Poster: Identifying Copyright Code in Training Corpus of Large Language Model

Lamia Hasan Rodoshi
University of Texas at Arlington
lxr6158@mavs.uta.edu

Faiza Tafannum
University of Texas at Arlington
fxt8308@mavs.uta.edu

Faysal Hossain Shezan
University of Texas at Arlington
faysal.shezan@uta.edu

***Abstract*—Large Language Models (LLMs) raise concerns about copyright infringement during training. They absorb vast amounts of data, potentially including copyrighted works. This paper proposes an interactive tool to investigate copyrighted content within LLM training data. Our tool uses prompts to reveal LLMs' knowledge of copyrighted material, focusing on code. We draw inspiration from interactive proof systems to assess whether an LLM can reproduce specific code. Our method successfully identified structural similarities between ChatGPT responses and GitHub code (93 out of 130 files). This highlights the tool's potential for detecting copyrighted materials in LLM training data and underscores the ethical and legal complexities involved.**

1. Introduction

Investigating copyright use in Large Language Model (LLM) training is critical because large language models are essentially sponges, soaking up massive amounts of data to learn. This data, while often publicly available, can include copyrighted creative works like books, articles, or code snippets. If LLMs cannot distinguish copyrighted material, they might inadvertently learn from it during training. This raises concerns about copyright infringement, potentially leading to legal issues and financial repercussions for developers. Additionally, it creates an ethical gray area regarding how much copyrighted material an LLM can absorb without infringing on creativity or originality. Existing works face challenges in identifying copyright materials when LLM works in a black-box environment.

This paper proposes an interactive tool for probing LLMs to detect potential use of copyrighted content in their training data. The tool facilitates communication with the LLM through carefully crafted prompts designed to exploit behavioral patterns that might indicate exposure to copyrighted material. Specifically, we investigate whether the LLM can generate outputs that demonstrably match the copyrighted content. As a test case, we focus on identifying copyright protected code within the LLM's training data.

In this work, we focus on eidetic memorization (also knowns as photographic memory) [1] of the large language model. Inspired from the interactive proof systems, we define a model $\mathcal{M}_\theta$ knows a code $\mathcal{C}$, if $\mathcal{C}$ can be extracted from $\mathcal{M}_\theta$ by interactive with the model. During the interaction, $\mathcal{M}_\theta$ can provide either full code snippet which exactly match with the copyright code, or can provide partial copyright code.

Our approach identifies structural similarities between $\mathcal{R}$ and $\mathcal{G}$[1] to assist in potential copyright issue detection. We prioritize syntactic resemblance in coding style and organization, recognizing that functionality alone does not imply infringement. Our evaluation on 130 files across 18 projects, demonstrates the effectiveness of our method. By isolating matching code segments, we are able to identify 93 files (out of 130) being similar to the codes found in Github. This research highlights the potential of identifying copyright materials in training corpus of LLMs.

2. Background

We collect data from GitHub repositories with an emphasis on projects adopting non-commercial licenses. License selection is critical for developers and users, dictating the permissible use, modification, and redistribution of projects. Creative Commons provides various licensing options, including CC BY-NC (Creative Commons Attribution-NonCommercial) and CC BY-NC-SA (Creative Commons Attribution-NonCommercial-ShareAlike). These licenses permit distribution, remixing, and adaptation of work for non-commercial purposes with mandatory attribution. The 'ShareAlike' component (CC BY-NC-SA) further mandates that derivative works retain the same non-commercial licensing. These licenses suit those seeking open sharing for educational or non-profit use while preventing commercial exploitation. For this project, we consider the following restrictive licenses during data collection: CC-BY-NC, CC BY-NC-SA, CC BY-NC-ND, and the Educational Community License. Given that ChatGPT (V-4) is intended for commercial use, it should refrain from processing data protected by these restrictive licenses.

