Towards Trustworthy LLMs for Code: A Data-Centric Synergistic Auditing Framework

Chong Wang†, Zhenpeng Chen†*, Tianlin Li†, Yilun Zhang‡, Yang Liu†
†*Nanyang Technological University, Singapore*
‡*AIXpert*
{chong.wang, zhenpeng.chen, yangliu}@ntu.edu.sg, tianlin001@e.ntu.edu.sg, yilun@aixpert.io

***Abstract*—LLM-powered coding and development assistants have become prevalent to programmers' workflows. However, concerns about the trustworthiness of LLMs for code persist despite their widespread use. Much of the existing research focused on either training or evaluation, raising questions about whether stakeholders in training and evaluation align in their understanding of model trustworthiness and whether they can move toward a unified direction. In this paper, we propose a vision for a unified trustworthiness auditing framework, DATATRUST, which adopts a data-centric approach that synergistically emphasizes both training and evaluation data and their correlations. DATATRUST aims to connect model trustworthiness indicators in evaluation with data quality indicators in training. It autonomously inspects training data and evaluates model trustworthiness using synthesized data, attributing potential causes from specific evaluation data to corresponding training data and refining indicator connections. Additionally, a trustworthiness arena powered by DATATRUST will engage crowdsourced input and deliver quantitative outcomes. We outline the benefits that various stakeholders can gain from DATATRUST and discuss the challenges and opportunities it presents.**

I. TRUSTWORTHINESS AUDITING OF LLMS FOR CODE

Large language models (LLMs) for code [1]–[5] have demonstrated significant potential in supporting various stages of the software development lifecycle [6]–[14]. As a result, LLM-powered coding and development assistants are now widely integrated into programmers' daily workflows. A prominent example is GitHub Copilot [15], an LLM-based coding assistant adopted by over 77,000 businesses and downloaded more than 20.3 million times from the VSCode Plugin marketplace (data as of September 24, 2024).

Despite the widespread adoption of LLMs for code in real-world development, significant concerns persist regarding their trustworthiness, particularly in dimensions such as *robustness*, *security*, *timeliness*, *privacy*, *fairness*, etc. In addition to the inherent risks common to general-purpose LLMs (*e.g.*, jailbreaking threats) [16]–[18], LLMs for code introduce additional concerns specific to code and software development. For example, the code generated by these models may contain vulnerabilities or weaknesses that pose serious code security risks [19]. Therefore, advancing towards more trustworthy LLMs for code has become an increasingly pressing issue.

While state-of-the-art research has concentrated on specific trustworthiness dimensions in either *training* [3], [4] or *evaluation* [19]–[22], the modern development and application of LLMs have evolved into a complex, iterative process of model training (including fine-tuning) and evaluation, involving multiple stakeholders. *This raises crucial questions: Have stakeholders in training and evaluation aligned their understandings of model trustworthiness? How can they be systematically guided toward a unified direction rather than relying on heuristic attempts? To address these, a unified trustworthiness auditing framework is essential—one that synergistically integrates both training and evaluation processes along with their iterative cycles.* Specifically, we present a *data-centric vision* for this framework, named DATATRUST, which attempts to connect the trustworthiness indicators in evaluation to the quality indicators in training. Building on this foundation, DATATRUST autonomously conducts independent inspections of training data and evaluates model trustworthiness using synthesized data, attributing potential causes identified in specific evaluation data to their corresponding training data and aiding in the refinement of indicator connections. A trustworthiness arena powered by DATATRUST can be launched to further engage crowd sources and deliver quantitative and comparative auditing outcomes such as leaderboards.

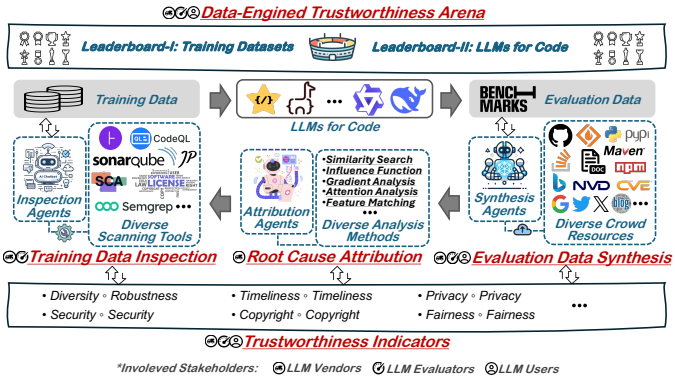

Fig. 1. Methodology Overview of DATATRUST.

