# ソースコード著作権侵害の過検知抑制に向けたLLMの記憶強度に基づくありふれた表現の識別

松林 勝[1,a)]　岩花 一輝[1]　小山 卓麻[1]　芝原 俊樹[1]　遠田 耕司[1]　千田 成樹[1]　大湊 健一郎[1]

**概要**：大規模言語モデル（LLM）が生成したソースコード（生成コード）の著作権侵害を検知する手法が提案されている．既存手法の多くは，生成コードと GitHub 等で公開されているソースコード（公開コード）の文字列の一致度が高い場合に著作権侵害を検知する．一方，誰が書いても同様になりやすい「ありふれた表現」の公開コードは著作権で保護されない．本研究では，生成コードがありふれた表現の公開コードと一致した場合に，既存手法ではそれを著作権侵害として過検知する可能性があることを実験的に証明する．加えて，この過検知の抑制を目的として，生成コードがありふれた表現か否かを識別する手法を提案する．一般的に，誰が書いても同様になりやすいありふれた表現のソースコードほど，学習データセットに頻出すると考えられる．そこで提案手法では，頻出する学習データほど LLM がより強く記憶するという従来の観測結果に基づいて，生成コードに対する LLM の記憶強度を用いてありふれた表現を識別する．評価では，公開データセットである The Stack のソースコードに対して，4 種類の OpenAI モデルの LLM-as-a-Judge によって，ありふれた表現か否かのラベルを付与した．そのデータセットに提案手法を適用した結果，Area Under the Curve (AUC) が 0.75 の性能でありふれた表現を識別できた．また，既存手法の検知結果に対して提案手法を適用することで，既存手法の検知率の低下を 1% に抑えつつ過検知率を 22%抑制でき，提案手法が過検知抑制に有効であることが示された．

**キーワード**：LLM，記憶強度，ソースコード，著作権侵害，ありふれた表現

# Identifying Commonplace Expression Based on LLM's Memorization Strength to Mitigate Over-Protection of Source Code Copyright

Masaru Matsubayashi[1,a)]　Kazuki Iwahana[1]　Takuma Koyama[1]　Toshiki Shibahara[1]
Kouji Tohda[1]　Masaki Chida[1]　Kenichiro Ominato[1]

***Abstract:*** There are many conventional methods for detecting copyright infringement in Large Language Model (LLM)-generated source code snippets. These methods detect potential infringement when generated code snippets closely match publicly available code snippets (e.g., on GitHub). However, many such snippets consist of "commonplace expressions" that are not protected by copyright. Thus, we empirically demonstrate that these methods can result in over-protection even when generated code snippets match commonplace expressions. In addition, we propose a method to determine whether generated code snippets are commonplace expressions to mitigate such over-protection. Our method utilizes an LLM's memorization strength, based on the assumption that code snippets consisting of commonplace expressions occur more frequently in the training data and are therefore more strongly memorized. For evaluation, we labeled code snippets from The Stack using an LLM-as-a-Judge setup with four OpenAI models. Our proposed method achieved an Area Under the Curve (AUC) of 0.75 for identifying commonplace expressions. Furthermore, our proposed method suppressed the over-detection rate of a conventional method by 22% while limiting the decrease in detection accuracy to only 1%.

***Keywords:*** LLM (Large Language Model), Memorization Strength, Source Code, Copyright Infringement, Commonplace Expression

## 1. はじめに

近年の大規模言語モデル（LLM）の進展，および GitHub Copilot [1] などの開発者を支援するツールの普及により，LLM を活用したソースコードの自動生成（コード生成）がソフトウェア開発の現場に浸透しつつある．Bain & Company の 2024 年の調査 [2] によると，既に開発者の 62%がコード生成をソフトウェア開発に活用している．

コード生成は開発効率の向上などの利便性をもたらす一方で，著作権侵害のリスクも内包する．ソースコードの著作権は，TRIPS 協定 [3] およびベルヌ条約 [4] により世界的に認められている．また，日本の著作権法 [5] の第 10 条第 1 項第 9 号においても，プログラムを著作物として保護することが明記されており，ソースコードもこの範疇に含まれる．したがって，もし LLM が第三者の著作物に類似するソースコードを生成し，開発者がそれを無断で利用した場合には，著作権侵害が成立するリスクが高い．

このリスクを低減するために，コード生成における著作権侵害を検知する手法 [1,6,7] が提案されている．既存手法は，LLM が生成したソースコード（生成コード）と GitHub 等で公開されているソースコード（公開コード）の文字列の一致度が高い場合に，著作権侵害を検知する．

一方，誰が書いても同様になりやすい「ありふれた表現」のソースコードは著作権で保護されない．ありふれた表現のソースコードの一例として，図 1 に示すような，Python で二つの数を加算する関数が挙げられる．この関数は，誰が書いても図 1 と同様の表現になりやすい．このようなありふれた表現の公開コードも多く存在するため，既存手法ではこれらと一致した生成コードを著作権侵害として過検知する可能性がある．仮に，このようなありふれた表現のソースコードまでもが著作権侵害として過検知されると，ソフトウェア開発の妨げになる恐れがある．

