103.

Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (eds.), *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pp. 1285–1298. ACM, 2017. doi: 10.1145/3133956.3134015. URL https://doi.org/10.1145/3133956.3134015.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P. Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen S. Meier-Hellstern,

Toju Duke, Lucas Dixon, Kun Zhang, Quoc V. Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. Glam: Efficient scaling of language models with mixture-of-experts. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5547–5569. PMLR, 2022. URL https://proceedings.mlr.press/v162/du22c.html.

Shubhang Shekhar Dvivedi, Vyshnav Vijay, Sai Leela Rahul Pujari, Shoumik Lodh, and Dhruv Kumar. A comparative analysis of large language models for code documentation generation. *CoRR*, abs/2312.10349, 2023. doi: 10.48550/ARXIV.2312.10349. URL https://doi.org/10.48550/arXiv.2312.10349.

Meryem Elallaoui, Khalid Nafil, and Raja Touahni. Automatic transformation of user stories into UML use case diagrams using NLP techniques. In Elhadi M. Shakshuki and Ansar-Ul-Haque Yasar (eds.), *The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018) / The 8th International Conference on Sustainable Energy Information Technology (SEIT 2018) / Affiliated Workshops, May 8-11, 2018, Porto, Portugal*, volume 130 of *Procedia Computer Science*, pp. 42–49. Elsevier, 2018. doi: 10.1016/J.PROCS.2018.04.010. URL https://doi.org/10.1016/j.procs.2018.04.010.

Aleksandra Eliseeva, Yaroslav Sokolov, Egor Bogomolov, Yaroslav Golubev, Danny Dig, and Timofey Bryksin. From commit message generation to history-aware commit message completion. In *38th IEEE/ACM International Conference on Automated Software Engineering, ASE 2023, Luxembourg, September 11-15, 2023*, pp. 723–735. IEEE, 2023. doi: 10.1109/ASE56229.2023.00078. URL https://doi.org/10.1109/ASE56229.2023.00078.

Jiahao Fan, Yi Li, Shaohua Wang, and Tien N. Nguyen. A C/C++ code vulnerability dataset with code changes and CVE summaries. In Sunghun Kim, Georgios Gousios, Sarah Nadi, and Joseph Hejderup (eds.), *MSR '20: 17th International Conference on Mining Software Repositories, Seoul, Republic of Korea, 29-30 June, 2020*, pp. 508–512. ACM, 2020. doi: 10.1145/3379597.3387501. URL https://doi.org/10.1145/3379597.3387501.

Zhiyu Fan, Xiang Gao, Martin Mirchev, Abhik Roychoudhury, and Shin Hwei Tan. Automated repair of programs from large language models. In *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*, pp. 1469–1481. IEEE, 2023. doi: 10.1109/ICSE48619.2023.00128. URL https://doi.org/10.1109/ICSE48619.2023.00128.

Chunrong Fang, Zixi Liu, Yangyang Shi, Jeff Huang, and Qingkai Shi. Functional code clone detection with syntax and semantics fusion learning. In Sarfraz Khurshid and Corina S. Pasareanu (eds.), *ISSTA '20: 29th ACM SIGSOFT International Symposium on Software Testing and Analysis, Virtual Event, USA, July 18-22, 2020*, pp. 516–527. ACM, 2020. doi: 10.1145/3395363.3397362. URL https://doi.org/10.1145/3395363.3397362.

Sen Fang, Youshuai Tan, Tao Zhang, and Yepang Liu. Self-attention networks for code search. *Inf. Softw. Technol.*, 134:106542, 2021. doi: 10.1016/J.INFSOF.2021.106542. URL https://doi.org/10.1016/j.infsof.2021.106542.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23:120:1–120:39, 2022. URL http://jmlr.org/papers/v23/21-0998.html.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: a theoretical perspective. *CoRR*, abs/2305.15408, 2023a. doi: 10.48550/ARXIV.2305.15408. URL https://doi.org/10.48550/arXiv.2305.15408.

K. J. Kevin Feng, Q. Vera Liao, Ziang Xiao, Jennifer Wortman Vaughan, Amy X. Zhang, and David W. McDonald. Canvil: Designerly adaptation for llm-powered user experiences. *CoRR*, abs/2401.09051, 2024. doi: 10.48550/ARXIV.2401.09051. URL https://doi.org/10.48550/arXiv.2401.09051.

Weixi Feng, Wanrong Zhu, Tsu-Jui Fu, Varun Jampani, Arjun R. Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023b. URL http://papers.nips.cc/paper_files/paper/2023/hash/3a7f9e485845dac27423375c934cb4db-Abstract-Conference.html.

Yu Feng, Ruben Martins, Osbert Bastani, and Isil Dillig. Program synthesis using conflict-driven learning. In Jeffrey S. Foster and Dan Grossman (eds.), *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2018, Philadelphia, PA, USA, June 18-22, 2018*, pp. 420–435. ACM, 2018. doi: 10.1145/3192366.3192382. URL https://doi.org/10.1145/3192366.3192382.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. Codebert: A pre-trained model for programming and natural languages. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pp. 1536–1547. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.139. URL https://doi.org/10.18653/v1/2020.findings-emnlp.139.

Patrick Fernandes, Miltiadis Allamanis, and Marc Brockschmidt. Structured neural summarization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=H1ersoRqtm.

Alessio Ferrari, Felice Dell'Orletta, Giorgio Oronzo Spagnolo, and Stefania Gnesi. Measuring and improving the completeness of natural language requirements. In Camille Salinesi and Inge van de Weerd (eds.), *Requirements Engineering: Foundation for Software Quality - 20th International Working Conference, REFSQ 2014, Essen, Germany, April 7-10, 2014. Proceedings*, volume 8396 of *Lecture Notes in Computer Science*, pp. 23–38. Springer, 2014. doi: 10.1007/978-3-319-05843-6_3. URL https://doi.org/10.1007/978-3-319-05843-6_3.

Alessio Ferrari, Sallam Abualhaija, and Chetan Arora. Model generation from requirements with llms: an exploratory study. *CoRR*, abs/2404.06371, 2024. doi: 10.48550/ARXIV.2404.06371. URL https://doi.org/10.48550/arXiv.2404.06371.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir R. Radev. Improving text-to-sql evaluation methodology. In Iryna Gurevych and Yusuke Miyao (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pp. 351–360. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-1033. URL https://aclanthology.org/P18-1033/.

Georgia Frantzeskou, Stephen G. MacDonell, Efstathios Stamatatos, Stelios Georgiou, and Stefanos Gritzalis. The significance of user-defined identifiers in java source code authorship identification. *Comput. Syst. Sci. Eng.*, 26(2), 2011.

Gordon Fraser and Andrea Arcuri. Evosuite: automatic test suite generation for object-oriented software. In Tibor Gyimóthy and Andreas Zeller (eds.), *SIGSOFT/FSE'11 19th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-19) and ESEC'11: 13th European Software Engineering Conference (ESEC-13), Szeged, Hungary, September 5-9, 2011*, pp. 416–419. ACM, 2011. doi: 10.1145/2025113.2025179. URL https://doi.org/10.1145/2025113.2025179.

Gordon Fraser and Andrea Arcuri. A large-scale evaluation of automated unit test generation using evosuite. *ACM Trans. Softw. Eng. Methodol.*, 24(2):8:1–8:42, 2014. doi: 10.1145/2685612. URL https://doi.org/10.1145/2685612.

Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Scott Yih, Luke Zettlemoyer, and Mike Lewis. Incoder: A generative model for code infilling and synthesis. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=hQwb-lbM6EL.

Lingyue Fu, Huacan Chai, Shuang Luo, Kounianhua Du, Weiming Zhang, Longteng Fan, Jiayi Lei, Renting Rui, Jianghao Lin, Yuchen Fang, Yifan Liu, Jingkuan Wang, Siyuan Qi, Kangning Zhang, Weinan Zhang, and Yong Yu. Codeapex: A bilingual programming evaluation benchmark for large language models. *CoRR*, abs/2309.01940, 2023. doi: 10.48550/arXiv.2309.01940. URL https://doi.org/10.48550/arXiv.2309.01940.

Michael Fu and Chakkrit Tantithamthavorn. Linevul: A transformer-based line-level vulnerability prediction. In *19th IEEE/ACM International Conference on Mining Software Repositories, MSR 2022, Pittsburgh, PA, USA, May 23-24, 2022*, pp. 608–620. ACM, 2022. doi: 10.1145/3524842.3528452. URL https://doi.org/10.1145/3524842.3528452.

Michael Fu, Chakkrit Tantithamthavorn, Trung Le, Van Nguyen, and Dinh Q. Phung. Vulrepair: a t5-based automated software vulnerability repair. In Abhik Roychoudhury, Cristian Cadar, and Miryung Kim (eds.), *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022, Singapore, Singapore, November 14-18, 2022*, pp. 935–947. ACM, 2022. doi: 10.1145/3540250.3549098. URL https://doi.org/10.1145/3540250.3549098.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. Scalable inference and training of context-rich syntactic translation models. In Nicoletta Calzolari, Claire Cardie, and Pierre Isabelle (eds.), *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*. The Association for Computer Linguistics, 2006. doi: 10.3115/1220175.1220296. URL https://aclanthology.org/P06-1121/.

Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. Towards robustness of text-to-sql models against synonym substitution. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 2505–2515. Association for Computational Linguistics, 2021a. doi: 10.18653/V1/2021.ACL-LONG.195. URL https://doi.org/10.18653/v1/2021.acl-long.195.

Yujian Gan, Xinyun Chen, and Matthew Purver. Exploring underexplored limitations of cross-domain text-to-sql generalization. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pp. 8926–8931. Association for Computational Linguistics, 2021b. doi: 10.18653/V1/2021.EMNLP-MAIN.702. URL https://doi.org/10.18653/v1/2021.emnlp-main.702.

Yujian Gan, Xinyun Chen, Qiuping Huang, and Matthew Purver. Measuring and improving compositional generalization in text-to-sql via component alignment. In Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz (eds.), *Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pp. 831–843. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.FINDINGS-NAACL.62. URL https://doi.org/10.18653/v1/2022.findings-naacl.62.

Chang Gao, Bowen Li, Wenxuan Zhang, Wai Lam, Binhua Li, Fei Huang, Luo Si, and Yongbin Li. Towards generalizable and robust text-to-sql parsing. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 2113–2125. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.FINDINGS-EMNLP.155. URL https://doi.org/10.18653/v1/2022.findings-emnlp.155.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. Text-to-sql empowered by large language models: A benchmark evaluation. *CoRR*, abs/2308.15363, 2023a. doi: 10.48550/ARXIV.2308.15363. URL https://doi.org/10.48550/arXiv.2308.15363.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027, 2021. URL https://arxiv.org/abs/2101.00027.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: program-aided language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10764–10799. PMLR, 2023b. URL https://proceedings.mlr.press/v202/gao23f.html.

Shuzheng Gao, Cuiyun Gao, Yulan He, Jichuan Zeng, Lunyiu Nie, Xin Xia, and Michael R. Lyu. Code structure-guided transformer for source code summarization. *ACM Trans. Softw. Eng. Methodol.*, 32(1): 23:1–23:32, 2023c. doi: 10.1145/3522674. URL https://doi.org/10.1145/3522674.

Yuexiu Gao and Chen Lyu. M2TS: multi-scale multi-modal approach based on transformer for source code summarization. In Ayushi Rastogi, Rosalia Tufano, Gabriele Bavota, Venera Arnaoudova, and Sonia Haiduc (eds.), *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension, ICPC 2022, Virtual Event, May 16-17, 2022*, pp. 24–35. ACM, 2022. doi: 10.1145/3524610.3527907. URL https://doi.org/10.1145/3524610.3527907.

Zeyu Gao, Hao Wang, Yuchen Zhou, Wenyu Zhu, and Chao Zhang. How far have we gone in vulnerability detection using large language models. *CoRR*, abs/2311.12420, 2023d. doi: 10.48550/ARXIV.2311.12420. URL https://doi.org/10.48550/arXiv.2311.12420.

Luca Gazzola, Daniela Micucci, and Leonardo Mariani. Automatic software repair: a survey. In Michel Chaudron, Ivica Crnkovic, Marsha Chechik, and Mark Harman (eds.), *Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, pp. 1219. ACM, 2018. doi: 10.1145/3180155.3182526. URL https://doi.org/10.1145/3180155.3182526.

Mingyang Geng, Shangwen Wang, Dezun Dong, Haotian Wang, Ge Li, Zhi Jin, Xiaoguang Mao, and Xiangke Liao. 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, ICSE 2024, Lisbon, Portugal, April 14-20, 2024*, pp. 39:1–39:13. ACM, 2024. doi: 10.1145/3597503.3608134. URL https://doi.org/10.1145/3597503.3608134.

Shalini Ghosh, Daniel Elenius, Wenchao Li, Patrick Lincoln, Natarajan Shankar, and Wilfried Steiner. ARSENAL: automatic requirements specification extraction from natural language. In Sanjai Rayadurgam and Oksana Tkachuk (eds.), *NASA Formal Methods - 8th International Symposium, NFM 2016, Minneapolis, MN, USA, June 7-9, 2016, Proceedings*, volume 9690 of *Lecture Notes in Computer Science*, pp. 41–46. Springer, 2016. doi: 10.1007/978-3-319-40648-0_4. URL https://doi.org/10.1007/978-3-319-40648-0_4.

Linyuan Gong, Mostafa Elhoushi, and Alvin Cheung. AST-T5: structure-aware pretraining for code generation and understanding. *CoRR*, abs/2401.03003, 2024. doi: 10.48550/ARXIV.2401.03003. URL https://doi.org/10.48550/arXiv.2401.03003.

M. Gopinath and Sibi Chakkaravarthy Sethuraman. A comprehensive survey on deep learning based malware detection techniques. *Comput. Sci. Rev.*, 47:100529, 2023. doi: 10.1016/J.COSREV.2022.100529. URL https://doi.org/10.1016/j.cosrev.2022.100529.

Claire Le Goues, Neal J. Holtschulte, Edward K. Smith, Yuriy Brun, Premkumar T. Devanbu, Stephanie Forrest, and Westley Weimer. The manybugs and introclass benchmarks for automated repair of C programs. *IEEE Trans. Software Eng.*, 41(12):1236–1256, 2015. doi: 10.1109/TSE.2015.2454513. URL https://doi.org/10.1109/TSE.2015.2454513.

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzmán, and Angela Fan. The flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Trans. Assoc. Comput. Linguistics*, 10:522–538, 2022. doi: 10.1162/TACL_A_00474. URL https://doi.org/10.1162/tacl_a_00474.

Luca Di Grazia and Michael Pradel. Code search: A survey of techniques for finding code. *ACM Comput. Surv.*, 55(11):220:1–220:31, 2023. doi: 10.1145/3565971. URL https://doi.org/10.1145/3565971.

Dejan Grubisic, Chris Cummins, Volker Seeker, and Hugh Leather. Compiler generated feedback for large language models. *CoRR*, abs/2403.14714, 2024a. doi: 10.48550/ARXIV.2403.14714. URL https://doi.org/10.48550/arXiv.2403.14714.

Dejan Grubisic, Volker Seeker, Gabriel Synnaeve, Hugh Leather, John M. Mellor-Crummey, and Chris Cummins. Priority sampling of large language models for compilers. In *Proceedings of the 4th Workshop on Machine Learning and Systems, EuroMLSys 2024, Athens, Greece, 22 April 2024*, pp. 91–97. ACM, 2024b. doi: 10.1145/3642970.3655831. URL https://doi.org/10.1145/3642970.3655831.

Alex Gu, Baptiste Rozière, Hugh Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida I. Wang. Cruxeval: A benchmark for code reasoning, understanding and execution. *CoRR*, abs/2401.03065, 2024. doi: 10.48550/ARXIV.2401.03065. URL https://doi.org/10.48550/arXiv.2401.03065.

Jian Gu, Zimin Chen, and Martin Monperrus. Multimodal representation for neural code search. In *IEEE International Conference on Software Maintenance and Evolution, ICSME 2021, Luxembourg, September 27 - October 1, 2021*, pp. 483–494. IEEE, 2021. doi: 10.1109/ICSME52107.2021.00049. URL https://doi.org/10.1109/ICSME52107.2021.00049.

Jiazhen Gu, Xuchuan Luo, Yangfan Zhou, and Xin Wang. Muffin: Testing deep learning libraries via neural architecture fuzzing. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pp. 1418–1430. ACM, 2022. doi: 10.1145/3510003.3510092. URL https://doi.org/10.1145/3510003.3510092.

Xiaodong Gu, Hongyu Zhang, Dongmei Zhang, and Sunghun Kim. Deepam: Migrate apis with multi-modal sequence to sequence learning. In Carles Sierra (ed.), *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pp. 3675–3681. ijcai.org, 2017. doi: 10.24963/IJCAI.2017/514. URL https://doi.org/10.24963/ijcai.2017/514.

Xiaodong Gu, Hongyu Zhang, and Sunghun Kim. Deep code search. In Michel Chaudron, Ivica Crnkovic, Marsha Chechik, and Mark Harman (eds.), *Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, pp. 933–944. ACM, 2018. doi: 10.1145/3180155.3180167. URL https://doi.org/10.1145/3180155.3180167.

Batu Guan, Yao Wan, Zhangqian Bi, Zheng Wang, Hongyu Zhang, Yulei Sui, Pan Zhou, and Lichao Sun. Codeip: A grammar-guided multi-bit watermark for large language models of code. *CoRR*, abs/2404.15639, 2024. doi: 10.48550/ARXIV.2404.15639. URL https://doi.org/10.48550/arXiv.2404.15639.

Vitor Guilherme and Auri Vincenzi. An initial investigation of chatgpt unit test generation capability. In Awdren L. Fontão, Débora M. B. Paiva, Hudson Borges, Maria Istela Cagnin, Patrícia Gomes Fernandes, Vanessa Borges, Silvana M. Melo, Vinicius H. S. Durelli, and Edna Dias Canedo (eds.), *8th Brazilian Symposium on Systematic and Automated Software Testing, SAST 2023, Campo Grande, MS, Brazil, September 25-29, 2023*, pp. 15–24. ACM, 2023. doi: 10.1145/3624032.3624035. URL https://doi.org/10.1145/3624032.3624035.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need. *CoRR*, abs/2306.11644, 2023. doi: 10.48550/arXiv.2306.11644. URL https://doi.org/10.48550/arXiv.2306.11644.

Tugçe Günes and Fatma Basak Aydemir. Automated goal model extraction from user stories using NLP. In Travis D. Breaux, Andrea Zisman, Samuel Fricker, and Martin Glinz (eds.), *28th IEEE International Requirements Engineering Conference, RE 2020, Zurich, Switzerland, August 31 - September 4, 2020*, pp. 382–387. IEEE, 2020. doi: 10.1109/RE48521.2020.00052. URL https://doi.org/10.1109/RE48521.2020.00052.

Chunxi Guo, Zhiliang Tian, Jintao Tang, Shasha Li, Zhihua Wen, Kaixuan Wang, and Ting Wang. Retrieval-augmented gpt-3.5-based text-to-sql framework with sample-aware prompting and dynamic revision chain. In Biao Luo, Long Cheng, Zheng-Guang Wu, Hongyi Li, and Chaojie Li (eds.), *Neural Information Processing - 30th International Conference, ICONIP 2023, Changsha, China, November 20-23, 2023, Proceedings, Part VI*, volume 14452 of *Lecture Notes in Computer Science*, pp. 341–356. Springer, 2023a. doi: 10.1007/978-981-99-8076-5_25. URL https://doi.org/10.1007/978-981-99-8076-5_25.

Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, Shujie Liu, Long Zhou, Nan Duan, Alexey Svyatkovskiy, Shengyu Fu, Michele Tufano, Shao Kun Deng, Colin B. Clement, Dawn Drain, Neel Sundaresan, Jian Yin, Daxin Jiang, and Ming Zhou. Graphcodebert: Pre-training code representations with data flow. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021a. URL https://openreview.net/forum?id=jLoC4ez43PZ.

Daya Guo, Shuai Lu, Nan Duan, Yanlin Wang, Ming Zhou, and Jian Yin. Unixcoder: Unified cross-modal pre-training for code representation. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 7212–7225. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.acl-long.499. URL https://doi.org/10.18653/v1/2022.acl-long.499.

Daya Guo, Canwen Xu, Nan Duan, Jian Yin, and Julian J. McAuley. Longcoder: A long-range pre-trained language model for code completion. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 12098–12107. PMLR, 2023b. URL https://proceedings.mlr.press/v202/guo23j.html.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. Deepseek-coder: When the large language model meets programming - the rise of code intelligence. *CoRR*, abs/2401.14196, 2024. doi: 10.48550/ARXIV.2401.14196. URL https://doi.org/10.48550/arXiv.2401.14196.

Haixuan Guo, Shuhan Yuan, and Xintao Wu. Logbert: Log anomaly detection via BERT. In *International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, July 18-22, 2021*, pp. 1–8. IEEE, 2021b. doi: 10.1109/IJCNN52387.2021.9534113. URL https://doi.org/10.1109/IJCNN52387.2021.9534113.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. Towards complex text-to-sql in cross-domain database with intermediate representation. In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 4524–4535. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1444. URL https://doi.org/10.18653/v1/p19-1444.

Qianyu Guo, Xiaofei Xie, Yi Li, Xiaoyu Zhang, Yang Liu, Xiaohong Li, and Chao Shen. Audee: Automated testing for deep learning frameworks. In *35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020, Melbourne, Australia, September 21-25, 2020*, pp. 486–498. IEEE, 2020. doi: 10.1145/3324884.3416571. URL https://doi.org/10.1145/3324884.3416571.

Anshul Gupta and Neel Sundaresan. Intelligent code reviews using deep learning, 2018.

Rahul Gupta, Soham Pal, Aditya Kanade, and Shirish K. Shevade. Deepfix: Fixing common C language errors by deep learning. In Satinder Singh and Shaul Markovitch (eds.), *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pp. 1345–1351. AAAI Press, 2017. doi: 10.1609/aaai.v31i1.10742. URL https://doi.org/10.1609/aaai.v31i1.10742.

Izzeddin Gur, Ofir Nachum, Yingjie Miao, Mustafa Safdari, Austin Huang, Aakanksha Chowdhery, Sharan Narang, Noah Fiedel, and Aleksandra Faust. Understanding HTML with large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pp. 2803–2821. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-EMNLP.185. URL https://doi.org/10.18653/v1/2023.findings-emnlp.185.

Emitza Guzman, Mohamed Ibrahim, and Martin Glinz. A little bird told me: Mining tweets for requirements and software evolution. In Ana Moreira, João Araújo, Jane Hayes, and Barbara Paech (eds.), *25th IEEE International Requirements Engineering Conference, RE 2017, Lisbon, Portugal, September 4-8, 2017*, pp. 11–20. IEEE Computer Society, 2017. doi: 10.1109/RE.2017.88. URL https://doi.org/10.1109/RE.2017.88.

Péter Gyimesi, Béla Vancsics, Andrea Stocco, Davood Mazinanian, Árpád Beszédes, Rudolf Ferenc, and Ali Mesbah. Bugsjs: a benchmark of javascript bugs. In *12th IEEE Conference on Software Testing, Validation and Verification, ICST 2019, Xi'an, China, April 22-27, 2019*, pp. 90–101. IEEE, 2019. doi: 10.1109/ICST.2019.00019. URL https://doi.org/10.1109/ICST.2019.00019.

Rajarshi Haldar and Julia Hockenmaier. Analyzing the performance of large language models on code summarization. *CoRR*, abs/2404.08018, 2024. doi: 10.48550/ARXIV.2404.08018. URL https://doi.org/10.48550/arXiv.2404.08018.

Patrick Haller, Jonas Golde, and Alan Akbik. PECC: problem extraction and coding challenges. In Nicoletta Calzolari, Min-Yen Kan, Véronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue (eds.), *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation, LREC/COLING 2024, 20-25 May, 2024, Torino, Italy*, pp. 12690–12699. ELRA and ICCL, 2024. URL https://aclanthology.org/2024.lrec-main.1111.

Sivana Hamer, Marcelo d'Amorim, and Laurie Williams. Just another copy and paste? comparing the security vulnerabilities of chatgpt generated code and stackoverflow answers. *CoRR*, abs/2403.15600, 2024. doi: 10.48550/ARXIV.2403.15600. URL https://doi.org/10.48550/arXiv.2403.15600.

Quinn Hanam, Fernando Santos De Mattos Brito, and Ali Mesbah. Discovering bug patterns in javascript. In Thomas Zimmermann, Jane Cleland-Huang, and Zhendong Su (eds.), *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2016, Seattle, WA, USA, November 13-18, 2016*, pp. 144–156. ACM, 2016. doi: 10.1145/2950290.2950308. URL https://doi.org/10.1145/2950290.2950308.

Hazim Hanif and Sergio Maffeis. Vulberta: Simplified source code pre-training for vulnerability detection. In *International Joint Conference on Neural Networks, IJCNN 2022, Padua, Italy, July 18-23, 2022*, pp. 1–8. IEEE, 2022. doi: 10.1109/IJCNN55064.2022.9892280. URL https://doi.org/10.1109/IJCNN55064.2022.9892280.

Yiyang Hao, Ge Li, Yongqiang Liu, Xiaowei Miao, He Zong, Siyuan Jiang, Yang Liu, and He Wei. Aixbench: A code generation benchmark dataset. *CoRR*, abs/2206.13179, 2022. doi: 10.48550/arXiv.2206.13179. URL https://doi.org/10.48550/arXiv.2206.13179.

Sakib Haque, Alexander LeClair, Lingfei Wu, and Collin McMillan. Improved automatic summarization of subroutines via attention to file context. In Sunghun Kim, Georgios Gousios, Sarah Nadi, and Joseph Hejderup (eds.), *MSR '20: 17th International Conference on Mining Software Repositories, Seoul, Republic of Korea, 29-30 June, 2020*, pp. 300–310. ACM, 2020. doi: 10.1145/3379597.3387449. URL https://doi.org/10.1145/3379597.3387449.

Nima Shiri Harzevili, Alvine Boaye Belle, Junjie Wang, Song Wang, Zhen Ming Jiang, and Nachiappan Nagappan. A survey on automated software vulnerability detection using machine learning and deep learning. *CoRR*, abs/2306.11673, 2023. doi: 10.48550/ARXIV.2306.11673. URL https://doi.org/10.48550/arXiv.2306.11673.

Mostafa Hassan, Caterina Urban, Marco Eilers, and Peter Müller. Maxsmt-based type inference for python 3. In Hana Chockler and Georg Weissenbacher (eds.), *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part II*, volume 10982 of *Lecture Notes in Computer Science*, pp. 12–19. Springer, 2018. doi: 10.1007/978-3-319-96142-2_2. URL https://doi.org/10.1007/978-3-319-96142-2_2.

Shirley Anugrah Hayati, Raphaël Olivier, Pravalika Avvaru, Pengcheng Yin, Anthony Tomasic, and Graham Neubig. Retrieval-based neural code generation. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 925–930. Association for Computational Linguistics, 2018. doi: 10.18653/V1/D18-1111. URL https://doi.org/10.18653/v1/d18-1111.

Moshe Hazoom, Vibhor Malik, and Ben Bogin. Text-to-sql in the wild: A naturally-occurring dataset based on stack exchange data. *CoRR*, abs/2106.05006, 2021. URL https://arxiv.org/abs/2106.05006.

Jingxuan He, Pesho Ivanov, Petar Tsankov, Veselin Raychev, and Martin T. Vechev. Debin: Predicting debug information in stripped binaries. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang (eds.), *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pp. 1667–1680. ACM, 2018. doi: 10.1145/3243734.3243866. URL https://doi.org/10.1145/3243734.3243866.

Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R. Lyu. Drain: An online log parsing approach with fixed depth tree. In Ilkay Altintas and Shiping Chen (eds.), *2017 IEEE International Conference on Web Services, ICWS 2017, Honolulu, HI, USA, June 25-30, 2017*, pp. 33–40. IEEE, 2017. doi: 10.1109/ICWS.2017.13. URL https://doi.org/10.1109/ICWS.2017.13.

Shilin He, Jieming Zhu, Pinjia He, and Michael R. Lyu. Loghub: A large collection of system log datasets towards automated log analytics. *CoRR*, abs/2008.06448, 2020. URL https://arxiv.org/abs/2008.06448.

Xinyi He, Jiaru Zou, Yun Lin, Mengyu Zhou, Shi Han, Zejian Yuan, and Dongmei Zhang. CONLINE: complex code generation and refinement with online searching and correctness testing. *CoRR*, abs/2403.13583, 2024. doi: 10.48550/ARXIV.2403.13583. URL https://doi.org/10.48550/arXiv.2403.13583.

Yichen He, Liran Wang, Kaiyi Wang, Yupeng Zhang, Hang Zhang, and Zhoujun Li. COME: commit message generation with modification embedding. In René Just and Gordon Fraser (eds.), *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2023, Seattle, WA, USA, July 17-21, 2023*, pp. 792–803. ACM, 2023a. doi: 10.1145/3597926.3598096. URL https://doi.org/10.1145/3597926.3598096.

Zhiwei He, Tian Liang, Wenxiang Jiao, Zhuosheng Zhang, Yujiu Yang, Rui Wang, Zhaopeng Tu, Shuming Shi, and Xing Wang. Exploring human-like translation strategy with large language models. *CoRR*, abs/2305.04118, 2023b. doi: 10.48550/ARXIV.2305.04118. URL https://doi.org/10.48550/arXiv.2305.04118.

Joy He-Yueya, Gabriel Poesia, Rose E. Wang, and Noah D. Goodman. Solving math word problems by combining language models with symbolic solvers. *CoRR*, abs/2304.09102, 2023. doi: 10.48550/ARXIV.2304.09102. URL https://doi.org/10.48550/arXiv.2304.09102.

Vincent J. Hellendoorn and Premkumar T. Devanbu. Are deep neural networks the best choice for modeling source code? In Eric Bodden, Wilhelm Schäfer, Arie van Deursen, and Andrea Zisman (eds.), *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, Paderborn, Germany, September 4-8, 2017*, pp. 763–773. ACM, 2017. doi: 10.1145/3106237.3106290. URL https://doi.org/10.1145/3106237.3106290.

Vincent J. Hellendoorn, Christian Bird, Earl T. Barr, and Miltiadis Allamanis. Deep learning type inference. In Gary T. Leavens, Alessandro Garcia, and Corina S. Pasareanu (eds.), *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2018, Lake Buena Vista, FL, USA, November 04-09, 2018*, pp. 152–162. ACM, 2018. doi: 10.1145/3236024.3236051. URL https://doi.org/10.1145/3236024.3236051.

Vincent J. Hellendoorn, Charles Sutton, Rishabh Singh, Petros Maniatis, and David Bieber. Global relational models of source code. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=B1lnbRNtwr.

Vincent J. Hellendoorn, Jason Tsay, Manisha Mukherjee, and Martin Hirzel. Towards automating code review at scale. In Diomidis Spinellis, Georgios Gousios, Marsha Chechik, and Massimiliano Di Penta (eds.), *ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, August 23-28, 2021*, pp. 1479–1482. ACM, 2021. doi: 10.1145/3468264.3473134. URL https://doi.org/10.1145/3468264.3473134.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, USA, June 24-27, 1990*. Morgan Kaufmann, 1990. URL https://aclanthology.org/H90-1021/.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS. In Joaquin Vanschoren and Sai-Kit Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021a. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/c24cd76e1ce41366a4bbe8a49b02a028-Abstract-round2.html.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021b. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Hatem Herchi and Wahiba Ben Abdessalem. From user requirements to UML class diagram. *CoRR*, abs/1211.0713, 2012. URL http://arxiv.org/abs/1211.0713.

Geert Heyman and Tom Van Cutsem. Neural code search revisited: Enhancing code snippet retrieval through natural language intent. *CoRR*, abs/2008.12193, 2020. URL https://arxiv.org/abs/2008.12193.

Abram Hindle, Earl T. Barr, Zhendong Su, Mark Gabel, and Premkumar T. Devanbu. On the naturalness of software. In Martin Glinz, Gail C. Murphy, and Mauro Pezzè (eds.), *34th International Conference on Software Engineering, ICSE 2012, June 2-9, 2012, Zurich, Switzerland*, pp. 837–847. IEEE Computer Society, 2012. doi: 10.1109/ICSE.2012.6227135. URL https://doi.org/10.1109/ICSE.2012.6227135.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

Thong Hoang, Hong Jin Kang, David Lo, and Julia Lawall. Cc2vec: distributed representations of code changes. In Gregg Rothermel and Doo-Hwan Bae (eds.), *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, pp. 518–529. ACM, 2020. doi: 10.1145/3377811.3380361. URL https://doi.org/10.1145/3377811.3380361.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/c1e2faff6f588870935f114ebe04a3e5-Abstract-Conference.html.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, and Chenglin Wu. Metagpt: Meta programming for multi-agent collaborative framework. *CoRR*, abs/2308.00352, 2023. doi: 10.48550/ARXIV.2308.00352. URL https://doi.org/10.48550/arXiv.2308.00352.