DATATRUST can deliver benefits to a range of stakeholders:

- *LLM Vendors.* For vendors controlling both training data and production LLMs, DATATRUST streamlines the development of trustworthy LLMs for code. It provides insights into the cycle of training data inspection, filtering, model training, evaluation, and root cause analysis. Furthermore, publishing auditing reports for both training data and models enhances trustworthiness transparency for downstream stakeholders.

* Zhenpeng Chen is the corresponding author

- *LLM Evaluators.* DATATRUST offers comprehensive trustworthiness evaluations, supporting continuous trustworthiness report (*e.g.*, leaderboards) updates for both commercial and open-source LLMs and training corpora. It provides valuable insights for downstream stakeholders in selecting models or data. Moreover, identified trustworthiness issues can help upstream stakeholders, like LLM vendors, conduct root cause analysis on their training data.
- *LLM Users.* DATATRUST engages LLM users in two ways: by enabling them to review auditing reports or leaderboards for insights into model trustworthiness, and by collecting trustworthiness issues encountered by users to create a crowd-sourced trustworthiness arena, which helps evaluators refine assessments and vendors perform root cause analysis.

This paper calls for collaboration between academic and industrial communities to refine the data-centric vision and tackle challenges in implementing the DATATRUST framework for trustworthy LLMs for code. The goal is to align understandings, standardize auditing processes, and enhance the transparency of LLM trustworthiness and their data, ultimately benefiting a wide range of stakeholders.

II. DATATRUST: METHODOLOGY AND CHALLENGES

Figure 1 presents an overview of DATATRUST's methodology. We start by compiling a comprehensive set of *Model Trustworthiness Indicators* across various dimensions and linking them to corresponding *Training Data Quality Indicators*. These indicators guide (i) the assessment of training data quality and (ii) the construction of thorough evaluation data. Based on this foundation, we design three key data-centric processes: *Training Data Inspection*, *Evaluation Data Synthesis*, and *Root Cause Attribution*. These processes iteratively operate during the training and evaluation phases of LLMs for code, continuously refining indicator connections while engaging multiple stakeholders. We will leverage existing scanning tools, crowd-sourced resources, and analysis methods, integrating them with advanced technologies like LLM-based agents. Additionally, we introduce a *Data-Driven Trustworthiness Arena* to engage users actively, enhancing the comparison and benchmarking of mainstream LLMs for code and their training datasets.

A. Trustworthiness Indicators and Connections

Each entry is represented as an indicator pair $\mathcal{T} \circ \mathcal{E}$, where $\mathcal{T}$ denotes a quality indicator for training data and $\mathcal{E}$ represents a trustworthiness indicator for model evaluation.

- *Diversity* $\circ$ *Robustness.* Training data often lacks diversity, leading to imbalanced distributions across domains, functionalities, and identifiers. This can raise model robustness concerns, such as inconsistent performance across domains [23] and vulnerabilities to adversarial attacks [24].
- *Security* $\circ$ *Security.* Security risks, including various software vulnerabilities are commonly found in open-source code repositories and may remain undetected for extended periods [19]. Consequently, the presence of insecure data samples in the training data might result in LLMs generating insecure outputs, such as vulnerable code.
- *Timeliness* $\circ$ *Timeliness.* Training data sourced from open-source code repositories over extended periods may include outdated information, such as the use of outdated code patterns, inactive libraries, and deprecated APIs [21]. This might cause the trained LLMs to generate outdated outputs.
- *Copyright* $\circ$ *Copyright.* Source files in training data may be subject to specific licensing terms or copied from other licensed repositories, raising two major copyright concerns. First, the training data might introduce the risk of license term violations, particularly when used for training commercial LLMs [25]. Second, the trained LLMs might generate outputs that raise user concerns regarding copyright [26].
- *Privacy* $\circ$ *Privacy.* Training data often contains hard-coded privacy-sensitive information, particularly software credentials (*e.g.*, API keys and passwords) and personally identifiable information (*e.g.*, names and addresses), which LLMs might inadvertently reproduce during inference [20], [27].
- *Fairness* $\circ$ *Fairness.* Machine learning algorithms are frequently reported to exhibit fairness issues related to protected demographic attributes [28]. LLMs for code, which are trained on historical code data, have been shown to perpetuate these biases, generating code/algorithms that may discriminate against certain demographic groups [29], [30].