3. Motivating Scenario

'Resources-Tortuino' is a popular project found in Github(https://github.com/PaulDance/Ressources-Tortuino), written in javascript language. This project is protected by CC-BY-NC-ND. This license restricts third-parties from changing the original content or using this for commercial purpose. We randomly select one code snippet from this

1. We refer ChatGPT response to $\mathcal{R}$ and copyright protected code to $\mathcal{G}$.

project as illustrated in Listing 1. The main objective of this project is to generate and initialize a dynamic menu structure on a web page which also includes optimize search functionality. Now, we aim to replicate this behavior using LLM (i.e., ChatGPT). To achieve that, we provide a series of prompts based on the response we get in the previous round, including- functionality, API/library usage, code executability etc.

ChatGPT generated the code snippet shown in Listing 2 after providing the prompts. Comparing Listings 1 and 2, we can identify many functional similarities, such as the HTML creation logic and how child nodes connect to parent nodes. However, the ChatGPT response has some calculation differences compared to the GitHub file, which could be addressed with more detailed prompts or instructions.

```
function makeTree(data, relPath){
 var result = '';
 if ('children' in data){
  result += '<ul>';
  for (var i in data.children){
   result += '<li><a href="'
          + relPath
          + data.children[i].url
          + '">'
          + data.children[i].name
          + '</a>';
   result += makeTree(
           data.children[i],
           relPath);
   result += '</li>';
  }
  result += '</ul>';
 }
 return result;
}
```

Listing 1: Github Code

```
function makeTree(data, basePath) {
 var result = '';
 if (data && data.children) {
  result += '<ul>';
  for (var i=0;i<data.children.length;i++){
   var child = data.children[i];
   var childPath = escapeHtml(
                   basePath + child.path
                   );
   result += '<li><a href="' + childPath
          +'">' + escapeHtml(child.name)
          + '</a>';
   result += makeTree(child, basePath);
   result += '</li>';
  }
  result += '</ul>';
 }
 return result;
}
```

Listing 2: ChatGPT Response

Listing 3: Motivating example for finding similar generated code

4. Dataset

In Github website, we use specific keywords (such as,. Creative Commons) and tags to search projects (following bredth-first strategy). In total, we collect 124 projects (53 projects contain 462 Python, 33 projects contain 545 JavaScript, 12 projects contain 381 Java, 8 projects contain 24 C#, 20 projects contain 130 C/C++ and 1 project contains 8 ruby files) that are copyright protected. For our evaluation, we randomly select 130 files from those 18 projects. In future, we will perform measurement study on the full dataset.

5. System Design

We start with extracting the important tokens (e.g., class name, function name) using rule-based approach from $\mathcal{G}$. However, the coverage of rules limits the performance of this approach. Also, this approach requires manual effort when we work with a new programming language. That's why we leverage ChatGPT to extract all of the important tokens from $\mathcal{G}$. From our investigation, we find that if we provide more contextual information to ChatGPT it helps them to provide us better result. That's why, we use those tokens to create a summary on $\mathcal{G}$ using ChatGPT.

Once the summarization is done, we create a new session with ChatGPT to automatically generate prompts. For this, we utilize both important tokens and summary (from previous step) of $\mathcal{G}$. We rely on these prompts because these sets cover every part of $\mathcal{G}$. So, these can instruct ChatGPT to generate such kind of code with particular logics that $\mathcal{G}$ focus on. Usually, the number of generated prompts ranges between 15-30.

#Total		#C		#C (w/ isolation)	
#Project	#Files	#Project	#Files	#Project	#Files
18	130	18	15	18	93

TABLE 1: Performance evaluation of our tool. C = Copyright Content.

Next, we evaluate the executability and compilability of $\mathcal{R}$. We implement the 'subprocess' module to establish a parent-child relationship between processes. In this context, the extracted code snippet from ChatGPT is passed through this module, functioning as a child process. The purpose is to assess the executability of the code, determining whether it can be run successfully or not. If the generated code is not runnable, our tool prompts ChatGPT to provide a corrected version based on the encountered errors. Then again, we check that the updated response code is runnable or not. We continue this process until we find an executable code. Once it gets an executable code it goes into the next step which is similarity checking.

Once we get response for each of the prompt, we calculate the similarity between $\mathcal{R}$ and $\mathcal{G}$. Like this way, we iterates all the prompts and select the one which achieves the best similarity score. And if the score exceeds the threshold value, then we mark that as potential sample which is used to train ChatGPT. We set the threshold value to 65, based on our analysis on 50 files.