Yang Hong, Chakkrit Tantithamthavorn, Patanamon Thongtanunam, and Aldeida Aleti. Commentfinder: a simpler, faster, more accurate code review comments recommendation. In Abhik Roychoudhury, Cristian Cadar, and Miryung Kim (eds.), *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022, Singapore, Singapore, November 14-18, 2022*, pp. 507–519. ACM, 2022. doi: 10.1145/3540250.3549119. URL https://doi.org/10.1145/3540250.3549119.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. Unnatural instructions: Tuning language models with (almost) no human labor. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 14409–14428. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.806. URL https://doi.org/10.18653/v1/2023.acl-long.806.

Iman Hosseini and Brendan Dolan-Gavitt. Beyond the C: retargetable decompilation using neural machine translation. *CoRR*, abs/2212.08950, 2022. doi: 10.48550/ARXIV.2212.08950. URL https://doi.org/10.48550/arXiv.2212.08950.

Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John C. Grundy, and Haoyu Wang. Large language models for software engineering: A systematic literature review. *CoRR*, abs/2308.10620, 2023. doi: 10.48550/ARXIV.2308.10620. URL https://doi.org/10.48550/arXiv.2308.10620.

Gang Hu, Min Peng, Yihan Zhang, Qianqian Xie, and Mengting Yuan. Neural joint attention code search over structure embeddings for software q&a sites. *J. Syst. Softw.*, 170:110773, 2020. doi: 10.1016/J.JSS.2020.110773. URL https://doi.org/10.1016/j.jss.2020.110773.

Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. Deep code comment generation. In Foutse Khomh, Chanchal K. Roy, and Janet Siegmund (eds.), *Proceedings of the 26th Conference on Program Comprehension, ICPC 2018, Gothenburg, Sweden, May 27-28, 2018*, pp. 200–210. ACM, 2018a. doi: 10.1145/3196321.3196334. URL https://doi.org/10.1145/3196321.3196334.

Xing Hu, Ge Li, Xin Xia, David Lo, Shuai Lu, and Zhi Jin. Summarizing source code with transferred API knowledge. In Jérôme Lang (ed.), *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pp. 2269–2275. ijcai.org, 2018b. doi: 10.24963/ijcai.2018/314. URL https://doi.org/10.24963/ijcai.2018/314.

Yang Hu, Umair Z. Ahmed, Sergey Mechtaev, Ben Leong, and Abhik Roychoudhury. Re-factoring based program repair applied to programming assignments. In *34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019, San Diego, CA, USA, November 11-15, 2019*, pp. 388–398. IEEE, 2019. doi: 10.1109/ASE.2019.00044. URL https://doi.org/10.1109/ASE.2019.00044.

Di Huang, Rui Zhang, Xing Hu, Xishan Zhang, Pengwei Jin, Nan Li, Zidong Du, Qi Guo, and Yunji Chen. Neural program synthesis with query. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022a. URL https://openreview.net/forum?id=NyJ2KIN8P17.

Dong Huang, Jianbo Dai, Han Weng, Puzhen Wu, Yuhao Qing, Jie M Zhang, Heming Cui, and Zhijiang Guo. Soap: Enhancing efficiency of generated code via self-optimization. 2024. URL https://doi.org/10.48550/arXiv.2405.15189.

Junjie Huang, Duyu Tang, Linjun Shou, Ming Gong, Ke Xu, Daxin Jiang, Ming Zhou, and Nan Duan. Cosqa: 20, 000+ web queries for code search and question answering. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 5690–5700. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.442. URL https://doi.org/10.18653/v1/2021.acl-long.442.

Junjie Huang, Chenglong Wang, Jipeng Zhang, Cong Yan, Haotian Cui, Jeevana Priya Inala, Colin B. Clement, Nan Duan, and Jianfeng Gao. Execution-based evaluation for data science code generation models. *CoRR*, abs/2211.09374, 2022b. doi: 10.48550/ARXIV.2211.09374. URL https://doi.org/10.48550/arXiv.2211.09374.

Kai Huang, Zhengzi Xu, Su Yang, Hongyu Sun, Xuejun Li, Zheng Yan, and Yuqing Zhang. A survey on automated program repair techniques. *CoRR*, abs/2303.18184, 2023a. doi: 10.48550/ARXIV.2303.18184. URL https://doi.org/10.48550/arXiv.2303.18184.

Qiao Huang, Xin Xia, Zhenchang Xing, David Lo, and Xinyu Wang. API method recommendation without worrying about the task-api knowledge gap. In Marianne Huchard, Christian Kästner, and Gordon Fraser (eds.), *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, Montpellier, France, September 3-7, 2018*, pp. 293–304. ACM, 2018. doi: 10.1145/3238147.3238191. URL https://doi.org/10.1145/3238147.3238191.

Yuan Huang, Nan Jia, Hao-Jie Zhou, Xiangping Chen, Zibin Zheng, and Mingdong Tang. Learning human-written commit messages to document code changes. *J. Comput. Sci. Technol.*, 35(6):1258–1277, 2020. doi: 10.1007/S11390-020-0496-0. URL https://doi.org/10.1007/s11390-020-0496-0.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *CoRR*, abs/2305.08322, 2023b. doi: 10.48550/arXiv.2305.08322. URL https://doi.org/10.48550/arXiv.2305.08322.

Faria Huq, Masum Hasan, Md. Mahim Anjum Haque, Sazan Mahbub, Anindya Iqbal, and Toufique Ahmed. Review4repair: Code review aided automatic program repairing. *Inf. Softw. Technol.*, 143:106765, 2022. doi: 10.1016/J.INFSOF.2021.106765. URL https://doi.org/10.1016/j.infsof.2021.106765.

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. Codesearchnet challenge: Evaluating the state of semantic code search. *CoRR*, abs/1909.09436, 2019. URL http://arxiv.org/abs/1909.09436.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. A comprehensive exploration on wikisql with table-aware word contextualization. *CoRR*, abs/1902.01069, 2019. URL http://arxiv.org/abs/1902.01069.

Mohd Ibrahim and Rodina Ahmad. Class diagram extraction from textual requirements using natural language processing (nlp) techniques. In *2010 Second International Conference on Computer Research and Development*, pp. 200–204, 2010. doi: 10.1109/ICCRD.2010.71.

Yoichi Ishibashi and Yoshimasa Nishimura. Self-organized agents: A llm multi-agent framework toward ultra large-scale code generation and optimization. *CoRR*, abs/2404.02183, 2024. doi: 10.48550/ARXIV.2404.02183. URL https://doi.org/10.48550/arXiv.2404.02183.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016. doi: 10.18653/v1/p16-1195. URL https://doi.org/10.18653/v1/p16-1195.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. Learning a neural semantic parser from user feedback. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pp. 963–973. Association for Computational Linguistics, 2017. doi: 10.18653/V1/P17-1089. URL https://doi.org/10.18653/v1/P17-1089.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. Mapping language to code in programmatic context. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 1643–1652. Association for Computational Linguistics, 2018. doi: 10.18653/v1/d18-1192. URL https://doi.org/10.18653/v1/d18-1192.

Srinivasan Iyer, Xi Victoria Lin, Ramakanth Pasunuru, Todor Mihaylov, Daniel Simig, Ping Yu, Kurt Shuster, Tianlu Wang, Qing Liu, Punit Singh Koura, Xian Li, Brian O'Horo, Gabriel Pereyra, Jeff Wang, Christopher Dewan, Asli Celikyilmaz, Luke Zettlemoyer, and Ves Stoyanov. OPT-IML: scaling language model instruction meta learning through the lens of generalization. *CoRR*, abs/2212.12017, 2022. doi: 10.48550/arXiv.2212.12017. URL https://doi.org/10.48550/arXiv.2212.12017.

Chirag Jain, Preethu Rose Anish, Amrita Singh, and Smita Ghaisas. A transformer-based approach for abstractive summarization of requirements from obligations in software engineering contracts. In Kurt Schneider, Fabiano Dalpiaz, and Jennifer Horkoff (eds.), *31st IEEE International Requirements Engineering Conference, RE 2023, Hannover, Germany, September 4-8, 2023*, pp. 169–179. IEEE, 2023. doi: 10.1109/RE57278.2023.00025. URL https://doi.org/10.1109/RE57278.2023.00025.

Naman Jain, Skanda Vaidyanath, Arun Shankar Iyer, Nagarajan Natarajan, Suresh Parthasarathy, Sriram K. Rajamani, and Rahul Sharma. Jigsaw: Large language models meet program synthesis. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pp. 1219–1231. ACM, 2022. doi: 10.1145/3510003.3510203. URL https://doi.org/10.1145/3510003.3510203.

Prithwish Jana, Piyush Jha, Haoyang Ju, Gautham Kishore, Aryan Mahajan, and Vijay Ganesh. Attention, compilation, and solver-based symbolic analysis are all you need. *CoRR*, abs/2306.06755, 2023. doi: 10.48550/ARXIV.2306.06755. URL https://doi.org/10.48550/arXiv.2306.06755.

Abhinav Jangda and Gaurav Anand. Predicting variable types in dynamically typed programming languages. *CoRR*, abs/1901.05138, 2019. URL http://arxiv.org/abs/1901.05138.

Kevin Jesse and Premkumar T. Devanbu. Manytypes4typescript: A comprehensive typescript dataset for sequence-based type inference. In *19th IEEE/ACM International Conference on Mining Software Repositories, MSR 2022, Pittsburgh, PA, USA, May 23-24, 2022*, pp. 294–298. ACM, 2022. doi: 10.1145/3524842.3528507. URL https://doi.org/10.1145/3524842.3528507.

Kevin Jesse, Premkumar T. Devanbu, and Toufique Ahmed. Learning type annotation: is big data enough? In Diomidis Spinellis, Georgios Gousios, Marsha Chechik, and Massimiliano Di Penta (eds.), *ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, August 23-28, 2021*, pp. 1483–1486. ACM, 2021. doi: 10.1145/3468264.3473135. URL https://doi.org/10.1145/3468264.3473135.

Kevin Jesse, Toufique Ahmed, Premkumar T. Devanbu, and Emily Morgan. Large language models and simple, stupid bugs. In *20th IEEE/ACM International Conference on Mining Software Repositories, MSR 2023, Melbourne, Australia, May 15-16, 2023*, pp. 563–575. IEEE, 2023. doi: 10.1109/MSR59073.2023.00082. URL https://doi.org/10.1109/MSR59073.2023.00082.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *CoRR*, abs/2310.06825, 2023a. doi: 10.48550/ARXIV.2310.06825. URL https://doi.org/10.48550/arXiv.2310.06825.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. *CoRR*, abs/2401.04088, 2024a. doi: 10.48550/ARXIV.2401.04088. URL https://doi.org/10.48550/arXiv.2401.04088.

Ellen Jiang, Kristen Olson, Edwin Toh, Alejandra Molina, Aaron Donsbach, Michael Terry, and Carrie J. Cai. Promptmaker: Prompt-based prototyping with large language models. In Simone D. J. Barbosa, Cliff Lampe, Caroline Appert, and David A. Shamma (eds.), *CHI '22: CHI Conference on Human Factors in Computing Systems, New Orleans, LA, USA, 29 April 2022 - 5 May 2022, Extended Abstracts*, pp. 35:1–35:8. ACM, 2022. doi: 10.1145/3491101.3503564. URL https://doi.org/10.1145/3491101.3503564.

Lin Jiang, Hui Liu, and He Jiang. Machine learning based recommendation of method names: How far are we. In *34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019, San Diego, CA, USA, November 11-15, 2019*, pp. 602–614. IEEE, 2019. doi: 10.1109/ASE.2019.00062. URL https://doi.org/10.1109/ASE.2019.00062.

Lingxiao Jiang, Ghassan Misherghi, Zhendong Su, and Stéphane Glondu. DECKARD: scalable and accurate tree-based detection of code clones. In *29th International Conference on Software Engineering (ICSE 2007), Minneapolis, MN, USA, May 20-26, 2007*, pp. 96–105. IEEE Computer Society, 2007. doi: 10.1109/ICSE.2007.30. URL https://doi.org/10.1109/ICSE.2007.30.

Nan Jiang, Thibaud Lutellier, and Lin Tan. CURE: code-aware neural machine translation for automatic program repair. In *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*, pp. 1161–1173. IEEE, 2021a. doi: 10.1109/ICSE43902.2021.00107. URL https://doi.org/10.1109/ICSE43902.2021.00107.

Nan Jiang, Kevin Liu, Thibaud Lutellier, and Lin Tan. Impact of code language models on automated program repair. In *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*, pp. 1430–1442. IEEE, 2023b. doi: 10.1109/ICSE48619.2023.00125. URL https://doi.org/10.1109/ICSE48619.2023.00125.

Nan Jiang, Chengxiao Wang, Kevin Liu, Xiangzhe Xu, Lin Tan, and Xiangyu Zhang. Nova$^+$: Generative language models for binaries. *CoRR*, abs/2311.13721, 2023c. doi: 10.48550/ARXIV.2311.13721. URL https://doi.org/10.48550/arXiv.2311.13721.

Siyuan Jiang and Collin McMillan. Towards automatic generation of short summaries of commits. In Giuseppe Scanniello, David Lo, and Alexander Serebrenik (eds.), *Proceedings of the 25th International Conference on Program Comprehension, ICPC 2017, Buenos Aires, Argentina, May 22-23, 2017*, pp. 320–323. IEEE Computer Society, 2017. doi: 10.1109/ICPC.2017.12. URL https://doi.org/10.1109/ICPC.2017.12.

Siyuan Jiang, Ameer Armaly, and Collin McMillan. Automatically generating commit messages from diffs using neural machine translation. In Grigore Rosu, Massimiliano Di Penta, and Tien N. Nguyen (eds.), *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, ASE 2017, Urbana, IL, USA, October 30 - November 03, 2017*, pp. 135–146. IEEE Computer Society, 2017. doi: 10.1109/ASE.2017.8115626. URL https://doi.org/10.1109/ASE.2017.8115626.

Xue Jiang, Zhuoran Zheng, Chen Lyu, Liang Li, and Lei Lyu. Treebert: A tree-based pre-trained model for programming language. In Cassio P. de Campos, Marloes H. Maathuis, and Erik Quaeghebeur (eds.), *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence, UAI 2021, Virtual Event, 27-30 July 2021*, volume 161 of *Proceedings of Machine Learning Research*, pp. 54–63. AUAI Press, 2021b. URL https://proceedings.mlr.press/v161/jiang21a.html.

Yufan Jiang, Qiaozhi He, Xiaomin Zhuang, and Zhihua Wu. Code comparison tuning for code large language models. *CoRR*, abs/2403.19121, 2024b. doi: 10.48550/ARXIV.2403.19121. URL https://doi.org/10.48550/arXiv.2403.19121.

Zhihan Jiang, Jinyang Liu, Zhuangbin Chen, Yichen Li, Junjie Huang, Yintong Huo, Pinjia He, Jiazhen Gu, and Michael R. Lyu. Lilac: Log parsing using llms with adaptive parsing cache. *CoRR*, abs/2310.01796, 2023d. doi: 10.48550/ARXIV.2310.01796. URL https://doi.org/10.48550/arXiv.2310.01796.

Zhihan Jiang, Jinyang Liu, Junjie Huang, Yichen Li, Yintong Huo, Jiazhen Gu, Zhuangbin Chen, Jieming Zhu, and Michael R. Lyu. A large-scale benchmark for log parsing. *CoRR*, abs/2308.10828, 2023e. doi: 10.48550/ARXIV.2308.10828. URL https://doi.org/10.48550/arXiv.2308.10828.

Mingsheng Jiao, Tingrui Yu, Xuan Li, Guanjie Qiu, Xiaodong Gu, and Beijun Shen. On the evaluation of neural code translation: Taxonomy and benchmark. *CoRR*, abs/2308.08961, 2023. doi: 10.48550/ARXIV.2308.08961. URL https://doi.org/10.48550/arXiv.2308.08961.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *CoRR*, abs/2310.06770, 2023. doi: 10.48550/ARXIV.2310.06770. URL https://doi.org/10.48550/arXiv.2310.06770.

Timo Johann, Christoph Stanik, Alireza M. Alizadeh B., and Walid Maalej. SAFE: A simple approach for feature extraction from app descriptions and app reviews. In Ana Moreira, João Araújo, Jane Hayes, and Barbara Paech (eds.), *25th IEEE International Requirements Engineering Conference, RE 2017, Lisbon, Portugal, September 4-8, 2017*, pp. 21–30. IEEE Computer Society, 2017. doi: 10.1109/RE.2017.71. URL https://doi.org/10.1109/RE.2017.71.

Harshit Joshi, José Pablo Cambronero Sánchez, Sumit Gulwani, Vu Le, Gust Verbruggen, and Ivan Radicek. Repair is nearly generation: Multilingual program repair with llms. In Brian Williams, Yiling Chen, and Jennifer Neville (eds.), *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pp. 5131–5140. AAAI Press, 2023. doi: 10.1609/AAAI.V37I4.25642. URL https://doi.org/10.1609/aaai.v37i4.25642.

Tae-Hwan Jung. Commitbert: Commit message generation using pre-trained programming language model. *CoRR*, abs/2105.14242, 2021. URL https://arxiv.org/abs/2105.14242.

René Just. The major mutation framework: efficient and scalable mutation analysis for java. In Corina S. Pasareanu and Darko Marinov (eds.), *International Symposium on Software Testing and Analysis, ISSTA '14, San Jose, CA, USA - July 21 - 26, 2014*, pp. 433–436. ACM, 2014. doi: 10.1145/2610384.2628053. URL https://doi.org/10.1145/2610384.2628053.

René Just, Darioush Jalali, and Michael D. Ernst. Defects4j: a database of existing faults to enable controlled testing studies for java programs. In Corina S. Pasareanu and Darko Marinov (eds.), *International Symposium on Software Testing and Analysis, ISSTA '14, San Jose, CA, USA - July 21 - 26, 2014*, pp. 437–440. ACM, 2014. doi: 10.1145/2610384.2628055. URL https://doi.org/10.1145/2610384.2628055.

Vaibhavi Kalgutkar, Ratinder Kaur, Hugo Gonzalez, Natalia Stakhanova, and Alina Matyukhina. Code authorship attribution: Methods and challenges. *ACM Comput. Surv.*, 52(1):3:1–3:36, 2019. doi: 10.1145/3292577. URL https://doi.org/10.1145/3292577.

Ashwin Kalyan, Abhishek Mohta, Oleksandr Polozov, Dhruv Batra, Prateek Jain, and Sumit Gulwani. Neural-guided deductive search for real-time program synthesis from examples. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rywDjg-RW.

Aditya Kanade, Petros Maniatis, Gogul Balakrishnan, and Kensen Shi. Learning and evaluating contextual embedding of source code. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5110–5121. PMLR, 2020. URL http://proceedings.mlr.press/v119/kanade20a.html.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. URL https://arxiv.org/abs/2001.08361.

Svetoslav Karaivanov, Veselin Raychev, and Martin T. Vechev. Phrase-based statistical translation of programming languages. In Andrew P. Black, Shriram Krishnamurthi, Bernd Bruegge, and Joseph N. Ruskiewicz (eds.), *Onward! 2014, Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software, part of SPLASH '14, Portland, OR, USA, October 20-24, 2014*, pp. 173–184. ACM, 2014. doi: 10.1145/2661136.2661148. URL https://doi.org/10.1145/2661136.2661148.

Rafael-Michael Karampatsis and Charles Sutton. How often do single-statement bugs occur?: The manysstubs4j dataset. In Sunghun Kim, Georgios Gousios, Sarah Nadi, and Joseph Hejderup (eds.), *MSR '20: 17th International Conference on Mining Software Repositories, Seoul, Republic of Korea, 29-30 June, 2020*, pp. 573–577. ACM, 2020. doi: 10.1145/3379597.3387491. URL https://doi.org/10.1145/3379597.3387491.

Rafael-Michael Karampatsis, Hlib Babii, Romain Robbes, Charles Sutton, and Andrea Janes. Big code != big vocabulary: open-vocabulary models for source code. In Gregg Rothermel and Doo-Hwan Bae (eds.), *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, pp. 1073–1085. ACM, 2020. doi: 10.1145/3377811.3380342. URL https://doi.org/10.1145/3377811.3380342.

Amir Hossein Kargaran, Nafiseh Nikeghbal, Abbas Heydarnoori, and Hinrich Schütze. Menucraft: Interactive menu system design with large language models. *CoRR*, abs/2303.04496, 2023. doi: 10.48550/ARXIV.2303.04496. URL https://doi.org/10.48550/arXiv.2303.04496.

Vineeth Kashyap, Jason Ruchti, Lucja Kot, Emma Turetsky, Rebecca Swords, Shih An Pan, Julien Henry, David Melski, and Eric M. Schulte. Automated customized bug-benchmark generation. In *19th International Working Conference on Source Code Analysis and Manipulation, SCAM 2019, Cleveland, OH, USA, September 30 - October 1, 2019*, pp. 103–114. IEEE, 2019. doi: 10.1109/SCAM.2019.00020. URL https://doi.org/10.1109/SCAM.2019.00020.

George Katsogiannis-Meimarakis and Georgia Koutrika. A survey on deep learning approaches for text-to-sql. *VLDB J.*, 32(4):905–936, 2023. doi: 10.1007/S00778-022-00776-8. URL https://doi.org/10.1007/s00778-022-00776-8.

Amol Kelkar, Rohan Relan, Vaishali Bhardwaj, Saurabh Vaichal, and Peter Relan. Bertrand-dr: Improving text-to-sql using a discriminative re-ranker. *CoRR*, abs/2002.00557, 2020. URL https://arxiv.org/abs/2002.00557.

Mohamad Khajezade, Fatemeh Hendijani Fard, and Mohamed S. Shehata. Evaluating few shot and contrastive learning methods for code clone detection. *CoRR*, abs/2204.07501, 2022. doi: 10.48550/ARXIV.2204.07501. URL https://doi.org/10.48550/arXiv.2204.07501.

Junaed Younus Khan and Gias Uddin. Automatic code documentation generation using GPT-3. In *37th IEEE/ACM International Conference on Automated Software Engineering, ASE 2022, Rochester, MI, USA, October 10-14, 2022*, pp. 174:1–174:6. ACM, 2022. doi: 10.1145/3551349.3559548. URL https://doi.org/10.1145/3551349.3559548.

Mohammad Abdullah Matin Khan, M. Saiful Bari, Xuan Long Do, Weishi Wang, Md. Rizwan Parvez, and Shafiq R. Joty. xcodeeval: A large scale multilingual multitask benchmark for code understanding, generation, translation and retrieval. *CoRR*, abs/2303.03004, 2023. doi: 10.48550/ARXIV.2303.03004. URL https://doi.org/10.48550/arXiv.2303.03004.

Ahmed Khanfir, Renzo Degiovanni, Mike Papadakis, and Yves Le Traon. Efficient mutation testing via pre-trained language models. *CoRR*, abs/2301.03543, 2023a. doi: 10.48550/arXiv.2301.03543. URL https://doi.org/10.48550/arXiv.2301.03543.

Ahmed Khanfir, Anil Koyuncu, Mike Papadakis, Maxime Cordy, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. ibir: Bug-report-driven fault injection. *ACM Trans. Softw. Eng. Methodol.*, 32(2): 33:1–33:31, 2023b. doi: 10.1145/3542946. URL https://doi.org/10.1145/3542946.

Kisub Kim, Dongsun Kim, Tegawendé F. Bissyandé, Eunjong Choi, Li Li, Jacques Klein, and Yves Le Traon. Facoy: a code-to-code search engine. In Michel Chaudron, Ivica Crnkovic, Marsha Chechik, and Mark Harman (eds.), *Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, pp. 946–957. ACM, 2018. doi: 10.1145/3180155.3180187. URL https://doi.org/10.1145/3180155.3180187.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rkgNKkHtvB.

Denis Kocetkov, Raymond Li, Loubna Ben allal, Jia LI, Chenghao Mou, Yacine Jernite, Margaret Mitchell, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro Von Werra, and Harm de Vries. The stack: 3 TB of permissively licensed source code. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=pxpbTdUEpD.

Li Kuang, Cong Zhou, and Xiaoxian Yang. Code comment generation based on graph neural network enhanced transformer model for code understanding in open-source software ecosystems. *Autom. Softw. Eng.*, 29(2):43, 2022. doi: 10.1007/S10515-022-00341-1. URL https://doi.org/10.1007/s10515-022-00341-1.

Sumith Kulal, Panupong Pasupat, Kartik Chandra, Mina Lee, Oded Padon, Alex Aiken, and Percy Liang. Spoc: Search-based pseudocode to code. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 11883–11894, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/7298332f04ac004a0ca44cc69ecf6f6b-Abstract.html.

Chinmay Kulkarni, Stefania Druga, Minsuk Chang, Alex Fiannaca, Carrie J. Cai, and Michael Terry. A word is worth a thousand pictures: Prompts as AI design material. *CoRR*, abs/2303.12647, 2023. doi: 10.48550/ARXIV.2303.12647. URL https://doi.org/10.48550/arXiv.2303.12647.

Ayush Kumar, Parth Nagarkar, Prabhav Nalhe, and Sanjeev Vijayakumar. Deep learning driven natural languages text to SQL query conversion: A survey. *CoRR*, abs/2208.04415, 2022. doi: 10.48550/ARXIV.2208.04415. URL https://doi.org/10.48550/arXiv.2208.04415.

Deeptimahanti Deva Kumar and Ratna Sanyal. Static uml model generator from analysis of requirements (sugar). In *2008 Advanced Software Engineering and Its Applications*, pp. 77–84, 2008. doi: 10.1109/ASEA.2008.25.

Marie-Anne Lachaux, Baptiste Rozière, Marc Szafraniec, and Guillaume Lample. DOBF: A deobfuscation pre-training objective for programming languages. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 14967–14979, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/7d6548bdc0082aacc950ed35e91fcccb-Abstract.html.

Jeremy Lacomis, Pengcheng Yin, Edward J. Schwartz, Miltiadis Allamanis, Claire Le Goues, Graham Neubig, and Bogdan Vasilescu. DIRE: A neural approach to decompiled identifier naming. In *34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019, San Diego, CA, USA, November 11-15, 2019*, pp. 628–639. IEEE, 2019. doi: 10.1109/ASE.2019.00064. URL https://doi.org/10.1109/ASE.2019.00064.

Shuvendu K. Lahiri, Aaditya Naik, Georgios Sakkas, Piali Choudhury, Curtis von Veh, Madanlal Musuvathi, Jeevana Priya Inala, Chenglong Wang, and Jianfeng Gao. Interactive code generation via test-driven user-intent formalization. *CoRR*, abs/2208.05950, 2022. doi: 10.48550/arXiv.2208.05950. URL https://doi.org/10.48550/arXiv.2208.05950.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-Tau Yih, Daniel Fried, Sida I. Wang, and Tao Yu. DS-1000: A natural and reliable benchmark for data science code generation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 18319–18345. PMLR, 2023. URL https://proceedings.mlr.press/v202/lai23b.html.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Max Landauer, Sebastian Onder, Florian Skopik, and Markus Wurzenberger. Deep learning for anomaly detection in log data: A survey. *CoRR*, abs/2207.03820, 2022. doi: 10.48550/ARXIV.2207.03820. URL https://doi.org/10.48550/arXiv.2207.03820.

Chris Lattner and Vikram S. Adve. LLVM: A compilation framework for lifelong program analysis & transformation. In *2nd IEEE / ACM International Symposium on Code Generation and Optimization (CGO 2004), 20-24 March 2004, San Jose, CA, USA*, pp. 75–88. IEEE Computer Society, 2004. doi: 10.1109/CGO.2004.1281665. URL https://doi.org/10.1109/CGO.2004.1281665.

Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, Leandro von Werra, Chenghao Mou, Eduardo González Ponferrada, Huu Nguyen, Jörg Frohberg, Mario Sasko, Quentin Lhoest, Angelina McMillan-Major, Gérard Dupont, Stella Biderman, Anna Rogers, Loubna Ben Allal, Francesco De Toni, Giada Pistilli, Olivier Nguyen, Somaieh Nikpoor, Maraim Masoud, Pierre Colombo, Javier de la Rosa, Paulo Villegas, Tristan Thrush, Shayne Longpre, Sebastian Nagel, Leon Weber, Manuel Muñoz, Jian Zhu, Daniel van Strien, Zaid Alyafeai, Khalid Almubarak, Minh Chien Vu, Itziar Gonzalez-Dios, Aitor Soroa, Kyle Lo, Manan Dey, Pedro Ortiz Suarez, Aaron Gokaslan, Shamik Bose, David Ifeoluwa Adelani, Long Phan, Hieu Tran, Ian Yu, Suhas Pai, Jenny Chim, Violette Lepercq, Suzana Ilic, Margaret Mitchell, Alexandra Sasha Luccioni, and Yacine Jernite. The bigscience ROOTS corpus: A 1.6tb composite multilingual dataset. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/ce9e92e3de2372a4b93353eb7f3dc0bd-Abstract-Datasets_and_Benchmarks.html.

Hugo Laurençon, Léo Tronchon, and Victor Sanh. Unlocking the conversion of web screenshots into HTML code with the websight dataset. *CoRR*, abs/2403.09029, 2024. doi: 10.48550/ARXIV.2403.09029. URL https://doi.org/10.48550/arXiv.2403.09029.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu-Hong Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/8636419dea1aa9fbd25fc4248e702da4-Abstract-Conference.html.

Van-Hoang Le and Hongyu Zhang. Log-based anomaly detection without log parsing. In *36th IEEE/ACM International Conference on Automated Software Engineering, ASE 2021, Melbourne, Australia, November 15-19, 2021*, pp. 492–504. IEEE, 2021. doi: 10.1109/ASE51524.2021.9678773. URL https://doi.org/10.1109/ASE51524.2021.9678773.

Van-Hoang Le and Hongyu Zhang. Log parsing with prompt-based few-shot learning. In *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*, pp. 2438–2449. IEEE, 2023a. doi: 10.1109/ICSE48619.2023.00204. URL https://doi.org/10.1109/ICSE48619.2023.00204.

Van-Hoang Le and Hongyu Zhang. Log parsing: How far can chatgpt go? In *38th IEEE/ACM International Conference on Automated Software Engineering, ASE 2023, Luxembourg, September 11-15, 2023*, pp. 1699–1704. IEEE, 2023b. doi: 10.1109/ASE56229.2023.00206. URL https://doi.org/10.1109/ASE56229.2023.00206.

Hugh Leather and Chris Cummins. Machine learning in compilers: Past, present and future. In *Forum for Specification and Design Languages, FDL 2020, Kiel, Germany, September 15-17, 2020*, pp. 1–8. IEEE, 2020. doi: 10.1109/FDL50818.2020.9232934. URL https://doi.org/10.1109/FDL50818.2020.9232934.

Alexander LeClair, Siyuan Jiang, and Collin McMillan. A neural model for generating natural language summaries of program subroutines. In Joanne M. Atlee, Tevfik Bultan, and Jon Whittle (eds.), *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, pp. 795–806. IEEE / ACM, 2019. doi: 10.1109/ICSE.2019.00087. URL https://doi.org/10.1109/ICSE.2019.00087.

Changyoon Lee, Yeon Seonwoo, and Alice Oh. CS1QA: A dataset for assisting code-based question answering in an introductory programming course. In Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pp. 2026–2040. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.NAACL-MAIN.148. URL https://doi.org/10.18653/v1/2022.naacl-main.148.

Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. Kaggledbqa: Realistic evaluation of text-to-sql parsers. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 2261–2273. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.ACL-LONG.176. URL https://doi.org/10.18653/v1/2021.acl-long.176.

Woosuk Lee, Kihong Heo, Rajeev Alur, and Mayur Naik. Accelerating search-based program synthesis using learned probabilistic models. In Jeffrey S. Foster and Dan Grossman (eds.), *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2018, Philadelphia, PA, USA, June 18-22, 2018*, pp. 436–449. ACM, 2018. doi: 10.1145/3192366.3192410. URL https://doi.org/10.1145/3192366.3192410.

Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua. Re-examining the role of schema linking in text-to-sql. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 6943–6954. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.EMNLP-MAIN.564. URL https://doi.org/10.18653/v1/2020.emnlp-main.564.

Caroline Lemieux and Koushik Sen. Fairfuzz: a targeted mutation strategy for increasing greybox fuzz testing coverage. In Marianne Huchard, Christian Kästner, and Gordon Fraser (eds.), *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, Montpellier, France, September 3-7, 2018*, pp. 475–485. ACM, 2018. doi: 10.1145/3238147.3238176. URL https://doi.org/10.1145/3238147.3238176.

Caroline Lemieux, Jeevana Priya Inala, Shuvendu K. Lahiri, and Siddhartha Sen. Codamosa: Escaping coverage plateaus in test generation with pre-trained large language models. In *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*, pp. 919–931. IEEE, 2023. doi: 10.1109/ICSE48619.2023.00085. URL https://doi.org/10.1109/ICSE48619.2023.00085.

Yichong Leng, Xu Tan, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. Unsupervised pivot translation for distant languages. In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 175–183. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1017. URL https://doi.org/10.18653/v1/p19-1017.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=qrwe7XHTmYb.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pp. 3045–3059. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.243. URL https://doi.org/10.18653/v1/2021.emnlp-main.243.

Keletso Letsholo, Liping Zhao, and Erol-Valeriu Chioasca. TRAM: A tool for transforming textual requirements into analysis models. In Ewen Denney, Tevfik Bultan, and Andreas Zeller (eds.), *2013 28th IEEE/ACM International Conference on Automated Software Engineering, ASE 2013, Silicon Valley, CA, USA, November 11-15, 2013*, pp. 738–741. IEEE, 2013. doi: 10.1109/ASE.2013.6693146. URL https://doi.org/10.1109/ASE.2013.6693146.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 7871–7880. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.703. URL https://doi.org/10.18653/v1/2020.acl-main.703.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/18abbeef8cfe9203fdf9053c9c4fe191-Abstract-Conference.html.