✏ ***Challenge 1.*** The complex working mechanisms of LLMs make it challenging to construct precise and fine-grained connections between model trustworthiness indicators and training data quality indicators. This challenge also motivates our call to action for the synergistic auditing framework that integrates both training and evaluation processes.

B. Training Data Inspection

Based on these indicators, DATATRUST automatically inspects the training data of LLMs for code to identify potential quality issues that could impact the model trustworthiness.

Integrating Diverse Scanning Tools. We can integrate a range of scanning and analysis tools to conduct targeted data inspections. These include code element extraction and semantics annotation for profiling data diversity, static vulnerability scanning for assessing security, software composition analysis (SCA) for tracking library and API versions, clone detection for analyzing license violations and establishing traceability, and regular expression generation for detecting privacy-sensitive information, fairness analysis, among others.

✏ ***Challenge 2.*** Scalability is a challenge when scanning large training corpora, which can contain millions of files (*e.g.*, 603 million in DeepSeek-Coder's corpus [3], [4]), leading to resource and time constraints. A practical solution is to sample a portion of the data for initial inspection, providing an approximate quality assessment.

LLM-based Inspection Agents. The results generated by existing scanning tools are often not directly usable as measurable quality indicators and need to be refined and aggregated. LLM-based agents are well-suited to handle this task. For example, an agent that explores the training corpus and understands code semantics through static analysis (*e.g.*, identifier extraction) is crucial for profiling data diversity.

Another agent is required to count deprecated libraries or APIs by cross-referencing online API documentation, based on library and API version tracing results from SCA tools.

✏ ***Challenge 3.*** Existing scanning tools have varied prerequisites, input/output formats, and configuration options, making integration challenging but offering innovation opportunities. To enhance integration, future efforts could focus on domain-specific languages (DSLs) for agent and tool communication, on-demand intelligent configuration selection for more flexible inspections, and aggregating results from different tools for more reliable decisions.

C. Evaluation Data Synthesis

To audit the trustworthiness of LLMs for code across various indicators, DATATRUST automatically synthesizes evaluation data and benchmarks by referencing a wide range of evolving resources.

Aggregating Diverse Crowd Resources. In addition to existing evaluations that rely on (semi-)manually curated benchmarks [23], DATATRUST dynamically generates comprehensive, up-to-date evaluation data by leveraging diverse real-time sources. These resources include open-source software platforms, API documentation, package management tools, developer Q&A forums, vulnerability databases, search engines, and social media. By gathering insightful information related to coding and software development, DATATRUST can inspire concrete test cases and deliver a more adaptable trustworthiness auditing. For example, it retrieves domain-specific coding task descriptions from GitHub to evaluate the domain robustness of the LLM under test (LLMUT). For security, recent malware and vulnerability reports are sourced from package managers, vulnerability databases, and social media. Additionally, discussions on other concerns across these resources inform evaluations of specific dimensions.

LLM-based Synthesis Agents. Diverse resources must be processed into *executable* test cases, which DATATRUST achieves using LLM-based agents. Each agent handles data synthesis for a specific evaluation dimension, with modules for automated data fetching (*e.g.*, vulnerability disclosures), test case generation (*e.g.*, probing prompts and oracles), and result inspection. For example, in deprecated API usage evaluation, the agent fetches release notes (*e.g.*, from Libraries.io [31]) to identify deprecated APIs, retrieves related code snippets from sources like GitHub, and generates prompts to test whether the LLM under test still uses the deprecated APIs.

✏ ***Challenge 4.*** Crowd resources, like informal text from Stack Overflow, are heterogeneous and challenging to integrate. This opens research opportunities, such as creating unified intermediate representations (IRs), developing adaptive crawling and parsing methods, and refining information fusion and conflict resolution. These tasks could leverage LLMs for web exploration, code generation (*e.g.*, HTML parsing code), and information summarization.