そこで本研究では，既存手法がありふれた表現の生成コードを著作権侵害として過検知する可能性があることを実験的に証明する．加えて，この過検知の抑制を目的として，生成コードがありふれた表現か否かを識別する手法を提案する．一般的に，誰が書いても同様になりやすいありふれた表現のソースコードほど，学習データセットに頻出すると考えられる．そこで提案手法では，頻出する学習データほど LLM がより強く記憶するという従来の観測結果 [8] に基づいて，生成コードに対する LLM の記憶強度を測定する．そして，その記憶強度が閾値を超えた場合に，その生成コードはありふれた表現であると識別する．

実験では，既存手法がありふれた表現の生成コードを著作権侵害として過検知する可能性があることの実験的な証明と，提案手法の有効性評価を行った．まず，公開データセットである The Stack [9] に含まれる 200〜500 文字の Python のソースコードを対象に，複数の OpenAI モデルの LLM-as-a-Judge によって，ありふれた表現か否かのラベル付けを行った．次に，そのデータセットに対して既存手法を適用した．その結果，過検知率が 0.74 となり，既存手法がありふれた表現の生成コードを著作権侵害として過検知する可能性があることを実証した．加えて，9 種類の LLM を用いて各ソースコードの記憶強度を測定し，提案手法の識別性能を評価した．その結果，Qwen2.5-Coder の 0.5B を用いた場合に Area Under the Curve (AUC) が 0.75 と最も高くなり，提案手法がありふれた表現の識別に有効であることが示された．また，既存手法の検知結果に対して提案手法を適用することで，既存手法の検知率の低下を 1% に抑えつつ過検知率を 22%抑制でき，提案手法が過検知抑制に有効であることも示された．

```python
def add(value1, value2):
    return value1 + value2
```

**図 1** ありふれた表現のソースコードの例

本論文の貢献は次の通りである．

- 既存手法がありふれた表現のソースコードを著作権侵害として過検知する可能性があることを実験的に証明した．
- 過検知の抑制を目的として，LLM の記憶強度に基づいてありふれた表現のソースコードを識別する手法を提案し，その有効性を示した．

2 章では，著作権法や著作権侵害を検知する手法，LLM の記憶強度を測定する手法について述べる．3 章では，予備実験を通して，既存手法がありふれた表現のソースコードを著作権侵害として過検知する可能性があることを実験的に証明する．4 章では，過検知の抑制を目的として，ありふれた表現を識別する提案手法について述べ，5 章でその有効性を評価する．6 章では本研究について議論し，7 章でまとめる．

なお，本論文は学術的な情報提供のみを目的とし，法的な助言または法的な根拠を提供するものではない．本論文のみに基づく法的判断や行動は避け，必ず法律の専門家に確認することを推奨する．

## 2. 関連研究

### 2.1 著作権法

著作権法は著作物を保護する法律である．著作物とは，日本の著作権法 [5] の第 2 条第 1 項第 1 号において，思想または感情を創作的に表現したものと定義されている．一方，図 1 に示すような，誰が書いても同様になりやすいありふれた表現のソースコードは著作物でないとされ，著作

[1] NTT 社会情報研究所
NTT Social Informatics Laboratories
[a)] mas.matsubayashi@ntt.com

権で保護されない．

既存の著作物に対する著作権侵害が認められるには，主に 2 つの要件を満たす必要がある．1 つ目の要件は，後発の作品が既存の著作物と同一，または類似していること（類似性）である．類似性の明確な判定基準は存在しないが，ドイツの Act on the Copyright Liability of Online Content Sharing Service Providers [10] では「160 文字超」の一致が判定基準の 1 つとして言及されている．2 つ目は，後発の作品が既存の著作物に依拠して複製等がなされたこと（依拠性）である．既存の著作物が LLM の学習に使われた場合は依拠性に該当する．

これらの要件に対して，類似性に基づいて著作権侵害を検知する手法や，メンバーシップ推論によって依拠性を検査する手法が数多く提案されている．それぞれの手法について，2.2 節と 2.3 節で述べる．

### 2.2 類似性に基づく著作権侵害検知

ソースコードや書籍を対象とした既存研究 [6, 7, 11, 12] では，後発の作品と既存の著作物の文字列的な類似度を Longest Common Subsequence や BLEU-4，ROUGE-L，Greedy String Tiling などを用いて計算している．そして，計算した類似度が閾値以上であれば著作権侵害として検知している．一方，GitHub 等の公開リポジトリには，著作権で保護されないありふれた表現の公開コードも多く存在する．そのため，これらと文字列的に一致した生成コードが与えられた場合に，既存手法はその生成コードを著作権侵害として過検知する可能性がある．