Chengshu Li, Jacky Liang, Andy Zeng, Xinyun Chen, Karol Hausman, Dorsa Sadigh, Sergey Levine, Li Fei-Fei, Fei Xia, and Brian Ichter. Chain of code: Reasoning with a language model-augmented code emulator. *CoRR*, abs/2312.04474, 2023a. doi: 10.48550/ARXIV.2312.04474. URL https://doi.org/10.48550/arXiv.2312.04474.

Fei Li and H. V. Jagadish. Constructing an interactive natural language interface for relational databases. *Proc. VLDB Endow.*, 8(1):73–84, 2014. doi: 10.14778/2735461.2735468. URL http://www.vldb.org/pvldb/vol8/p73-li.pdf.

Guanhua Li, Yijian Wu, Chanchal K. Roy, Jun Sun, Xin Peng, Nanjie Zhan, Bin Hu, and Jingyi Ma. SAGA: efficient and large-scale detection of near-miss clones with GPU acceleration. In Kostas Kontogiannis, Foutse Khomh, Alexander Chatzigeorgiou, Marios-Eleftherios Fokaefs, and Minghui Zhou (eds.), *27th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2020, London, ON, Canada, February 18-21, 2020*, pp. 272–283. IEEE, 2020a. doi: 10.1109/SANER48275.2020.9054832. URL https://doi.org/10.1109/SANER48275.2020.9054832.

Haochen Li, Chunyan Miao, Cyril Leung, Yanxian Huang, Yuan Huang, Hongyu Zhang, and Yanlin Wang. Exploring representation-level augmentation for code search. In Yoav Goldberg, Zornitsa Kozareva,

and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 4924–4936. Association for Computational Linguistics, 2022a. doi: 10.18653/v1/2022.emnlp-main.327. URL https://doi.org/10.18653/v1/2022.emnlp-main.327.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. CMMLU: measuring massive multitask language understanding in chinese. *CoRR*, abs/2306.09212, 2023b. doi: 10.48550/ARXIV.2306.09212. URL https://doi.org/10.48550/arXiv.2306.09212.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. RESDSQL: decoupling schema linking and skeleton parsing for text-to-sql. In Brian Williams, Yiling Chen, and Jennifer Neville (eds.), *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pp. 13067–13075. AAAI Press, 2023c. doi: 10.1609/AAAI.V37I11.26535. URL https://doi.org/10.1609/aaai.v37i11.26535.

Heng-Yi Li, Shu-Ting Shi, Ferdian Thung, Xuan Huo, Bowen Xu, Ming Li, and David Lo. Deepreview: Automatic code review using deep multi-instance learning. In Qiang Yang, Zhi-Hua Zhou, Zhiguo Gong, Min-Ling Zhang, and Sheng-Jun Huang (eds.), *Advances in Knowledge Discovery and Data Mining - 23rd Pacific-Asia Conference, PAKDD 2019, Macau, China, April 14-17, 2019, Proceedings, Part II*, volume 11440 of *Lecture Notes in Computer Science*, pp. 318–330. Springer, 2019a. doi: 10.1007/978-3-030-16145-3_25. URL https://doi.org/10.1007/978-3-030-16145-3_25.

Hongyu Li, Seohyun Kim, and Satish Chandra. Neural code search evaluation dataset. *CoRR*, abs/1908.09804, 2019b. URL http://arxiv.org/abs/1908.09804.

Jia Li, Chongyang Tao, Zhi Jin, Fang Liu, Jia Allen Li, and Ge Li. ZC3: zero-shot cross-language code clone detection. *CoRR*, abs/2308.13754, 2023d. doi: 10.48550/ARXIV.2308.13754. URL https://doi.org/10.48550/arXiv.2308.13754.

Jia Li, Yunfei Zhao, Yongmin Li, Ge Li, and Zhi Jin. Acecoder: Utilizing existing code to enhance code generation, 2023e.

Jia Li, Ge Li, Xuanming Zhang, Yihong Dong, and Zhi Jin. Evocodebench: An evolving code generation benchmark aligned with real-world code repositories. *CoRR*, abs/2404.00599, 2024a. doi: 10.48550/ARXIV.2404.00599. URL https://doi.org/10.48550/arXiv.2404.00599.

Jian Li, Yue Wang, Michael R. Lyu, and Irwin King. Code completion with neural attention and pointer networks. In Jérôme Lang (ed.), *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pp. 4159–4165. ijcai.org, 2018a. doi: 10.24963/ijcai.2018/578. URL https://doi.org/10.24963/ijcai.2018/578.

Jinyang Li, Binyuan Hui, Reynold Cheng, Bowen Qin, Chenhao Ma, Nan Huo, Fei Huang, Wenyu Du, Luo Si, and Yongbin Li. Graphix-t5: Mixing pre-trained transformers with graph-aware layers for text-to-sql parsing. In Brian Williams, Yiling Chen, and Jennifer Neville (eds.), *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pp. 13076–13084. AAAI Press, 2023f. doi: 10.1609/AAAI.V37I11.26536. URL https://doi.org/10.1609/aaai.v37i11.26536.

Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chen-Chuan Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can LLM already serve as A database interface? A big bench for large-scale database grounded text-to-sqls. *CoRR*, abs/2305.03111, 2023g. doi: 10.48550/ARXIV.2305.03111. URL https://doi.org/10.48550/arXiv.2305.03111.

Junlong Li, Yiheng Xu, Lei Cui, and Furu Wei. Markuplm: Pre-training of text and markup language for visually rich document understanding. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 6078–6087. Association for Computational Linguistics, 2022b. doi: 10.18653/V1/2022.ACL-LONG.420. URL https://doi.org/10.18653/v1/2022.acl-long.420.

Kaixin Li, Yuchen Tian, Qisheng Hu, Ziyang Luo, and Jing Ma. Mmcode: Evaluating multi-modal code large language models with visually rich programming problems. *CoRR*, abs/2404.09486, 2024b. doi: 10.48550/ARXIV.2404.09486. URL https://doi.org/10.48550/arXiv.2404.09486.

Lingwei Li, Li Yang, Huaxi Jiang, Jun Yan, Tiejian Luo, Zihan Hua, Geng Liang, and Chun Zuo. AUGER: automatically generating review comments with pre-training models. In Abhik Roychoudhury, Cristian Cadar, and Miryung Kim (eds.), *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022, Singapore, Singapore, November 14-18, 2022*, pp. 1009–1021. ACM, 2022c. doi: 10.1145/3540250.3549099. URL https://doi.org/10.1145/3540250.3549099.

Raymond Li, Loubna Ben allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia LI, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Joel Lamy-Poirier, Joao Monteiro, Nicolas Gontier, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Ben Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason T Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Urvashi Bhattacharyya, Wenhao Yu, Sasha Luccioni, Paulo Villegas, Fedor Zhdanov, Tony Lee, Nadav Timor, Jennifer Ding, Claire S Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro Von Werra, and Harm de Vries. Starcoder: may the source be with you! *Transactions on Machine Learning Research*, 2023h. ISSN 2835-8856. URL https://openreview.net/forum?id=KoFOg41haE. Reproducibility Certification.

Ruitong Li, Gang Hu, and Min Peng. Hierarchical embedding for code search in software q&a sites. In *2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, United Kingdom, July 19-24, 2020*, pp. 1–10. IEEE, 2020b. doi: 10.1109/IJCNN48605.2020.9207101. URL https://doi.org/10.1109/IJCNN48605.2020.9207101.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-lm improves controllable text generation. In *NeurIPS*, 2022d. URL http://papers.nips.cc/paper_files/paper/2022/hash/1be5bc25d50895ee656b8c2d9eb89d6a-Abstract-Conference.html.

Xiaonan Li, Yeyun Gong, Yelong Shen, Xipeng Qiu, Hang Zhang, Bolun Yao, Weizhen Qi, Daxin Jiang, Weizhu Chen, and Nan Duan. Coderetriever: A large scale contrastive pre-training method for code search. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 2898–2910. Association for Computational Linguistics, 2022e. doi: 10.18653/v1/2022.emnlp-main.187. URL https://doi.org/10.18653/v1/2022.emnlp-main.187.

Xiaonan Li, Daya Guo, Yeyun Gong, Yun Lin, Yelong Shen, Xipeng Qiu, Daxin Jiang, Weizhu Chen, and Nan Duan. Soft-labeled contrastive pre-training for function-level code representation. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 118–129. Association for Computational Linguistics, 2022f. doi: 10.18653/v1/2022.findings-emnlp.9. URL https://doi.org/10.18653/v1/2022.findings-emnlp.9.

Xuan Li, Zerui Wang, Qianxiang Wang, Shoumeng Yan, Tao Xie, and Hong Mei. Relationship-aware code search for javascript frameworks. In Thomas Zimmermann, Jane Cleland-Huang, and Zhendong Su (eds.), *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering,*

FSE 2016, Seattle, WA, USA, November 13-18, 2016, pp. 690–701. ACM, 2016. doi: 10.1145/2950290.2950341. URL https://doi.org/10.1145/2950290.2950341.

Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. Mapping natural language instructions to mobile UI action sequences. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 8198–8210. Association for Computational Linguistics, 2020c. doi: 10.18653/V1/2020.ACL-MAIN.729. URL https://doi.org/10.18653/v1/2020.acl-main.729.

Yi Li, Shaohua Wang, Tien N. Nguyen, and Son Van Nguyen. Improving bug detection via context-based code representation learning and attention-based neural networks. *Proc. ACM Program. Lang.*, 3(OOPSLA): 162:1–162:30, 2019c. doi: 10.1145/3360588. URL https://doi.org/10.1145/3360588.

Yi Li, Shaohua Wang, and Tien N. Nguyen. Dlfix: context-based code transformation learning for automated program repair. In Gregg Rothermel and Doo-Hwan Bae (eds.), *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, pp. 602–614. ACM, 2020d. doi: 10.1145/3377811.3380345. URL https://doi.org/10.1145/3377811.3380345.

Yi Li, Shaohua Wang, and Tien N. Nguyen. Vulnerability detection with fine-grained interpretations. In Diomidis Spinellis, Georgios Gousios, Marsha Chechik, and Massimiliano Di Penta (eds.), *ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, August 23-28, 2021*, pp. 292–303. ACM, 2021a. doi: 10.1145/3468264.3468597. URL https://doi.org/10.1145/3468264.3468597.

Yi Li, Shaohua Wang, and Tien N. Nguyen. A context-based automated approach for method name consistency checking and suggestion. In *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*, pp. 574–586. IEEE, 2021b. doi: 10.1109/ICSE43902.2021.00060. URL https://doi.org/10.1109/ICSE43902.2021.00060.

Yi Li, Shaohua Wang, and Tien N. Nguyen. DEAR: A novel deep learning-based approach for automated program repair. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pp. 511–523. ACM, 2022g. doi: 10.1145/3510003.3510177. URL https://doi.org/10.1145/3510003.3510177.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need II: phi-1.5 technical report. *CoRR*, abs/2309.05463, 2023i. doi: 10.48550/arXiv.2309.05463. URL https://doi.org/10.48550/arXiv.2309.05463.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022h. doi: 10.1126/science.abq1158. URL https://www.science.org/doi/abs/10.1126/science.abq1158.

Zehan Li, Jianfei Zhang, Chuantao Yin, Yuanxin Ouyang, and Wenge Rong. Procqa: A large-scale community-based programming question answering dataset for code search. *CoRR*, abs/2403.16702, 2024c. doi: 10.48550/ARXIV.2403.16702. URL https://doi.org/10.48550/arXiv.2403.16702.

Zhen Li, Deqing Zou, Shouhuai Xu, Xinyu Ou, Hai Jin, Sujuan Wang, Zhijun Deng, and Yuyi Zhong. Vuldeepecker: A deep learning-based system for vulnerability detection. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018b. URL https://www.ndss-symposium.org/wp-content/uploads/2018/02/ndss2018_03A-2_Li_paper.pdf.

Zhen Li, Deqing Zou, Shouhuai Xu, Zhaoxuan Chen, Yawei Zhu, and Hai Jin. Vuldeelocator: A deep learning-based fine-grained vulnerability detector. *IEEE Trans. Dependable Secur. Comput.*, 19(4):2821–2837, 2022i. doi: 10.1109/TDSC.2021.3076142. URL https://doi.org/10.1109/TDSC.2021.3076142.

Zhen Li, Deqing Zou, Shouhuai Xu, Hai Jin, Yawei Zhu, and Zhaoxuan Chen. Sysevr: A framework for using deep learning to detect software vulnerabilities. *IEEE Trans. Dependable Secur. Comput.*, 19(4):2244–2258, 2022j. doi: 10.1109/TDSC.2021.3051525. URL https://doi.org/10.1109/TDSC.2021.3051525.

Zhiyu Li, Shuai Lu, Daya Guo, Nan Duan, Shailesh Jannu, Grant Jenks, Deep Majumder, Jared Green, Alexey Svyatkovskiy, Shengyu Fu, and Neel Sundaresan. Automating code review activities by large-scale pre-training. In Abhik Roychoudhury, Cristian Cadar, and Miryung Kim (eds.), *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022, Singapore, Singapore, November 14-18, 2022*, pp. 1035–1047. ACM, 2022k. doi: 10.1145/3540250.3549081. URL https://doi.org/10.1145/3540250.3549081.

Ruigang Liang, Ying Cao, Peiwei Hu, and Kai Chen. Neutron: an attention-based neural decompiler. *Cybersecur.*, 4(1):5, 2021. doi: 10.1186/S42400-021-00070-0. URL https://doi.org/10.1186/s42400-021-00070-0.

Dianshu Liao, Shidong Pan, Xiaoyu Sun, Xiaoxue Ren, Qing Huang, Zhenchang Xing, Huan Jin, and Qinying Li. A^3-codgen: A repository-level code generation framework for code reuse with local-aware, global-aware, and third-party-library-aware. *CoRR*, abs/2312.05772, 2023. doi: 10.48550/ARXIV.2312.05772. URL https://doi.org/10.48550/arXiv.2312.05772.

Derrick Lin, James Koppel, Angela Chen, and Armando Solar-Lezama. Quixbugs: a multi-lingual program repair benchmark set based on the quixey challenge. In Gail C. Murphy (ed.), *Proceedings Companion of the 2017 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity, SPLASH 2017, Vancouver, BC, Canada, October 23 - 27, 2017*, pp. 55–56. ACM, 2017. doi: 10.1145/3135932.3135941. URL https://doi.org/10.1145/3135932.3135941.

Feng Lin, Dong Jae Kim, and Tse-Husn Chen. When llm-based code generation meets the software development process. *CoRR*, abs/2403.15852, 2024. doi: 10.48550/ARXIV.2403.15852. URL https://doi.org/10.48550/arXiv.2403.15852.

Guanjun Lin, Jun Zhang, Wei Luo, Lei Pan, Yang Xiang, Olivier Y. de Vel, and Paul Montague. Cross-project transfer representation learning for vulnerable function discovery. *IEEE Trans. Ind. Informatics*, 14(7): 3289–3297, 2018a. doi: 10.1109/TII.2018.2821768. URL https://doi.org/10.1109/TII.2018.2821768.

Guanjun Lin, Wei Xiao, Jun Zhang, and Yang Xiang. Deep learning-based vulnerable function detection: A benchmark. In Jianying Zhou, Xiapu Luo, Qingni Shen, and Zhen Xu (eds.), *Information and Communications Security - 21st International Conference, ICICS 2019, Beijing, China, December 15-17, 2019, Revised Selected Papers*, volume 11999 of *Lecture Notes in Computer Science*, pp. 219–232. Springer, 2019. doi: 10.1007/978-3-030-41579-2_13. URL https://doi.org/10.1007/978-3-030-41579-2_13.

Guanjun Lin, Jun Zhang, Wei Luo, Lei Pan, Olivier Y. de Vel, Paul Montague, and Yang Xiang. Software vulnerability discovery via learning multi-domain knowledge bases. *IEEE Trans. Dependable Secur. Comput.*, 18(5):2469–2485, 2021. doi: 10.1109/TDSC.2019.2954088. URL https://doi.org/10.1109/TDSC.2019.2954088.

Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *AI Open*, 3:111–132, 2022. doi: 10.1016/J.AIOPEN.2022.10.001. URL https://doi.org/10.1016/j.aiopen.2022.10.001.

Xi Victoria Lin, Chenglong Wang, Luke Zettlemoyer, and Michael D. Ernst. Nl2bash: A corpus and semantic parser for natural language interface to the linux operating system. In Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Kôiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asunción Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga (eds.), *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. European Language Resources Association (ELRA), 2018b. URL http://www.lrec-conf.org/proceedings/lrec2018/summaries/1021.html.

Zhenghao Lin, Yeyun Gong, Yelong Shen, Tong Wu, Zhihao Fan, Chen Lin, Nan Duan, and Weizhu Chen. Text generation with diffusion language models: A pre-training approach with continuous paragraph

denoise. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 21051–21064. PMLR, 2023. URL https://proceedings.mlr.press/v202/lin23d.html.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, Fumin Wang, and Andrew W. Senior. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016. doi: 10.18653/V1/P16-1057. URL https://doi.org/10.18653/v1/p16-1057.

Xiang Ling, Lingfei Wu, Saizhuo Wang, Gaoning Pan, Tengfei Ma, Fangli Xu, Alex X. Liu, Chunming Wu, and Shouling Ji. Deep graph matching and searching for semantic code retrieval. *ACM Trans. Knowl. Discov. Data*, 15(5):88:1–88:21, 2021. doi: 10.1145/3447571. URL https://doi.org/10.1145/3447571.

Aiwei Liu, Xuming Hu, Lijie Wen, and Philip S. Yu. A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability. *CoRR*, abs/2303.13547, 2023a. doi: 10.48550/ARXIV.2303.13547. URL https://doi.org/10.48550/arXiv.2303.13547.

Bingchang Liu, Chaoyu Chen, Cong Liao, Zi Gong, Huan Wang, Zhichao Lei, Ming Liang, Dajun Chen, Min Shen, Hailian Zhou, Hang Yu, and Jianguo Li. Mftcoder: Boosting code llms with multitask fine-tuning. *CoRR*, abs/2311.02303, 2023b. doi: 10.48550/ARXIV.2311.02303. URL https://doi.org/10.48550/arXiv.2311.02303.

Chao Liu, Xin Xia, David Lo, Zhiwei Liu, Ahmed E. Hassan, and Shanping Li. Codematcher: Searching code based on sequential semantics of important query words. *ACM Trans. Softw. Eng. Methodol.*, 31(1): 12:1–12:37, 2022a. doi: 10.1145/3465403. URL https://doi.org/10.1145/3465403.

Chenxiao Liu and Xiaojun Wan. Codeqa: A question answering dataset for source code comprehension. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pp. 2618–2632. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.FINDINGS-EMNLP.223. URL https://doi.org/10.18653/v1/2021.findings-emnlp.223.

Fang Liu, Ge Li, Yunfei Zhao, and Zhi Jin. Multi-task learning based pre-trained language model for code completion. In *35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020, Melbourne, Australia, September 21-25, 2020*, pp. 473–485. IEEE, 2020. doi: 10.1145/3324884.3416591. URL https://doi.org/10.1145/3324884.3416591.

Fang Liu, Ge Li, Zhiyi Fu, Shuai Lu, Yiyang Hao, and Zhi Jin. Learning to recommend method names with global context. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pp. 1294–1306. ACM, 2022b. doi: 10.1145/3510003.3510154. URL https://doi.org/10.1145/3510003.3510154.

Fang Liu, Jia Li, and Li Zhang. Syntax and domain aware model for unsupervised program translation. In *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*, pp. 755–767. IEEE, 2023c. doi: 10.1109/ICSE48619.2023.00072. URL https://doi.org/10.1109/ICSE48619.2023.00072.

Fang Liu, Yang Liu, Lin Shi, Houkun Huang, Ruifeng Wang, Zhen Yang, and Li Zhang. Exploring and evaluating hallucinations in llm-powered code generation. *CoRR*, abs/2404.00971, 2024. doi: 10.48550/ARXIV.2404.00971. URL https://doi.org/10.48550/arXiv.2404.00971.

Jiate Liu, Yiqin Zhu, Kaiwen Xiao, Qiang Fu, Xiao Han, Wei Yang, and Deheng Ye. RLTF: reinforcement learning from unit test feedback. *CoRR*, abs/2307.04349, 2023d. doi: 10.48550/ARXIV.2307.04349. URL https://doi.org/10.48550/arXiv.2307.04349.

Jiawei Liu, Jinkun Lin, Fabian Ruffy, Cheng Tan, Jinyang Li, Aurojit Panda, and Lingming Zhang. Nnsmith: Generating diverse and valid test cases for deep learning compilers. In Tor M. Aamodt, Natalie D. Enright Jerger, and Michael M. Swift (eds.), *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, ASPLOS 2023, Vancouver, BC, Canada, March 25-29, 2023*, pp. 530–543. ACM, 2023e. doi: 10.1145/3575693.3575707. URL https://doi.org/10.1145/3575693.3575707.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *CoRR*, abs/2305.01210, 2023f. doi: 10.48550/arXiv.2305.01210. URL https://doi.org/10.48550/arXiv.2305.01210.

Kui Liu, Dongsun Kim, Tegawendé F. Bissyandé, Tae-young Kim, Kisub Kim, Anil Koyuncu, Suntae Kim, and Yves Le Traon. Learning to spot and refactor inconsistent method names. In Joanne M. Atlee, Tevfik Bultan, and Jon Whittle (eds.), *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, pp. 1–12. IEEE / ACM, 2019a. doi: 10.1109/ICSE.2019.00019. URL https://doi.org/10.1109/ICSE.2019.00019.

Kui Liu, Anil Koyuncu, Dongsun Kim, and Tegawendé F. Bissyandé. Tbar: revisiting template-based automated program repair. In Dongmei Zhang and Anders Møller (eds.), *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2019, Beijing, China, July 15-19, 2019*, pp. 31–42. ACM, 2019b. doi: 10.1145/3293882.3330577. URL https://doi.org/10.1145/3293882.3330577.

Mingjie Liu, Nathaniel Ross Pinckney, Brucek Khailany, and Haoxing Ren. Verilogeval: Evaluating large language models for verilog code generation. *CoRR*, abs/2309.07544, 2023g. doi: 10.48550/ARXIV.2309.07544. URL https://doi.org/10.48550/arXiv.2309.07544.

Qin Liu, Zihe Liu, Hongming Zhu, Hongfei Fan, Bowen Du, and Yu Qian. Generating commit messages from diffs using pointer-generator network. In Margaret-Anne D. Storey, Bram Adams, and Sonia Haiduc (eds.), *Proceedings of the 16th International Conference on Mining Software Repositories, MSR 2019, 26-27 May 2019, Montreal, Canada*, pp. 299–309. IEEE / ACM, 2019c. doi: 10.1109/MSR.2019.00056. URL https://doi.org/10.1109/MSR.2019.00056.

Shang Liu, Wenji Fang, Yao Lu, Qijun Zhang, Hongce Zhang, and Zhiyao Xie. Rtlcoder: Outperforming GPT-3.5 in design RTL generation with our open-source dataset and lightweight solution. *CoRR*, abs/2312.08617, 2023h. doi: 10.48550/ARXIV.2312.08617. URL https://doi.org/10.48550/arXiv.2312.08617.

Shangqing Liu, Cuiyun Gao, Sen Chen, Lun Yiu Nie, and Yang Liu. ATOM: commit message generation based on abstract syntax tree and hybrid ranking. *IEEE Trans. Software Eng.*, 48(5):1800–1817, 2022c. doi: 10.1109/TSE.2020.3038681. URL https://doi.org/10.1109/TSE.2020.3038681.

Shangqing Liu, Xiaofei Xie, Jing Kai Siow, Lei Ma, Guozhu Meng, and Yang Liu. Graphsearchnet: Enhancing gnns via capturing global dependencies for semantic code search. *IEEE Trans. Software Eng.*, 49(4):2839–2855, 2023i. doi: 10.1109/TSE.2022.3233901. URL https://doi.org/10.1109/TSE.2022.3233901.

Tianyang Liu, Canwen Xu, and Julian J. McAuley. Repobench: Benchmarking repository-level code autocompletion systems. *CoRR*, abs/2306.03091, 2023j. doi: 10.48550/ARXIV.2306.03091. URL https://doi.org/10.48550/arXiv.2306.03091.

Xiaoyu Liu, Jinu Jang, Neel Sundaresan, Miltiadis Allamanis, and Alexey Svyatkovskiy. Adaptivepaste: Code adaptation through learning semantics-aware variable usage representations. *CoRR*, abs/2205.11023, 2022d. doi: 10.48550/arXiv.2205.11023. URL https://doi.org/10.48550/arXiv.2205.11023.

Yilun Liu, Shimin Tao, Weibin Meng, Jingyu Wang, Wenbing Ma, Yanqing Zhao, Yuhang Chen, Hao Yang, Yanfei Jiang, and Xun Chen. Logprompt: Prompt engineering towards zero-shot and interpretable log analysis. *CoRR*, abs/2308.07610, 2023k. doi: 10.48550/ARXIV.2308.07610. URL https://doi.org/10.48550/arXiv.2308.07610.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019d. URL http://arxiv.org/abs/1907.11692.

Yudong Liu, Xu Zhang, Shilin He, Hongyu Zhang, Liqun Li, Yu Kang, Yong Xu, Minghua Ma, Qingwei Lin, Yingnong Dang, Saravan Rajmohan, and Dongmei Zhang. Uniparser: A unified log parser for heterogeneous log data. In Frédérique Laforest, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman, and Lionel Médini (eds.), *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pp. 1893–1901. ACM, 2022e. doi: 10.1145/3485447.3511993. URL https://doi.org/10.1145/3485447.3511993.

Yuliang Liu, Xiangru Tang, Zefan Cai, Junjie Lu, Yichi Zhang, Yanjun Shao, Zexuan Deng, Helan Hu, Zengxian Yang, Kaikai An, Ruijun Huang, Shuzheng Si, Sheng Chen, Haozhe Zhao, Zhengliang Li, Liang Chen, Yiming Zong, Yan Wang, Tianyu Liu, Zhiwei Jiang, Baobao Chang, Yujia Qin, Wangchunshu Zhou, Yilun Zhao, Arman Cohan, and Mark Gerstein. Ml-bench: Large language models leverage open-source libraries for machine learning tasks. *CoRR*, abs/2311.09835, 2023l. doi: 10.48550/ARXIV.2311.09835. URL https://doi.org/10.48550/arXiv.2311.09835.

Zhe Liu, Chunyang Chen, Junjie Wang, Mengzhuo Chen, Boyu Wu, Xing Che, Dandan Wang, and Qing Wang. Chatting with GPT-3 for zero-shot human-like mobile automated GUI testing. *CoRR*, abs/2305.09434, 2023m. doi: 10.48550/ARXIV.2305.09434. URL https://doi.org/10.48550/arXiv.2305.09434.

Zhijie Liu, Yutian Tang, Xiapu Luo, Yuming Zhou, and Liang Feng Zhang. No need to lift a finger anymore? assessing the quality of code generation by chatgpt. *CoRR*, abs/2308.04838, 2023n. doi: 10.48550/ARXIV.2308.04838. URL https://doi.org/10.48550/arXiv.2308.04838.

Zhongxin Liu, Xin Xia, Ahmed E. Hassan, David Lo, Zhenchang Xing, and Xinyu Wang. Neural-machine-translation-based commit message generation: how far are we? In Marianne Huchard, Christian Kästner, and Gordon Fraser (eds.), *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, Montpellier, France, September 3-7, 2018*, pp. 373–384. ACM, 2018. doi: 10.1145/3238147.3238190. URL https://doi.org/10.1145/3238147.3238190.

Fan Long and Martin C. Rinard. Automatic patch generation by learning correct code. In Rastislav Bodík and Rupak Majumdar (eds.), *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, pp. 298–312. ACM, 2016. doi: 10.1145/2837614.2837617. URL https://doi.org/10.1145/2837614.2837617.

Pablo Loyola, Edison Marrese-Taylor, and Yutaka Matsuo. A neural architecture for generating natural language descriptions from source code changes. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pp. 287–292. Association for Computational Linguistics, 2017. doi: 10.18653/V1/P17-2045. URL https://doi.org/10.18653/v1/P17-2045.

Pablo Loyola, Edison Marrese-Taylor, Jorge A. Balazs, Yutaka Matsuo, and Fumiko Satoh. Content aware source code change description generation. In Emiel Krahmer, Albert Gatt, and Martijn Goudbeek (eds.), *Proceedings of the 11th International Conference on Natural Language Generation, Tilburg University, The Netherlands, November 5-8, 2018*, pp. 119–128. Association for Computational Linguistics, 2018. doi: 10.18653/V1/W18-6513. URL https://doi.org/10.18653/v1/w18-6513.

Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, Tianyang Liu, Max Tian, Denis Kocetkov, Arthur Zucker, Younes Belkada, Zijian Wang, Qian Liu, Dmitry Abulkhanov, Indraneil Paul, Zhuang Li, Wen-Ding Li, Megan Risdal, Jia Li, Jian Zhu, Terry Yue Zhuo, Evgenii Zheltonozhskii, Nii Osae Osae Dade, Wenhao Yu, Lucas Krauß, Naman Jain, Yixuan Su, Xuanli He, Manan Dey, Edoardo Abati, Yekun Chai, Niklas Muennighoff, Xiangru Tang, Muhtasham Oblokulov, Christopher Akiki, Marc Marone,

Chenghao Mou, Mayank Mishra, Alex Gu, Binyuan Hui, Tri Dao, Armel Zebaze, Olivier Dehaene, Nicolas Patry, Canwen Xu, Julian J. McAuley, Han Hu, Torsten Scholak, Sébastien Paquet, Jennifer Robinson, Carolyn Jane Anderson, Nicolas Chapados, and et al. Starcoder 2 and the stack v2: The next generation. *CoRR*, abs/2402.19173, 2024. doi: 10.48550/ARXIV.2402.19173. URL https://doi.org/10.48550/arXiv.2402.19173.

Junyi Lu, Lei Yu, Xiaojia Li, Li Yang, and Chun Zuo. Llama-reviewer: Advancing code review automation with large language models through parameter-efficient fine-tuning (practical experience report). *CoRR*, abs/2308.11148, 2023a. doi: 10.48550/arXiv.2308.11148. URL https://doi.org/10.48550/arXiv.2308.11148.

Meili Lu, Xiaobing Sun, Shaowei Wang, David Lo, and Yucong Duan. Query expansion via wordnet for effective code search. In Yann-Gaël Guéhéneuc, Bram Adams, and Alexander Serebrenik (eds.), *22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering, SANER 2015, Montreal, QC, Canada, March 2-6, 2015*, pp. 545–549. IEEE Computer Society, 2015. doi: 10.1109/SANER.2015.7081874. URL https://doi.org/10.1109/SANER.2015.7081874.

Mengmeng Lu and Peng Liang. Automatic classification of non-functional requirements from augmented app user reviews. In Emilia Mendes, Steve Counsell, and Kai Petersen (eds.), *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, EASE 2017, Karlskrona, Sweden, June 15-16, 2017*, pp. 344–353. ACM, 2017. doi: 10.1145/3084226.3084241. URL https://doi.org/10.1145/3084226.3084241.

Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin B. Clement, Dawn Drain, Daxin Jiang, Duyu Tang, Ge Li, Lidong Zhou, Linjun Shou, Long Zhou, Michele Tufano, Ming Gong, Ming Zhou, Nan Duan, Neel Sundaresan, Shao Kun Deng, Shengyu Fu, and Shujie Liu. Codexglue: A machine learning benchmark dataset for code understanding and generation. In Joaquin Vanschoren and Sai-Kit Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/c16a5320fa475530d9583c34fd356ef5-Abstract-round1.html.

Siyang Lu, Xiang Wei, Yandong Li, and Liqiang Wang. Detecting anomaly in big data system logs using convolutional neural network. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress, DASC/PiCom/DataCom/CyberSciTech 2018, Athens, Greece, August 12-15, 2018*, pp. 151–158. IEEE Computer Society, 2018. doi: 10.1109/DASC/PICOM/DATACOM/CYBERSCITEC.2018.00037. URL https://doi.org/10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00037.

Yangyang Lu, Ge Li, Zelong Zhao, Linfeng Wen, and Zhi Jin. Learning to infer API mappings from API documents. In Gang Li, Yong Ge, Zili Zhang, Zhi Jin, and Michael Blumenstein (eds.), *Knowledge Science, Engineering and Management - 10th International Conference, KSEM 2017, Melbourne, VIC, Australia, August 19-20, 2017, Proceedings*, volume 10412 of *Lecture Notes in Computer Science*, pp. 237–248. Springer, 2017. doi: 10.1007/978-3-319-63558-3_20. URL https://doi.org/10.1007/978-3-319-63558-3_20.

Yao Lu, Shang Liu, Qijun Zhang, and Zhiyao Xie. RTLLM: an open-source benchmark for design RTL generation with large language model. *CoRR*, abs/2308.05345, 2023b. doi: 10.48550/ARXIV.2308.05345. URL https://doi.org/10.48550/arXiv.2308.05345.

