Poster: Intellectual Property Infringement Assessment of Code Language Models

Zhiyuan Yu*, Yuhao Wu*, Ning Zhang*, Chenguang Wang*, Yevgeniy Vorobeychik*, Chaowei Xiao†
*Washington University in St. Louis
†Arizona State University
{yu.zhiyuan, yuhao.wu, zhang.ning, chenguangwang, yvorobeychik}@wustl.edu, xiaocw@asu.edu

***Abstract*—Recent advances in large language models (LMs) have facilitated their ability to synthesize programming code. However, they have also raised concerns about intellectual property (IP) rights violations. Despite the significance of this issue, it has been relatively less explored. In this paper, we aim to bridge the gap by presenting CODEIPPROMPT[1], a platform for automatic evaluation of the extent to which code language models may reproduce licensed programs. It comprises two key components: prompts constructed from a licensed code database to elicit LMs to generate IP-violating code, and a measurement tool to evaluate the extent of IP violation of code LMs. We conducted an extensive evaluation of existing open-source code LMs and commercial products and revealed the prevalence of IP violations in all these models. Our study provides a testbed for evaluating the IP violation issues of code generation platforms and stresses the need for a better mitigation strategy.**

***Index Terms*—Intellectual Property, Code Language Model**

I. INTRODUCTION

The recent advancements in large language models such as GPT-4 have brought about revolutionary changes in the field of natural language processing. These models have demonstrated the ability to generate content that closely resembles human-created materials, leading to the emergence of a new form of content known as Artificial Intelligence Generated Content (AIGC). An important application of AIGC is code generation, however, the use of AI-generated code also raises legal and ethical concerns. A key issue is the potential violation of IP rights. As code generative models are trained on open-source repositories, people found that they can produce programs that are similar or even identical to existing ones without compliance with corresponding licenses [1], [2]. As such, it is likely more concerning that users may be at risk of inadvertently violating the IP rights of original works.

In this work, we introduce CODEIPPROMPT [3], the first automated testing framework to evaluate the extent to which code language models will generate potentially IP-violating code, enabled by two key functionalities: extracting function signature and accompanying comments from licensed code to construct prompts, and measuring the extent of IP violation with code plagiarism similarity scores.

To conduct a comprehensive evaluation, we collected over 4M real-world licensed repositories to evaluate eight models across five programming languages. The results showed that the issue is prevalent for these models, as most of them are capable of generating code strongly resembling portions of licensed software within 50 prompted code generations. Our findings highlight the challenges in mitigating this issue and the pressing need to reconsider the data used for training.

[1]Project website: https://sites.google.com/view/codeipprompt/.

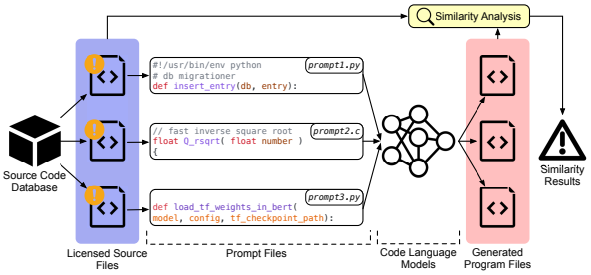

Fig. 1: Workflow of CODEIPPROMPT.

II. METHODOLOGY

The core design of CODEIPPROMPT is depicted in Fig. 1. To create a comprehensive dataset for evaluation, we compiled a collection of licensed code repositories pbtained from GitHub. From the sampled licensed code, we extracted function signatures and accompanying comments to serve as prompts, and the resulting generated code is subsequently compared to the original program to calculate similarity scores. In this work, we only focus on IP infringement of permissive and copyleft licenses.

A. Constructing Prompts from Licensed Code

Real-world Data as Foundation of Prompts. We used a three-step data collection process. First, we used the GitHub REST API to gather information from repositories with varying licenses and programming languages. Second, we implemented a parser to examine metadata, download repositories, and categorize entries in a metadata database. Lastly, we compressed and uploaded resources to cloud storage, post-processed to filter and extract target programs. As a result, programs from 4,075,553 open-source repositories hosted on GitHub with 34 mainstream licenses are collected.

Constructing Prompts from Data. As we focused on functions or classes as units of evaluation, each prompt is limited to a single function signature and its accompanying comment. To handle the diverse syntactic structures in various programming languages and avoid disruption from irrelevant components

TABLE I: Data statistics of the constructed prompts.

# Prompts	**Total**	**Permissive**	**Weak Copyleft**		**Strong Copyleft**	
		478.7K	190.1K		280.4K	
	949.3K	**C**	**C++**	**C#**	**Python**	**Java**
		24.9K	24.9K	126.4K	127.8K	591.6K
# Tokens	**Avg.**	**Permissive**	**Weak Copyleft**		**Strong Copyleft**	
		$13.2_{7.3}$	$13.4_{6.8}$		$13.1_{7.3}$	
	$13.2_{7.2}$	**C**	**C++**	**C#**	**Python**	**Java**
		$19.5_{10.6}$	$17.0_{10.1}$	$14.5_{8.3}$	$12.2_{6.6}$	$12.7_{6.5}$

(e.g., variable names or comments containing keywords), we compiled various regular expressions to identify elements such as comments, functions, and classes. With individual lines of code as units, these regular expressions were used to match and extract target code snippets while preserving the original code context. We sampled source files representing copyleft and permissively licensed code and derived prompts across five programming languages (i.e. Python, C, C++, C#, Java) with varying lengths. After removing empty prompts and those containing special characters, we obtained around 950K prompts with the statistics summarized in Table I.