### 2.3 メンバーシップ推論による依拠性の検査

既存手法は，LLM の応答内容に基づく手法 [13, 14] と，LLM の記憶強度に基づく手法 [8, 15] に大別できる．前者の手法は，学習に使われた著作物を LLM が逐語的に再生成できる特性を利用する．具体的には，著作物の冒頭部分を LLM に与え，その続きを LLM が正確に再現できるか否かを基に，メンバーシップ推論を行う．一方で後者は，学習に使われた著作物を LLM が強く記憶している特性を利用する．具体的には，著作物の各トークンの生成確率や perplexity の大きさを基に LLM の記憶強度を測定して，メンバーシップ推論を行う．

LLM の記憶強度に基づく手法のうち，本研究との関連が大きいものとして，Min-K% Prob [8] が提案されている．この手法では，学習に使われたか否かを検査したいトークン列を $x = x_1, x_2, \ldots, x_n$，各トークン $x_i$ の生成確率を $p(x_i \mid x_1, \ldots, x_{i-1})$ としたとき，生成確率が低い K% のトークンの平均値を以下の式で計算する．

$$\text{Min-K\% Prob}(x) = \frac{1}{E} \sum_{x_i \in \text{Min-K\%}(x)} p(x_i \mid x_1, \ldots, x_{i-1}). \tag{1}$$

なお，$x_i \in \text{Min-K\%}(x)$ は生成確率の低い K%のトークンを抽出する操作であり，$E$ は抽出したトークンの数である．この手法はシンプルでありながら，Carlini ら [15] の perplexity ベースの手法よりも高精度であることが報告されている [8]．また，学習データセットに頻出するデータほど，そうでないデータと比べて Min-K% Probが大きくなることが示されている [8]．

## 3. 予備実験

既存の著作権侵害検知手法がありふれた表現のソースコードを著作権侵害として過検知する可能性があることを実験的に証明する．

### 3.1 実験設定

**データセット** 著者らの知る限り，ありふれた表現のソースコードの過検知率を評価できるデータセットは存在しない．そこで，公開データセットである The Stack [9] を利用した独自のデータセットを構築した．The Stack は 2015 年 1 月 1 日から 2022 年 3 月 31 日までに GitHub で公開された 30 以上のプログラミング言語のソースコードで構成される．予備実験ではまず，この The Stack に含まれるもののうち，コメントなどを除いた実際のソースコード部分が 200〜500 文字である Python のソースコードを 7,000 件抽出した．200 文字以上としたのは，ドイツの Act on the Copyright Liability of Online Content Sharing Service Providers [10] が判定基準の 1 つとしている「160 文字超」に確実に該当し得る長さのソースコードを対象とするためである．一方，500 文字以下に制限したのは，ありふれた表現とそうでない表現の混在による評価の複雑性を緩和するためである．

抽出したソースコードには，それがありふれた表現か否かを示すラベルが付与されていない．そこで，抽出したソースコードを図 2 のプロンプトとともに OpenAI の GPT-4.1，GPT-4o，o3，o1 それぞれに入力し，「ありふれた表現」か「そうでない表現（保護され得る表現）」かを判定させた．その際，GPT-4.1 と GPT-4o の Temperature を 0 に設定し，固定的な出力が得られるようにした．それ以外のモデルでは Temperature を設定できないため，デフォルトの設定で，ある程度のランダム性をもった出力を得た．そして，表 1 に示すラベルを表 1 の条件を満たすサンプルに付与し，表 1 に記載の件数だけ抽出して，最終的なデータセットとして利用した．この時，データセットのラベルの品質を高めるために，より多くのモデルで判定結果が一致したサンプルを最優先で抽出した．

**既存の著作権侵害検知手法** 研究倫理の観点から具体的な名称についての言及は避けるが，一般的な市場サービスが提供する著作権侵害検知機能（既存の著作権侵害検知機能）

```
# Role
  * You are an experienced engineer.
  * You are also an expert in copyright law.

# Task
  * You will be given a piece of source code as input.
  * Return 'unprotectable' if the input code is
    boilerplate or commonly written in the same way
    by most programmers, meaning it is not eligible
    for copyright protection.
  * Otherwise, return 'protectable' if the code
    demonstrates originality and is therefore
    eligible for copyright protection.

# Input
  {code}
```

**図 2** ラベル付けプロンプト

**表 1** ラベルの付与条件と内訳

| ラベル | 条件 | 件数 | 計 |
|---|---|---|---|
| ありふれた表現 | 全モデルの判定が左記で一致 | 500 | 500 |
| | 全モデルの判定が左記で一致 | 45 | |
| 保護され得る表現 | 3 モデルの判定が左記で一致 | 431 | 500 |
| | 2 モデルの判定が左記で一致 | 24 | |

**表 2** TP，FN，FP，TN の定義

| ラベル | 検知 | 非検知 |
|---|---|---|
| 保護され得る表現 | True Positive (TP) | False Negative (FN) |
| ありふれた表現 | False Positive (FP) | True Negative (TN) |

を利用した．既存の著作権侵害検知機能は，The Stack に含まれるものよりも最近までにインターネット上で公開された公開コードと一致するソースコードを検出できる．