Yuwen Lu, Ziang Tong, Qinyi Zhao, Chengzhi Zhang, and Toby Jia-Jun Li. UI layout generation with llms guided by UI grammar. *CoRR*, abs/2310.15455, 2023c. doi: 10.48550/ARXIV.2310.15455. URL https://doi.org/10.48550/arXiv.2310.15455.

Sifei Luan, Di Yang, Celeste Barnaby, Koushik Sen, and Satish Chandra. Aroma: code recommendation via structural code search. *Proc. ACM Program. Lang.*, 3(OOPSLA):152:1–152:28, 2019. doi: 10.1145/3360578. URL https://doi.org/10.1145/3360578.

Francesca Lucchetti and Arjun Guha. Activation steering for robust type prediction in codellms. *CoRR*, abs/2404.01903, 2024. doi: 10.48550/ARXIV.2404.01903. URL https://doi.org/10.48550/arXiv.2404.01903.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct. *CoRR*, abs/2306.08568, 2023. doi: 10.48550/arXiv.2306.08568. URL https://doi.org/10.48550/arXiv.2306.08568.

Thibaud Lutellier, Hung Viet Pham, Lawrence Pang, Yitong Li, Moshi Wei, and Lin Tan. Coconut: combining context-aware neural translation models using ensemble for program repair. In Sarfraz Khurshid and Corina S. Pasareanu (eds.), *ISSTA '20: 29th ACM SIGSOFT International Symposium on Software Testing and Analysis, Virtual Event, USA, July 18-22, 2020*, pp. 101–114. ACM, 2020. doi: 10.1145/3395363.3397369. URL https://doi.org/10.1145/3395363.3397369.

Fei Lv, Hongyu Zhang, Jian-Guang Lou, Shaowei Wang, Dongmei Zhang, and Jianjun Zhao. Codehow: Effective code search based on API understanding and extended boolean model (E). In Myra B. Cohen, Lars Grunske, and Michael Whalen (eds.), *30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, Lincoln, NE, USA, November 9-13, 2015*, pp. 260–270. IEEE Computer Society, 2015. doi: 10.1109/ASE.2015.42. URL https://doi.org/10.1109/ASE.2015.42.

Qin Lyu, Kaushik Chakrabarti, Shobhit Hathi, Souvik Kundu, Jianwen Zhang, and Zheng Chen. Hybrid ranking network for text-to-sql. *CoRR*, abs/2008.04759, 2020. URL https://arxiv.org/abs/2008.04759.

Zeyang Ma, An Ran Chen, Dong Jae Kim, Tse-Hsun Chen, and Shaowei Wang. Llmparser: An exploratory study on using large language models for log parsing. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, ICSE 2024, Lisbon, Portugal, April 14-20, 2024*, pp. 99:1–99:13. ACM, 2024. doi: 10.1145/3597503.3639150. URL https://doi.org/10.1145/3597503.3639150.

Marcos Macedo, Yuan Tian, Filipe Roseiro Côgo, and Bram Adams. Exploring the impact of the output format on the evaluation of large language models for code translation. *CoRR*, abs/2403.17214, 2024. doi: 10.48550/ARXIV.2403.17214. URL https://doi.org/10.48550/arXiv.2403.17214.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Fernanda Madeiral, Simon Urli, Marcelo de Almeida Maia, and Martin Monperrus. BEARS: an extensible java bug benchmark for automatic program repair studies. In Xinyu Wang, David Lo, and Emad Shihab (eds.), *26th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2019, Hangzhou, China, February 24-27, 2019*, pp. 468–478. IEEE, 2019. doi: 10.1109/SANER.2019.8667991. URL https://doi.org/10.1109/SANER.2019.8667991.

Parvez Mahbub, Naz Zarreen Oishie, and S. M. Rafizul Haque. Authorship identification of source code segments written by multiple authors using stacking ensemble method. *CoRR*, abs/2212.05610, 2022. doi: 10.48550/arXiv.2212.05610. URL https://doi.org/10.48550/arXiv.2212.05610.

Yubo Mai, Zhipeng Gao, Xing Hu, Lingfeng Bao, Yu Liu, and Jianling Sun. Are human rules necessary? generating reusable apis with cot reasoning and in-context learning. 2024. URL https://doi.org/10.48550/arXiv.2405.03509.

Rabee Sohail Malik, Jibesh Patra, and Michael Pradel. Nl2type: inferring javascript function types from natural language information. In Joanne M. Atlee, Tevfik Bultan, and Jon Whittle (eds.), *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, pp. 304–315. IEEE / ACM, 2019. doi: 10.1109/ICSE.2019.00045. URL https://doi.org/10.1109/ICSE.2019.00045.

Aniketh Malyala, Katelyn Zhou, Baishakhi Ray, and Saikat Chakraborty. On ml-based program translation: Perils and promises. In *45th IEEE/ACM International Conference on Software Engineering: New Ideas and Emerging Results, NIER@ICSE, Melbourne, Australia, May 14-20, 2023*, pp. 60–65. IEEE, 2023. doi: 10.1109/ICSE-NIER58687.2023.00017. URL https://doi.org/10.1109/ICSE-NIER58687.2023.00017.

Zhenyu Mao, Jialong Li, Munan Li, and Kenji Tei. Multi-role consensus through llms discussions for vulnerability detection. *CoRR*, abs/2403.14274, 2024. doi: 10.48550/ARXIV.2403.14274. URL https://doi.org/10.48550/arXiv.2403.14274.

Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. Neo: A learned query optimizer. *Proc. VLDB Endow.*, 12(11):1705–1718, 2019. doi: 10.14778/3342263.3342644. URL http://www.vldb.org/pvldb/vol12/p1705-marcus.pdf.

Ryan Marcus, Parimarjan Negi, Hongzi Mao, Nesime Tatbul, Mohammad Alizadeh, and Tim Kraska. Bao: Learning to steer query optimizers. *CoRR*, abs/2004.03814, 2020. URL https://arxiv.org/abs/2004.03814.

Antonio Mastropaolo, Simone Scalabrino, Nathan Cooper, David Nader-Palacio, Denys Poshyvanyk, Rocco Oliveto, and Gabriele Bavota. Studying the usage of text-to-text transfer transformer to support code-related tasks. In *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*, pp. 336–347. IEEE, 2021. doi: 10.1109/ICSE43902.2021.00041. URL https://doi.org/10.1109/ICSE43902.2021.00041.

George Mathew and Kathryn T. Stolee. Cross-language code search using static and dynamic analyses. In Diomidis Spinellis, Georgios Gousios, Marsha Chechik, and Massimiliano Di Penta (eds.), *ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, August 23-28, 2021*, pp. 205–217. ACM, 2021. doi: 10.1145/3468264.3468538. URL https://doi.org/10.1145/3468264.3468538.

Deep Mehta, Kartik Rawool, Subodh Gujar, and Bowen Xu. Automated devops pipeline generation for code repositories using large language models. *CoRR*, abs/2312.13225, 2023. doi: 10.48550/ARXIV.2312.13225. URL https://doi.org/10.48550/arXiv.2312.13225.

Weibin Meng, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi Zhang, Shimin Tao, Pei Sun, and Rong Zhou. Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In Sarit Kraus (ed.), *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 4739–4745. ijcai.org, 2019. doi: 10.24963/IJCAI.2019/658. URL https://doi.org/10.24963/ijcai.2019/658.

Aditya Krishna Menon, Omer Tamuz, Sumit Gulwani, Butler W. Lampson, and Adam Kalai. A machine learning framework for programming by example. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pp. 187–195. JMLR.org, 2013. URL http://proceedings.mlr.press/v28/menon13.html.

Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski,

Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, and et al. Gemma: Open models based on gemini research and technology. *CoRR*, abs/2403.08295, 2024. doi: 10.48550/ARXIV.2403.08295. URL https://doi.org/10.48550/arXiv.2403.08295.

Manel Mezghani, Juyeon Kang, and Florence Sèdes. Industrial requirements classification for redundancy and inconsistency detection in SEMIOS. In Guenther Ruhe, Walid Maalej, and Daniel Amyot (eds.), *26th IEEE International Requirements Engineering Conference, RE 2018, Banff, AB, Canada, August 20-24, 2018*, pp. 297–303. IEEE Computer Society, 2018. doi: 10.1109/RE.2018.00037. URL https://doi.org/10.1109/RE.2018.00037.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=r1gs9JgRZ.

Amir M. Mir, Evaldas Latoskinas, and Georgios Gousios. Manytypes4py: A benchmark python dataset for machine learning-based type inference. In *18th IEEE/ACM International Conference on Mining Software Repositories, MSR 2021, Madrid, Spain, May 17-19, 2021*, pp. 585–589. IEEE, 2021. doi: 10.1109/MSR52588.2021.00079. URL https://doi.org/10.1109/MSR52588.2021.00079.

Amir M. Mir, Evaldas Latoskinas, Sebastian Proksch, and Georgios Gousios. Type4py: Practical deep similarity learning-based type inference for python. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pp. 2241–2252. ACM, 2022. doi: 10.1145/3510003.3510124. URL https://doi.org/10.1145/3510003.3510124.

Facundo Molina, Marcelo d'Amorim, and Nazareno Aguirre. Fuzzing class specifications. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pp. 1008–1020. ACM, 2022. doi: 10.1145/3510003.3510120. URL https://doi.org/10.1145/3510003.3510120.

Martin Monperrus. Automatic software repair: A bibliography. *ACM Comput. Surv.*, 51(1):17:1–17:24, 2018. doi: 10.1145/3105906. URL https://doi.org/10.1145/3105906.

Martin Monperrus, Matias Martinez, He Ye, Fernanda Madeiral, Thomas Durieux, and Zhongxing Yu. Megadiff: A dataset of 600k java source code changes categorized by diff size. *CoRR*, abs/2108.04631, 2021. URL https://arxiv.org/abs/2108.04631.

Lili Mou, Ge Li, Lu Zhang, Tao Wang, and Zhi Jin. Convolutional neural networks over tree structures for programming language processing. In Dale Schuurmans and Michael P. Wellman (eds.), *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pp. 1287–1293. AAAI Press, 2016. doi: 10.1609/aaai.v30i1.10139. URL https://doi.org/10.1609/aaai.v30i1.10139.

Fangwen Mu, Xiao Chen, Lin Shi, Song Wang, and Qing Wang. Developer-intent driven code comment generation. In *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*, pp. 768–780. IEEE, 2023a. doi: 10.1109/ICSE48619.2023.00073. URL https://doi.org/10.1109/ICSE48619.2023.00073.

Fangwen Mu, Lin Shi, Song Wang, Zhuohao Yu, Binquan Zhang, Chenxue Wang, Shichao Liu, and Qing Wang. Clarifygpt: Empowering llm-based code generation with intention clarification. *CoRR*, abs/2310.10996, 2023b. doi: 10.48550/ARXIV.2310.10996. URL https://doi.org/10.48550/arXiv.2310.10996.

Priyanka Mudgal and Rita H. Wouhaybi. An assessment of chatgpt on log data. *CoRR*, abs/2309.07938, 2023. doi: 10.48550/ARXIV.2309.07938. URL https://doi.org/10.48550/arXiv.2309.07938.

Niklas Muennighoff, Qian Liu, Armel Randy Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro Von Werra, and Shayne Longpre. Octopack: Instruction tuning code large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mw1PWNSWZP.

Khurram Murad, Syed Noor-ul-Hassan Shirazi, Yousaf Bin Zikria, and Nassar Ikram. Evading virus detection using code obfuscation. In Tai-Hoon Kim, Young-Hoon Lee, Byeong Ho Kang, and Dominik Slezak (eds.), *Future Generation Information Technology - Second International Conference, FGIT 2010, Jeju Island, Korea, December 13-15, 2010. Proceedings*, volume 6485 of *Lecture Notes in Computer Science*, pp. 394–401. Springer, 2010. doi: 10.1007/978-3-642-17569-5_39. URL https://doi.org/10.1007/978-3-642-17569-5_39.

Bardia Nadimi and Hao Zheng. A multi-expert large language model architecture for verilog code generation. *CoRR*, abs/2404.08029, 2024. doi: 10.48550/ARXIV.2404.08029. URL https://doi.org/10.48550/arXiv.2404.08029.

Kawser Wazed Nafi, Tonny Shekha Kar, Banani Roy, Chanchal K. Roy, and Kevin A. Schneider. CLCDSA: cross language code clone detection using syntactical features and API documentation. In *34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019, San Diego, CA, USA, November 11-15, 2019*, pp. 1026–1037. IEEE, 2019. doi: 10.1109/ASE.2019.00099. URL https://doi.org/10.1109/ASE.2019.00099.

Tasuku Nakagawa, Yoshiki Higo, and Shinji Kusumoto. NIL: large-scale detection of large-variance clones. In Diomidis Spinellis, Georgios Gousios, Marsha Chechik, and Massimiliano Di Penta (eds.), *ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, August 23-28, 2021*, pp. 830–841. ACM, 2021. doi: 10.1145/3468264.3468564. URL https://doi.org/10.1145/3468264.3468564.

Linyong Nan, Yilun Zhao, Weijin Zou, Narutatsu Ri, Jaesung Tae, Ellen Zhang, Arman Cohan, and Dragomir Radev. Enhancing few-shot text-to-sql capabilities of large language models: A study on prompt design strategies. *CoRR*, abs/2305.12586, 2023. doi: 10.48550/arXiv.2305.12586. URL https://doi.org/10.48550/arXiv.2305.12586.

Sasho Nedelkoski, Jasmin Bogatinovski, Alexander Acker, Jorge Cardoso, and Odej Kao. Self-supervised log parsing. In Yuxiao Dong, Dunja Mladenic, and Craig Saunders (eds.), *Machine Learning and Knowledge Discovery in Databases: Applied Data Science Track - European Conference, ECML PKDD 2020, Ghent, Belgium, September 14-18, 2020, Proceedings, Part IV*, volume 12460 of *Lecture Notes in Computer Science*, pp. 122–138. Springer, 2020a. doi: 10.1007/978-3-030-67667-4_8. URL https://doi.org/10.1007/978-3-030-67667-4_8.

Sasho Nedelkoski, Jasmin Bogatinovski, Alexander Acker, Jorge Cardoso, and Odej Kao. Self-attentive classification-based anomaly detection in unstructured logs. In Claudia Plant, Haixun Wang, Alfredo Cuzzocrea, Carlo Zaniolo, and Xindong Wu (eds.), *20th IEEE International Conference on Data Mining, ICDM 2020, Sorrento, Italy, November 17-20, 2020*, pp. 1196–1201. IEEE, 2020b. doi: 10.1109/ICDM50108.2020.00148. URL https://doi.org/10.1109/ICDM50108.2020.00148.

Anh Tuan Nguyen, Tung Thanh Nguyen, and Tien N. Nguyen. Lexical statistical machine translation for language migration. In Bertrand Meyer, Luciano Baresi, and Mira Mezini (eds.), *Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE'13, Saint Petersburg, Russian Federation, August 18-26, 2013*, pp. 651–654. ACM, 2013. doi: 10.1145/2491411.2494584. URL https://doi.org/10.1145/2491411.2494584.

Anh Tuan Nguyen, Tung Thanh Nguyen, and Tien N. Nguyen. Divide-and-conquer approach for multi-phase statistical migration for source code (T). In Myra B. Cohen, Lars Grunske, and Michael Whalen (eds.), *30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, Lincoln, NE, USA, November 9-13, 2015*, pp. 585–596. IEEE Computer Society, 2015. doi: 10.1109/ASE.2015.74. URL https://doi.org/10.1109/ASE.2015.74.

Hoan Anh Nguyen, Tien N. Nguyen, Danny Dig, Son Nguyen, Hieu Tran, and Michael Hilton. Graph-based mining of in-the-wild, fine-grained, semantic code change patterns. In Joanne M. Atlee, Tevfik Bultan, and Jon Whittle (eds.), *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, pp. 819–830. IEEE / ACM, 2019. doi: 10.1109/ICSE.2019.00089. URL https://doi.org/10.1109/ICSE.2019.00089.

Nhan Nguyen and Sarah Nadi. An empirical evaluation of github copilot's code suggestions. In *19th IEEE/ACM International Conference on Mining Software Repositories, MSR 2022, Pittsburgh, PA, USA, May 23-24, 2022*, pp. 1–5. ACM, 2022. doi: 10.1145/3524842.3528470. URL https://doi.org/10.1145/3524842.3528470.

Son Nguyen, Hung Phan, Trinh Le, and Tien N. Nguyen. Suggesting natural method names to check name consistencies. In Gregg Rothermel and Doo-Hwan Bae (eds.), *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, pp. 1372–1384. ACM, 2020. doi: 10.1145/3377811.3380926. URL https://doi.org/10.1145/3377811.3380926.

Thieu Nguyen, Satoru Kobayashi, and Kensuke Fukuda. Logdtl: Network log template generation with deep transfer learning. In Toufik Ahmed, Olivier Festor, Yacine Ghamri-Doudane, Joon-Myung Kang, Alberto E. Schaeffer Filho, Abdelkader Lahmadi, and Edmundo R. M. Madeira (eds.), *17th IFIP/IEEE International Symposium on Integrated Network Management, IM 2021, Bordeaux, France, May 17-21, 2021*, pp. 848–853. IEEE, 2021. URL https://ieeexplore.ieee.org/document/9464068.

Trong Duc Nguyen, Anh Tuan Nguyen, and Tien N. Nguyen. Mapping API elements for code migration with vector representations. In Laura K. Dillon, Willem Visser, and Laurie A. Williams (eds.), *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016 - Companion Volume*, pp. 756–758. ACM, 2016. doi: 10.1145/2889160.2892661. URL https://doi.org/10.1145/2889160.2892661.

Van-Anh Nguyen, Dai Quoc Nguyen, Van Nguyen, Trung Le, Quan Hung Tran, and Dinh Phung. Regvd: Revisiting graph neural networks for vulnerability detection. In *44th IEEE/ACM International Conference on Software Engineering: Companion Proceedings, ICSE Companion 2022, Pittsburgh, PA, USA, May 22-24, 2022*, pp. 178–182. ACM/IEEE, 2022. doi: 10.1145/3510454.3516865. URL https://doi.org/10.1145/3510454.3516865.

Daniel Nichols, Joshua Hoke Davis, Zhaojun Xie, Arjun Rajaram, and Abhinav Bhatele. Can large language models write parallel code? *CoRR*, abs/2401.12554, 2024. doi: 10.48550/ARXIV.2401.12554. URL https://doi.org/10.48550/arXiv.2401.12554.

Lun Yiu Nie, Cuiyun Gao, Zhicong Zhong, Wai Lam, Yang Liu, and Zenglin Xu. Coregen: Contextualized code representation learning for commit message generation. *Neurocomputing*, 459:97–107, 2021. doi: 10.1016/J.NEUCOM.2021.05.039. URL https://doi.org/10.1016/j.neucom.2021.05.039.

Pengyu Nie, Rahul Banerjee, Junyi Jessy Li, Raymond J. Mooney, and Milos Gligoric. Learning deep semantics for test completion. In *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*, pp. 2111–2123. IEEE, 2023. doi: 10.1109/ICSE48619.2023.00178. URL https://doi.org/10.1109/ICSE48619.2023.00178.

Erik Nijkamp, Hiroaki Hayashi, Caiming Xiong, Silvio Savarese, and Yingbo Zhou. Codegen2: Lessons for training llms on programming and natural languages. *CoRR*, abs/2305.02309, 2023a. doi: 10.48550/ARXIV.2305.02309. URL https://doi.org/10.48550/arXiv.2305.02309.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. Codegen: An open large language model for code with multi-turn program synthesis. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023b. URL https://openreview.net/pdf?id=iaYcJKpY2B_.

Georgios Nikitopoulos, Konstantina Dritsa, Panos Louridas, and Dimitris Mitropoulos. Crossvul: a cross-language vulnerability dataset with commit data. In Diomidis Spinellis, Georgios Gousios, Marsha Chechik,

and Massimiliano Di Penta (eds.), *ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, August 23-28, 2021*, pp. 1565–1569. ACM, 2021. doi: 10.1145/3468264.3473122. URL https://doi.org/10.1145/3468264.3473122.

Vikram Nitin, Anthony Saieva, Baishakhi Ray, and Gail Kaiser. DIRECT : A transformer-based model for decompiled identifier renaming. In Royi Lachmy, Ziyu Yao, Greg Durrett, Milos Gligoric, Junyi Jessy Li, Ray Mooney, Graham Neubig, Yu Su, Huan Sun, and Reut Tsarfaty (eds.), *Proceedings of the 1st Workshop on Natural Language Processing for Programming (NLP4Prog 2021)*, pp. 48–57, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.nlp4prog-1.6. URL https://aclanthology.org/2021.nlp4prog-1.6.

Changan Niu, Chuanyi Li, Vincent Ng, Jidong Ge, Liguo Huang, and Bin Luo. Spt-code: Sequence-to-sequence pre-training for learning source code representations. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pp. 1–13. ACM, 2022. doi: 10.1145/3510003.3510096. URL https://doi.org/10.1145/3510003.3510096.

Changan Niu, Chuanyi Li, Vincent Ng, Dongxiao Chen, Jidong Ge, and Bin Luo. An empirical comparison of pre-trained models of source code. In *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*, pp. 2136–2148. IEEE, 2023. doi: 10.1109/ICSE48619.2023.00180. URL https://doi.org/10.1109/ICSE48619.2023.00180.

Changan Niu, Ting Zhang, Chuanyi Li, Bin Luo, and Vincent Ng. On evaluating the efficiency of source code generated by llms. *CoRR*, abs/2404.06041, 2024. doi: 10.48550/ARXIV.2404.06041. URL https://doi.org/10.48550/arXiv.2404.06041.

Yu Nong, Rainy Sharma, Abdelwahab Hamou-Lhadj, Xiapu Luo, and Haipeng Cai. Open science in software engineering: A study on deep learning-based vulnerability detection. *IEEE Trans. Software Eng.*, 49(4): 1983–2005, 2023. doi: 10.1109/TSE.2022.3207149. URL https://doi.org/10.1109/TSE.2022.3207149.

Maxwell I. Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models. *CoRR*, abs/2112.00114, 2021. URL https://arxiv.org/abs/2112.00114.

Augustus Odena, Catherine Olsson, David G. Andersen, and Ian J. Goodfellow. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4901–4911. PMLR, 2019. URL http://proceedings.mlr.press/v97/odena19a.html.

Saeyoon Oh and Shin Yoo. Csa-trans: Code structure aware transformer for ast. *CoRR*, abs/2404.05767, 2024. doi: 10.48550/ARXIV.2404.05767. URL https://doi.org/10.48550/arXiv.2404.05767.

Wonseok Oh and Hakjoo Oh. Pyter: effective program repair for python type errors. In Abhik Roychoudhury, Cristian Cadar, and Miryung Kim (eds.), *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022, Singapore, Singapore, November 14-18, 2022*, pp. 922–934. ACM, 2022. doi: 10.1145/3540250.3549130. URL https://doi.org/10.1145/3540250.3549130.

Milos Ojdanic, Aayush Garg, Ahmed Khanfir, Renzo Degiovanni, Mike Papadakis, and Yves Le Traon. Syntactic vs. semantic similarity of artificial and real faults in mutation testing studies. *CoRR*, abs/2112.14508, 2021. URL https://arxiv.org/abs/2112.14508.

Milos Ojdanic, Ahmed Khanfir, Aayush Garg, Renzo Degiovanni, Mike Papadakis, and Yves Le Traon. On comparing mutation testing tools through learning-based mutant selection. In *IEEE/ACM International Conference on Automation of Software Test, AST 2023, Melbourne, Australia, May 15-16, 2023*, pp. 35–46. IEEE, 2023. doi: 10.1109/AST58925.2023.00008. URL https://doi.org/10.1109/AST58925.2023.00008.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. doi: 10.48550/arXiv.2303.08774. URL https://doi.org/10.48550/arXiv.2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Rangeet Pan, Ali Reza Ibrahimzada, Rahul Krishna, Divya Sankar, Lambert Pouguem Wassi, Michele Merler, Boris Sobolev, Raju Pavuluri, Saurabh Sinha, and Reyhaneh Jabbarvand. Understanding the effectiveness of large language models in code translation. *CoRR*, abs/2308.03109, 2023. doi: 10.48550/ARXIV.2308.03109. URL https://doi.org/10.48550/arXiv.2308.03109.

Xinglu Pan, Chenxiao Liu, Yanzhen Zou, Tao Xie, and Bing Xie. MESIA: understanding and leveraging supplementary nature of method-level comments for automatic comment generation. *CoRR*, abs/2403.17357, 2024. doi: 10.48550/ARXIV.2403.17357. URL https://doi.org/10.48550/arXiv.2403.17357.

Irene Vlassi Pandi, Earl T. Barr, Andrew D. Gordon, and Charles Sutton. Opttyper: Probabilistic type inference by optimising logical and natural constraints. *CoRR*, abs/2004.00348, 2020. URL https://arxiv.org/abs/2004.00348.

Annibale Panichella, Fitsum Meshesha Kifetew, and Paolo Tonella. Automated test case generation as a many-objective optimisation problem with dynamic selection of the targets. *IEEE Trans. Software Eng.*, 44(2):122–158, 2018. doi: 10.1109/TSE.2017.2663435. URL https://doi.org/10.1109/TSE.2017.2663435.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pp. 311–318. ACL, 2002. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040/.

Nikhil Parasaram, Huijie Yan, Boyu Yang, Zineb Flahy, Abriele Qudsi, Damian Ziaber, Earl Barr, and Sergey Mechtaev. The fact selection problem in llm-based program repair. *CoRR*, abs/2404.05520, 2024. doi: 10.48550/ARXIV.2404.05520. URL https://doi.org/10.48550/arXiv.2404.05520.

Emilio Parisotto, Abdel-rahman Mohamed, Rishabh Singh, Lihong Li, Dengyong Zhou, and Pushmeet Kohli. Neuro-symbolic program synthesis. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=rJ0JwFcex.

Md. Rizwan Parvez, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. Building language models for text with named entities. In Iryna Gurevych and Yusuke Miyao (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pp. 2373–2383. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-1221. URL https://aclanthology.org/P18-1221/.

Md. Rizwan Parvez, Wasi Uddin Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. Retrieval augmented code generation and summarization. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pp. 2719–2734. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.FINDINGS-EMNLP.232. URL https://doi.org/10.18653/v1/2021.findings-emnlp.232.

Jibesh Patra and Michael Pradel. Semantic bug seeding: a learning-based approach for creating realistic bugs. In Diomidis Spinellis, Georgios Gousios, Marsha Chechik, and Massimiliano Di Penta (eds.), *ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations*

of Software Engineering, Athens, Greece, August 23-28, 2021, pp. 906–918. ACM, 2021. doi: 10.1145/3468264.3468623. URL https://doi.org/10.1145/3468264.3468623.

Rishov Paul, Md. Mohib Hossain, Masum Hasan, and Anindya Iqbal. Automated program repair based on code review: How do pre-trained transformer models perform? *CoRR*, abs/2304.07840, 2023. doi: 10.48550/ARXIV.2304.07840. URL https://doi.org/10.48550/arXiv.2304.07840.

Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. Asleep at the keyboard? assessing the security of github copilot's code contributions. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, pp. 754–768. IEEE, 2022. doi: 10.1109/SP46214.2022.9833571. URL https://doi.org/10.1109/SP46214.2022.9833571.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon LLM: outperforming curated corpora with web data, and web data only. *CoRR*, abs/2306.01116, 2023. doi: 10.48550/arXiv.2306.01116. URL https://doi.org/10.48550/arXiv.2306.01116.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. *CoRR*, abs/2309.00071, 2023a. doi: 10.48550/ARXIV.2309.00071. URL https://doi.org/10.48550/arXiv.2309.00071.

Yun Peng, Cuiyun Gao, Zongjie Li, Bowei Gao, David Lo, Qirun Zhang, and Michael R. Lyu. Static inference meets deep learning: A hybrid type inference approach for python. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pp. 2019–2030. ACM, 2022. doi: 10.1145/3510003.3510038. URL https://doi.org/10.1145/3510003.3510038.

Yun Peng, Shuzheng Gao, Cuiyun Gao, Yintong Huo, and Michael R. Lyu. Domain knowledge matters: Improving prompts with fix templates for repairing python type errors. *CoRR*, abs/2306.01394, 2023b. doi: 10.48550/ARXIV.2306.01394. URL https://doi.org/10.48550/arXiv.2306.01394.

Yun Peng, Chaozheng Wang, Wenxuan Wang, Cuiyun Gao, and Michael R. Lyu. Generative type inference for python. *CoRR*, abs/2307.09163, 2023c. doi: 10.48550/ARXIV.2307.09163. URL https://doi.org/10.48550/arXiv.2307.09163.

Daniel Perez and Shigeru Chiba. Cross-language clone detection by learning over abstract syntax trees. In Margaret-Anne D. Storey, Bram Adams, and Sonia Haiduc (eds.), *Proceedings of the 16th International Conference on Mining Software Repositories, MSR 2019, 26-27 May 2019, Montreal, Canada*, pp. 518–528. IEEE / ACM, 2019. doi: 10.1109/MSR.2019.00078. URL https://doi.org/10.1109/MSR.2019.00078.

Neil Perry, Megha Srivastava, Deepak Kumar, and Dan Boneh. Do users write more insecure code with AI assistants? In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda (eds.), *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pp. 2785–2799. ACM, 2023. doi: 10.1145/3576915.3623157. URL https://doi.org/10.1145/3576915.3623157.

Jannik Pewny and Thorsten Holz. Evilcoder: automated bug insertion. In Stephen Schwab, William K. Robertson, and Davide Balzarotti (eds.), *Proceedings of the 32nd Annual Conference on Computer Security Applications, ACSAC 2016, Los Angeles, CA, USA, December 5-9, 2016*, pp. 214–225. ACM, 2016. URL http://dl.acm.org/citation.cfm?id=2991103.

Hung Dang Phan, Anh Tuan Nguyen, Trong Duc Nguyen, and Tien N. Nguyen. Statistical migration of API usages. In Sebastián Uchitel, Alessandro Orso, and Martin P. Robillard (eds.), *Proceedings of the 39th International Conference on Software Engineering, ICSE 2017, Buenos Aires, Argentina, May 20-28, 2017 - Companion Volume*, pp. 47–50. IEEE Computer Society, 2017. doi: 10.1109/ICSE-C.2017.17. URL https://doi.org/10.1109/ICSE-C.2017.17.

Huy N. Phan, Hoang N. Phan, Tien N. Nguyen, and Nghi D. Q. Bui. Repohyper: Better context retrieval is all you need for repository-level code completion. *CoRR*, abs/2403.06095, 2024. doi: 10.48550/ARXIV.2403.06095. URL https://doi.org/10.48550/arXiv.2403.06095.

Juan Altmayer Pizzorno and Emery D. Berger. Coverup: Coverage-guided llm-based test generation. *CoRR*, abs/2403.16218, 2024. doi: 10.48550/ARXIV.2403.16218. URL https://doi.org/10.48550/arXiv.2403.16218.

Serena Elisa Ponta, Henrik Plate, Antonino Sabetta, Michele Bezzi, and Cédric Dangremont. A manually-curated dataset of fixes to vulnerabilities of open-source software. In Margaret-Anne D. Storey, Bram Adams, and Sonia Haiduc (eds.), *Proceedings of the 16th International Conference on Mining Software Repositories, MSR 2019, 26-27 May 2019, Montreal, Canada*, pp. 383–387. IEEE / ACM, 2019. doi: 10.1109/MSR.2019.00064. URL https://doi.org/10.1109/MSR.2019.00064.

Mohammadreza Pourreza and Davood Rafiei. DIN-SQL: decomposed in-context learning of text-to-sql with self-correction. *CoRR*, abs/2304.11015, 2023. doi: 10.48550/ARXIV.2304.11015. URL https://doi.org/10.48550/arXiv.2304.11015.

Michael Pradel and Koushik Sen. Deepbugs: a learning approach to name-based bug detection. *Proc. ACM Program. Lang.*, 2(OOPSLA):147:1–147:25, 2018. doi: 10.1145/3276517. URL https://doi.org/10.1145/3276517.

Michael Pradel, Parker Schuh, and Koushik Sen. Typedevil: Dynamic type inconsistency analysis for javascript. In Antonia Bertolino, Gerardo Canfora, and Sebastian G. Elbaum (eds.), *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 1*, pp. 314–324. IEEE Computer Society, 2015. doi: 10.1109/ICSE.2015.51. URL https://doi.org/10.1109/ICSE.2015.51.

Michael Pradel, Georgios Gousios, Jason Liu, and Satish Chandra. Typewriter: neural type prediction with search-based validation. In Prem Devanbu, Myra B. Cohen, and Thomas Zimmermann (eds.), *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, pp. 209–220. ACM, 2020. doi: 10.1145/3368089.3409715. URL https://doi.org/10.1145/3368089.3409715.

Julian Aron Prenner and Romain Robbes. Runbugrun - an executable dataset for automated program repair. *CoRR*, abs/2304.01102, 2023. doi: 10.48550/ARXIV.2304.01102. URL https://doi.org/10.48550/arXiv.2304.01102.

Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=R8sQPpGCv0.