6. Experiment Design and Evaluation

Our primary objective is to identify syntactic similarities between $\mathcal{R}$ and $\mathcal{G}$, regardless of semantic overlap. This focus on structure emphasizes coding style and organization. While similar functionality does not imply copyright infringement, our approach prioritizes the alignment of specific approaches and syntax. Even with functionally identical elements, distinct implementations will be appropriately differentiated.

To evaluate our approach, we assess similarity and completeness. We randomly select 130 files from 18 projects for validation (Table 1). We evaluate various similarity metrics such as Levenshtein Distance, Jaccard Similarity, and Euclidean Distance. Cosine similarity proved most effective, as it captures structural resemblance despite potential renaming of variables or parameters. To improve completeness, we isolate matching code segments, leading to higher accuracy (93/130 files). Without isolation, 15/130 files exhibits the usage of copyright content. Our evaluation demonstrates the value of this approach for precise function summarization in program analysis. Our tool can also be extended to other domains with the necessary modifications (modifying the logic of similarity checking for image to pixel value).

References

[1] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*. 2633–2650.

Poster: Identifying Copyright Code in Training Corpus of Large Language Model

Lamia Hasan Rodoshi
University of Texas at Arlington
lxr6158@mavs.uta.edu

Faiza Tafannum
University of Texas at Arlington
fxt8308@mavs.uta.edu

Faysal Hossain Shezan
University of Texas at Arlington
faysal.shezan@uta.edu

1. Introduction

From Large Language Models' (LLMs) vast amounts of training data, it is possible that,

- LLMs might not differentiate copyrighted content
- They learn from it during training, leading to legal and ethical concerns

Our interactive tool,

- Carefully crafts prompts that are used to detect if the LLM generates outputs matching copyrighted content
- Explores Eidetic memorization [1]
- Defines a model M_θ as knowing a code C if C can be extracted from M_θ through interactive engagement with the model

During the interaction, M_θ can provide either:

- Full code snippet exactly matching with the copyright code
- Or Partial copyright code

In our approach,

- Structural similarities between ChatGPT response and copyright protected code is identified
- Syntactic resemblance in coding style and organization prioritized
- 130 files across 18 projects are evaluated
- 93 files are similar to codes found on GitHub.

2. Background on Copyright License

- Considered restrictive licenses during data collection:
  - CC-BY-NC (Creative Commons Attribution-NonCommercial)
  - CC BY-NC-SA (Creative Commons Attribution-NonCommercial-ShareAlike)
  - CC BY-NC-ND (Creative Commons Attribution NonCommercial NoDerivs)
  - Educational Community License
- These licenses permit distribution, remixing, and adaptation for non-commercial purposes like education or non-profit use with mandatory attribution.
- ChatGPT (V-4) should avoid processing data protected by these licenses as it is intended for commercial use.

3. Motivation

- `**Resources-Tortuino**' is a popular project found in Github (https://github.com/PaulDance/Ressources-Tortuino), written in javascript language, protected by CC-BY-NC-ND license.
- We randomly select one code snippet from this project to check whether LLM (i.e., ChatGPT) replicate the GitHub files content or not.
- Figure 1 and 2 represent higher similarity in the functionalities between response of ChatGPT and GitHub file's code snippet.

```
function makeTree(data, relPath){
    var result = '';
    if ('children' in data){
        result += '<ul>';
        for (var i in data.children){
            result += '<li><a href="'
                    + relPath
                    + data.children[i].url
                    + '">'
                    + data.children[i].name
                    + '</a>';
            result += makeTree(
                    data.children[i],
                     relPath);
            result += '</li>';
        }
        result += '</ul>';
    }
    return result;
}
```

Figure 1: GitHub Code

```
function makeTree(data, basePath) {
    var result = '';
    if (data && data.children) {
        result += '<ul>';
        for (var
i=0;i<data.children.length;i++){
            var child = data.children[i];
            var childPath = escapeHtml(
            basePath + child.path);
            result += '<li><a href="'
                    + childPath +'">'
                    + escapeHtml(child.name)
                    + '</a>';
            result += makeTree(child, basePath);
            result += '</li>';
        }
        result += '</ul>';
    }
    return result;
}
```

Figure 2: ChatGPT Response

4. System Design

We have mentioned the GitHub code as *'G'* and ChatGPT Response as *'R'*.