**実験方法** 最初に，データセットの各ソースコードを既存の著作権侵害検知機能に入力した．それに対して既存の著作権侵害検知機能は，入力に一致する公開コードの有無を検知結果として出力した．この検知結果と入力ソースコードのラベルを基に，表 2 に示す通り TP と FN，FP，TN を定義した．そして，著作権侵害の検知率 $TPR = \frac{TP}{TP+FN}$ と過検知率 $FPR = \frac{FP}{FP+TN}$ を評価した．

### 3.2 結果と考察

TPR が 0.81，FPR が 0.74 となった．この結果より，既存の著作権侵害検知機能のように，生成コードと公開コードの一致度合に基づいて著作権侵害を検知する既存手法では，ありふれた表現の生成コードを著作権侵害として過検知してしまう可能性が示された．

## 4. 提案手法

既存手法がありふれた表現のソースコードを著作権侵害として過検知する可能性があることを 3 章で実験的に証明した．それに対して本章では，著作権侵害の過検知の抑制を目的として，生成コードがありふれた表現か否かを識別する手法を提案する．一般的に，誰が書いても同様になりやすいありふれた表現のソースコードほど，学習データセットに頻出すると考えられる．そこで提案手法では，頻出する学習データほど LLM がより強く記憶するという従来の観測結果 [8] に基づいて，生成コードに対する LLM の記憶強度を用いてありふれた表現を識別する．

### 4.1 全体像

提案手法の全体像を図 3 に示す．前提として，任意の LLM や GitHub Copilot [1] 等のサービスによってソースコードが生成され，その生成コードの著作権侵害が既存の著作権侵害検知機能によって検知されているとする．提案手法は，著作権侵害として検知された生成コードのみを受け取り，その生成コードの記憶強度を任意の LLM で測定する．記憶強度の測定については 4.2 節で詳説する．最後に，測定した記憶強度が閾値 ($thr$) 以上であれば，その生成コードはありふれた表現であり，著作権侵害ではないと再判定することで過検知を抑制する．

### 4.2 記憶強度の測定

記憶強度は Min-K% Prob [8] に加え，本節で述べる Max-K% Probにより測定する．これらの測定手法の全体像を図 4 に示す．Min-K% Probは，LLM の生成確率の低いトークンを対象にその平均を測定する．一方で，誰が書いても同様になりやすいありふれた表現は，学習データセットに頻出し，LLM の生成確率が高くなると考えられる．そこで，より直接的にありふれた表現のトークンを対象とした記憶強度の平均を計算するために，提案手法では以下の式で定義する Max-K% Probも導入する．

$$\text{Max-K\% Prob}(x) = \frac{1}{E} \sum_{x_i \in \text{Max-K\%}(x)} p(x_i \mid x_1, \ldots, x_{i-1}). \tag{2}$$

なお，単に生成確率が最大のトークンのみや，生成確率の高い一定数のトークンを記憶強度の測定対象とするのではなく，Max-K%を測定対象とするのは，提案手法の頑健性を高めるためである．ソースコードはプログラミング言語の文法的制約の下で記述されるため，例えば図 1 の “, ” や “ : ” のように，自動的に決定されるトークンが複数存在する．このような決定的なトークンの生成確率は極めて高くなりやすく，ソースコードが長くなるほどそのようなトークンの絶対数は増加する．そしてこれは，そのソースコードがありふれた表現か否かによらない．よって，生成確率が最大のトークンのみや，生成確率の高い一定数のトークンを測定対象とする場合，ソースコードが長くなるほど前述のような決定的なトークンで測定対象が支配され，ありふれた表現を正しく識別できなくなる可能性が高い．それに対して Max-K%は，ソースコードの長さに応じて測定対象とするトークンの絶対数を調整できる．これにより，前述のような決定的なトークンで測定対象が支配されることを回避し，ありふれた表現の識別の頑健性を高められる．