D. Root Cause Attribution

The trustworthiness issues identified during evaluation are often connected to specific instances within the training data. We use root cause attribution to link trustworthiness issues identified during evaluation to problematic instances in the training data. This is crucial for systematic trustworthiness auditing and developing trustworthy LLMs for code because it (i) refines the connections between model trustworthiness and training data quality, (ii) addresses incomplete training data inspections by identifying previously unknown problematic instances, and (iii) provides clearer insights into which training instances significantly impact the model's trustworthiness when combined with direct inspection results.

Adopting Diverse Attribution Methods. Various instance-level attribution methods exist to assess the correlation between evaluation instances and training instances. These methods include influence functions [32], similarity search [33], gradient-based analysis [34], attention analysis [35], each with its own strengths and weaknesses, and some may overlap in functionality. In addition to instance-level attribution, the identified issues can also facilitate pattern-level attribution, enabling the identification of specific categories of unknown problematic training data. For each identified issues, we analyze the common patterns contributing to the issues and translate them into specific, lightweight inspection rules (*e.g.*, CodeQL [36] queries for particular types of vulnerabilities). This also acts as an incremental inspection mechanism, addressing scalability limitations in training data inspection.

LLM-based Attribution Agents. To leverage the advantages of these diverse methods, we employ LLM-based agents to perform comprehensive analyses based on the outputs of these methods, ultimately making informed decisions through the LLM's understanding and summarization capabilities for both code and natural language. For instance, if a code security issue is detected during evaluation (such as the recurrence of a newly disclosed vulnerability in LLMUT-generated code), an attribution agent for code security first utilizes existing instance-level attribution methods to identify training instances (such as code snippets or functions) that may have contributed to the issue. Subsequently, the agent determines the final instances related to the issue by comparing the code patterns with the identified vulnerability patterns, leveraging its code understanding capabilities.

✏ ***Challenge 5.*** Although instance-level attribution methods like influence functions and gradient-based analysis have shown promise in other domains [33], [37], their performance in code-related data analysis remains untested. This creates risks when applying them for root cause attribution in LLMs for code, but also presents opportunities. Additionally, effectively combining instance-level analysis with pattern-level attribution for incremental training data inspection is a challenge. A mechanism is needed to ensure the accuracy of patterns extracted from evaluation issues, guiding more reliable pattern-level attribution.

E. Data-Engined Trustworthiness Arena

Based on the framework outlined above, DATATRUST can also provide a data-driven trustworthiness arena inspired by the Chatbot Arena [38]. This arena features two leaderboards: one

for LLMs for code and another for training data. It continuously maintains and updates these leaderboards by applying the three data-centric processes to various LLMs for code and their corresponding training datasets when applicable. For LLMs with non-open-sourced training data, DATATRUST can still rank them on the LLMs for code leaderboard using evaluation data synthesis alone.

Engaging Crowd User Interactions. Additionally, DATATRUST provides interfaces for community contributions like Chatbot Arena, related to test case creation, oracle validation, and result confirmation. The key to this initiative is designing suitable interaction paradigms—such as engaging mini-games or daily coding tasks—to minimize participant difficulties and reduce manual efforts for users. To facilitate this, LLM-based agents are employed to offer guidance and assistance for crafting and manipulating test inputs, as well as to automatically convert human-involved tasks into user-friendly or user-transparently information formats. For instance, if the data synthesis agent for code security generates a test input and identifies a recurring vulnerability in the code produced by the LLMUT, an assistance agent in the arena can gather additional relevant information about this vulnerability from diverse resources and present it as a concise, readable checklist for community contributors.

✏ ***Challenge 6.*** Unlike the Chatbot Arena, where tasks are simple for general users, our trustworthiness arena faces a higher participation threshold due to the complexity of certain dimensions. To address this, we need to design more accessible interaction paradigm by simplifying or restructuring the tasks. For example, instead of relying solely on yes/no binary annotations for potentially vulnerable code, we can provide a checklist of low-level safeguard operations (*e.g.*, index range validation) and ask users to verify each item.