B. Benckmarking IP Violation with Similarity Score

Similarity Score Calculation. To identify IP violations, we adapted and incorporated JPlag [4] and Dolos [5], two of the most widely recognized code plagiarism detection tools that have been utilized as expert witness evidence in lawsuits. Both tools take a set of programs and compare them pairwise, producing a similarity score ranging from 0 to 1 for each pair. JPlag uses lexical analysis and string tiling to compare programs, while Dolos converts them into abstract syntax trees (ASTs) and calculates similarity based on the coverage of unique AST fingerprints. Therefore, their generated similarity scores cover different types of code plagiarism, and we took the maximum between the two as the result. In this study, we consider a similarity score > 0.5 as potential plagiarism. It serves as a quantitative indicator within the framework and can be adjusted based on individual cases.

Benchmark Metrics for IP Infringement. For each sampled prompt, ten code generations was conducted for each code language model. The maximum score was utilized as the similarity score for the respective prompt. We then performed bootstrapping by sampling $n = 50$ code generations 1K times. Two metrics were used to characterize the models: (1) the *Expected Maximum* (EM) similarity calculated by the mean of the maximum scores from 1K bootstrapped samples; and (2) the *Empirical Probability* (EP) measured as the mean probability of generating code with score > 0.5 at least once in the samples. In the present context, the EM score measures the worst-case scenario in which generated code is highly similar to existing code, while the EP reflects the frequency at which the model generates potentially IP-violating code.

III. Evaluation

Using CodeIPPrompt, we evaluated the real-world risks of generating IP-violating code on 8 code generation language models, comprising 6 open-source models and 2 commercial products. They cover a wide range of architectures including GPT-3 (i.e., Copilot and Codex), GPT-2 (i.e., CodeParrot), encoder-decoder(i.e., CodeRL), autoregressive transformer (i.e., CodeGen), etc. The names of the models under these frameworks are Copilot, code-davinci-002, CodeParrot-110M and CodeParrot-1.5B, CodeT5-large and CodeT5-large-ntp-py, CodeGen-350M and CodeGen-2.7B. In the study, we followed the original settings of code language models and employed nucleus sampling with top-p where $p = 0.95$.

TABLE II: Evaluation results with prompts.

	Copilot	Codex	CodeT5-large	CodeT5-large-ntp-py
EM	$0.62_{0.25}$	$0.64_{0.20}$	$0.11_{0.21}$	$0.92_{0.12}$
EP	0.64	0.75	0.04	0.99
	CodeGen-350M	**CodeGen-2.7B**	**CodeParrot-110M**	**CodeParrot-1.5B**
EM	$0.73_{0.15}$	$0.81_{0.22}$	$0.55_{0.23}$	$0.55_{0.22}$
EP	0.94	0.88	0.53	0.52

The results are presented in Table II. We observed that most models can generate potential IP-violating code within 50 generations with a relatively high probability greater than 0.5. We additionally evaluated ChatGPT built on GPT-3.5, with the constructed prompts to ask them to act as code generation models. It achieves EM of $0.66_{0.17}$ and EP of 0.67. This performance is similar to the Copilot and Codex models. We speculate the potential reason is that these models are trained on similar datasets. The CodeT5-large model exhibits the lowest similarity scores. Manual qualitative analysis revealed that this was because many generated code snippets were not meaningful. In contrast, the CodeT5-large-ntp-py model, which was fine-tuned on Python programs with additional data, demonstrated significantly higher EM and EP.

IV. Conclusion

We present CodeIPPrompt, a generalizable platform for evaluating the extent to which language models can reproduce learned code, which can result in potential intellectual property infringement. Using the framework, we analyzed multiple state-of-the-art models and commercial products, and investigated potential strategies for addressing the issue. With this work, we aim to shed light on the landscape of IP protection in generated code by language models.

Acknowledgments

This work was partially supported by the NSF (CNS-1916926, CNS-2238635), ARO (W911NF2010141), and Intel.

References

[1] T. Davis, "Copilot emits large chunks of my copyrighted code." https://twitter.com/docsparse/status/1581461734665367554, October 2022.

[2] S. Karpinski, "Copilot autocompletes the fast inverse square root implementation from quake iii." https://twitter.com/stefankarpinski/status/1410971061181681674, July 2021.

[3] Z. Yu *et al.*, "Codeipprompt: Intellectual property infringement assessment of code language models," in *International conference on machine learning*, PMLR, 2023.

[4] L. Prechelt *et al.*, *JPlag: Finding plagiarisms among a set of programs*.

[5] R. Maertens *et al.*, "Dolos: Language-agnostic plagiarism detection in source code," *J. Comput. Assist. Learn.*, 2022.