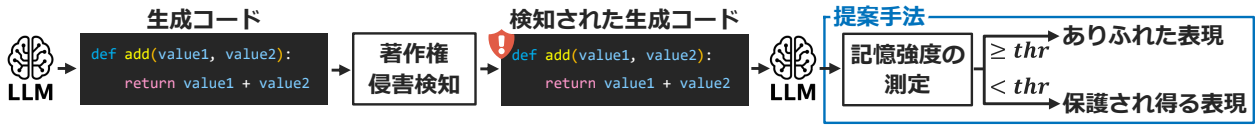


**図 3** 提案手法の全体像

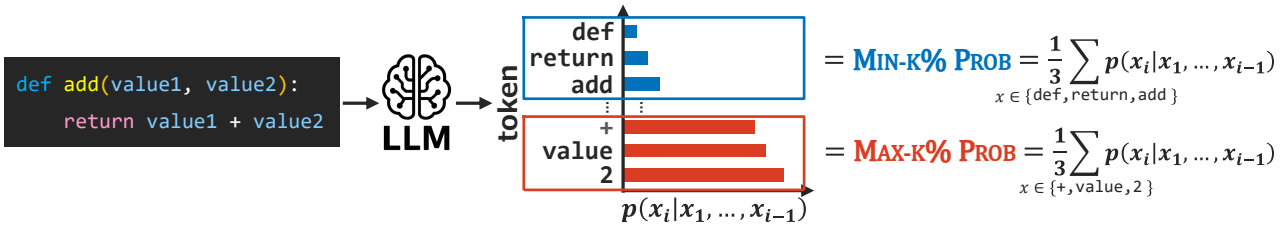


**図 4** MIN-K% PROBと MAX-K% PROB

**表 3** 検知後データセットの内訳

| ラベル | 条件 | 件数 | 計 |
|---|---|---|---|
| ありふれた表現 | 全モデルの判定が左記で一致 | 368 | 368 |
| | 全モデルの判定が左記で一致 | 35 | |
| 保護され得る表現 | 3 モデルの判定が左記で一致 | 354 | 404 |
| | 2 モデルの判定が左記で一致 | 15 | |

**表 4** 提案手法のパラメータ設定

| パラメータ | 値 |
|---|---|
| モデル | Mamba-Codestral-7B-v0.1[*1], CodeLlama-7b-hf[*2], CodeLlama-13b-hf[*3], CodeLlama-34b-hf[*4], CodeLlama-70b-hf[*5], Qwen2.5-Coder-0.5B[*6], Qwen2.5-Coder-7B[*7], Qwen2.5-Coder-14B[*8], Qwen2.5-Coder-32B[*9] |
| 記憶強度の測定手法 | MIN-K% PROB, MAX-K% PROB |
| $K$ | 10, 20, ..., 100 |
| $thr$ | 0.00, 0.01, ... 1.00 |

## 5. 評価

提案手法のありふれた表現の識別性能の評価 (評価 1) と，提案手法や比較手法の追加による既存の著作権侵害検知機能の FPR の抑制効果の比較評価 (評価 2) を行った．

### 5.1 実験設定

**データセット** 以下 2 種類のデータセットを利用した．

- 検知前データセット：3.1 節で構築したデータセットである．このデータセットの内訳は表 1 の通りである．
- 検知後データセット：検知前データセットを既存の著作権侵害検知機能に入力し，著作権侵害と検知されたソースコードのみを抽出したデータセットである．このデータセットの内訳を表 3 に示す．

**提案手法の実装** 表 4 に示す 9 種類のオープンソースなモデルを利用して，MIN-K% PROBもしくは MAX-K% PROB により記憶強度を測定した．また，MIN-K% PROBと MAX-K% PROBのパラメータである $K$，およびありふれた表現の識別に利用する閾値 $thr$ を表 4 に示す通り複数パターン用意した．そして，これらを網羅的に組み合わせた提案手法を実装し，それぞれの性能を評価した．

**比較手法** OpenAI の o4-mini と Qwen2.5-Coder-0.5B に対して，図 2 のプロンプトを入力することでありふれた表現の識別を行う LLM-as-a-Judge ベースの手法を用意した．o4-mini を選択した理由は，評価実施時点で最新かつ経済的な推論モデルであり，ラベル付けのプロセスに関与していないモデルであったためである．一方で，Qwen2.5-Coder-0.5B を選択した理由は，5.2 節に示す通り，そのモデルを記憶強度の測定に利用した際の提案手法の性能が最も高かったためである．

**実験方法** 評価 1 では，検知後データセットに対して提案手法を適用し，ありふれた表現と識別されたものを著作権侵害なし，そうでないものを著作権侵害ありと判定した．そして，著作権侵害の検知率 (TPR) と過検知率 (FPR) を計算した．この結果に対して，検知後データセットを基準とした提案手法の ROC (Receiver Operating Characteristic) 曲線の描画と AUC の計算を行い，ありふれた表現の識別性能を評価した．なお，評価 1 における TPR と FPR の値域はともに [0.0, 1.0] であり，ありふれた表現を正しく識別できるほど TPR は高く，FPR は低くなる．

評価 2 では，検知後データセットに対して提案手法と比較手法を適用し，検知前データセットを基準とした TPR

[*1] https://huggingface.co/mistralai/Mamba-Codestral-7B-v0.1
[*2] https://huggingface.co/codellama/CodeLlama-7b-hf
[*3] https://huggingface.co/codellama/CodeLlama-13b-hf
[*4] https://huggingface.co/codellama/CodeLlama-34b-hf
[*5] https://huggingface.co/codellama/CodeLlama-70b-hf
[*6] https://huggingface.co/Qwen/Qwen2.5-Coder-0.5B
[*7] https://huggingface.co/Qwen/Qwen2.5-Coder-7B
[*8] https://huggingface.co/Qwen/Qwen2.5-Coder-14B
[*9] https://huggingface.co/Qwen/Qwen2.5-Coder-32B