III. RELATED WORKS

Trustworthiness issues in general LLMs have garnered significant attention from both academia and industry. Existing research has introduced various principles and dimensions of trustworthiness and benchmarked several mainstream LLMs [16], [18], [39]. These dimensions often include *safety/security*, *fairness*, *robustness*, *privacy*, *machine ethics*, *transparency*, *accountability*, and *regulations and laws*. Since LLMs for code are frequently built by fine-tuning general LLMs or by training on both text and code corpora, they inherently inherit these broader trustworthiness concerns. For instance, LLMs for code may be vulnerable to adversarial attacks, which can undermine their robustness [24].

Within the software engineering community, trustworthiness in LLMs for code is also becoming an increasingly important issue. Lo [40] has called for trustworthy and synergistic AI for Software Engineering (AI4SE), offering a systematic overview of many open challenges and opportunities. Yang *et al.* [22] revisit the dimensions of trustworthiness in LLMs for code, covering aspects such as *robustness*, *security*, *privacy*, *explainability*, *efficiency*, and *usability*, and suggest initial enhancement principles focusing on training data. Spiess *et al.* [41] have introduced correctness calibration techniques for LLMs for code to improve the trustworthiness of their outputs. Additionally, other studies address specific trustworthiness dimensions and application areas, such as trustworthy program synthesis [42], backdoor-trigger taxonomy [43], and trustworthy code summarization [44].

These works provide a strong foundation for realizing our unified auditing framework of DATATRUST. Our primary objective is to align understanding, standardize auditing processes, and enhance the transparency of the trustworthiness of LLMs and their data, ultimately benefiting a broad spectrum of stakeholders.

IV. SUMMARY

In this paper, we propose a vision for a unified trustworthiness auditing framework, DATATRUST, which adopts a data-centric approach that synergistically emphasizes the relationship between training and evaluation data. DATATRUST seeks to connect model trustworthiness indicators in evaluation with data quality indicators in training. It autonomously inspects training data, evaluates model trustworthiness using synthesized data, and attributes potential causes from specific evaluation data to their corresponding training data while refining indicator connections. DATATRUST can achieve *extensibility* through the integration of diverse tools and resources, as well as *evolvability* by incorporating real-world information and knowledge. Nevertheless, open challenges remain, particularly concerning tool integration, resource aggregation, method adoption, and interaction paradigms, all of which present valuable opportunities for future research.

V. FUTURE PLANS

We propose several actionable plans with achievable timelines. First, we will focus on a subset of DATATRUST to initiate exploration and implementation, starting with a prototype. This subset will include a few key indicators, such as security and copyright, applied to open-source LLMs for code (*e.g.*, StarCoder) alongside open-source training data, and a well-established application task such as code generation. Second, we will implement the three data-centric auditing processes, using the selected indicators, LLMs, and application task, designing approaches to address the challenges identified earlier. Third, we will launch an initial arena platform to engage participants, providing continuously updated leaderboards. Finally, we will refine the methodology and complete DATATRUST through collaboration and feedback from the broader community.

ACKNOWLEDGEMENT

This research is supported by the National Research Foundation, Singapore, the Cyber Security Agency under its National Cybersecurity R&D Programme (NCRP25-P04-TAICeN), and DSO National Laboratories under the AI Singapore Programme (AISG2-GC-2023-008). It is also supported by the National Research Foundation, Prime Minister's Office, Singapore under the Campus for Research Excellence and Technological Enterprise (CREATE) programme.

REFERENCES

[1] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[2] B. Roziere, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, R. Sauvestre, T. Remez *et al.*, "Code llama: Open foundation models for code," *arXiv preprint arXiv:2308.12950*, 2023.

[3] D. Guo, Q. Zhu, D. Yang, Z. Xie, K. Dong, W. Zhang, G. Chen, X. Bi, Y. Wu, Y. Li *et al.*, "Deepseek-coder: When the large language model meets programming–the rise of code intelligence," *arXiv preprint arXiv:2401.14196*, 2024.

[4] Q. Zhu, D. Guo, Z. Shao, D. Yang, P. Wang, R. Xu, Y. Wu, Y. Li, H. Gao, S. Ma *et al.*, "Deepseek-coder-v2: Breaking the barrier of closed-source models in code intelligence," *arXiv preprint arXiv:2406.11931*, 2024.

