一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report

LLMを用いたコードクローン検出における識別子名の影響分析

井上龍太郎†　肥後　芳樹†

† 大阪大学大学院情報科学研究科
大阪府吹田市山田丘 1-5
E-mail: †{ry-inoue,higo}@ist.osaka-u.ac.jp

あらまし　コードクローンはソフトウェアの保守性を低下させる要因であり，効率的な検出が重要である．LLM（大規模言語モデル）はコードクローン検出の有力な手段として注目されているが，LLM がどの情報に依存してクローンを検出しているかは明らかではない．本研究では，LLM によるコードクローン検出において識別子名が検出精度に与える影響を分析することで，識別子情報に起因する誤検出を 抑制するための知見を得る．機能等価なメソッドペアで構成されるデータセット FEMPDataset を用い，メソッド名，メソッド引数名，変数名を意味のない文字列に置換したデータセットを複数のパターン作成した．作成したデータセットに対して 6 種類の LLM でクローン検出精度を評価し，マスクの有無による性能変化を比較した．

キーワード　コードクローン，LLM（大規模言語モデル），FEMPDataset

1. ま え が き

コードクローンとは，ソースコード中に存在する互いに一致または類似した部分を持つコード片のことである [1]．コードクローンに対するコーディングは一貫した変更が必要となることがあり，一貫性のない変更は変更もれによって欠陥の原因となる [2]．このように，コードクローンの存在はソースコードの修正に大きな障害となってしまい，システムの保守性を大きく損ねてしまう．このため，コードクローンを効率的に検出する必要がある．コードクローンは構文的な類似度によって Type-1 から Type-4 の 4 つに分類される．特に Type-4 のコードクローンは構文的に一致する箇所が少なく，検出が難しいとされる．

コードクローン検出を目的として，多くのツールが開発されてきた．字句解析やメトリクスなどの静的解析を用いた CCFinder [3] や NiCad [4] といったツールは，構文的な類似度の高いコードクローン検出で高い精度を示す．しかし，構文的な類似度の低いコードクローン検出では精度が低いという課題がある．

また，ASTNN [5] や CodeBERT [6] などの機械学習を使用したコードクローン検出手法も提案されている．これらのツールは静的解析を用いたツールと比較して，構文的な類似度の低い Type-3 や Type-4 のコードクローン検出において高い精度を示している [7].

近年は，機械学習に加えて大規模言語モデル（LLM）を用いたコードクローン検出手法が注目されている．LLM は自然言語処理の分野で高い成果を上げており，コードクローン検出などのタスクにおいても有望な手法とされている [8]．Google，OpenAI，Microsoft などの企業は多種多様な LLM を開発，公開しており，Google の Gemini [9] や Gemma3 [10]，OpenAI の GPT-4 [11]，Microsoft の Phi-4 [12]，などが代表的な例である．これらの LLM は自然言語処理だけでなく，コード生成やコードクローン検出などのタスクにも利用されている [8].

Almatrafi らの研究 [13] では，few-shot instruction tuning を適用した GPT-4 が，Type-4 クローン検出において F1 スコア 0.93 を達成した．この値は，NiCad などの静的解析ベースのツールが示す精度を大きく上回る．また，Type-1 から Type-4 の全てを含む全体評価においても，GPT-4 は F1 スコア 0.87 を記録している．

また，我々は 3 つの LLM に対して FEMPDataset を用いたファインチューニングを行い，コードクローン検出の精度の向上が認められた [14]．また，静的解析ツールにおいて検出が難しかった異構造のコードクローンに対する検出において，LLM が有用であることを示した．

しかしながら，多くの LLM は自然言語処理を対象に開発されており，もっともらしい回答する事でユーザーの期待に応えるように訓練されている．入力される単語の意味が LLM の出力に影響を与えることから，コードの構造や論理のみならずメソッド名や変数名などの識別子名がコードクローン検出に影響を与えている可能性がある．

そこで本研究では，LLM を用いたコードクローン検出において，識別子情報に起因する誤検出を抑制するための知見を得る事を目的に，識別子名が検出精度に与える影響を分析する．メソッド名，メソッドパラメータ名，変数名の 3 種類に識別子名を分類し，これらの識別子名を意味の持たない名前に置換したデータセットを作成した．以降，識別子名を意味をもたない名前に置換する処理をマスク処理と呼ぶ．作成したデータセットに対して 6 種類の LLM でクローン検出精度を評価し，マスクの有無による性能変化を比較した．