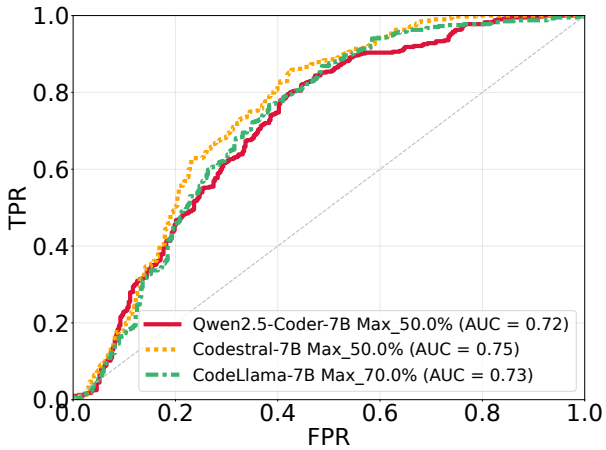


**図 5** モデルファミリー間での ROC 曲線と AUC の比較

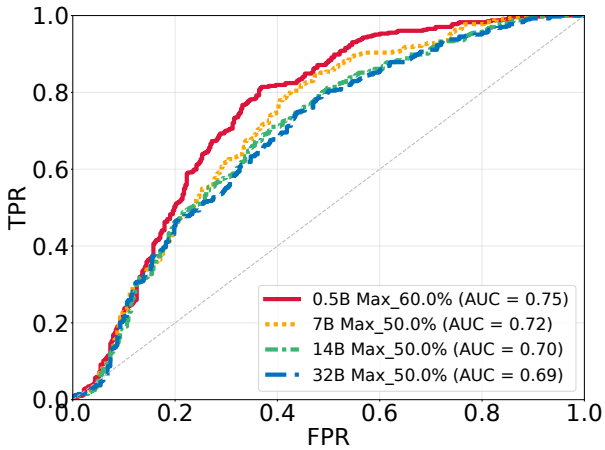


**図 6** サイズ間での ROC 曲線と AUC の比較（Qwen2.5-Coder）

と FPR を評価した．この評価の目的は，既存の著作権侵害検知機能の後に提案手法や比較手法を適用することで，既存の著作権侵害検知機能の FPR をどの程度抑制できるかを評価することである．なお，評価 2 における TPR と FPR の値域はそれぞれ，最初に適用する既存の著作権侵害検知機能の TPR と FPR を上限値とする [0.0, 0.81] と [0.0, 0.74] となる．

### 5.2 結果と考察

**評価 1-1：モデルファミリー間での識別性能の比較** モデルサイズを 7B で揃え，モデルファミリーのみを変更した場合の ROC 曲線と AUC を図 5 に示す．なお，図 5 の ROC 曲線と AUC には，各モデルファミリーにおいて AUC が最大となるパラメータの組み合わせの結果のみを示している．図 5 より，$K = 50$ 付近の Max-K% Probが共通的に最も有効であった．それらの AUC はモデルファミリーによらず 0.7 を超え，概ね 0.73 前後となった．このことから，提案手法はモデルファミリーによらずありふれた表現の識別に有効だと考える．

**評価 1-2：モデルサイズ間での識別性能の比較** Qwen2.5 と CodeLlama それぞれで，モデルサイズのみを変更した場合の ROC 曲線と AUC を図 6 と図 7 に示す．なお，図 6 と図 7 の ROC 曲線と AUC には，各モデルサイズにおいて AUC が最大となるパラメータの組み合わせの結果のみを示している．図 6 と図 7 より，モデルサイズによらず，$K = 50$ 付近の Max-K% Probが共通的に最も有効であった．また，小規模なモデルほど AUC が高まる傾向が見られた．このことから，提案手法では小規模なモデルほどありふれた表現の識別に有効だと考える．

なお，小規模なモデルほど AUC が高くなったのは，大規模なモデルほど記憶容量が大きく，ありふれた表現か否かに関わらず全てのソースコードを同程度に強く記憶できるためと考える．そう考える根拠として，Qwen2.5 におけるモデルサイズと，各ソースコードの Max-50% Probの平均値の関係を表 5 に示す．表 5 に示す Max-50% Probの平均値は，ありふれた表現のラベルを持つソースコードとそうでないソースコードで分けて計算した．また，それらの差分値も表 5 に示す．表 5 に示す通り，モデルサイズが大規模になるほど Max-K% Probの平均値が増加し，差分値は減少した．これは，大規模なモデルほど，ありふれた表現か否かによらず全てのソースコードを同程度に強く記憶できることを示していると考える．

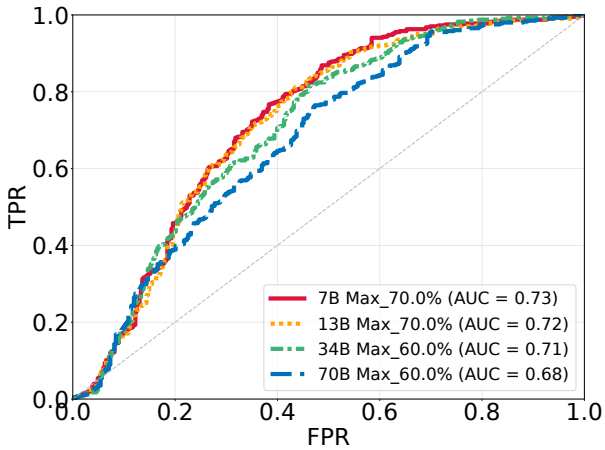


**図 7** サイズ間での ROC 曲線と AUC の比較（CodeLlama）

**表 5** Qwen2.5-Coder のサイズと Max-50% Probの平均値

| モデルサイズ | Max-50% Probの平均値 | | 差分値 |
|---|---|---|---|
| | ありふれた表現 | 保護され得る表現 | |
| 7B | 0.968 | 0.945 | 0.023 |
| 14B | 0.973 | 0.954 | 0.019 |
| 32B | 0.975 | 0.957 | 0.018 |