Yewen Pu, Karthik Narasimhan, Armando Solar-Lezama, and Regina Barzilay. sk_p: a neural program corrector for moocs. In Eelco Visser (ed.), *Companion Proceedings of the 2016 ACM SIGPLAN International Conference on Systems, Programming, Languages and Applications: Software for Humanity, SPLASH 2016, Amsterdam, Netherlands, October 30 - November 4, 2016*, pp. 39–40. ACM, 2016. doi: 10.1145/2984043.2989222. URL https://doi.org/10.1145/2984043.2989222.

Ruchir Puri, David S. Kung, Geert Janssen, Wei Zhang, Giacomo Domeniconi, Vladimir Zolotov, Julian Dolby, Jie Chen, Mihir R. Choudhury, Lindsey Decker, Veronika Thost, Luca Buratti, Saurabh Pujar, Shyam Ramji, Ulrich Finkler, Susan Malaika, and Frederick Reiss. Codenet: A large-scale AI for code dataset for learning a diversity of coding tasks. In Joaquin Vanschoren and Sai-Kit Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/a5bfc9e07964f8dddeb95fc584cd965d-Abstract-round2.html.

Jiaxing Qi, Shaohan Huang, Zhongzhi Luan, Carol J. Fung, Hailong Yang, and Depei Qian. Loggpt: Exploring chatgpt for log-based anomaly detection. *CoRR*, abs/2309.01189, 2023. doi: 10.48550/ARXIV.2309.01189. URL https://doi.org/10.48550/arXiv.2309.01189.

Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. Communicative agents for software development. *CoRR*, abs/2307.07924, 2023. doi: 10.48550/ARXIV.2307.07924. URL https://doi.org/10.48550/arXiv.2307.07924.

Bowen Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binhua Li, Ruiying Geng, Rongyu Cao, Jian Sun, Luo Si, Fei Huang, and Yongbin Li. A survey on text-to-sql parsing: Concepts, methods, and future directions. *CoRR*, abs/2208.13629, 2022a. doi: 10.48550/ARXIV.2208.13629. URL https://doi.org/10.48550/arXiv.2208.13629.

Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosformer: Rethinking softmax in attention. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022b. URL https://openreview.net/forum?id=Bl8CQrx2Up4.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. Abstract syntax networks for code generation and semantic parsing. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pp. 1139–1149. Association for Computational Linguistics, 2017. doi: 10.18653/V1/P17-1105. URL https://doi.org/10.18653/v1/P17-1105.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, H. Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant M. Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew J. Johnson, Blake A. Hechtman, Laura Weidinger, Iason Gabriel, William S. Isaac, Edward Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling language models: Methods, analysis & insights from training gopher. *CoRR*, abs/2112.11446, 2021. URL https://arxiv.org/abs/2112.11446.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Abir Rahali and Moulay A. Akhloufi. Malbertv2: Code aware bert-based model for malware identification. *Big Data Cogn. Comput.*, 7(2):60, 2023. doi: 10.3390/BDCC7020060. URL https://doi.org/10.3390/bdcc7020060.

Md Mahbubur Rahman, Ira Ceka, Chengzhi Mao, Saikat Chakraborty, Baishakhi Ray, and Wei Le. Towards causal deep learning for vulnerability detection. *CoRR*, abs/2310.07958, 2023. doi: 10.48550/ARXIV.2310.07958. URL https://doi.org/10.48550/arXiv.2310.07958.

Baishakhi Ray, Vincent J. Hellendoorn, Saheel Godhane, Zhaopeng Tu, Alberto Bacchelli, and Premkumar T. Devanbu. On the "naturalness" of buggy code. In Laura K. Dillon, Willem Visser, and Laurie A. Williams (eds.), *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016*, pp. 428–439. ACM, 2016. doi: 10.1145/2884781.2884848. URL https://doi.org/10.1145/2884781.2884848.

Veselin Raychev, Martin T. Vechev, and Eran Yahav. Code completion with statistical language models. In Michael F. P. O'Boyle and Keshav Pingali (eds.), *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '14, Edinburgh, United Kingdom - June 09 - 11, 2014*, pp. 419–428. ACM, 2014. doi: 10.1145/2594291.2594321. URL https://doi.org/10.1145/2594291.2594321.

Veselin Raychev, Martin T. Vechev, and Andreas Krause. Predicting program properties from "big code". In Sriram K. Rajamani and David Walker (eds.), *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, pp. 111–124. ACM, 2015. doi: 10.1145/2676726.2677009. URL https://doi.org/10.1145/2676726.2677009.

Veselin Raychev, Pavol Bielik, and Martin T. Vechev. Probabilistic model for code with decision trees. In Eelco Visser and Yannis Smaragdakis (eds.), *Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2016, part of SPLASH 2016, Amsterdam, The Netherlands, October 30 - November 4, 2016*, pp. 731–747. ACM, 2016a. doi: 10.1145/2983990.2984041. URL https://doi.org/10.1145/2983990.2984041.

Veselin Raychev, Pavol Bielik, Martin T. Vechev, and Andreas Krause. Learning programs from noisy data. In Rastislav Bodík and Rupak Majumdar (eds.), *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, pp. 761–774. ACM, 2016b. doi: 10.1145/2837614.2837671. URL https://doi.org/10.1145/2837614.2837671.

Shuo Ren, Daya Guo, Shuai Lu, Long Zhou, Shujie Liu, Duyu Tang, Neel Sundaresan, Ming Zhou, Ambrosio Blanco, and Shuai Ma. Codebleu: a method for automatic evaluation of code synthesis. *CoRR*, abs/2009.10297, 2020. URL https://arxiv.org/abs/2009.10297.

Cedric Richter and Heike Wehrheim. TSSB-3M: mining single statement bugs at massive scale. In *19th IEEE/ACM International Conference on Mining Software Repositories, MSR 2022, Pittsburgh, PA, USA, May 23-24, 2022*, pp. 418–422. ACM, 2022. doi: 10.1145/3524842.3528505. URL https://doi.org/10.1145/3524842.3528505.

Niklas Risse and Marcel Böhme. Limits of machine learning for automatic vulnerability detection. *CoRR*, abs/2306.17193, 2023. doi: 10.48550/ARXIV.2306.17193. URL https://doi.org/10.48550/arXiv.2306.17193.

Marcel Robeer, Garm Lucassen, Jan Martijn E. M. van der Werf, Fabiano Dalpiaz, and Sjaak Brinkkemper. Automated extraction of conceptual models from user stories via NLP. In *24th IEEE International Requirements Engineering Conference, RE 2016, Beijing, China, September 12-16, 2016*, pp. 196–205. IEEE Computer Society, 2016. doi: 10.1109/RE.2016.40. URL https://doi.org/10.1109/RE.2016.40.

Alex Robinson. Sketch2code: Generating a website from a paper mockup. *CoRR*, abs/1905.13750, 2019. URL http://arxiv.org/abs/1905.13750.

Alberto D. Rodriguez, Katherine R. Dearstyne, and Jane Cleland-Huang. Prompts matter: Insights and strategies for prompt engineering in automated software traceability. In Kurt Schneider, Fabiano Dalpiaz, and Jennifer Horkoff (eds.), *31st IEEE International Requirements Engineering Conference, RE 2023 - Workshops, Hannover, Germany, September 4-5, 2023*, pp. 455–464. IEEE, 2023. doi: 10.1109/REW57809.2023.00087. URL https://doi.org/10.1109/REW57809.2023.00087.

Krishna Ronanki, Beatriz Cabrero Daniel, Jennifer Horkoff, and Christian Berger. Requirements engineering using generative AI: prompts and prompting patterns. *CoRR*, abs/2311.03832, 2023. doi: 10.48550/ARXIV.2311.03832. URL https://doi.org/10.48550/arXiv.2311.03832.

Subhajit Roy, Awanish Pandey, Brendan Dolan-Gavitt, and Yu Hu. Bug synthesis: challenging bug-finding tools with deep faults. In Gary T. Leavens, Alessandro Garcia, and Corina S. Pasareanu (eds.), *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2018, Lake Buena Vista, FL, USA, November*

04-09, 2018, pp. 224–234. ACM, 2018. doi: 10.1145/3236024.3236084. URL https://doi.org/10.1145/3236024.3236084.

Baptiste Rozière, Marie-Anne Lachaux, Lowik Chanussot, and Guillaume Lample. Unsupervised translation of programming languages. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/ed23fbf18c2cd35f8c7f8de44f85c08d-Abstract.html.

Baptiste Rozière, Jie Zhang, François Charton, Mark Harman, Gabriel Synnaeve, and Guillaume Lample. Leveraging automated unit tests for unsupervised code translation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=cmt-6KtR4c4.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton-Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code. *CoRR*, abs/2308.12950, 2023. doi: 10.48550/arXiv.2308.12950. URL https://doi.org/10.48550/arXiv.2308.12950.

Jingqing Ruan, Yihong Chen, Bin Zhang, Zhiwei Xu, Tianpeng Bao, Guoqing Du, Shiwei Shi, Hangyu Mao, Xingyu Zeng, and Rui Zhao. TPTU: task planning and tool usage of large language model-based AI agents. *CoRR*, abs/2308.03427, 2023. doi: 10.48550/ARXIV.2308.03427. URL https://doi.org/10.48550/arXiv.2308.03427.

Ohad Rubin and Jonathan Berant. Smbop: Semi-autoregressive bottom-up semantic parsing. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pp. 311–324. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.29. URL https://doi.org/10.18653/v1/2021.naacl-main.29.

Rebecca L. Russell, Louis Y. Kim, Lei H. Hamilton, Tomo Lazovich, Jacob Harer, Onur Ozdemir, Paul M. Ellingwood, and Marc W. McConley. Automated vulnerability detection in source code using deep representation learning. In M. Arif Wani, Mehmed M. Kantardzic, Moamar Sayed Mouchaweh, João Gama, and Edwin Lughofer (eds.), *17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018, Orlando, FL, USA, December 17-20, 2018*, pp. 757–762. IEEE, 2018. doi: 10.1109/ICMLA.2018.00120. URL https://doi.org/10.1109/ICMLA.2018.00120.

Saksham Sachdev, Hongyu Li, Sifei Luan, Seohyun Kim, Koushik Sen, and Satish Chandra. Retrieval on source code: a neural code search. In Justin Gottschlich and Alvin Cheung (eds.), *Proceedings of the 2nd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages, MAPL@PLDI 2018, Philadelphia, PA, USA, June 18-22, 2018*, pp. 31–41. ACM, 2018. doi: 10.1145/3211346.3211353. URL https://doi.org/10.1145/3211346.3211353.

Caitlin Sadowski, Jeffrey van Gogh, Ciera Jaspan, Emma Söderberg, and Collin Winter. Tricorder: Building a program analysis ecosystem. In Antonia Bertolino, Gerardo Canfora, and Sebastian G. Elbaum (eds.), *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 1*, pp. 598–608. IEEE Computer Society, 2015. doi: 10.1109/ICSE.2015.76. URL https://doi.org/10.1109/ICSE.2015.76.

Ripon K. Saha, Yingjun Lyu, Wing Lam, Hiroaki Yoshida, and Mukul R. Prasad. Bugs.jar: a large-scale, diverse dataset of real-world java bugs. In Andy Zaidman, Yasutaka Kamei, and Emily Hill (eds.), *Proceedings of the 15th International Conference on Mining Software Repositories, MSR 2018, Gothenburg,*

Sweden, May 28-29, 2018, pp. 10–13. ACM, 2018. doi: 10.1145/3196398.3196473. URL https://doi.org/10.1145/3196398.3196473.

Muhammet Sahin and Serif Bahtiyar. A survey on malware detection with deep learning. In Siddika Berna Örs and Atilla Elçi (eds.), *SIN 2020: 13th International Conference on Security of Information and Networks, Virtual Event / Istanbul, Turkey, November 4-6, 2020*, pp. 34:1–34:6. ACM, 2020. doi: 10.1145/3433174.3433609. URL https://doi.org/10.1145/3433174.3433609.

Anthony Saieva, Saikat Chakraborty, and Gail E. Kaiser. On contrastive learning of semantic similarity forcode to code search. *CoRR*, abs/2305.03843, 2023. doi: 10.48550/ARXIV.2305.03843. URL https://doi.org/10.48550/arXiv.2305.03843.

Abhishek Sainani, Preethu Rose Anish, Vivek Joshi, and Smita Ghaisas. Extracting and classifying requirements from software engineering contracts. In Travis D. Breaux, Andrea Zisman, Samuel Fricker, and Martin Glinz (eds.), *28th IEEE International Requirements Engineering Conference, RE 2020, Zurich, Switzerland, August 31 - September 4, 2020*, pp. 147–157. IEEE, 2020. doi: 10.1109/RE48521.2020.00026. URL https://doi.org/10.1109/RE48521.2020.00026.

Vaibhav Saini, Farima Farmahinifarahani, Yadong Lu, Pierre Baldi, and Cristina V. Lopes. Oreo: detection of clones in the twilight zone. In Gary T. Leavens, Alessandro Garcia, and Corina S. Pasareanu (eds.), *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2018, Lake Buena Vista, FL, USA, November 04-09, 2018*, pp. 354–365. ACM, 2018. doi: 10.1145/3236024.3236026. URL https://doi.org/10.1145/3236024.3236026.

Hitesh Sajnani, Vaibhav Saini, Jeffrey Svajlenko, Chanchal K. Roy, and Cristina V. Lopes. Sourcerercc: scaling code clone detection to big-code. In Laura K. Dillon, Willem Visser, and Laurie A. Williams (eds.), *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016*, pp. 1157–1168. ACM, 2016. doi: 10.1145/2884781.2884877. URL https://doi.org/10.1145/2884781.2884877.

Pasquale Salza, Christoph Schwizer, Jian Gu, and Harald C. Gall. On the effectiveness of transfer learning for code search. *IEEE Trans. Software Eng.*, 49(4):1804–1822, 2023. doi: 10.1109/TSE.2022.3192755. URL https://doi.org/10.1109/TSE.2022.3192755.

Gustavo Sandoval, Hammond Pearce, Teo Nys, Ramesh Karri, Siddharth Garg, and Brendan Dolan-Gavitt. Lost at C: A user study on the security implications of large language model code assistants. In Joseph A. Calandrino and Carmela Troncoso (eds.), *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pp. 2205–2222. USENIX Association, 2023. URL https://www.usenix.org/conference/usenixsecurity23/presentation/sandoval.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=9Vrb9D0WI4.

Laboni Sarker, Mara Downing, Achintya Desai, and Tevfik Bultan. Syntactic robustness for llm-based code generation. *CoRR*, abs/2404.01535, 2024. doi: 10.48550/ARXIV.2404.01535. URL https://doi.org/10.48550/arXiv.2404.01535.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilic, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella

Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, and et al. BLOOM: A 176b-parameter open-access multilingual language model. *CoRR*, abs/2211.05100, 2022. doi: 10.48550/arXiv.2211.05100. URL https://doi.org/10.48550/arXiv.2211.05100.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Yacmpz84TH.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: parsing incrementally for constrained auto-regressive decoding from language models. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pp. 9895–9901. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.779. URL https://doi.org/10.18653/v1/2021.emnlp-main.779.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Roei Schuster, Congzheng Song, Eran Tromer, and Vitaly Shmatikov. You autocomplete me: Poisoning vulnerabilities in neural code completion. In Michael D. Bailey and Rachel Greenstadt (eds.), *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pp. 1559–1575. USENIX Association, 2021. URL https://www.usenix.org/conference/usenixsecurity21/presentation/schuster.

Max Schäfer, Sarah Nadi, Aryaz Eghbali, and Frank Tip. An empirical evaluation of using large language models for automated unit test generation, 2023.

Marija Selakovic, Michael Pradel, Rezwana Karim, and Frank Tip. Test generation for higher-order functions in dynamic languages. *Proc. ACM Program. Lang.*, 2(OOPSLA):161:1–161:27, 2018. doi: 10.1145/3276531. URL https://doi.org/10.1145/3276531.

Sachith Seneviratne, Ridwan Shariffdeen, Sanka Rasnayaka, and Nuran Kasthuriarachchi. Self-supervised vision transformers for malware detection. *IEEE Access*, 10:103121–103135, 2022. doi: 10.1109/ACCESS.2022.3206445. URL https://doi.org/10.1109/ACCESS.2022.3206445.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016. doi: 10.18653/v1/p16-1009. URL https://doi.org/10.18653/v1/p16-1009.

Ramin Shahbazi and Fatemeh Hendijani Fard. Apicontext2com: Code comment generation by incorporating pre-defined API documentation. In *31st IEEE/ACM International Conference on Program Comprehension, ICPC 2023, Melbourne, Australia, May 15-16, 2023*, pp. 13–24. IEEE, 2023. doi: 10.1109/ICPC58990.2023.00012. URL https://doi.org/10.1109/ICPC58990.2023.00012.

Sina Shamshiri. Automated unit test generation for evolving software. In Elisabetta Di Nitto, Mark Harman, and Patrick Heymans (eds.), *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, Bergamo, Italy, August 30 - September 4, 2015*, pp. 1038–1041. ACM, 2015. doi: 10.1145/2786805.2803196. URL https://doi.org/10.1145/2786805.2803196.

Rishab Sharma, Fuxiang Chen, and Fatemeh H. Fard. LAMNER: code comment generation using character language model and named entity recognition. In Ayushi Rastogi, Rosalia Tufano, Gabriele Bavota, Venera Arnaoudova, and Sonia Haiduc (eds.), *Proceedings of the 30th IEEE/ACM International Conference on*

Program Comprehension, ICPC 2022, Virtual Event, May 16-17, 2022, pp. 48–59. ACM, 2022. doi: 10.1145/3524610.3527924. URL https://doi.org/10.1145/3524610.3527924.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In Marilyn A. Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pp. 464–468. Association for Computational Linguistics, 2018. doi: 10.18653/V1/N18-2074. URL https://doi.org/10.18653/v1/n18-2074.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 922–938. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.75. URL https://doi.org/10.18653/v1/2021.acl-long.75.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *CoRR*, abs/1911.02150, 2019. URL http://arxiv.org/abs/1911.02150.

Dongdong She, Kexin Pei, Dave Epstein, Junfeng Yang, Baishakhi Ray, and Suman Jana. NEUZZ: efficient fuzzing with neural program smoothing. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pp. 803–817. IEEE, 2019. doi: 10.1109/SP.2019.00052. URL https://doi.org/10.1109/SP.2019.00052.

Dongdong She, Rahul Krishna, Lu Yan, Suman Jana, and Baishakhi Ray. Mtfuzz: fuzzing with a multi-task neural network. In Prem Devanbu, Myra B. Cohen, and Thomas Zimmermann (eds.), *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, pp. 737–749. ACM, 2020. doi: 10.1145/3368089.3409723. URL https://doi.org/10.1145/3368089.3409723.

Xinyu She, Yue Liu, Yanjie Zhao, Yiling He, Li Li, Chakkrit Tantithamthavorn, Zhan Qin, and Haoyu Wang. Pitfalls in language models for code intelligence: A taxonomy and survey. *CoRR*, abs/2310.17903, 2023. doi: 10.48550/ARXIV.2310.17903. URL https://doi.org/10.48550/arXiv.2310.17903.

Bo Shen, Jiaxin Zhang, Taihong Chen, Daoguang Zan, Bing Geng, An Fu, Muhan Zeng, Ailun Yu, Jichuan Ji, Jingyang Zhao, Yuenan Guo, and Qianxiang Wang. Pangu-coder2: Boosting large language models for code with ranking feedback. *CoRR*, abs/2307.14936, 2023. doi: 10.48550/arXiv.2307.14936. URL https://doi.org/10.48550/arXiv.2307.14936.

Chaochen Shi, Borui Cai, Yao Zhao, Longxiang Gao, Keshav Sood, and Yong Xiang. Coss: Leveraging statement semantics for code summarization. *IEEE Trans. Software Eng.*, 49(6):3472–3486, 2023a. doi: 10.1109/TSE.2023.3256362. URL https://doi.org/10.1109/TSE.2023.3256362.

Ensheng Shi, Yanlin Wang, Wei Tao, Lun Du, Hongyu Zhang, Shi Han, Dongmei Zhang, and Hongbin Sun. RACE: retrieval-augmented commit message generation. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 5520–5530. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.EMNLP-MAIN.372. URL https://doi.org/10.18653/v1/2022.emnlp-main.372.

Ensheng Shi, Yanlin Wang, Wenchao Gu, Lun Du, Hongyu Zhang, Shi Han, Dongmei Zhang, and Hongbin Sun. Cocosoda: Effective contrastive learning for code search. In *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*, pp. 2198–2210. IEEE, 2023b. doi: 10.1109/ICSE48619.2023.00185. URL https://doi.org/10.1109/ICSE48619.2023.00185.

Peng Shi, Patrick Ng, Zhiguo Wang, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Cícero Nogueira dos Santos, and Bing Xiang. Learning contextual representations for semantic parsing with generation-augmented pre-training. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 13806–13814. AAAI Press, 2021. doi: 10.1609/AAAI.V35I15.17627. URL https://doi.org/10.1609/aaai.v35i15.17627.

Shu-Ting Shi, Ming Li, David Lo, Ferdian Thung, and Xuan Huo. Automatic code review by learning the revision of source code. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 4910–4917. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33014910. URL https://doi.org/10.1609/aaai.v33i01.33014910.

Tianze Shi, Chen Zhao, Jordan L. Boyd-Graber, Hal Daumé III, and Lillian Lee. On the potential of lexico-logical alignments for semantic parsing to SQL queries. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pp. 1849–1864. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.FINDINGS-EMNLP.167. URL https://doi.org/10.18653/v1/2020.findings-emnlp.167.

Samiha Shimmi and Mona Rahimi. Leveraging code-test co-evolution patterns for automated test case recommendation. In *IEEE/ACM International Conference on Automation of Software Test, AST@ICSE 2022, Pittsburgh, PA, USA, May 21-22, 2022*, pp. 65–76. ACM/IEEE, 2022. doi: 10.1145/3524481.3527222. URL https://doi.org/10.1145/3524481.3527222.

Jiho Shin, Sepehr Hashtroudi, Hadi Hemmati, and Song Wang. Domain adaptation for deep unit test case generation. *CoRR*, abs/2308.08033, 2023. doi: 10.48550/ARXIV.2308.08033. URL https://doi.org/10.48550/arXiv.2308.08033.

Parshin Shojaee, Aneesh Jain, Sindhu Tipirneni, and Chandan K. Reddy. Execution-based code generation using deep reinforcement learning. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=0XBuaxqEcG.

Disha Shrivastava, Denis Kocetkov, Harm de Vries, Dzmitry Bahdanau, and Torsten Scholak. Repofusion: Training code models to understand your repository. *CoRR*, abs/2306.10998, 2023a. doi: 10.48550/ARXIV.2306.10998. URL https://doi.org/10.48550/arXiv.2306.10998.

Disha Shrivastava, Hugo Larochelle, and Daniel Tarlow. Repository-level prompt generation for large language models of code. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31693–31715. PMLR, 2023b. URL https://proceedings.mlr.press/v202/shrivastava23a.html.

Jianhang Shuai, Ling Xu, Chao Liu, Meng Yan, Xin Xia, and Yan Lei. Improving code search with co-attentive representation learning. In *ICPC '20: 28th International Conference on Program Comprehension, Seoul, Republic of Korea, July 13-15, 2020*, pp. 196–207. ACM, 2020. doi: 10.1145/3387904.3389269. URL https://doi.org/10.1145/3387904.3389269.

Chenglei Si, Yanzhe Zhang, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. Design2code: How far are we from automating front-end engineering? *CoRR*, abs/2403.03163, 2024. doi: 10.48550/ARXIV.2403.03163. URL https://doi.org/10.48550/arXiv.2403.03163.

Mukul Singh, José Cambronero, Sumit Gulwani, Vu Le, Carina Negreanu, and Gust Verbruggen. Codefusion: A pre-trained diffusion model for code generation, 2023.

Jing Kai Siow, Cuiyun Gao, Lingling Fan, Sen Chen, and Yang Liu. CORE: automating review recommendation for code changes. In Kostas Kontogiannis, Foutse Khomh, Alexander Chatzigeorgiou, Marios-Eleftherios Fokaefs, and Minghui Zhou (eds.), *27th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2020, London, ON, Canada, February 18-21, 2020*, pp. 284–295. IEEE, 2020. doi: 10.1109/SANER48275.2020.9054794. URL https://doi.org/10.1109/SANER48275.2020.9054794.

Aishwarya Sivaraman, Rui Abreu, Andrew Scott, Tobi Akomolede, and Satish Chandra. Mining idioms in the wild. In *44th IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice, ICSE (SEIP) 2022, Pittsburgh, PA, USA, May 22-24, 2022*, pp. 187–196. IEEE, 2022. doi: 10.1109/ICSE-SEIP55303.2022.9794062. URL https://doi.org/10.1109/ICSE-SEIP55303.2022.9794062.

Saleh Soltan, Shankar Ananthakrishnan, Jack FitzGerald, Rahul Gupta, Wael Hamza, Haidar Khan, Charith Peris, Stephen Rawls, Andy Rosenbaum, Anna Rumshisky, Chandana Satya Prakash, Mukund Sridhar, Fabian Triefenbach, Apurv Verma, Gökhan Tür, and Prem Natarajan. Alexatm 20b: Few-shot learning using a large-scale multilingual seq2seq model. *CoRR*, abs/2208.01448, 2022. doi: 10.48550/arXiv.2208.01448. URL https://doi.org/10.48550/arXiv.2208.01448.

Nikita Sorokin, Dmitry Abulkhanov, Sergey I. Nikolenko, and Valentin Malykh. Cct-code: Cross-consistency training for multilingual clone detection and code search. *CoRR*, abs/2305.11626, 2023. doi: 10.48550/ARXIV.2305.11626. URL https://doi.org/10.48550/arXiv.2305.11626.

Davit Soselia, Khalid Saifullah, and Tianyi Zhou. Learning ui-to-code reverse generator using visual critic without rendering. *CoRR*, abs/2305.14637, 2023. doi: 10.48550/ARXIV.2305.14637. URL https://doi.org/10.48550/arXiv.2305.14637.

Benjamin Steenhoek, Md Mahbubur Rahman, Richard Jiles, and Wei Le. An empirical study of deep learning models for vulnerability detection. In *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*, pp. 2237–2248. IEEE, 2023a. doi: 10.1109/ICSE48619.2023.00188. URL https://doi.org/10.1109/ICSE48619.2023.00188.

Benjamin Steenhoek, Michele Tufano, Neel Sundaresan, and Alexey Svyatkovskiy. Reinforcement learning from automatic feedback for high-quality unit test generation. *CoRR*, abs/2310.02368, 2023b. doi: 10.48550/ARXIV.2310.02368. URL https://doi.org/10.48550/arXiv.2310.02368.

Benjamin Steenhoek, Md Mahbubur Rahman, Monoshi Kumar Roy, Mirza Sanjida Alam, Earl T. Barr, and Wei Le. A comprehensive study of the capabilities of large language models for vulnerability detection. *CoRR*, abs/2403.17218, 2024. doi: 10.48550/ARXIV.2403.17218. URL https://doi.org/10.48550/arXiv.2403.17218.

Chia-Yi Su and Collin McMillan. Semantic similarity loss for neural source code summarization. *CoRR*, abs/2308.07429, 2023a. doi: 10.48550/ARXIV.2308.07429. URL https://doi.org/10.48550/arXiv.2308.07429.

Chia-Yi Su and Collin McMillan. Distilled GPT for source code summarization. *CoRR*, abs/2308.14731, 2023b. doi: 10.48550/ARXIV.2308.14731. URL https://doi.org/10.48550/arXiv.2308.14731.

Chia-Yi Su, Aakash Bansal, Vijayanta Jain, Sepideh Ghanavati, and Collin McMillan. A language model of java methods with train/test deduplication. *CoRR*, abs/2305.08286, 2023a. doi: 10.48550/ARXIV.2305.08286. URL https://doi.org/10.48550/arXiv.2305.08286.

Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/J.NEUCOM.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Yiming Su, Chengcheng Wan, Utsav Sethi, Shan Lu, Madan Musuvathi, and Suman Nath. Hotgpt: How to make software documentation more useful with a large language model? In Malte Schwarzkopf, Andrew

Baumann, and Natacha Crooks (eds.), *Proceedings of the 19th Workshop on Hot Topics in Operating Systems, HOTOS 2023, Providence, RI, USA, June 22-24, 2023*, pp. 87–93. ACM, 2023b. doi: 10.1145/3593856.3595910. URL https://doi.org/10.1145/3593856.3595910.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. Learning to map context-dependent sentences to executable formal queries. In Marilyn A. Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pp. 2238–2249. Association for Computational Linguistics, 2018. doi: 10.18653/v1/n18-1203. URL https://doi.org/10.18653/v1/n18-1203.

Ruoxi Sun, Sercan Ö. Arik, Hootan Nakhost, Hanjun Dai, Rajarishi Sinha, Pengcheng Yin, and Tomas Pfister. Sql-palm: Improved large language model adaptation for text-to-sql. *CoRR*, abs/2306.00739, 2023a. doi: 10.48550/ARXIV.2306.00739. URL https://doi.org/10.48550/arXiv.2306.00739.

Weisong Sun, Chunrong Fang, Yuchen Chen, Guanhong Tao, Tingxu Han, and Quanjun Zhang. Code search based on context-aware code translation. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pp. 388–400. ACM, 2022. doi: 10.1145/3510003.3510140. URL https://doi.org/10.1145/3510003.3510140.

Weisong Sun, Chunrong Fang, Yudu You, Yun Miao, Yi Liu, Yuekang Li, Gelei Deng, Shenghan Huang, Yuchen Chen, Quanjun Zhang, Hanwei Qian, Yang Liu, and Zhenyu Chen. Automatic code summarization via chatgpt: How far are we? *CoRR*, abs/2305.12865, 2023b. doi: 10.48550/ARXIV.2305.12865. URL https://doi.org/10.48550/arXiv.2305.12865.

Yuqiang Sun, Daoyuan Wu, Yue Xue, Han Liu, Wei Ma, Lyuye Zhang, Miaolei Shi, and Yang Liu. Llm4vuln: A unified evaluation framework for decoupling and enhancing llms' vulnerability reasoning. *CoRR*, abs/2401.16185, 2024. doi: 10.48550/ARXIV.2401.16185. URL https://doi.org/10.48550/arXiv.2401.16185.

Zeyu Sun, Qihao Zhu, Yingfei Xiong, Yican Sun, Lili Mou, and Lu Zhang. Treegen: A tree-based transformer architecture for code generation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 8984–8991. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6430. URL https://doi.org/10.1609/aaai.v34i05.6430.

Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. *CoRR*, abs/2303.08128, 2023. doi: 10.48550/ARXIV.2303.08128. URL https://doi.org/10.48550/arXiv.2303.08128.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 3104–3112, 2014. URL https://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html.

Jeffrey Svajlenko and Chanchal K. Roy. A survey on the evaluation of clone detection performance and benchmarking. *CoRR*, abs/2006.15682, 2020. URL https://arxiv.org/abs/2006.15682.

Jeffrey Svajlenko, Judith F. Islam, Iman Keivanloo, Chanchal Kumar Roy, and Mohammad Mamun Mia. Towards a big data curated benchmark of inter-project code clones. In *30th IEEE International Conference on Software Maintenance and Evolution, Victoria, BC, Canada, September 29 - October 3, 2014*, pp. 476–480. IEEE Computer Society, 2014. doi: 10.1109/ICSME.2014.77. URL https://doi.org/10.1109/ICSME.2014.77.

Alexey Svyatkovskiy, Ying Zhao, Shengyu Fu, and Neel Sundaresan. Pythia: Ai-assisted code completion system. In Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis

(eds.), *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pp. 2727–2735. ACM, 2019. doi: 10.1145/3292500.3330699. URL https://doi.org/10.1145/3292500.3330699.

Alexey Svyatkovskiy, Shao Kun Deng, Shengyu Fu, and Neel Sundaresan. Intellicode compose: code generation using transformer. In Prem Devanbu, Myra B. Cohen, and Thomas Zimmermann (eds.), *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, pp. 1433–1443. ACM, 2020. doi: 10.1145/3368089.3417058. URL https://doi.org/10.1145/3368089.3417058.

Marc Szafraniec, Baptiste Rozière, Hugh Leather, Patrick Labatut, François Charton, and Gabriel Synnaeve. Code translation with compiler representations. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=XomEU3eNeSQ.

Florian Tambon, Arghavan Moradi Dakhel, Amin Nikanjam, Foutse Khomh, Michel C. Desmarais, and Giuliano Antoniol. Bugs in large language models generated code: An empirical study. *CoRR*, abs/2403.08937, 2024. doi: 10.48550/ARXIV.2403.08937. URL https://doi.org/10.48550/arXiv.2403.08937.

Hanzhuo Tan, Qi Luo, Jing Li, and Yuqun Zhang. Llm4decompile: Decompiling binary code with large language models. *CoRR*, abs/2403.05286, 2024. doi: 10.48550/ARXIV.2403.05286. URL https://doi.org/10.48550/arXiv.2403.05286.

Shin Hwei Tan, Jooyong Yi, Yulis, Sergey Mechtaev, and Abhik Roychoudhury. Codeflaws: a programming competition benchmark for evaluating automated program repair tools. In Sebastián Uchitel, Alessandro Orso, and Martin P. Robillard (eds.), *Proceedings of the 39th International Conference on Software Engineering, ICSE 2017, Buenos Aires, Argentina, May 20-28, 2017 - Companion Volume*, pp. 180–182. IEEE Computer Society, 2017. doi: 10.1109/ICSE-C.2017.76. URL https://doi.org/10.1109/ICSE-C.2017.76.