[5] R. Li, L. B. Allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. Li, J. Chim *et al.*, "Starcoder: may the source be with you!" *arXiv preprint arXiv:2305.06161*, 2023.

[6] J. Liu, K. Wang, Y. Chen, X. Peng, Z. Chen, L. Zhang, and Y. Lou, "Large language model-based agents for software engineering: A survey," *arXiv preprint arXiv:2409.02977*, 2024.

[7] Y. Huang, W. Zhong, E. Shi, M. Yang, J. Chen, H. Li, Y. Ma, Q. Wang, Z. Zheng, and Y. Wang, "Agents in software engineering: Survey, landscape, and vision," *arXiv preprint arXiv:2409.09030*, 2024.

[8] D. Jin, Z. Jin, X. Chen, and C. Wang, "Mare: Multi-agents collaboration framework for requirements engineering," *arXiv preprint arXiv:2405.03256*, 2024.

[9] K. Ronanki, B. Cabrero-Daniel, J. Horkoff, and C. Berger, "Requirements engineering using generative ai: Prompts and prompting patterns," in *Generative AI for Effective Software Development*. Springer, 2024, pp. 109–127.

[10] H. Jin, L. Huang, H. Cai, J. Yan, B. Li, and H. Chen, "From llms to llm-based agents for software engineering: A survey of current, challenges and future," *arXiv preprint arXiv:2408.02479*, 2024.

[11] J. Liu, C. S. Xia, Y. Wang, and L. Zhang, "Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[12] K. Tamberg and H. Bahsi, "Harnessing large language models for software vulnerability detection: A comprehensive benchmarking study," *arXiv preprint arXiv:2405.15614*, 2024.

[13] Z. Ma, A. R. Chen, D. J. Kim, T.-H. Chen, and S. Wang, "Llmparser: An exploratory study on using large language models for log parsing," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 2024, pp. 1–13.

[14] B. Desai and K. Patil, "Reinforcement learning-based load balancing with large language models and edge intelligence for dynamic cloud environments," *Journal of Innovative Technologies*, vol. 6, no. 1, pp. 1–13, 2023.

[15] Github copilot. [Online]. Available: https://github.com/features/copilot

[16] B. Wang, W. Chen, H. Pei, C. Xie, M. Kang, C. Zhang, C. Xu, Z. Xiong, R. Dutta, R. Schaeffer *et al.*, "Decodingtrust: A comprehensive assessment of trustworthiness in gpt models." in *NeurIPS*, 2023.

[17] Y. Zeng, K. Klyman, A. Zhou, Y. Yang, M. Pan, R. Jia, D. Song, P. Liang, and B. Li, "Ai risk categorization decoded (air 2024): From government regulations to corporate policies," *arXiv preprint arXiv:2406.17864*, 2024.

[18] L. Sun, Y. Huang, H. Wang, S. Wu, Q. Zhang, C. Gao, Y. Huang, W. Lyu, Y. Zhang, X. Li *et al.*, "Trustllm: Trustworthiness in large language models," *arXiv preprint arXiv:2401.05561*, 2024.

[19] H. Pearce, B. Ahmad, B. Tan, B. Dolan-Gavitt, and R. Karri, "Asleep at the keyboard? assessing the security of github copilot's code contributions," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 754–768.

[20] Y. Huang, Y. Li, W. Wu, J. Zhang, and M. R. Lyu, "Your code secret belongs to me: Neural code completion tools can memorize hard-coded credentials," *Proceedings of the ACM on Software Engineering*, vol. 1, no. FSE, pp. 2515–2537, 2024.

[21] C. Wang, K. Huang, J. Zhang, Y. Feng, L. Zhang, Y. Liu, and X. Peng, "How and why llms use deprecated apis in code completion? an empirical study," *arXiv preprint arXiv:2406.09834*, 2024.

[22] Z. Yang, Z. Sun, T. Z. Yue, P. Devanbu, and D. Lo, "Robustness, security, privacy, explainability, efficiency, and usability of large language models for code," *arXiv preprint arXiv:2403.07506*, 2024.

[23] T. Y. Zhuo, M. C. Vu, J. Chim, H. Hu, W. Yu, Widyasari *et al.*, "Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions," *arXiv preprint arXiv:2406.15877*, 2024.