**評価 2：FPR の抑制効果** 著作権で保護され得るソースコードをありふれた表現と誤判定すると著作権侵害のリスクを高める．そこで，提案手法の追加による既存の著作権侵害検知機能の TPR の低下量を −0.01（TPR@0.80）と −0.05（TPR@0.75）の最小限に抑えるように閾値 $thr$ を

**表 6** 高 TPR における提案手法の FPR とその抑制量

| 手法 | FPR (抑制量) | |
|---|---|---|
| | TPR@0.80 (−0.01) | TPR@0.75 (−0.05) |
| Codestral-7B-v0.1 | **0.52 (−0.22)** | 0.43 (−0.31) |
| Qwen2.5-Coder-0.5B | 0.60 (−0.14) | **0.41 (−0.33)** |
| Qwen2.5-Coder-7B | 0.58 (−0.16) | 0.54 (−0.20) |
| Qwen2.5-Coder-14B | 0.74 (−0.00) | 0.57 (−0.17) |
| Qwen2.5-Coder-32B | 0.74 (−0.00) | 0.55 (−0.19) |
| CodeLlama-7b-hf | 0.65 (−0.09) | 0.43 (−0.31) |
| CodeLlama-13b-hf | 0.66 (−0.08) | 0.49 (−0.25) |
| CodeLlama-34b-hf | 0.74 (−0.00) | 0.51 (−0.23) |
| CodeLlama-70b-hf | 0.74 (−0.00) | 0.47 (−0.27) |

**表 7** 比較手法の結果

| モデル | TPR (低下量) | FPR (抑制量) |
|---|---|---|
| o4-mini | 0.14 (−0.67) | 0.00 (−0.74) |
| Qwen2.5-Coder-0.5B | 0.02 (−0.79) | 0.01 (−0.73) |

設定した際の，FPR と FPR の抑制量を表 6 に示す．表 6 より，TPR の低下量を −0.01 に抑えた際には，Codestral-7B-v0.1 が FPR を最も抑制でき，その抑制量は −0.22 となった．また，TPR の低下量を −0.05 に抑えた際には，Qwen2.5-Coder-0.5B が FPR を最も抑制でき，その抑制量は −0.33 となった．一方，表 7 には既存の著作権侵害検知機能の後に比較手法を追加した際の TPR と FPR，およびそれらの低下量と抑制量を示す．表 7 より，比較手法は FPR を大きく抑制できたものの，TPR も過剰に低下させてしまった．これらの結果より，提案手法には改善の余地があるものの，既存の著作権侵害検知機能の過検知の抑制に有効だと考える．

## 6. 議論

**モデルの選択** 提案手法には小規模かつオープンソースな任意のモデルを選択すれば良いと考える．5.2 節で述べたとおり，提案手法では小規模モデルを利用する方が有効性が高まる．また，モデルファミリーによる性能差は小さい．この傾向は，GitHub Copilot 等がソースコードの生成に利用しているクローズドな LLM にも共通していると考える．よって，コード生成に利用された LLM の記憶強度を直接的に測定できない場合でも，任意の小規模かつオープンソースなモデルで代替できると考える．これは，提案手法を実用化する上での大きな優位性と考える．

**提案手法の改善** 提案手法は一定の有効性を持つと考えるが，その性能には改善の余地がある．そこで今後は，提案手法の性能の改善を目指す．その方法の一つとして，Min-K% Probや Max-K% Prob以外の，提案手法により適した記憶強度の測定方法の特定と活用が有効と考える．また，記憶強度の測定に用いる LLM の内部パラメータの観測や，その LLM の fine-tuning，複数の LLM でのアンサンブルも有効と考える．

**データセットの改善** 本稿では，200〜500 文字の Python のソースコードに限定したデータセットを構築して評価を行った．今後は，その他のプログラミング言語や，より長いソースコードを対象とした評価も行えるようにデータセットを改善する．また，本稿におけるソースコードへのラベル付けは，ラベルの信頼度を高めるために，OpenAI の 4 種類のモデルの判定の完全一致や多数決を取ることで行った．一方で，より信頼度の高いラベルを付与するには，現在のラベル付けプロセスを改善する必要があると考える．今後は，OpenAI 以外が提供するモデルの追加や，法律の専門家によるラベルの確認と修正などを組み合わせたラベル付けを行うことで，ラベルの信頼度を改善する．