Lappoon R. Tang and Raymond J. Mooney. Automated construction of database interfaces: Intergrating statistical and relational learning for semantic parsing. In Hinrich Schütze and Keh-Yih Su (eds.), *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, EMNLP 2000, Hong Kong, October 7-8, 2000*, pp. 133–141. Association for Computational Linguistics, 2000. doi: 10.3115/1117794.1117811. URL https://aclanthology.org/W00-1317/.

Yutian Tang, Zhijie Liu, Zhichao Zhou, and Xiapu Luo. Chatgpt vs SBST: A comparative assessment of unit test suite generation. *CoRR*, abs/2307.00588, 2023a. doi: 10.48550/ARXIV.2307.00588. URL https://doi.org/10.48550/arXiv.2307.00588.

Ze Tang, Xiaoyu Shen, Chuanyi Li, Jidong Ge, Liguo Huang, Zheling Zhu, and Bin Luo. Ast-trans: Code summarization with efficient tree-structured attention. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pp. 150–162. ACM, 2022. doi: 10.1145/3510003.3510224. URL https://doi.org/10.1145/3510003.3510224.

Zilu Tang, Mayank Agarwal, Alex Shypula, Bailin Wang, Derry Wijaya, Jie Chen, and Yoon Kim. Explain-then-translate: An analysis on improving program translation with self-generated explanations. *CoRR*, abs/2311.07070, 2023b. doi: 10.48550/ARXIV.2311.07070. URL https://doi.org/10.48550/arXiv.2311.07070.

Shimin Tao, Weibin Meng, Yimeng Chen, Yichen Zhu, Ying Liu, Chunning Du, Tao Han, Yongpeng Zhao, Xiangguang Wang, and Hao Yang. Logstamp: Automatic online log parsing based on sequence labelling. *SIGMETRICS Perform. Evaluation Rev.*, 49(4):93–98, 2022. doi: 10.1145/3543146.3543168. URL https://doi.org/10.1145/3543146.3543168.

Wei Tao, Yanlin Wang, Ensheng Shi, Lun Du, Shi Han, Hongyu Zhang, Dongmei Zhang, and Wenqiang Zhang. On the evaluation of commit message generation models: An experimental study. In *IEEE International Conference on Software Maintenance and Evolution, ICSME 2021, Luxembourg, September*

27 - October 1, 2021, pp. 126–136. IEEE, 2021. doi: 10.1109/ICSME52107.2021.00018. URL https://doi.org/10.1109/ICSME52107.2021.00018.

Wei Tao, Yucheng Zhou, Wenqiang Zhang, and Yu Cheng. MAGIS: llm-based multi-agent framework for github issue resolution. *CoRR*, abs/2403.17927, 2024. doi: 10.48550/ARXIV.2403.17927. URL https://doi.org/10.48550/arXiv.2403.17927.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Comput. Surv.*, 55(6):109:1–109:28, 2023a. doi: 10.1145/3530811. URL https://doi.org/10.1145/3530811.

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. UL2: unifying language learning paradigms. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023b. URL https://openreview.net/pdf?id=6ruVLB727MC.

CodeGemma Team. Codegemma: Open code models based on gemma, 2024. URL https://storage.googleapis.com/deepmind-media/gemma/codegemma_report.pdf.

Shailja Thakur, Baleegh Ahmad, Zhenxing Fan, Hammond Pearce, Benjamin Tan, Ramesh Karri, Brendan Dolan-Gavitt, and Siddharth Garg. Benchmarking large language models for automated verilog RTL code generation. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2023, Antwerp, Belgium, April 17-19, 2023*, pp. 1–6. IEEE, 2023a. doi: 10.23919/DATE56975.2023.10137086. URL https://doi.org/10.23919/DATE56975.2023.10137086.

Shailja Thakur, Baleegh Ahmad, Hammond Pearce, Benjamin Tan, Brendan Dolan-Gavitt, Ramesh Karri, and Siddharth Garg. Verigen: A large language model for verilog code generation. *CoRR*, abs/2308.00708, 2023b. doi: 10.48550/ARXIV.2308.00708. URL https://doi.org/10.48550/arXiv.2308.00708.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Kathleen S. Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed H. Chi, and Quoc Le. Lamda: Language models for dialog applications. *CoRR*, abs/2201.08239, 2022. URL https://arxiv.org/abs/2201.08239.

Kiran Thorat, Jiahui Zhao, Yaotian Liu, Hongwu Peng, Xi Xie, Bin Lei, Jeff Zhang, and Caiwen Ding. Advanced large language model (llm)-driven verilog development: Enhancing power, performance, and area optimization in code synthesis. *CoRR*, abs/2312.01022, 2023. doi: 10.48550/ARXIV.2312.01022. URL https://doi.org/10.48550/arXiv.2312.01022.

Runchu Tian, Yining Ye, Yujia Qin, Xin Cong, Yankai Lin, Yinxu Pan, Yesai Wu, Zhiyuan Liu, and Maosong Sun. Debugbench: Evaluating debugging capability of large language models. *CoRR*, abs/2401.04621, 2024. doi: 10.48550/ARXIV.2401.04621. URL https://doi.org/10.48550/arXiv.2401.04621.

Saurabh Tiwari, Deepti Ameta, and Asim Banerjee. An approach to identify use case scenarios from textual requirements specification. In Ravindra Naik, Santonu Sarkar, Thomas T. Hildebrandt, Atul Kumar, and Richa Sharma (eds.), *Proceedings of the 12th Innovations on Software Engineering Conference (formerly known as India Software Engineering Conference), ISEC 2019, Pune, India, February 14-16, 2019*, pp. 5:1–5:11. ACM, 2019. doi: 10.1145/3299771.3299774. URL https://doi.org/10.1145/3299771.3299774.

David A. Tomassi, Naji Dmeiri, Yichen Wang, Antara Bhowmick, Yen-Chuan Liu, Premkumar T. Devanbu, Bogdan Vasilescu, and Cindy Rubio-González. Bugswarm: mining and continuously growing a dataset of reproducible failures and fixes. In Joanne M. Atlee, Tevfik Bultan, and Jon Whittle (eds.), *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, pp. 339–349. IEEE / ACM, 2019. doi: 10.1109/ICSE.2019.00048. URL https://doi.org/10.1109/ICSE.2019.00048.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023a. doi: 10.48550/arXiv.2302.13971. URL https://doi.org/10.48550/arXiv.2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023b. doi: 10.48550/arXiv.2307.09288. URL https://doi.org/10.48550/arXiv.2307.09288.

Hieu Tran, Ngoc M. Tran, Son Nguyen, Hoan Nguyen, and Tien N. Nguyen. Recovering variable names for minified code with usage contexts. In Joanne M. Atlee, Tevfik Bultan, and Jon Whittle (eds.), *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, pp. 1165–1175. IEEE / ACM, 2019. doi: 10.1109/ICSE.2019.00119. URL https://doi.org/10.1109/ICSE.2019.00119.

Immanuel Trummer. Codexdb: Synthesizing code for query processing from natural language instructions using GPT-3 codex. *Proc. VLDB Endow.*, 15(11):2921–2928, 2022. URL https://www.vldb.org/pvldb/vol15/p2921-trummer.pdf.

Yun-Da Tsai, Mingjie Liu, and Haoxing Ren. Rtlfixer: Automatically fixing RTL syntax errors with large language models. *CoRR*, abs/2311.16543, 2023. doi: 10.48550/ARXIV.2311.16543. URL https://doi.org/10.48550/arXiv.2311.16543.

Chen Tsfaty and Michael Fire. Malicious source code detection using transformer. *CoRR*, abs/2209.07957, 2022. doi: 10.48550/ARXIV.2209.07957. URL https://doi.org/10.48550/arXiv.2209.07957.

Zhaopeng Tu, Zhendong Su, and Premkumar T. Devanbu. On the localness of software. In Shing-Chi Cheung, Alessandro Orso, and Margaret-Anne D. Storey (eds.), *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, (FSE-22), Hong Kong, China, November 16 - 22, 2014*, pp. 269–280. ACM, 2014. doi: 10.1145/2635868.2635875. URL https://doi.org/10.1145/2635868.2635875.

Michele Tufano, Jevgenija Pantiuchina, Cody Watson, Gabriele Bavota, and Denys Poshyvanyk. On learning meaningful code changes via neural machine translation. In Joanne M. Atlee, Tevfik Bultan, and Jon Whittle (eds.), *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, pp. 25–36. IEEE / ACM, 2019a. doi: 10.1109/ICSE.2019.00021. URL https://doi.org/10.1109/ICSE.2019.00021.

Michele Tufano, Cody Watson, Gabriele Bavota, Massimiliano Di Penta, Martin White, and Denys Poshyvanyk. An empirical study on learning bug-fixing patches in the wild via neural machine translation. *ACM Trans. Softw. Eng. Methodol.*, 28(4):19:1–19:29, 2019b. doi: 10.1145/3340544. URL https://doi.org/10.1145/3340544.

Michele Tufano, Cody Watson, Gabriele Bavota, Massimiliano Di Penta, Martin White, and Denys Poshyvanyk. Learning how to mutate source code from bug-fixes. In *2019 IEEE International Conference on Software Maintenance and Evolution, ICSME 2019, Cleveland, OH, USA, September 29 - October 4, 2019*, pp. 301–312. IEEE, 2019c. doi: 10.1109/ICSME.2019.00046. URL https://doi.org/10.1109/ICSME.2019.00046.

Michele Tufano, Jason Kimko, Shiya Wang, Cody Watson, Gabriele Bavota, Massimiliano Di Penta, and Denys Poshyvanyk. Deepmutation: a neural mutation tool. In Gregg Rothermel and Doo-Hwan Bae (eds.), *ICSE '20: 42nd International Conference on Software Engineering, Companion Volume, Seoul, South Korea, 27 June - 19 July, 2020*, pp. 29–32. ACM, 2020. doi: 10.1145/3377812.3382146. URL https://doi.org/10.1145/3377812.3382146.

Michele Tufano, Dawn Drain, Alexey Svyatkovskiy, Shao Kun Deng, and Neel Sundaresan. Unit test case generation with transformers and focal context, 2021a.

Michele Tufano, Dawn Drain, Alexey Svyatkovskiy, and Neel Sundaresan. Generating accurate assert statements for unit test cases using pretrained transformers. In *IEEE/ACM International Conference on Automation of Software Test, AST@ICSE 2022, Pittsburgh, PA, USA, May 21-22, 2022*, pp. 54–64. ACM/IEEE, 2022a. doi: 10.1145/3524481.3527220. URL https://doi.org/10.1145/3524481.3527220.

Rosalia Tufano, Luca Pascarella, Michele Tufano, Denys Poshyvanyk, and Gabriele Bavota. Towards automating code review activities. In *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*, pp. 163–174. IEEE, 2021b. doi: 10.1109/ICSE43902.2021.00027. URL https://doi.org/10.1109/ICSE43902.2021.00027.

Rosalia Tufano, Simone Masiero, Antonio Mastropaolo, Luca Pascarella, Denys Poshyvanyk, and Gabriele Bavota. Using pre-trained models to boost code review automation. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pp. 2291–2302. ACM, 2022b. doi: 10.1145/3510003.3510621. URL https://doi.org/10.1145/3510003.3510621.

Lewis Tunstall, Leandro von Werra, and Thomas Wolf. *Natural Language Processing with Transformers: Building Language Applications with Hugging Face*. O'Reilly Media, Incorporated, 2022. ISBN 1098103246. URL https://books.google.ch/books?id=7hhyzgEACAAJ.

Saad Ullah, Mingji Han, Saurabh Pujar, Hammond Pearce, Ayse K. Coskun, and Gianluca Stringhini. Can large language models identify and reason about security vulnerabilities? not yet. *CoRR*, abs/2312.12575, 2023. doi: 10.48550/ARXIV.2312.12575. URL https://doi.org/10.48550/arXiv.2312.12575.

Tim van Dam, Frank van der Heijden, Philippe de Bekker, Berend Nieuwschepen, Marc Otten, and Maliheh Izadi. Investigating the performance of language models for completing code in functional programming languages: a haskell case study. *CoRR*, abs/2403.15185, 2024. doi: 10.48550/ARXIV.2403.15185. URL https://doi.org/10.48550/arXiv.2403.15185.

Marko Vasic, Aditya Kanade, Petros Maniatis, David Bieber, and Rishabh Singh. Neural program repair by jointly learning to localize and repair. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=ByloJ20qtm.

Bogdan Vasilescu, Casey Casalnuovo, and Premkumar T. Devanbu. Recovering clear, natural identifiers from obfuscated JS names. In Eric Bodden, Wilhelm Schäfer, Arie van Deursen, and Andrea Zisman (eds.), *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, Paderborn, Germany, September 4-8, 2017*, pp. 683–693. ACM, 2017. doi: 10.1145/3106237.3106289. URL https://doi.org/10.1145/3106237.3106289.

Mario Linares Vásquez, Luis Fernando Cortes-Coy, Jairo Aponte, and Denys Poshyvanyk. Changescribe: A tool for automatically generating commit messages. In Antonia Bertolino, Gerardo Canfora, and Sebastian G. Elbaum (eds.), *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 2*, pp. 709–712. IEEE Computer Society, 2015. doi: 10.1109/ICSE.2015.229. URL https://doi.org/10.1109/ICSE.2015.229.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Vasudev Vikram, Caroline Lemieux, and Rohan Padhye. Can large language models write good property-based tests? *CoRR*, abs/2307.04346, 2023. doi: 10.48550/ARXIV.2307.04346. URL https://doi.org/10.48550/arXiv.2307.04346.

Johannes Villmow, Jonas Depoix, and Adrian Ulges. ConTest: A unit test completion benchmark featuring context. In Royi Lachmy, Ziyu Yao, Greg Durrett, Milos Gligoric, Junyi Jessy Li, Ray Mooney, Graham Neubig, Yu Su, Huan Sun, and Reut Tsarfaty (eds.), *Proceedings of the 1st Workshop on Natural Language Processing for Programming (NLP4Prog 2021)*, pp. 17–25, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.nlp4prog-1.2. URL https://aclanthology.org/2021.nlp4prog-1.2.

Yao Wan, Zhou Zhao, Min Yang, Guandong Xu, Haochao Ying, Jian Wu, and Philip S. Yu. Improving automatic source code summarization via deep reinforcement learning. In Marianne Huchard, Christian Kästner, and Gordon Fraser (eds.), *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, Montpellier, France, September 3-7, 2018*, pp. 397–407. ACM, 2018. doi: 10.1145/3238147.3238206. URL https://doi.org/10.1145/3238147.3238206.

Yao Wan, Guanghua Wan, Shijie Zhang, Hongyu Zhang, Yulei Sui, Pan Zhou, Hai Jin, and Lichao Sun. Does your neural code completion model use my code? a membership inference approach. *CoRR*, abs/2404.14296, 2024. doi: 10.48550/ARXIV.2404.14296. URL https://doi.org/10.48550/arXiv.2404.14296.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Tal Linzen, Grzegorz Chrupala, and Afra Alishahi (eds.), *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018*, pp. 353–355. Association for Computational Linguistics, 2018a. doi: 10.18653/v1/w18-5446. URL https://doi.org/10.18653/v1/w18-5446.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 3261–3275, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/4496bf24afe7fab6f046bf4923da8de6-Abstract.html.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. RAT-SQL: relation-aware schema encoding and linking for text-to-sql parsers. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 7567–7578. Association for Computational Linguistics, 2020a. doi: 10.18653/v1/2020.acl-main.677. URL https://doi.org/10.18653/v1/2020.acl-main.677.

Bailin Wang, Wenpeng Yin, Xi Victoria Lin, and Caiming Xiong. Learning to synthesize data for semantic parsing. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pp. 2760–2766. Association for Computational Linguistics, 2021a. doi: 10.18653/V1/2021.NAACL-MAIN.220. URL https://doi.org/10.18653/v1/2021.naacl-main.220.

Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. Screen2words: Automatic mobile UI summarization with multimodal learning. In Jeffrey Nichols, Ranjitha Kumar, and Michael Nebeling (eds.), *UIST '21: The 34th Annual ACM Symposium on User Interface Software and Technology, Virtual Event, USA, October 10-14, 2021*, pp. 498–510. ACM, 2021b. doi: 10.1145/3472749.3474765. URL https://doi.org/10.1145/3472749.3474765.

Bryan Wang, Gang Li, and Yang Li. Enabling conversational interaction with mobile UI using large language models. In Albrecht Schmidt, Kaisa Väänänen, Tesh Goyal, Per Ola Kristensson, Anicia Peters, Stefanie Mueller, Julie R. Williamson, and Max L. Wilson (eds.), *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, CHI 2023, Hamburg, Germany, April 23-28, 2023*, pp. 432:1–432:17. ACM, 2023a. doi: 10.1145/3544548.3580895. URL https://doi.org/10.1145/3544548.3580895.

Chunhui Wang, Fabrizio Pastore, and Lionel C. Briand. Automated generation of constraints from use case specifications to support system testing. In *11th IEEE International Conference on Software Testing, Verification and Validation, ICST 2018, Västerås, Sweden, April 9-13, 2018*, pp. 23–33. IEEE Computer Society, 2018b. doi: 10.1109/ICST.2018.00013. URL https://doi.ieeecomputersociety.org/10.1109/ICST.2018.00013.

Deze Wang, Zhouyang Jia, Shanshan Li, Yue Yu, Yun Xiong, Wei Dong, and Xiangke Liao. Bridging pre-trained models and downstream tasks for source code understanding. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pp. 287–298. ACM, 2022a. doi: 10.1145/3510003.3510062. URL https://doi.org/10.1145/3510003.3510062.

Haoye Wang, Xin Xia, David Lo, Qiang He, Xinyu Wang, and John Grundy. Context-aware retrieval-based deep commit message generation. *ACM Trans. Softw. Eng. Methodol.*, 30(4):56:1–56:30, 2021c. doi: 10.1145/3464689. URL https://doi.org/10.1145/3464689.

Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, and Qing Wang. Software testing with large language models: Survey, landscape, and vision. *CoRR*, abs/2307.07221, 2023b. doi: 10.48550/ARXIV.2307.07221. URL https://doi.org/10.48550/arXiv.2307.07221.

Ke Wang and Zhendong Su. Blended, precise semantic program embeddings. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2020, pp. 121–134, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450376136. doi: 10.1145/3385412.3385999. URL https://doi.org/10.1145/3385412.3385999.

Liran Wang, Xunzhu Tang, Yichen He, Changyu Ren, Shuhua Shi, Chaoran Yan, and Zhoujun Li. Delving into commit-issue correlation to enhance commit message generation models. In *38th IEEE/ACM International Conference on Automated Software Engineering, ASE 2023, Luxembourg, September 11-15, 2023*, pp. 710–722. IEEE, 2023c. doi: 10.1109/ASE56229.2023.00050. URL https://doi.org/10.1109/ASE56229.2023.00050.

Pengcheng Wang, Jeffrey Svajlenko, Yanzhao Wu, Yun Xu, and Chanchal K. Roy. Ccaligner: a token based large-gap clone detector. In Michel Chaudron, Ivica Crnkovic, Marsha Chechik, and Mark Harman (eds.), *Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, pp. 1066–1077. ACM, 2018c. doi: 10.1145/3180155.3180179. URL https://doi.org/10.1145/3180155.3180179.

Ping Wang, Tian Shi, and Chandan K. Reddy. Text-to-sql generation for question answering on electronic medical records. In Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen (eds.), *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pp. 350–361. ACM / IW3C2, 2020b. doi: 10.1145/3366423.3380120. URL https://doi.org/10.1145/3366423.3380120.

Qiaosi Wang, Michael Madaio, Shaun K. Kane, Shivani Kapania, Michael Terry, and Lauren Wilcox. Designing responsible AI: adaptations of UX practice to meet responsible AI challenges. In Albrecht Schmidt, Kaisa Väänänen, Tesh Goyal, Per Ola Kristensson, Anicia Peters, Stefanie Mueller, Julie R. Williamson, and Max L. Wilson (eds.), *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, CHI 2023, Hamburg, Germany, April 23-28, 2023*, pp. 249:1–249:16. ACM, 2023d. doi: 10.1145/3544548.3581278. URL https://doi.org/10.1145/3544548.3581278.

Qifan Wang, Yi Fang, Anirudh Ravula, Fuli Feng, Xiaojun Quan, and Dongfang Liu. Webformer: The web-page transformer for structure information extraction. In Frédérique Laforest, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman, and Lionel Médini (eds.), *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pp. 3124–3133. ACM, 2022b. doi: 10.1145/3485447.3512032. URL https://doi.org/10.1145/3485447.3512032.

Shangwen Wang, Ming Wen, Bo Lin, Yepang Liu, Tegawendé F. Bissyandé, and Xiaoguang Mao. Pre-implementation method name prediction for object-oriented programming. *ACM Trans. Softw. Eng. Methodol.*, 32(6), sep 2023e. ISSN 1049-331X. doi: 10.1145/3597203. URL https://doi.org/10.1145/3597203.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *CoRR*, abs/2006.04768, 2020c. URL https://arxiv.org/abs/2006.04768.

Song Wang, Devin Chollak, Dana Movshovitz-Attias, and Lin Tan. Bugram: bug detection with n-gram language models. In David Lo, Sven Apel, and Sarfraz Khurshid (eds.), *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, ASE 2016, Singapore, September 3-7, 2016*, pp. 708–719. ACM, 2016a. doi: 10.1145/2970276.2970341. URL https://doi.org/10.1145/2970276.2970341.

Song Wang, Taiyue Liu, and Lin Tan. Automatically learning semantic features for defect prediction. In Laura K. Dillon, Willem Visser, and Laurie A. Williams (eds.), *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016*, pp. 297–308. ACM, 2016b. doi: 10.1145/2884781.2884804. URL https://doi.org/10.1145/2884781.2884804.

Thomas Wang, Adam Roberts, Daniel Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, Julien Launay, and Colin Raffel. What language model architecture and pretraining objective works best for zero-shot generalization? In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 22964–22984. PMLR, 2022c. URL https://proceedings.mlr.press/v162/wang22u.html.

Wenhan Wang, Ge Li, Bo Ma, Xin Xia, and Zhi Jin. Detecting code clones with graph neural network and flow-augmented abstract syntax tree. In Kostas Kontogiannis, Foutse Khomh, Alexander Chatzigeorgiou, Marios-Eleftherios Fokaefs, and Minghui Zhou (eds.), *27th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2020, London, ON, Canada, February 18-21, 2020*, pp. 261–271. IEEE, 2020d. doi: 10.1109/SANER48275.2020.9054857. URL https://doi.org/10.1109/SANER48275.2020.9054857.

Wenhua Wang, Yuqun Zhang, Zhengran Zeng, and Guandong Xu. Transˆ3: A transformer-based framework for unifying code summarization and code search. *CoRR*, abs/2003.03238, 2020e. URL https://arxiv.org/abs/2003.03238.

Xin Wang, Yasheng Wang, Fei Mi, Pingyi Zhou, Yao Wan, Xiao Liu, Li Li, Hao Wu, Jin Liu, and Xin Jiang. Syncobert: Syntax-guided multi-modal contrastive pre-training for code representation, 2021d.

Xin Wang, Yasheng Wang, Yao Wan, Fei Mi, Yitong Li, Pingyi Zhou, Jin Liu, Hao Wu, Xin Jiang, and Qun Liu. Compilable neural code generation with compiler feedback. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 9–19. Association for Computational Linguistics, 2022d. doi: 10.18653/V1/2022.FINDINGS-ACL.2. URL https://doi.org/10.18653/v1/2022.findings-acl.2.

Xin Wang, Yasheng Wang, Yao Wan, Jiawei Wang, Pingyi Zhou, Li Li, Hao Wu, and Jin Liu. CODE-MVP: learning to represent source code from multiple views with contrastive pre-training. In Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz (eds.), *Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pp. 1066–1077. Association for Computational Linguistics, 2022e. doi: 10.18653/v1/2022.findings-naacl.80. URL https://doi.org/10.18653/v1/2022.findings-naacl.80.

Xingyao Wang, Hao Peng, Reyhaneh Jabbarvand, and Heng Ji. Leti: Learning to generate from textual interactions. *CoRR*, abs/2305.10314, 2023f. doi: 10.48550/ARXIV.2305.10314. URL https://doi.org/10.48550/arXiv.2305.10314.

Xuheng Wang, Xu Zhang, Liqun Li, Shilin He, Hongyu Zhang, Yudong Liu, Lingling Zheng, Yu Kang, Qingwei Lin, Yingnong Dang, Saravanakumar Rajmohan, and Dongmei Zhang. SPINE: a scalable log parser with feedback guidance. In Abhik Roychoudhury, Cristian Cadar, and Miryung Kim (eds.), *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022, Singapore, Singapore, November 14-18, 2022*, pp. 1198–1208. ACM, 2022f. doi: 10.1145/3540250.3549176. URL https://doi.org/10.1145/3540250.3549176.

Yanlin Wang, Ensheng Shi, Lun Du, Xiaodi Yang, Yuxuan Hu, Shi Han, Hongyu Zhang, and Dongmei Zhang. Cocosum: Contextual code summarization with multi-relational graph neural network. *CoRR*, abs/2107.01933, 2021e. URL https://arxiv.org/abs/2107.01933.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. Super-naturalinstructions: Generalization via declarative instructions on 1600+ NLP tasks. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 5085–5109. Association for Computational Linguistics, 2022g. URL https://aclanthology.org/2022.emnlp-main.340.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 13484–13508. Association for Computational Linguistics, 2023g. doi: 10.18653/v1/2023.acl-long.754. URL https://doi.org/10.18653/v1/2023.acl-long.754.

Yue Wang, Weishi Wang, Shafiq R. Joty, and Steven C. H. Hoi. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pp. 8696–8708. Association for Computational Linguistics, 2021f. doi: 10.18653/v1/2021.emnlp-main.685. URL https://doi.org/10.18653/v1/2021.emnlp-main.685.

Yue Wang, Hung Le, Akhilesh Deepak Gotmare, Nghi D. Q. Bui, Junnan Li, and Steven C. H. Hoi. Codet5+: Open code large language models for code understanding and generation. *CoRR*, abs/2305.07922, 2023h. doi: 10.48550/arXiv.2305.07922. URL https://doi.org/10.48550/arXiv.2305.07922.

Zan Wang, Ming Yan, Junjie Chen, Shuang Liu, and Dongdi Zhang. Deep learning library testing via effective model generation. In Prem Devanbu, Myra B. Cohen, and Thomas Zimmermann (eds.), *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, pp. 788–799. ACM, 2020f. doi: 10.1145/3368089.3409761. URL https://doi.org/10.1145/3368089.3409761.

Zheng Wang and Michael F. P. O'Boyle. Machine learning in compiler optimisation. *CoRR*, abs/1805.03441, 2018. URL http://arxiv.org/abs/1805.03441.

Zhiruo Wang, Shuyan Zhou, Daniel Fried, and Graham Neubig. Execution-based evaluation for open-domain code generation. *CoRR*, abs/2212.10481, 2022h. doi: 10.48550/ARXIV.2212.10481. URL https://doi.org/10.48550/arXiv.2212.10481.

Zhiruo Wang, Grace Cuenca, Shuyan Zhou, Frank F. Xu, and Graham Neubig. Mconala: A benchmark for code generation from multiple natural languages. In Andreas Vlachos and Isabelle Augenstein (eds.), *Findings of the Association for Computational Linguistics: EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*, pp. 265–273. Association for Computational Linguistics, 2023i. doi: 10.18653/V1/2023.FINDINGS-EACL.20. URL https://doi.org/10.18653/v1/2023.findings-eacl.20.

Cody Watson, Michele Tufano, Kevin Moran, Gabriele Bavota, and Denys Poshyvanyk. On learning meaningful assert statements for unit test cases. In Gregg Rothermel and Doo-Hwan Bae (eds.), *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, pp. 1398–1409. ACM, 2020. doi: 10.1145/3377811.3380429. URL https://doi.org/10.1145/3377811.3380429.

Anjiang Wei, Yinlin Deng, Chenyuan Yang, and Lingming Zhang. Free lunch for testing: Fuzzing deep-learning libraries from open source. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pp. 995–1007. ACM, 2022a. doi: 10.1145/3510003.3510041. URL https://doi.org/10.1145/3510003.3510041.

Bolin Wei, Ge Li, Xin Xia, Zhiyi Fu, and Zhi Jin. Code generation as a dual task of code summarization. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 6559–6569, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/e52ad5c9f751f599492b4f087ed7ecfc-Abstract.html.

Huihui Wei and Ming Li. Supervised deep features for software functional clone detection by exploiting lexical and syntactical information in source code. In Carles Sierra (ed.), *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pp. 3034–3040. ijcai.org, 2017. doi: 10.24963/ijcai.2017/423. URL https://doi.org/10.24963/ijcai.2017/423.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022b. URL https://openreview.net/forum?id=gEZrGCozdqR.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022c. ISSN 2835-8856. URL https://openreview.net/forum?id=yzkSU5zdwD. Survey Certification.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022d. URL http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Jiayi Wei, Maruth Goyal, Greg Durrett, and Isil Dillig. Lambdanet: Probabilistic type inference using graph neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=Hkx6hANtwH.

Jiayi Wei, Greg Durrett, and Isil Dillig. Typet5: Seq2seq type inference using static analysis. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=4TyNEhI2GdN.

Yuanbo Wen, Qi Guo, Qiang Fu, Xiaqing Li, Jianxing Xu, Yanlin Tang, Yongwei Zhao, Xing Hu, Zidong Du, Ling Li, Chao Wang, Xuehai Zhou, and Yunji Chen. Babeltower: Learning to auto-parallelized program translation. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23685–23700. PMLR, 2022. URL https://proceedings.mlr.press/v162/wen22b.html.

Martin Weyssow, Houari A. Sahraoui, and Eugene Syriani. Recommending metamodel concepts during modeling activities with pre-trained language models. *Softw. Syst. Model.*, 21(3):1071–1089, 2022. doi: 10.1007/S10270-022-00975-5. URL https://doi.org/10.1007/s10270-022-00975-5.

Jules White, Sam Hays, Quchen Fu, Jesse Spencer-Smith, and Douglas C. Schmidt. Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. *CoRR*, abs/2303.07839, 2023. doi: 10.48550/ARXIV.2303.07839. URL https://doi.org/10.48550/arXiv.2303.07839.

Martin White, Christopher Vendome, Mario Linares Vásquez, and Denys Poshyvanyk. Toward deep learning software repositories. In Massimiliano Di Penta, Martin Pinzger, and Romain Robbes (eds.), *12th IEEE/ACM Working Conference on Mining Software Repositories, MSR 2015, Florence, Italy, May 16-17, 2015*, pp. 334–345. IEEE Computer Society, 2015. doi: 10.1109/MSR.2015.38. URL https://doi.org/10.1109/MSR.2015.38.

Martin White, Michele Tufano, Christopher Vendome, and Denys Poshyvanyk. Deep learning code fragments for code clone detection. In David Lo, Sven Apel, and Sarfraz Khurshid (eds.), *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, ASE 2016, Singapore, September 3-7, 2016*, pp. 87–98. ACM, 2016. doi: 10.1145/2970276.2970326. URL https://doi.org/10.1145/2970276.2970326.

Ratnadira Widyasari, Sheng Qin Sim, Camellia Lok, Haodi Qi, Jack Phan, Qijin Tay, Constance Tan, Fiona Wee, Jodie Ethelda Tan, Yuheng Yieh, Brian Goh, Ferdian Thung, Hong Jin Kang, Thong Hoang, David Lo, and Eng Lieh Ouh. Bugsinpy: a database of existing bugs in python programs to enable controlled testing and debugging studies. In Prem Devanbu, Myra B. Cohen, and Thomas Zimmermann (eds.), *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, pp. 1556–1560. ACM, 2020. doi: 10.1145/3368089.3417943. URL https://doi.org/10.1145/3368089.3417943.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256, 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Edmund Wong, Jinqiu Yang, and Lin Tan. Autocomment: Mining question and answer sites for automatic comment generation. In Ewen Denney, Tevfik Bultan, and Andreas Zeller (eds.), *2013 28th IEEE/ACM International Conference on Automated Software Engineering, ASE 2013, Silicon Valley, CA, USA, November 11-15, 2013*, pp. 562–567. IEEE, 2013. doi: 10.1109/ASE.2013.6693113. URL https://doi.org/10.1109/ASE.2013.6693113.

Edmund Wong, Taiyue Liu, and Lin Tan. Clocom: Mining existing source code for automatic comment generation. In Yann-Gaël Guéhéneuc, Bram Adams, and Alexander Serebrenik (eds.), *22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering, SANER 2015, Montreal, QC, Canada, March 2-6, 2015*, pp. 380–389. IEEE Computer Society, 2015. doi: 10.1109/SANER.2015.7081848. URL https://doi.org/10.1109/SANER.2015.7081848.

Wai Kin Wong, Huaijin Wang, Zongjie Li, Zhibo Liu, Shuai Wang, Qiyi Tang, Sen Nie, and Shi Wu. Refining decompiled C code with large language models. *CoRR*, abs/2310.06530, 2023. doi: 10.48550/ARXIV.2310.06530. URL https://doi.org/10.48550/arXiv.2310.06530.