[24] Z. Yang, J. Shi, J. He, and D. Lo, "Natural attack for pre-trained models of code," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 1482–1493.

[25] N. Rahman and E. Santacana, "Beyond fair use: Legal risk evaluation for training llms on copyrighted text," in *ICML Workshop on Generative AI and Law*, 2023.

[26] T. Li, Q. Liu, T. Pang, C. Du, Q. Guo, Y. Liu, and M. Lin, "Purifying large language models by ensembling a small language model," *arXiv preprint arXiv:2402.14845*, 2024.

[27] Y. Nie, C. Wang, K. Wang, G. Xu, G. Xu, and H. Wang, "Decoding secret memorization in code llms through token-level characterization," *arXiv preprint arXiv:2410.08858*, 2024.

[28] Z. Chen, J. M. Zhang, M. Hort, M. Harman, and F. Sarro, "Fairness testing: A comprehensive survey and analysis of trends," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 5, pp. 1–59, 2024.

[29] D. Huang, Q. Bu, J. Zhang, X. Xie, J. Chen, and H. Cui, "Bias assessment and mitigation in llm-based code generation," *arXiv preprint arXiv:2309.14345*, 2023.

[30] Y. Liu, X. Chen, Y. Gao, Z. Su, F. Zhang, D. Zan, J.-G. Lou, P.-Y. Chen, and T.-Y. Ho, "Uncovering and quantifying social biases in code generation," *Advances in Neural Information Processing Systems*, vol. 36, pp. 2368–2380, 2023.

[31] Libraries.io - security & maintenance data for open source software. [Online]. Available: https://libraries.io/

[32] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *International conference on machine learning*. PMLR, 2017, pp. 1885–1894.

[33] P. Pezeshkpour, S. Jain, B. C. Wallace, and S. Singh, "An empirical comparison of instance attribution methods for nlp," *arXiv preprint arXiv:2104.04128*, 2021.

[34] G. Pruthi, F. Liu, S. Kale, and M. Sundararajan, "Estimating training data influence by tracing gradient descent," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19 920–19 930, 2020.

[35] Y. Hao, L. Dong, F. Wei, and K. Xu, "Self-attention attribution: Interpreting information interactions inside transformer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 14, 2021, pp. 12 963–12 971.

[36] Codeql - industry-leading semantic code analysis engine for discovering vulnerabilities. [Online]. Available: https://codeql.github.com/

[37] Z. Dai and D. K. Gifford, "Training data attribution for diffusion models," *arXiv preprint arXiv:2306.02174*, 2023.

[38] W.-L. Chiang, L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, H. Zhang, B. Zhu, M. Jordan, J. E. Gonzalez *et al.*, "Chatbot arena: An open platform for evaluating llms by human preference," *arXiv preprint arXiv:2403.04132*, 2024.

[39] J. Hong, J. Duan, C. Zhang, Z. Li, C. Xie, K. Lieberman, J. Diffenderfer, B. Bartoldson, A. Jaiswal, K. Xu *et al.*, "Decoding compressed trust: Scrutinizing the trustworthiness of efficient llms under compression," *arXiv preprint arXiv:2403.15447*, 2024.

[40] D. Lo, "Trustworthy and synergistic artificial intelligence for software engineering: Vision and roadmaps," in *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*. IEEE, 2023, pp. 69–85.

[41] C. Spiess, D. Gros, K. S. Pai, M. Pradel, M. R. I. Rabin, A. Alipour, S. Jha, P. Devanbu, and T. Ahmed, "Calibration and correctness of language models for code," *arXiv preprint arXiv:2402.02047*, 2024.

[42] D. Key, W.-D. Li, and K. Ellis, "Toward trustworthy neural program synthesis," *arXiv preprint arXiv:2210.00848*, 2022.

[43] A. Hussain, M. R. I. Rabin, T. Ahmed, B. Xu, P. Devanbu, and M. A. Alipour, "Trojans in large language models of code: A critical review through a trigger-based taxonomy," *arXiv preprint arXiv:2405.02828*, 2024.

[44] Y. Virk, P. Devanbu, and T. Ahmed, "Enhancing trust in llm-generated code summaries with calibrated confidence scores," *arXiv preprint arXiv:2404.19318*, 2024.