**法的解釈** ありふれた表現に該当するか否かの判断基準は，各国の著作権法や判例の解釈に依存しており，明確かつ普遍的なものは存在しない．したがって，3 章の予備実験の結果は，生成コードと公開コードの一致度合に基づいて著作権侵害を検知する既存手法や市場技術が必ずしも過検知を起こすと断定するものではなく，あくまで 1 つの可能性を示したに過ぎない．また，提案手法は一つの技術的アプローチに過ぎず，本稿で示した結果が法的判断においてそのまま適用可能であることを意味するものではない．

一方で，提案手法で計算する Min-K% Prob等の記憶強度には，生成コードの著作権侵害の有無を評価する際の補助的な情報として活用する余地があると考える．これらを活用することで，開発者や法務担当者が生成コードの著作権侵害リスクをより慎重かつ合理的に評価することが可能になることを期待する．

## 7. おわりに

LLM が生成したソースコードの著作権侵害を検知する既存手法では，著作権で保護されない「ありふれた表現」のソースコードを著作権侵害として過検知する可能性があることを実験的に証明した．加えて，この過検知の抑制を目的として，LLM の記憶強度に基づいてありふれた表現のソースコードを識別する手法を提案した．評価では，公開データセットである The Stack のソースコードに対して，4 種類の OpenAI モデルの LLM-as-a-Judge によって，ありふれた表現か否かのラベルを付与した．そのデータセットに提案手法を適用した結果，AUC = 0.75 の性能でありふれた表現を識別でき，提案手法の有効性を確認した．また，既存手法の検知結果に対して提案手法を適用することで，既存手法の検知率の低下を 1% に抑えつつ過検知率を 22%抑制でき，提案手法が過検知抑制に有効であることも示された．特筆すべき点として，記憶強度の測定に用いる LLM はモデルファミリーに依存せず，小規模なモデルが適していることが明らかになった．これは，提案手法を実用化する上での大きな優位性と考える．今後は，提案手法

やラベル付けプロセスの改善，様々なプログラミング言語の様々な長さのソースコードを対象とした評価を行う．

## 参考文献


[1] GitHub, Inc. Github copilot: Your ai pair programmer. https://github.com/features/copilot. (Accessed on Aug. 7, 2025).

[2] Bain & Company. Technology report 2024 technology meets the moment as ai delivers results. https://www.bain.com/globalassets/noindex/2024/bain_report_technology_report_2024.pdf, 2024. (Accessed on Aug. 7, 2025).

[3] 特許庁. Trips 協定 3. https://www.jpo.go.jp/system/laws/gaikoku/trips/chap3.html\#law10. (Accessed on Aug. 7, 2025).

[4] 文部科学省. 1971 年 7 月 24 日にパリで改正された万国著作権条約 (仮訳). https://www.mext.go.jp/unesco/009/003/011.pdf. (Accessed on Aug. 7, 2025).

[5] 著作権法（昭和四十五年法律第四十八号）. https://laws.e-gov.go.jp/law/345AC0000000048/20251001_507AC0000000027#Mp-Ch_2-Se_1. (Accessed on Aug. 7, 2025).

[6] Weiwei Xu, Kai Gao, Hao He, and Minghui Zhou. Licoeval: Evaluating llms on license compliance in code generation. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, pp. 1665–1677, 2025.

[7] Zhiyuan Yu, Yuhao Wu, Ning Zhang, Chenguang Wang, Yevgeniy Vorobeychik, and Chaowei Xiao. CodeIP-Prompt: Intellectual property infringement assessment of code language models. In *Proceedings of the 40th International Conference on Machine Learning*, Vol. 202 of *Proceedings of Machine Learning Research*, pp. 40373–40389, 2023.

[8] Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. Detecting pretraining data from large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[9] Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro von Werra, and Harm de Vries. The stack: 3 tb of permissively licensed source code. *Preprint*, 2022.

[10] Federal Ministry of Justice and Consumer Protection (BMJV). Act on the copyright liability of online content sharing service providers (urheberrechts-diensteanbieter-gesetz – urhdag). https://www.gesetze-im-internet.de/englisch_urhdag/. (Accessed on Aug. 7, 2025).

[11] Felix B Mueller, Rebekka Görge, Anna K Bernzen, Janna C Pirk, and Maximilian Poretschkin. Llms and memorization: On quality and specificity of copyright compliance. *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, Vol. 7, No. 1, pp. 984–996, 2024.

[12] Xiaoze Liu, Ting Sun, Tianyang Xu, Feijie Wu, Cunxiang Wang, Xiaoqian Wang, and Jing Gao. SHIELD: Evaluation and defense strategies for copyright compliance in LLM text generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1640–1670, 2024.

[13] Weijie Zhao, Huajie Shao, Zhaozhuo Xu, Suzhen Duan, and Denghui Zhang. Measuring copyright risks of large language model via partial information probing, 2024.

[14] André Vicente Duarte, Xuandong Zhao, Arlindo L. Oliveira, and Lei Li. DE-COP: Detecting copyrighted content in language models training data. In *Proceedings of the 41st International Conference on Machine Learning*, Vol. 235 of *Proceedings of Machine Learning Research*, pp. 11940–11956, 2024.

[15] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 2633–2650, 2021.