Chengyue Wu, Yixiao Ge, Qiushan Guo, Jiahao Wang, Zhixuan Liang, Zeyu Lu, Ying Shan, and Ping Luo. Plot2code: A comprehensive benchmark for evaluating multi-modal large language models in code generation from scientific plots. 2024a. URL https://doi.org/10.48550/arXiv.2405.07990.

Di Wu, Wasi Uddin Ahmad, Dejiao Zhang, Murali Krishna Ramanathan, and Xiaofei Ma. Repoformer: Selective retrieval for repository-level code completion. *CoRR*, abs/2403.10059, 2024b. doi: 10.48550/ARXIV.2403.10059. URL https://doi.org/10.48550/arXiv.2403.10059.

Hongqiu Wu, Hai Zhao, and Min Zhang. Code summarization with structure-induced transformer. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pp. 1078–1090. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.findings-acl.93. URL https://doi.org/10.18653/v1/2021.findings-acl.93.

Jie JW Wu and Fatemeh H Fard. Benchmarking the communication competence of code generation for llms and llm agent. *arXiv preprint arXiv:2406.00215*, 2024. URL https://doi.org/10.48550/arXiv.2406.00215.

Ming Wu, Pengcheng Wang, Kangqi Yin, Haoyu Cheng, Yun Xu, and Chanchal K. Roy. Lvmapper: A large-variance clone detector using sequencing alignment approach. *IEEE Access*, 8:27986–27997, 2020. doi: 10.1109/ACCESS.2020.2971545. URL https://doi.org/10.1109/ACCESS.2020.2971545.

Mingyuan Wu, Ling Jiang, Jiahong Xiang, Yuqun Zhang, Guowei Yang, Huixin Ma, Sen Nie, Shi Wu, Heming Cui, and Lingming Zhang. Evaluating and improving neural program-smoothing-based fuzzing. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pp. 847–858. ACM, 2022. doi: 10.1145/3510003.3510089. URL https://doi.org/10.1145/3510003.3510089.

Chunqiu Steven Xia and Lingming Zhang. Less training, more repairing please: revisiting automated program repair via zero-shot learning. In Abhik Roychoudhury, Cristian Cadar, and Miryung Kim (eds.), *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022, Singapore, Singapore, November 14-18, 2022*, pp. 959–971. ACM, 2022. doi: 10.1145/3540250.3549101. URL https://doi.org/10.1145/3540250.3549101.

Chunqiu Steven Xia and Lingming Zhang. Conversational automated program repair. *CoRR*, abs/2301.13246, 2023. doi: 10.48550/ARXIV.2301.13246. URL https://doi.org/10.48550/arXiv.2301.13246.

Chunqiu Steven Xia, Matteo Paltenghi, Jia Le Tian, Michael Pradel, and Lingming Zhang. Fuzz4all: Universal fuzzing with large language models. *CoRR*, abs/2308.04748, 2023a. doi: 10.48550/ARXIV.2308.04748. URL https://doi.org/10.48550/arXiv.2308.04748.

Chunqiu Steven Xia, Yuxiang Wei, and Lingming Zhang. Automated program repair in the era of large pre-trained language models. In *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*, pp. 1482–1494. IEEE, 2023b. doi: 10.1109/ICSE48619.2023.00129. URL https://doi.org/10.1109/ICSE48619.2023.00129.

Chenhao Xie, Wenhao Huang, Jiaqing Liang, Chengsong Huang, and Yanghua Xiao. Webke: Knowledge extraction from semi-structured web with pre-trained markup language model. In Gianluca Demartini, Guido Zuccon, J. Shane Culpepper, Zi Huang, and Hanghang Tong (eds.), *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, pp. 2211–2220. ACM, 2021a. doi: 10.1145/3459637.3482491. URL https://doi.org/10.1145/3459637.3482491.

Danning Xie, Yitong Li, Mijung Kim, Hung Viet Pham, Lin Tan, Xiangyu Zhang, and Michael W. Godfrey. Docter: documentation-guided fuzzing for testing deep learning API functions. In Sukyoung Ryu and Yannis Smaragdakis (eds.), *ISSTA '22: 31st ACM SIGSOFT International Symposium on Software Testing and Analysis, Virtual Event, South Korea, July 18 - 22, 2022*, pp. 176–188. ACM, 2022a. doi: 10.1145/3533767.3534220. URL https://doi.org/10.1145/3533767.3534220.

Rui Xie, Wei Ye, Jinan Sun, and Shikun Zhang. Exploiting method names to improve code summarization: A deliberation multi-task learning approach. In *29th IEEE/ACM International Conference on Program Comprehension, ICPC 2021, Madrid, Spain, May 20-21, 2021*, pp. 138–148. IEEE, 2021b. doi: 10.1109/ICPC52881.2021.00022. URL https://doi.org/10.1109/ICPC52881.2021.00022.

Rui Xie, Zhengran Zeng, Zhuohao Yu, Chang Gao, Shikun Zhang, and Wei Ye. Codeshell technical report. *CoRR*, abs/2403.15747, 2024. doi: 10.48550/ARXIV.2403.15747. URL https://doi.org/10.48550/arXiv.2403.15747.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 602–631. Association for Computational Linguistics, 2022b. doi: 10.18653/v1/2022.emnlp-main.39. URL https://doi.org/10.18653/v1/2022.emnlp-main.39.

Yutao Xie, Jiayi Lin, Hande Dong, Lei Zhang, and Zhonghai Wu. A survey of deep code search. *CoRR*, abs/2305.05959, 2023a. doi: 10.48550/arXiv.2305.05959. URL https://doi.org/10.48550/arXiv.2305.05959.

Zhuokui Xie, Yinghao Chen, Chen Zhi, Shuiguang Deng, and Jianwei Yin. Chatunitest: a chatgpt-based automated unit test generation tool. *CoRR*, abs/2305.04764, 2023b. doi: 10.48550/ARXIV.2305.04764. URL https://doi.org/10.48550/arXiv.2305.04764.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=CfXh93NDgH.

Frank F. Xu, Uri Alon, Graham Neubig, and Vincent Josua Hellendoorn. A systematic evaluation of large language models of code. In Swarat Chaudhuri and Charles Sutton (eds.), *MAPS@PLDI 2022: 6th ACM SIGPLAN International Symposium on Machine Programming, San Diego, CA, USA, 13 June 2022*, pp. 1–10. ACM, 2022. doi: 10.1145/3520312.3534862. URL https://doi.org/10.1145/3520312.3534862.

Ling Xu, Huanhuan Yang, Chao Liu, Jianhang Shuai, Meng Yan, Yan Lei, and Zhou Xu. Two-stage attention-based model for code search with textual and structural features. In *28th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2021, Honolulu, HI, USA, March 9-12, 2021*, pp. 342–353. IEEE, 2021a. doi: 10.1109/SANER50967.2021.00039. URL https://doi.org/10.1109/SANER50967.2021.00039.

Shengbin Xu, Yuan Yao, Feng Xu, Tianxiao Gu, Hanghang Tong, and Jian Lu. Commit message generation for source code changes. In Sarit Kraus (ed.), *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 3975–3981. ijcai.org, 2019a. doi: 10.24963/IJCAI.2019/552. URL https://doi.org/10.24963/ijcai.2019/552.

Sihan Xu, Sen Zhang, Weijing Wang, Xinya Cao, Chenkai Guo, and Jing Xu. Method name suggestion with hierarchical attention networks. In Manuel V. Hermenegildo and Atsushi Igarashi (eds.), *Proceedings of the 2019 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, PEPM@POPL 2019, Cascais, Portugal, January 14-15, 2019*, pp. 10–21. ACM, 2019b. doi: 10.1145/3294032.3294079. URL https://doi.org/10.1145/3294032.3294079.

Xiangzhe Xu, Zhuo Zhang, Shiwei Feng, Yapeng Ye, Zian Su, Nan Jiang, Siyuan Cheng, Lin Tan, and Xiangyu Zhang. Lmpa: Improving decompilation by synergy of large language model and program analysis. *CoRR*, abs/2306.02546, 2023. doi: 10.48550/ARXIV.2306.02546. URL https://doi.org/10.48550/arXiv.2306.02546.

Xiaojun Xu, Chang Liu, and Dawn Song. Sqlnet: Generating structured queries from natural language without reinforcement learning. *CoRR*, abs/1711.04436, 2017. URL http://arxiv.org/abs/1711.04436.

Yichen Xu and Yanqiao Zhu. A survey on pretrained language models for neural code intelligence. *CoRR*, abs/2212.10079, 2022. doi: 10.48550/arXiv.2212.10079. URL https://doi.org/10.48550/arXiv.2212.10079.

Zhaogui Xu, Xiangyu Zhang, Lin Chen, Kexin Pei, and Baowen Xu. Python probabilistic type inference with natural language support. In Thomas Zimmermann, Jane Cleland-Huang, and Zhendong Su (eds.), *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2016, Seattle, WA, USA, November 13-18, 2016*, pp. 607–618. ACM, 2016. doi: 10.1145/2950290.2950343. URL https://doi.org/10.1145/2950290.2950343.

Zhifeng Xu, Xianjin Fang, and Gaoming Yang. Malbert: A novel pre-training method for malware detection. *Comput. Secur.*, 111:102458, 2021b. doi: 10.1016/J.COSE.2021.102458. URL https://doi.org/10.1016/j.cose.2021.102458.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pp. 483–498. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.41. URL https://doi.org/10.18653/v1/2021.naacl-main.41.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. Sqlizer: query synthesis from natural language. *Proc. ACM Program. Lang.*, 1(OOPSLA):63:1–63:26, 2017. doi: 10.1145/3133887. URL https://doi.org/10.1145/3133887.

Mohammad A. Yahya and Dae-Kyoo Kim. Cross-language source code clone detection using deep learning with infercode. *CoRR*, abs/2205.04913, 2022. doi: 10.48550/arXiv.2205.04913. URL https://doi.org/10.48550/arXiv.2205.04913.

Shuhan Yan, Hang Yu, Yuting Chen, Beijun Shen, and Lingxiao Jiang. Are the code snippets what we are searching for? A benchmark and an empirical study on code search with natural-language queries. In Kostas Kontogiannis, Foutse Khomh, Alexander Chatzigeorgiou, Marios-Eleftherios Fokaefs, and Minghui Zhou (eds.), *27th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2020, London, ON, Canada, February 18-21, 2020*, pp. 344–354. IEEE, 2020. doi: 10.1109/SANER48275.2020.9054840. URL https://doi.org/10.1109/SANER48275.2020.9054840.

Weixiang Yan, Yuchen Tian, Yunzhe Li, Qian Chen, and Wen Wang. Codetransocean: A comprehensive multilingual benchmark for code translation. *CoRR*, abs/2310.04951, 2023. doi: 10.48550/ARXIV.2310.04951. URL https://doi.org/10.48550/arXiv.2310.04951.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, Fan Yang, Fei Deng, Feng Wang, Feng Liu, Guangwei Ai, Guosheng Dong, Haizhou Zhao, Hang Xu, Haoze Sun, Hongda Zhang, Hui Liu, Jiaming Ji, Jian Xie, Juntao Dai, Kun Fang, Lei Su, Liang Song, Lifeng Liu, Liyun Ru, Luyao Ma, Mang Wang, Mickel Liu, MingAn Lin, Nuolan Nie, Peidong Guo, Ruiyang Sun, Tao Zhang, Tianpeng Li, Tianyu Li, Wei Cheng, Weipeng Chen, Xiangrong Zeng, Xiaochuan Wang, Xiaoxi Chen, Xin Men, Xin Yu, Xuehai Pan, Yanjun Shen, Yiding Wang, Yiyu Li, Youxin Jiang, Yuchen Gao, Yupeng Zhang, Zenan Zhou, and Zhiying Wu. Baichuan 2: Open large-scale language models. *CoRR*, abs/2309.10305, 2023a. doi: 10.48550/arXiv.2309.10305. URL https://doi.org/10.48550/arXiv.2309.10305.

Chen Yang, Junjie Chen, Bin Lin, Jianyi Zhou, and Ziqi Wang. Enhancing llm-based test generation for hard-to-cover branches via program analysis. *CoRR*, abs/2404.04966, 2024a. doi: 10.48550/ARXIV.2404.04966. URL https://doi.org/10.48550/arXiv.2404.04966.

Chenyuan Yang, Yinlin Deng, Runyu Lu, Jiayi Yao, Jiawei Liu, Reyhaneh Jabbarvand, and Lingming Zhang. White-box compiler fuzzing empowered by large language models. *CoRR*, abs/2310.15991, 2023b. doi: 10.48550/ARXIV.2310.15991. URL https://doi.org/10.48550/arXiv.2310.15991.

Chenyuan Yang, Yinlin Deng, Jiayi Yao, Yuxing Tu, Hanchi Li, and Lingming Zhang. Fuzzing automatic differentiation in deep-learning libraries. In *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*, pp. 1174–1186. IEEE, 2023c. doi: 10.1109/ICSE48619.2023.00105. URL https://doi.org/10.1109/ICSE48619.2023.00105.

John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. Intercode: Standardizing and benchmarking interactive coding with execution feedback. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023d. URL http://papers.nips.cc/paper_files/paper/2023/hash/4b175d846fb008d540d233c188379ff9-Abstract-Datasets_and_Benchmarks.html.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *CoRR*, abs/2405.15793, 2024b. doi: 10.48550/ARXIV.2405.15793. URL https://doi.org/10.48550/arXiv.2405.15793.

Wei Yang, Peng Xu, and Yanshuai Cao. Hierarchical neural data synthesis for semantic parsing. *CoRR*, abs/2112.02212, 2021. URL https://arxiv.org/abs/2112.02212.

Xianjun Yang, Kexun Zhang, Haifeng Chen, Linda R. Petzold, William Yang Wang, and Wei Cheng. Zero-shot detection of machine-generated codes. *CoRR*, abs/2310.05103, 2023e. doi: 10.48550/ARXIV.2310.05103. URL https://doi.org/10.48550/arXiv.2310.05103.

Yanping Yang, Ling Xu, Meng Yan, Zhou Xu, and Zhongyang Deng. A naming pattern based approach for method name recommendation. In *IEEE 33rd International Symposium on Software Reliability Engineering, ISSRE 2022, Charlotte, NC, USA, October 31 - Nov. 3, 2022*, pp. 344–354. IEEE, 2022. doi: 10.1109/ISSRE55969.2022.00041. URL https://doi.org/10.1109/ISSRE55969.2022.00041.

Ziyu Yao, Daniel S. Weld, Wei-Peng Chen, and Huan Sun. Staqc: A systematically mined question-code dataset from stack overflow. In Pierre-Antoine Champin, Fabien Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis (eds.), *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pp. 1693–1703. ACM, 2018. doi: 10.1145/3178876.3186081. URL https://doi.org/10.1145/3178876.3186081.

Michihiro Yasunaga and Percy Liang. Graph-based, self-supervised program repair from diagnostic feedback. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10799–10808. PMLR, 2020. URL http://proceedings.mlr.press/v119/yasunaga20a.html.

Michihiro Yasunaga and Percy Liang. Break-it-fix-it: Unsupervised learning for program repair. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11941–11952. PMLR, 2021. URL http://proceedings.mlr.press/v139/yasunaga21a.html.

Ming-Ho Yee and Arjun Guha. Do machine learning models produce typescript types that type check? In Karim Ali and Guido Salvaneschi (eds.), *37th European Conference on Object-Oriented Programming, ECOOP 2023, July 17-21, 2023, Seattle, Washington, United States*, volume 263 of *LIPIcs*, pp. 37:1–37:28. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi: 10.4230/LIPIcs.ECOOP.2023.37. URL https://doi.org/10.4230/LIPIcs.ECOOP.2023.37.

Burak Yetistiren, Isik Özsoy, Miray Ayerdem, and Eray Tüzün. Evaluating the code quality of ai-assisted code generation tools: An empirical study on github copilot, amazon codewhisperer, and chatgpt. *CoRR*, abs/2304.10778, 2023. doi: 10.48550/ARXIV.2304.10778. URL https://doi.org/10.48550/arXiv.2304.10778.

Pengcheng Yin and Graham Neubig. A syntactic neural model for general-purpose code generation. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pp. 440–450. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1041. URL https://doi.org/10.18653/v1/P17-1041.

Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. Learning to mine aligned code and natural language pairs from stack overflow. In Andy Zaidman, Yasutaka Kamei, and Emily Hill (eds.), *Proceedings of the 15th International Conference on Mining Software Repositories, MSR 2018, Gothenburg, Sweden, May 28-29, 2018*, pp. 476–486. ACM, 2018. doi: 10.1145/3196398.3196408. URL https://doi.org/10.1145/3196398.3196408.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. Tabert: Pretraining for joint understanding of textual and tabular data. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 8413–8426. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.ACL-MAIN.745. URL https://doi.org/10.18653/v1/2020.acl-main.745.

Ying Yin, Yuhai Zhao, Yiming Sun, and Chen Chen. Automatic code review by learning the structure information of code graph. *Sensors*, 23(5):2551, 2023. doi: 10.3390/s23052551. URL https://doi.org/10.3390/s23052551.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, Kaidong Yu, Peng Liu, Qiang Liu, Shawn Yue, Senbin Yang, Shiming Yang, Tao Yu, Wen Xie, Wenhao Huang, Xiaohui Hu, Xiaoyi Ren, Xinyao Niu, Pengcheng Nie, Yuchi Xu, Yudong Liu, Yue Wang, Yuxuan Cai, Zhenyu Gu, Zhiyuan Liu, and Zonghong Dai. Yi: Open foundation models by 01.ai. *CoRR*, abs/2403.04652, 2024. doi: 10.48550/ARXIV.2403.04652. URL https://doi.org/10.48550/arXiv.2403.04652.

Hao Yu, Wing Lam, Long Chen, Ge Li, Tao Xie, and Qianxiang Wang. Neural detection of semantic code clones via tree-based convolution. In Yann-Gaël Guéhéneuc, Foutse Khomh, and Federica Sarro (eds.), *Proceedings of the 27th International Conference on Program Comprehension, ICPC 2019, Montreal, QC, Canada, May 25-31, 2019*, pp. 70–80. IEEE / ACM, 2019a. doi: 10.1109/ICPC.2019.00021. URL https://doi.org/10.1109/ICPC.2019.00021.

Hao Yu, Bo Shen, Dezhi Ran, Jiaxin Zhang, Qi Zhang, Yuchi Ma, Guangtai Liang, Ying Li, Tao Xie, and Qianxiang Wang. Codereval: A benchmark of pragmatic code generation with generative pre-trained models. *CoRR*, abs/2302.00288, 2023a. doi: 10.48550/arXiv.2302.00288. URL https://doi.org/10.48550/arXiv.2302.00288.

Siyu Yu, Ningjiang Chen, Yifan Wu, and Wensheng Dou. Self-supervised log parsing using semantic contribution difference. *J. Syst. Softw.*, 200:111646, 2023b. doi: 10.1016/J.JSS.2023.111646. URL https://doi.org/10.1016/j.jss.2023.111646.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir R. Radev. Typesql: Knowledge-based type-aware neural text-to-sql generation. In Marilyn A. Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pp. 588–594. Association for Computational Linguistics, 2018a. doi: 10.18653/v1/n18-2093. URL https://doi.org/10.18653/v1/n18-2093.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir R. Radev. Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 1653–1663. Association for Computational Linguistics, 2018b. doi: 10.18653/v1/d18-1193. URL https://doi.org/10.18653/v1/d18-1193.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 3911–3921. Association for Computational Linguistics, 2018c. doi: 10.18653/v1/d18-1425. URL https://doi.org/10.18653/v1/d18-1425.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander R. Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter S. Lasecki, and Dragomir R. Radev. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 1962–1979. Association for Computational Linguistics, 2019b. doi: 10.18653/v1/D19-1204. URL https://doi.org/10.18653/v1/D19-1204.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir R. Radev. Sparc: Cross-domain semantic parsing in context. In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 4511–4523. Association for Computational Linguistics, 2019c. doi: 10.18653/v1/p19-1443. URL https://doi.org/10.18653/v1/p19-1443.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir R. Radev, Richard Socher, and Caiming Xiong. Grappa: Grammar-augmented pre-training for table semantic parsing. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=kyaIeYj4zZ.

Xiaojing Yu, Tianlong Chen, Zhengjie Yu, Huiyu Li, Yang Yang, Xiaoqian Jiang, and Anxiao Jiang. Dataset and enhanced model for eligibility criteria-to-sql semantic parsing. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asunción Moreno, Jan Odijk, and Stelios Piperidis (eds.), *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020*, pp. 5829–5837. European Language Resources Association, 2020. URL https://aclanthology.org/2020.lrec-1.714/.

Zhaojian Yu, Xin Zhang, Ning Shang, Yangyu Huang, Can Xu, Yishujie Zhao, Wenxiang Hu, and Qiufeng Yin. Wavecoder: Widespread and versatile enhanced instruction tuning with refined data generation. *CoRR*, abs/2312.14187, 2023c. doi: 10.48550/ARXIV.2312.14187. URL https://doi.org/10.48550/arXiv.2312.14187.

Zhiqiang Yuan, Junwei Liu, Qiancheng Zi, Mingwei Liu, Xin Peng, and Yiling Lou. Evaluating instruction-tuned large language models on code comprehension and generation. *CoRR*, abs/2308.01240, 2023a. doi: 10.48550/arXiv.2308.01240. URL https://doi.org/10.48550/arXiv.2308.01240.

Zhiqiang Yuan, Yiling Lou, Mingwei Liu, Shiji Ding, Kaixin Wang, Yixuan Chen, and Xin Peng. No more manual tests? evaluating and improving chatgpt for unit test generation. *CoRR*, abs/2305.04207, 2023b. doi: 10.48550/ARXIV.2305.04207. URL https://doi.org/10.48550/arXiv.2305.04207.

Nusrat Zahan, Philipp Burckhardt, Mikola Lysenko, Feross Aboukhadijeh, and Laurie A. Williams. Shifting the lens: Detecting malware in npm ecosystem with large language models. *CoRR*, abs/2403.12196, 2024. doi: 10.48550/ARXIV.2403.12196. URL https://doi.org/10.48550/arXiv.2403.12196.

Daoguang Zan, Bei Chen, Dejian Yang, Zeqi Lin, Minsu Kim, Bei Guan, Yongji Wang, Weizhu Chen, and Jian-Guang Lou. CERT: continual pre-training on sketches for library-oriented code generation. In Luc De Raedt (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pp. 2369–2375. ijcai.org, 2022. doi: 10.24963/ijcai.2022/329. URL https://doi.org/10.24963/ijcai.2022/329.

Daoguang Zan, Bei Chen, Fengji Zhang, Dianjie Lu, Bingchao Wu, Bei Guan, Yongji Wang, and Jian-Guang Lou. Large language models meet nl2code: A survey. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 7443–7464. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.411. URL https://doi.org/10.18653/v1/2023.acl-long.411.

Daoguang Zan, Ailun Yu, Wei Liu, Dong Chen, Bo Shen, Wei Li, Yafen Yao, Yongshun Gong, Xiaolin Chen, Bei Guan, Zhiguang Yang, Yongji Wang, Qianxiang Wang, and Lizhen Cui. Codes: Natural language to code repository via multi-layer sketch. *CoRR*, abs/2403.16443, 2024. doi: 10.48550/ARXIV.2403.16443. URL https://doi.org/10.48550/arXiv.2403.16443.

John M. Zelle and Raymond J. Mooney. Learning to parse database queries using inductive logic programming. In William J. Clancey and Daniel S. Weld (eds.), *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, USA, August 4-8, 1996, Volume 2*, pp. 1050–1055. AAAI Press / The MIT Press, 1996. URL http://www.aaai.org/Library/AAAI/1996/aaai96-156.php.

Jichuan Zeng, Xi Victoria Lin, Steven C. H. Hoi, Richard Socher, Caiming Xiong, Michael R. Lyu, and Irwin King. Photon: A robust cross-domain text-to-sql system. In Asli Celikyilmaz and Tsung-Hsien Wen (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ACL 2020, Online, July 5-10, 2020*, pp. 204–214. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.ACL-DEMOS.24. URL https://doi.org/10.18653/v1/2020.acl-demos.24.

Chenyuan Zhang, Hao Liu, Jiutian Zeng, Kejing Yang, Yuhong Li, and Hui Li. Prompt-enhanced software vulnerability detection using chatgpt. *CoRR*, abs/2308.12697, 2023a. doi: 10.48550/ARXIV.2308.12697. URL https://doi.org/10.48550/arXiv.2308.12697.

Chunyan Zhang, Junchao Wang, Qinglei Zhou, Ting Xu, Ke Tang, Hairen Gui, and Fudong Liu. A survey of automatic source code summarization. *Symmetry*, 14(3):471, 2022. doi: 10.3390/SYM14030471. URL https://doi.org/10.3390/sym14030471.

Dejiao Zhang, Wasi Uddin Ahmad, Ming Tan, Hantian Ding, Ramesh Nallapati, Dan Roth, Xiaofei Ma, and Bing Xiang. Code representation learning at scale. *CoRR*, abs/2402.01935, 2024a. doi: 10.48550/ARXIV.2402.01935. URL https://doi.org/10.48550/arXiv.2402.01935.

Fengji Zhang, Bei Chen, Yue Zhang, Jin Liu, Daoguang Zan, Yi Mao, Jian-Guang Lou, and Weizhu Chen. Repocoder: Repository-level code completion through iterative retrieval and generation. *CoRR*, abs/2303.12570, 2023b. doi: 10.48550/ARXIV.2303.12570. URL https://doi.org/10.48550/arXiv.2303.12570.

Haibo Zhang and Kouichi Sakurai. A survey of software clone detection from security perspective. *IEEE Access*, 9:48157–48173, 2021. doi: 10.1109/ACCESS.2021.3065872. URL https://doi.org/10.1109/ACCESS.2021.3065872.

Jian Zhang, Xu Wang, Hongyu Zhang, Hailong Sun, Kaixuan Wang, and Xudong Liu. A novel neural source code representation based on abstract syntax tree. In Joanne M. Atlee, Tevfik Bultan, and Jon Whittle

(eds.), *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, pp. 783–794. IEEE / ACM, 2019a. doi: 10.1109/ICSE.2019.00086. URL https://doi.org/10.1109/ICSE.2019.00086.

Jiansong Zhang and Nora M. El-Gohary. Integrating semantic nlp and logic reasoning into a unified system for fully-automated code checking. *Automation in Construction*, 73:45–57, 2017. ISSN 0926-5805. doi: https://doi.org/10.1016/j.autcon.2016.08.027. URL https://www.sciencedirect.com/science/article/pii/S0926580516301819.

Jingxuan Zhang, He Jiang, Zhilei Ren, and Xin Chen. Recommending apis for API related questions in stack overflow. *IEEE Access*, 6:6205–6219, 2018. doi: 10.1109/ACCESS.2017.2777845. URL https://doi.org/10.1109/ACCESS.2017.2777845.

Kechi Zhang, Zhuo Li, Jia Li, Ge Li, and Zhi Jin. Self-edit: Fault-aware code editor for code generation. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 769–787. Association for Computational Linguistics, 2023c. doi: 10.18653/V1/2023.ACL-LONG.45. URL https://doi.org/10.18653/v1/2023.acl-long.45.

Quanjun Zhang, Chunrong Fang, Yuxiang Ma, Weisong Sun, and Zhenyu Chen. A survey of learning-based automated program repair. *CoRR*, abs/2301.03270, 2023d. doi: 10.48550/arXiv.2301.03270. URL https://doi.org/10.48550/arXiv.2301.03270.

Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir R. Radev. Editing-based SQL query generation for cross-domain context-dependent questions. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 5337–5348. Association for Computational Linguistics, 2019b. doi: 10.18653/v1/D19-1537. URL https://doi.org/10.18653/v1/D19-1537.

Shudan Zhang, Hanlin Zhao, Xiao Liu, Qinkai Zheng, Zehan Qi, Xiaotao Gu, Xiaohan Zhang, Yuxiao Dong, and Jie Tang. Naturalcodebench: Examining coding performance mismatch on humaneval and natural user prompts. 2024b. URL https://doi.org/10.48550/arXiv.2405.04520.

Tianzhu Zhang, Han Qiu, Gabriele Castellano, Myriana Rifai, Chung Shue Chen, and Fabio Pianese. System log parsing: A survey. *IEEE Trans. Knowl. Data Eng.*, 35(8):8596–8614, 2023e. doi: 10.1109/TKDE.2022.3222417. URL https://doi.org/10.1109/TKDE.2022.3222417.

Xinyu Zhang, Siddharth Muralee, Sourag Cherupattamoolayil, and Aravind Machiry. On the effectiveness of large language models for github workflows. *CoRR*, abs/2403.12446, 2024c. doi: 10.48550/ARXIV.2403.12446. URL https://doi.org/10.48550/arXiv.2403.12446.

Xu Zhang, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie, Xinsheng Yang, Qian Cheng, Ze Li, Junjie Chen, Xiaoting He, Randolph Yao, Jian-Guang Lou, Murali Chintalapati, Furao Shen, and Dongmei Zhang. Robust log-based anomaly detection on unstable log data. In Marlon Dumas, Dietmar Pfahl, Sven Apel, and Alessandra Russo (eds.), *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2019, Tallinn, Estonia, August 26-30, 2019*, pp. 807–817. ACM, 2019c. doi: 10.1145/3338906.3338931. URL https://doi.org/10.1145/3338906.3338931.

Ying Zhang, Wenjia Song, Zhengjie Ji, Danfeng Yao, and Na Meng. How well does LLM generate security tests? *CoRR*, abs/2310.00710, 2023f. doi: 10.48550/ARXIV.2310.00710. URL https://doi.org/10.48550/arXiv.2310.00710.

Ziyin Zhang, Yikang Liu, Weifang Huang, Junyu Mao, Rui Wang, and Hai Hu. MELA: multilingual evaluation of linguistic acceptability. *CoRR*, abs/2311.09033, 2023g. doi: 10.48550/ARXIV.2311.09033. URL https://doi.org/10.48550/arXiv.2311.09033.

Ziyin Zhang, Lizhen Xu, Zhaokun Jiang, Hongkun Hao, and Rui Wang. Multiple-choice questions are efficient and robust llm evaluators. 2024d. URL https://doi.org/10.48550/arXiv.2405.11966.

Gang Zhao and Jeff Huang. Deepsim: deep learning code functional similarity. In Gary T. Leavens, Alessandro Garcia, and Corina S. Pasareanu (eds.), *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2018, Lake Buena Vista, FL, USA, November 04-09, 2018*, pp. 141–151. ACM, 2018. doi: 10.1145/3236024.3236068. URL https://doi.org/10.1145/3236024.3236068.

Liang Zhao, Hexin Cao, and Yunsong Zhao. GP: context-free grammar pre-training for text-to-sql parsers. *CoRR*, abs/2101.09901, 2021. URL https://arxiv.org/abs/2101.09901.

Liang Zhao, Xiaocheng Feng, Xiachong Feng, Bing Qin, and Ting Liu. Length extrapolation of transformers: A survey from the perspective of position encoding. *CoRR*, abs/2312.17044, 2023. doi: 10.48550/ARXIV.2312.17044. URL https://doi.org/10.48550/arXiv.2312.17044.

Liping Zhao, Waad Alhoshan, Alessio Ferrari, Keletso J. Letsholo, Muideen A. Ajagbe, Erol-Valeriu Chioasca, and Riza Theresa Batista-Navarro. Natural language processing (NLP) for requirements engineering: A systematic mapping study. *CoRR*, abs/2004.01099, 2020. URL https://arxiv.org/abs/2004.01099.

Qinkai Zheng, Xiao Xia, Xu Zou, Yuxiao Dong, Shan Wang, Yufei Xue, Lei Shen, Zihan Wang, Andi Wang, Yang Li, Teng Su, Zhilin Yang, and Jie Tang. Codegeex: A pre-trained model for code generation with multilingual benchmarking on humaneval-x. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pp. 5673–5684, New York, NY, USA, 2023a. Association for Computing Machinery. ISBN 9798400701030. doi: 10.1145/3580305.3599790. URL https://doi.org/10.1145/3580305.3599790.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. Opencodeinterpreter: Integrating code generation with execution and refinement. *CoRR*, abs/2402.14658, 2024. doi: 10.48550/ARXIV.2402.14658. URL https://doi.org/10.48550/arXiv.2402.14658.

Yunhui Zheng, Saurabh Pujar, Burn L. Lewis, Luca Buratti, Edward A. Epstein, Bo Yang, Jim Laredo, Alessandro Morari, and Zhong Su. D2A: A dataset built for ai-based vulnerability detection methods using differential analysis. In *43rd IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice, ICSE (SEIP) 2021, Madrid, Spain, May 25-28, 2021*, pp. 111–120. IEEE, 2021. doi: 10.1109/ICSE-SEIP52600.2021.00020. URL https://doi.org/10.1109/ICSE-SEIP52600.2021.00020.

Zibin Zheng, Kaiwen Ning, Jiachi Chen, Yanlin Wang, Wenqing Chen, Lianghong Guo, and Weicheng Wang. Towards an understanding of large language models in software engineering tasks. *CoRR*, abs/2308.11396, 2023b. doi: 10.48550/ARXIV.2308.11396. URL https://doi.org/10.48550/arXiv.2308.11396.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103, 2017. URL http://arxiv.org/abs/1709.00103.