Figure 3 represents the workflow of our tool where we initially extract key tokens from *G* using a rule-based approach, but it limited rule coverage and manual effort for new languages prompt us to leverage ChatGPT. Providing contextual information to ChatGPT yields better results, aiding in token extraction and subsequent summarization of *G*. We then use these tokens to generate 15-30 prompts covering all aspects of *G*, relying on subprocess for executability and compilability assessment of *R*'s generated code. If the code is not runnable, ChatGPT is prompted for corrections until executable code is obtained. Subsequently, similarity between *R* and *G* is calculated for each prompt, with the best-scoring prompt selected. Samples exceeding a similarity threshold are marked for copyright code in training ChatGPT.

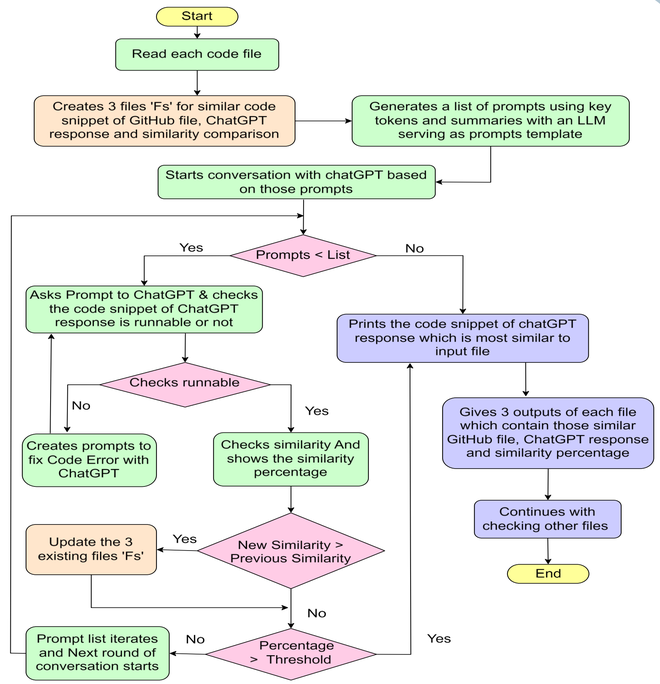

Figure 3: Workflow of Our Tool

5. Experiment Design and Evaluation

- Identify syntactic similarities between *R* and *G* prioritizing coding style and organization over semantic overlap.
- Prioritize aligning syntax and approaches, acknowledging functional similarities despite differing implementations.

Total		#C		#C (w/ isolation)	
Projects	Files	Projects	Files	Projects	Files
18	130	18	15	18	93

TABLE 1: Performance evaluation of our tool. C = Copyright Content.

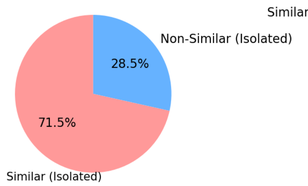

Figure 4: Files % w/ Isolation

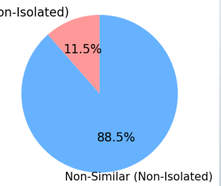

Figure 5: Files % w/o Isolation

- Prior to isolation, our analysis reveal a lower similarity percentage of 11.5% among the 130 total files. However, after isolation, the similarity percentage surges to 71.5%, underscoring the effectiveness of our method for informed assessments of potential copyright concerns.

6. Conclusion

- Introduces an interactive tool for exploring potential copyright infringement in LLM training data, focusing on code.
- Successfully identifies 93 out of 130 files with matching code segments, showcasing the tool's potential for detecting copyrighted materials in LLM training data.

7. Acknowledgement

We want to thank Microsoft Azure for supporting us with Azure credits and the anonymous reviewer for providing constructive feedback.

8. References

[1] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In 30th USENIX Security Symposium (USENIX Security 21).