Victor Zhong, Mike Lewis, Sida I. Wang, and Luke Zettlemoyer. Grounded adaptation for zero-shot executable semantic parsing. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 6869–6882. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.EMNLP-MAIN.558. URL https://doi.org/10.18653/v1/2020.emnlp-main.558.

Wenkang Zhong, Chuanyi Li, Jidong Ge, and Bin Luo. Neural program repair : Systems, challenges and solutions. In *Internetware 2022: 13th Asia-Pacific Symposium on Internetware, Hohhot, China, June 11 - 12, 2022*, pp. 96–106. ACM, 2022. doi: 10.1145/3545258.3545268. URL https://doi.org/10.1145/3545258.3545268.

Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, and Hongsheng Li. Solving challenging math word problems using GPT-4 code interpreter with code-based self-verification. *CoRR*, abs/2308.07921, 2023a. doi: 10.48550/ARXIV.2308.07921. URL https://doi.org/10.48550/arXiv.2308.07921.

Bingzhe Zhou, Xinying Wang, Shengbin Xu, Yuan Yao, Minxue Pan, Feng Xu, and Xiaoxing Ma. Hybrid API migration: A marriage of small API mapping models and large language models. In Hong Mei, Jian Lv, Zhi Jin, Xuandong Li, Xiaohu Yang, and Xin Xia (eds.), *Proceedings of the 14th Asia-Pacific Symposium on Internetware, Internetware 2023, Hangzhou, China, August 4-6, 2023*, pp. 12–21. ACM, 2023b. doi: 10.1145/3609437.3609466. URL https://doi.org/10.1145/3609437.3609466.

Xin Zhou, Sicong Cao, Xiaobing Sun, and David Lo. Large language model for vulnerability detection and repair: Literature review and the road ahead. *CoRR*, abs/2404.02525, 2024. doi: 10.48550/ARXIV.2404.02525. URL https://doi.org/10.48550/arXiv.2404.02525.

Yaqin Zhou, Shangqing Liu, Jing Kai Siow, Xiaoning Du, and Yang Liu. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 10197–10207, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/49265d2447bc3bbfe9e76306ce40a31f-Abstract.html.

Ziyi Zhou, Huiqun Yu, Guisheng Fan, Zijie Huang, and Kang Yang. Towards retrieval-based neural code summarization: A meta-learning approach. *IEEE Trans. Software Eng.*, 49(4):3008–3031, 2023c. doi: 10.1109/TSE.2023.3238161. URL https://doi.org/10.1109/TSE.2023.3238161.

Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, and Michael R. Lyu. Tools and benchmarks for automated log parsing. In Helen Sharp and Mike Whalen (eds.), *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE (SEIP) 2019, Montreal, QC, Canada, May 25-31, 2019*, pp. 121–130. IEEE / ACM, 2019. doi: 10.1109/ICSE-SEIP.2019.00021. URL https://doi.org/10.1109/ICSE-SEIP.2019.00021.

Ming Zhu, Aneesh Jain, Karthik Suresh, Roshan Ravindran, Sindhu Tipirneni, and Chandan K. Reddy. Xlcost: A benchmark dataset for cross-lingual code intelligence. *CoRR*, abs/2206.08474, 2022a. doi: 10.48550/ARXIV.2206.08474. URL https://doi.org/10.48550/arXiv.2206.08474.

Ming Zhu, Karthik Suresh, and Chandan K. Reddy. Multilingual code snippets training for program translation. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pp. 11783–11790. AAAI Press, 2022b. doi: 10.1609/AAAI.V36I10.21434. URL https://doi.org/10.1609/aaai.v36i10.21434.

Qihao Zhu, Zeyu Sun, Yuan-an Xiao, Wenjie Zhang, Kang Yuan, Yingfei Xiong, and Lu Zhang. A syntax-guided edit decoder for neural program repair. In Diomidis Spinellis, Georgios Gousios, Marsha Chechik, and Massimiliano Di Penta (eds.), *ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, August 23-28, 2021*, pp. 341–353. ACM, 2021. doi: 10.1145/3468264.3468544. URL https://doi.org/10.1145/3468264.3468544.

Qihao Zhu, Daya Guo, Zhihong Shao, Dejian Yang, Peiyi Wang, Runxin Xu, Y Wu, Yukun Li, Huazuo Gao, Shirong Ma, et al. Deepseek-coder-v2: Breaking the barrier of closed-source models in code intelligence. *arXiv preprint arXiv:2406.11931*, 2024. URL https://doi.org/10.48550/arXiv.2406.11931.

Terry Yue Zhuo, Armel Zebaze, Nitchakarn Suppattarachai, Leandro von Werra, Harm de Vries, Qian Liu, and Niklas Muennighoff. Astraios: Parameter-efficient instruction tuning code large language models. *CoRR*, abs/2401.00788, 2024. doi: 10.48550/ARXIV.2401.00788. URL https://doi.org/10.48550/arXiv.2401.00788.

Deqing Zou, Sujuan Wang, Shouhuai Xu, Zhen Li, and Hai Jin. μvuldeepecker: A deep learning-based system for multiclass vulnerability detection. *IEEE Trans. Dependable Secur. Comput.*, 18(5):2224–2236, 2021. doi: 10.1109/TDSC.2019.2942930. URL https://doi.org/10.1109/TDSC.2019.2942930.

A Benchmarks for Downstrem Tasks

Table 5, 6, 7, 8, 9, 10 list benchmark datasets for code downstream tasks. For the size of these datasets, we use self-reported number whenever available.

Table 5: Benchmarks for text-to-SQL generation.

Task	Date	Benchmark	Source	Size
Text-to-SQL	1990	ATIS	Hemphill et al. (1990); Dahl et al. (1994)	11508
	1996	GeoQuery	Zelle & Mooney (1996)	877
	2000	Restaurants	Tang & Mooney (2000)	378
	2014-09	MAS	Li & Jagadish (2014)	196
	2017-02	Yelp	Yaghmazadeh et al. (2017)	128
	2017-02	IMDb	Yaghmazadeh et al. (2017)	131
	2017-04	Scholar	Iyer et al. (2017)	816
	2017-08	WikiSQL	Zhong et al. (2017)	80654
	2018-06	Advising	Finegan-Dollak et al. (2018)	4570
	2018-09	Spider	Yu et al. (2018c)	10181
	2019-06	SParC	Yu et al. (2019c)	12726
	2019-07	MIMICSQL	Wang et al. (2020b)	10000
	2019-09	CoSQL	Yu et al. (2019b)	15598
	2020-05	Criteria-to-SQL	Yu et al. (2020)	2003
	2020-10	Squall	Shi et al. (2020)	11276
	2020-10	Spider-Realistic	Deng et al. (2021)	508
	2021-06	Spider-Syn	Gan et al. (2021a)	8034
	2021-06	SEDE	Hazoom et al. (2021)	12023
	2021-06	KaggleDBQA	Lee et al. (2021)	400
	2021-09	Spider-DK	Gan et al. (2021b)	535
	2022-05	Spider-SS	Gan et al. (2022)	8034
	2022-05	Spider-CG	Gan et al. (2022)	45599
	2023-05	BIRD	Li et al. (2023g)	12751

Table 6: Benchmarks for program synthesis. JS is short for JavaScript. *Automatically mined/human-annotated. †These are 1749 prompts for 48 problems. ‡10538 prompts for 1420 methods. ◇Machine-generated/human-annotated prompts.

Task	Date	Benchmark	Source	Size	Language
Program Synthesis	2018-02	NL2Bash	Lin et al. (2018b)	9305	Bash
	2018-08	CONCODE	Iyer et al. (2018)	104K	Java
	2019-10	JuICe	Agashe et al. (2019)	*1.5M/3725	Python
	2021-05	APPS	Hendrycks et al. (2021a)	10000	Python
	2021-07	HumanEval	Chen et al. (2021b)	164	Python
	2021-08	MBPP	Austin et al. (2021)	974	Python
	2021-08	MathQA-Python	Austin et al. (2021)	23914	Python
	2021-08	PlotCoder	Chen et al. (2021c)	40797	Python
	2022-01	DSP	Chandel et al. (2022)	1119	Python
	2022-02	CodeContests	Li et al. (2022h)	13610	C++, Python, Java
	2022-03	MCoNaLa	Wang et al. (2023i)	896	Python
	2022-06	AixBench	Hao et al. (2022)	336	Java
	2022-10	MBXP	Athiwaratkun et al. (2023)	12.4K	13
	2022-10	Multilingual HumanEval	Athiwaratkun et al. (2023)	1.9K	12
	2022-10	MathQA-X	Athiwaratkun et al. (2023)	5.6K	Python, Java, JS
	2022-11	ExeDS	Huang et al. (2022b)	534	Python
	2022-11	DS-1000	Lai et al. (2023)	1000	Python
	2022-12	ODEX	Wang et al. (2022h)	945	Python
	2023-02	CoderEval	Yu et al. (2023a)	460	Python, Java
	2023-03	xCodeEval	Khan et al. (2023)	5.5M	11
	2023-03	HumanEval-X	Zheng et al. (2023a)	820	Python, C++, Java, JS, Go
	2023-06	StudentEval	Babe et al. (2023)	†1749	Python
	2023-06	DotPrompts	Agrawal et al. (2023)	‡10538	Java
	2023-09	CodeApex	Fu et al. (2023)	476	C++
	2023-09	VerilogEval	Liu et al. (2023g)	◇8645/156	Verilog
	2023-11	ML-Bench	Liu et al. (2023l)	10040	Bash
	2024-01	ParEval	Nichols et al. (2024)	420	C++, HIP, CUDA
	2024-04	MMCode	Li et al. (2024b)	3548	Python
	2024-04	PECC	Haller et al. (2024)	2396	Python
	2024-05	NaturalCodeBench	Zhang et al. (2024b)	402	Python, Java
	2024-05	Plot2Code	Wu et al. (2024a)	132	Python
	2024-05	MHPP	Dai et al. (2024b)	140	Python

Table 7: Benchmarks for code translation and program repair. JS is short for JavaScript. *These are pairwise sample counts. For example, HumanEval-X includes 164 programs, each implemented in 5 languages, totaling $164 \times (5 \times 4 \ / \ 2) = 1640$ translation pairs. †These are code change datasets, and only a subset therein concern bug-fixing.

Task	Date	Benchmark	Source	Size	Language
Code Translation	2020-06	GeeksforGeeks	Rozière et al. (2020)	1.4K	C++, Java, Python
	2021-02	CodeTrans	Lu et al. (2021)	11.8K	Java, C#
	2021-08	Avatar	Ahmad et al. (2023)	9515	Java, Python
	2022-06	CoST	Zhu et al. (2022b)	*132K	C++, Java, Python, C#, JS, PHP, C
	2022-06	XLCoST	Zhu et al. (2022a)	*567K	C++, Java, Python, C#, JS, PHP, C
	2023-03	xCodeEval	Khan et al. (2023)	*5.6M	11
	2023-03	HumanEval-X	Zheng et al. (2023a)	*1640	Python, C++, Java, JS, Go
	2023-08	G-TransEval	Jiao et al. (2023)	*4000	C++, Java, C#, JS, Python
	2023-10	CodeTransOcean	Yan et al. (2023)	270.5K	45
Program Repair	2014-07	Defects4J	Just et al. (2014)	357	Java
	2015-12	ManyBugs	Goues et al. (2015)	185	C
	2015-12	IntroClass	Goues et al. (2015)	998	C
	2016-11	BugAID	Hanam et al. (2016)	105K	JS
	2017-02	DeepFix	Gupta et al. (2017)	6971	C
	2017-05	Codeflaws	Tan et al. (2017)	3902	C
	2017-10	QuixBugs	Lin et al. (2017)	80	Java, Python
	2018-05	Bugs.jar	Saha et al. (2018)	1158	Java
	2018-12	BFP	Chakraborty et al. (2022b)	124K	Java
	2019-01	Bears	Madeiral et al. (2019)	251	Java
	2019-01	unnamed	Tufano et al. (2019a)	†21.8K	Java
	2019-04	BugsJS	Gyimesi et al. (2019)	453	JS
	2019-05	BugSwarm	Tomassi et al. (2019)	1827/1264	Java/Python
	2019-05	CPatMiner	Nguyen et al. (2019)	†17K	Java
	2019-05	ManySStuBs4J	Karampatsis & Sutton (2020)	154K	Java
	2019-11	Refactory	Hu et al. (2019)	1783	Python
	2020-07	CoCoNut	Lutellier et al. (2020)	24M	Java, JS, C, Python
	2020-10	Review4Repair	Huq et al. (2022)	58021	Java
	2020-11	BugsInPy	Widyasari et al. (2020)	493	Python
	2021-07	TFix	Berabi et al. (2021)	105K	JS
	2021-08	Megadiff	Monperrus et al. (2021)	†663K	Java
	2022-01	SSB/TSSB	Richter & Wehrheim (2022)	9M/3M	Python
	2022-10	FixJS	Csuvik & Vidács (2022)	324K	JS
	2022-11	TypeBugs	Oh & Oh (2022)	93	Python
	2023-03	xCodeEval	Khan et al. (2023)	4.7M	11
	2023-04	RunBugRun	Prenner & Robbes (2023)	450K	C, C++, Java, Python, JS, Ruby, Go, PHP
	2023-08	HumanEvalPack	Muennighoff et al. (2024)	984	Python, JS, Go, Java, C++, Rust
	2024-01	DebugBench	Tian et al. (2024)	4253	C++, Java, Python

Table 8: Benchmarks for code summarization, code completion, commit message generation, and defect/vulnerability detection. JS is short for JavaScript. *The task of code completion can be evaluated on any source code corpus, so we only list a few widely used benchmarks here. For cross-file code completion please refer to Table 10. †With/without verb-direct object filter.

Task	Date	Benchmark	Source	Size	Language
Code Summarization	2016-08	CODE-NN	Iyer et al. (2016)	66K/32K	C#/SQL
	2017-07	unnamed	Barone & Sennrich (2017)	150K	Python
	2018-05	DeepCom	Hu et al. (2018a)	588K	Java
	2018-07	TL-CodeSum	Hu et al. (2018b)	411K	Java
	2018-11	unnamed	Wan et al. (2018)	109K	Python
	2019-02	unnamed	LeClair et al. (2019)	2.1M	Java
	2019-09	CodeSearchNet	Husain et al. (2019)	2.3M	Go, JS, Python, PHP, Java, Ruby
	2023-08	HumanEvalPack	Muennighoff et al. (2024)	984	Python, JS, Go, Java, C++, Rust
*Code Completion	2013-05	GitHub Java Corpus	Allamanis & Sutton (2013)	2.1M	Java
	2016-10	Py150	Raychev et al. (2016a)	150K	Python
	2016-10	JS150	Raychev et al. (2016a)	150K	JS
	2023-06	LCC	Guo et al. (2023b)	360K	Python, Java, C#
Commit Message Generation	2017-03	unnamed	Jiang & McMillan (2017)	509K	Java
	2017-04	CommitGen	Loyola et al. (2017)	153K	Python, C++, Java, JS
	2017-08	CommitGen	Jiang et al. (2017)	†32K/75K	Java
	2018-09	NNGen	Liu et al. (2018)	27K	Java
	2019-05	PtrGNCMsg	Liu et al. (2019c)	64.9K	Java
	2019-08	CoDiSum	Xu et al. (2019a)	90.7K	Java
	2019-12	ATOM	Liu et al. (2022c)	160K	Java
	2021-05	CommitBERT	Jung (2021)	346K	Python, PHP, Go, Java, JS, Ruby
	2021-07	MCMD	Tao et al. (2021)	2.25M	Java, C#, C++, Python, JS
	2021-07	CoRec	Wang et al. (2021c)	107K	Java
	2023-07	ExGroFi	Wang et al. (2023c)	19263	Java
	2023-08	CommitChronicle	Eliseeva et al. (2023)	10.7M	20
Defect (Vulnerability) Detection	2018-01	CGD	Li et al. (2018b)	62K	C, C++
	2018-07	Draper VDISC	Russell et al. (2018)	12.8M	C, C++
	2018-07	SySeVR	Li et al. (2022j)	15591	C, C++
	2018-04	unnamed	Lin et al. (2018a)	32988	C, C++
	2019-02	unnamed	Ponta et al. (2019)	624	Java
	2019-09	Devign	Zhou et al. (2019)	48687	C
	2019-11	unnamed	Lin et al. (2021)	170K	C, C++
	2019-12	GREAT	Hellendoorn et al. (2020)	2.8M	Python
	2020-01	MVD	Zou et al. (2021)	182K	C, C++
	2020-02	unnamed	Lin et al. (2019)	1471	C
	2020-09	ReVeal	Chakraborty et al. (2022c)	18K	C
	2020-09	Big-Vul	Fan et al. (2020)	265K	C, C++
	2021-02	D2A	Zheng et al. (2021)	1.3M	C, C++
	2021-05	PyPIBugs	Allamanis et al. (2021)	2374	Python
	2021-07	CVEfixes	Bhandari et al. (2021)	5495	27
	2021-08	CrossVul	Nikitopoulos et al. (2021)	27476	40+
	2023-04	DiverseVul	Chen et al. (2023g)	349K	C, C++
	2023-06	VulnPatchPairs	Risse & Böhme (2023)	26K	C
	2023-11	VulBench	Gao et al. (2023d)	455	C

Table 9: Benchmarks for code retrieval, code reasoning, type inference, and clone detection/code search,. JS is short for JavaScript. *These benchmarks include a large number of automatically constructed samples, and a small set of human-annotated samples. †These are general-domain reasoning benchmarks, and only a subset therein concern programming, algorithms, and other topics related to computer science. ‡These are project counts (or, in the case of Cassano et al. (2023c), file counts). Yee & Guha (2023) propose to measure project-level type check rate instead of type prediction accuracy for TypeScript. ◇These are 21K/24K Java/Python methods for 576 programming problems.

Task	Date	Benchmark	Source	Size	Language
Code Retrieval	2018-03	StaQC	Yao et al. (2018)	268K	Python, SQL
	2018-05	DeepCS	Gu et al. (2018)	16M	Java
	2018-05	CoNaLa	Yin et al. (2018)	*600K/2.9K	Python
	2019-08	unnamed	Li et al. (2019b)	287	Java
	2019-09	CodeSearchNet	Husain et al. (2019)	*2.3M/99	Go, JS, Python, PHP, Java, Ruby
	2020-02	CosBench	Yan et al. (2020)	52	Java
	2020-08	SO-DS	Heyman & Cutsem (2020)	2.2K	Python
	2020-10	FB-Java	Ling et al. (2021)	249K	Java
	2021-02	AdvTest	Lu et al. (2021)	280K	Python
	2021-02	WebQueryTest	Lu et al. (2021)	1K	Python
	2021-05	CoSQA	Huang et al. (2021)	21K	Python
	2024-03	ProCQA	Li et al. (2024c)	5.2M	C, C++, Java, Python, Ruby, Lisp, JS, C#, Go, Rust, PHP
Code Reasoning	2020-09	MMLU	Hendrycks et al. (2021b)	†15908	
	2021-09	CodeQA	Liu & Wan (2021)	120K/70K	Java/Python
	2022-10	CS1QA	Lee et al. (2022)	9237	
	2023-05	C-Eval	Huang et al. (2023b)	†13948	
	2023-06	CMMLU	Li et al. (2023b)	†11528	
	2023-09	CodeApex	Fu et al. (2023)	250	C++
	2024-01	CRUXEval	Gu et al. (2024)	800	Python
	2024-05	PythonIO	Zhang et al. (2024d)	2650	Python
Type Inference	2019-12	TypeWriter OSS	Pradel et al. (2020)	208K	Python
	2020-04	Typilus	Allamanis et al. (2020)	252K	Python
	2020-04	LambdaNet	Wei et al. (2020)	‡300	TypeScript
	2021-04	ManyTypes4Py	Mir et al. (2021)	869K	Python
	2022-10	ManyTypes4TypeScript	Jesse & Devanbu (2022)	9.1M	TypeScript
	2023-02	TypeWeaver	Yee & Guha (2023)	‡513	TypeScript
	2023-03	BetterTypes4Py	Wei et al. (2023)	608K	Python
	2023-03	InferTypes4Py	Wei et al. (2023)	4.6K	Python
	2023-05	OpenTau	Cassano et al. (2023c)	‡744	TypeScript
Clone Detection / Code Search	2014-09	BigCloneBench	Svajlenko et al. (2014)	6M	Java
	2014-09	POJ-104	Mou et al. (2016)	52K	C, C++
	2019-05	unnamed	Perez & Chiba (2019)	◇21K/24K	Java/Python
	2019-11	CLCDSA	Nafi et al. (2019)	78K	Java, C#, Python

Table 10: Benchmarks for unit test/assertion generation, log parsing, and repository level coding. *LogHub (2023) is an annotated subset of LogHub (2018). †Line Completion/API Invocation Completion/Function Completion. ‡Retrieval/Completion/Pipeline. ⋆File count. ◇Migration/Temporal Edit.

Task	Date	Benchmark	Source	Size	Language
Unit Test / Assertion Generation	2014-12	SF110	Fraser & Arcuri (2014)	24K	Java
	2020-09	Method2Test	Tufano et al. (2021a)	781K	Java
	2021-08	ConTest	Villmow et al. (2021)	365K	Java
Log Parsing	2018-11	LogHub (2018)	Zhu et al. (2019); He et al. (2020)	379M	
	2023-08	LogHub (2023)	Jiang et al. (2023e)	*50.4M	
Repository-Level Coding	2023-03	RepoEval	Zhang et al. (2023b)	†1600/1600/373	Python
	2023-06	RepoBench	Liu et al. (2023j)	‡890K/9M/43K	Python, Java
	2023-06	PragmaticCode	Agrawal et al. (2023)	⋆880	Java
	2023-06	Stack-Repo	Shrivastava et al. (2023a)	816K	Java
	2023-09	CodePlan	Bairi et al. (2023)	◇645/21	◇C#/Python
	2023-10	SWE-Bench	Jimenez et al. (2023)	2294	Python
	2023-10	CrossCodeEval	Ding et al. (2023)	9928	Python, Java, TypeScript, C#
	2024-03	EvoCodeBench	Li et al. (2024a)	275	Python

B Results on Downstream Benchmarks

For the benchmarks listed in Appendix A, we also list code language models' reported performance on some of the most widely used ones, including:

- NL-code search (Table 11): CodeSearchNet (Husain et al., 2019), CosQA (Huang et al., 2021), and AdvTest (Lu et al., 2021), all measured by Mean Reciprocal Rank (MRR);
- Clone detection (Table 12): BigCloneBench (Svajlenko et al., 2014), measured by F1, and POJ-104 (Mou et al., 2016), measured by Mean Average Precision; the train/test splits of both two datasets are provided by Lu et al. (2021);
- Defect detection (Table 13): Devign (Zhou et al., 2019), measured by accuracy; the train/test split is provided by Lu et al. (2021);
- Code summarization (Table 14): CodeSearchNet (Husain et al., 2019), measured by BLEU; the train/test split is provided by Lu et al. (2021);
- Code translation (Table 15, 16): CodeTrans (Lu et al., 2021), measured by BLEU and CodeBLEU, and Transcoder (Rozière et al., 2020), measured by pass@1;
- Code reasoning (Table 17, 18): CRUXEval (Gu et al., 2024), measured by pass@1, and PythonIO, measured by accuracy.

Table 11: Mean Reciprocal Rank of NL-code search performance on CodeSearchNet, AdvTest, and CosQA.

	Ruby	JS	Go	Python	Java	PHP	Average	AdvTest	CosQA	source
RoBERTa	58.7	51.7	85.0	58.7	59.9	56.0	61.7	18.3	60.3	Lu et al. (2021)↓
CodeBERT	67.9	62.0	88.2	67.2	67.6	62.8	69.3	27.2	65.7	
GraphCodeBERT	70.3	64.4	89.7	69.2	69.1	64.9	71.3	35.2	68.4	Guo et al. (2021a)
SynCoBERT	72.2	67.7	91.3	72.4	72.3	67.8	74.0	38.1	69.6	Wang et al. (2021d)
SPT-Code	70.1	64.1	89.5	69.9	70.0	65.1	71.5			Niu et al. (2022)
CodeRetriever	77.1	71.9	92.4	75.8	76.5	70.8	77.4	46.9	75.4	Li et al. (2022e)
UniXcoder	74.0	68.4	91.5	72.0	72.6	67.6	74.4	41.3	70.1	Guo et al. (2022)↓
PLBART	67.5	61.6	88.7	66.3	66.3	61.1	68.5	34.7	65.0	
CodeT5$_{base}$	71.9	65.5	88.8	69.8	68.6	64.5	71.5	39.3	67.8	
CoCoSoDa	81.8	76.4	92.1	75.7	76.3	70.3	78.8			Shi et al. (2023b)
Code-MVP								40.4	72.1	Wang et al. (2022e)
SCodeR	77.5	72.0	92.7	74.2	74.8	69.2	76.7	45.5	74.5	Li et al. (2022f)
CodeT5+$_{large}$	78.0	71.3	92.7	75.8	76.2	70.1	77.4	44.7	74.0	Wang et al. (2023h)↓
CodeGen-multi 350M	66.0	62.2	90.0	68.6	70.1	63.9	70.1	34.8	64.8	

Table 12: Clone detection performance on BigCloneBench (F1) and POJ-104 (Mean Average Precision). Works in the two blocks use different implementations and report different baseline scores.

	BigCloneBench (F1)	POJ-104 (MAP)	source
RoBERTa	94.9	79.96	Lu et al. (2021)↓
CodeBERT	96.5	84.29	
DOBF	96.5		Lachaux et al. (2021)
PLBART	97.2		Ahmad et al. (2021)
GraphCodeBERT	97.1	85.16	Wang et al. (2021d)↓
SynCoBERT	97.4	88.24	
CodeT5$_{base}$	97.2		Wang et al. (2021f)
RoBERTa	91.3	76.67	Guo et al. (2022)
CodeBERT	94.1	82.67	Guo et al. (2021a)↓
GraphCodeBERT	95.0	85.16	
DISCO	94.4	83.32	Ding et al. (2022a)
PLBART	93.6	86.27	Guo et al. (2022)↓
CodeT5$_{base}$	95.0	88.65	
UniXcoder	95.2	90.52	
SCodeR	95.3	92.45	Li et al. (2022f)

Table 13: Accuracy of defect detection on Devign. pt: pretrained. ft: finetuned.

	Accuracy	pt	ft	prompt	source
Transformer	61.6		✓	-	Ahmad et al. (2021)
RoBERTa	61.0	✓	✓	-	Lu et al. (2021)↓
CodeBERT	62.1	✓	✓	-	
PLBART	63.2	✓	✓	-	Ahmad et al. (2021)
GraphCodeBERT	63.2	✓	✓	-	Wang et al. (2021d)↓
SynCoBERT	64.5	✓	✓	-	
CodeT5$_{base}$	65.8	✓	✓	-	Wang et al. (2021f)
DISCO	64.4	✓	✓	-	Ding et al. (2022a)
VulBERTa	64.8	✓	✓	-	Hanif & Maffeis (2022)
Instruct-CodeGen 16B	47.8	✓		0-shot	Yuan et al. (2023a)↓
CodeAlpaca 7B	51.9	✓		0-shot	
Vicuna 7B	54.0	✓		0-shot	
WizardCoder 15B	54.4	✓		0-shot	

Table 14: BLEU scores of code summarization on CodeSearchNet, using the CodeXGLUE split (top) and the original split (bottom). pt: pretrained. ft: finetuned.

	Ruby	JS	Go	Python	Java	PHP	Average	pt	ft	prompt	source
Transformer	11.18	11.59	16.38	15.81	16.26	22.12	15.56		✓	-	Lu et al. (2021)↓
RoBERTa	11.17	11.90	17.72	18.14	16.47	24.02	16.57	✓	✓	-	
CodeBERT	12.16	14.90	18.07	19.06	17.65	25.16	17.83	✓	✓	-	
PLBART	14.11	15.56	18.91	19.30	18.45	23.58	18.32	✓	✓	-	Ahmad et al. (2021)
CodeT5$_{base}$	15.69	16.24	19.76	20.36	20.46	26.09	19.77	✓	✓	-	Wang et al. (2021f)
UniXcoder	14.87	15.85	19.07	19.13	20.31	26.54	19.30	✓	✓	-	Guo et al. (2022)
InCoder				18.27				✓		0-shot	Fried et al. (2023)
NatGen	15.38	16.00	19.43	20.09	20.38	26.00	19.55	✓	✓	-	Chakraborty et al. (2022a)
CodeT5+$_{large}$	15.63	17.93	19.64	20.47	20.83	26.39	20.15	✓	✓	-	Wang et al. (2023h)
StarCoder				21.99				✓		0-shot	Li et al. (2023h)
ChatGPT				10.28				✓		0-shot	Sun et al. (2023b)
PaLM 2				19.23				✓		few-shot	Haldar & Hockenmaier (2024)↓
LLaMA 2 70B				22.41				✓		few-shot	
Mastropaolo et al.	7.8	9.0	15.3	9.7	13.5	18.4	12.3	✓	✓	-	Niu et al. (2022)↓
CugLM	7.4	10.0	17.0	11.4	13.2	17.8	12.8	✓	✓	-	
TreeBERT	7.4	10.3	17.1	11.1	13.8	18.0	12.9	✓	✓	-	
CodeBERT	8.3	10.0	16.0	10.7	13.6	20.1	13.1	✓	✓	-	
GraphCodeBERT	8.3	10.9	16.5	11.0	14.5	20.0	13.5	✓	✓	-	
SPT-Code	8.4	12.8	18.8	12.8	16.8	20.4	15.0	✓	✓	-	

Table 15: BLEU and CodeBLEU scores of code translation on CodeTrans.

	Java→C#		C#→Java		source
	BLEU	CodeBLEU	BLEU	CodeBLEU	
Transformer	55.8	63.7	50.5	61.6	Lu et al. (2021)↓
CodeBERT	79.9	85.1	72.1	79.4	
GraphCodeBERT	80.6		72.6		Guo et al. (2021a)
PLBART	83.0	87.9	78.3	85.3	Ahmad et al. (2021)
SynCoBERT	80.8	84.8	76.5	82.2	Wang et al. (2021d)
CodeT5$_{base}$	84.0	87.8	79.9	84.4	Wang et al. (2021f)
NatGen		88.1		85.2	Chakraborty et al. (2022a)

Table 16: Pass@1 scores of code translation on TransCoder evaluation set.

	C++ → Java	C++ → Py	Java → C++	Java → Py	Py → C++	Py → Java	source
TransCoder	60.9	44.5	80.9	35.0	32.2	24.7	Rozière et al. (2020)
DOBF				40.6		46.6	Lachaux et al. (2021)
TransCoder-ST	66.7	61.1	84.1	67.8	52.2	56.7	Rozière et al. (2022)
TransCoder-IR	62.9		74.5				Szafraniec et al. (2023)
LaMDA		30.2					Chowdhery et al. (2023)↓
PaLM		51.8					
PaLM Coder		55.1					
StarCoder (self-debug)		70.0 (76.6)					Chen et al. (2023f)↓
Codex (self-debug)		80.4 (92.5)					
GPT-3.5 (self-debug)		89.1 (92.7)					
GPT-4 (self-debug)		77.3 (90.4)					

Table 17: Pass@1 scores of code reasoning on CRUXEval input and output prediction. DeepSeek-Coder-V2 is a Mixture-of-Experts model with a total of 236B parameters, where 21B are activated for each token.

	size	input	output	source
Phi-1	1.3B	13.9	23.3	Gu et al. (2024)↓
Phi-1.5	1.3B	24.1	27.1	
Mistral	7B	36.0	31.7	
StarCoder$_{Base}$	16B	31.6	33.3	
CodeLLaMA (CoT)	34B	46.5 (50.4)	41.1 (46.0)	
DeepSeek-Coder$_{Instruct}$	33B	47.4	44.0	
GPT-3.5 (CoT)		49.2 (49.1)	50.0 (63.3)	
GPT-4 (CoT)	-	67.1 (74.8)	63.4 (81.9)	
StarCoder2	16B	48.1	47.1	Lozhkov et al. (2024)
Codestral + CoT	22B	48.0	60.6	Zhu et al. (2024)↓
LLaMA-3$_{Instruct}$ + CoT	70B	61.1	64.3	
DeepSeek-Coder-V2$_{Instruct}$ + CoT	236B (21B)	70.0	75.1	
Gemini-1.5-Pro + CoT	-	67.0	77.5	
Claude-3-Opus + CoT	-	73.4	82.0	
GPT-4o + CoT	-	77.4	88.7	

Table 18: Accuracy of code reasoning performance on PythonIO, extracted from Zhang et al. (2024d).

	size	accuracy
Phi-2	2.7B	29.8
Phi-3	3.8B	38.2
ChatGLM3$_{Base}$	6B	27.7
Gemma	7B	30.5
Mistral	7B	31.7
Qwen1.5	7B	32.8
LLaMA-3$_{Instruct}$	8B	39.0
Flan-T5	11B	29.3
LLaMA-2	13B	26.6
Qwen1.5	14B	40.9
LLaMA-2	70B	38.2
LLaMA-3$_{Instruct}$	70B	70.1
Qwen1.5	72B	50.4