実験の結果，多くのモデルで識別子名をマスクした場合で

もクローン検出性能が維持，あるいは向上する傾向が見られた．特に，メソッド名のみをマスク処理を行わない場合に性能が低下する傾向が確認されたことから，メソッド名が判定に与える影響が大きいと結論付けた．さらに高性能なモデルである GPT-4.1 と Phi-4 を対象に，メソッド名の類似度と判定の変化を分析した結果，両モデルともにメソッド名の字句的・意味的類似度に強く影響を受けていることが明らかになった．メソッド名の類似度はクローンを正しく判定する手がかりとなる一方で，構造的に類似していない非クローンをクローンと誤判定する主要な原因にもなっていた．

2. 準　　備

本章では，研究の背景として，コードクローンの定義と分類，実験で使用するデータセット，評価指標について述べる．

2.1 コードクローン

コードクローンとは，ソースコード中に存在する互いに一致または類似した部分を持つコード片のことである [1]．コードクローンは，コピーアンドペーストなどの既存コードの再利用や，類似した機能を大規模システム内に再び実装することによって作成される [15]．また，コードクローンの関係にあるコード片の組をクローンペアと呼ぶ．

2.1.1 コードクローンの分類

Roy らは，コードクローンを類似度にもとづいて以下の 4 種類に分類，定義した [16]．

- Type-1（T1）：改行・スペース・コメントなどのレイアウトの違いを除いて一致するコードクローン．
- Type-2（T2）：Type-1 に加えて，識別子，リテラル，型の違いを除いて一致するコードクローン．
- Type-3（T3）：Type-2 に加えて，文の変更・挿入・削除などの違いを除いて一致するコードクローン．
- Type-4（T4）：構文的な違いを持つが，同じ処理を行うコードクローン．

2.2 FEMPDataset

FEMPDataset [17] は異構造で機能等価なメソッドを集めたデータセットである．テストケースの相互実行により機能等価なメソッドペアの候補を抽出し，人が目視で真に機能等価なメソッドペアを選別することで，正確なデータセットを構築している．FEMPDataset に含まれる機能等価なメソッドペアと機能等価でないメソッドペアは，コードクローン検出ツールにおいて正しく分類することが求められる．

2.3 LLM（大規模言語モデル）

LLM は，大規模なコーパスを用いて学習された言語モデルである．2017 年に Transformer [18] が発表されて以来，モデルの大規模化と学習データ量の増加が進んでいる．LLM は自然言語処理だけでなく，ソフトウェア工学を含む様々な研究分野で活用されている．

2.4 検出精度指標

コードクローン検出の性能指標として，Recall，Precision，Accuracy を用いる．各指標は，真陽性（TP），偽陽性（FP），偽陰性（FN），真陰性（TN）の数を用いて以下のように定義される．

Recall　実際にクローンであるペアのうち，正しくクローンと判定された割合．

$$Recall = \frac{TP}{TP + FN}$$

Precision　クローンと判定されたペアのうち，実際にクローンであった割合．

$$Precision = \frac{TP}{TP + FP}$$

Accuracy　全てのペアに対し，正しく判定を行った割合．

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

2.5 類似度指標

本研究では，識別子名の類似度を評価する指標として，Jaccard 係数，Cosine 類似度，単語出現頻度の 3 つの指標を使用する．

2.5.1 Jaccard 係数

Jaccard 係数は，2 つの集合 A と B が与えられたときに，その共通要素の数を和集合の要素数で割ることで算出される指標であり，以下の式で表される．

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

値は 0 から 1 の範囲をとり，1 に近いほど集合の類似度が高いことを示す．本研究では，メソッド名を CamelCase や SnakeCase で単語に分割し，2 つのメソッド名で共通する単語の割合を計算する．「getUserName」と「getUserData」では，「get」と「User」が共通するため，類似度が高くなる．

2.5.2 Cosine 類似度

Cosine 類似度は，2 つのベクトル A と B がなす角度のコサインを計算することで，その方向の類似度を測る指標であり，以下の式で表される．

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

値は-1 から 1 の範囲をとり，1 に近いほどベクトルが同じ方向を向いており，類似度が高いことを示す．本研究では，メソッド名を意味的な特徴を捉えたベクトルに変換し，2 つのベクトルの Cosine 類似度を計算することで，意味的な類似度を評価する．Jaccard 係数と異なり，単語が異なっても意味が近い場合（例：「calculate」と「compute」）に類似度が高くなる．ベクトル化に使用するエンベディングモデルは「codet5p-110m-embedding」を使用した．

2.5.3 単語出現頻度

メソッド名に含まれる単語がデータセット全体でどの程度使用されているかを表す指標である．頻出単語ほど一般的で，珍しい単語ほどそのメソッドの特徴を表すと考えられる．クローンペアと非クローンペアのそれぞれにおいて，メソッド名に含まれる単語の出現回数をカウントし，その回数の平均を各メソッドごとに計算する．メソッドペアに含まれる各メソッドの出現回数の平均を，メソッドペア全体の単語出現頻度とする．

3. 関 連 研 究

本章では，LLM を利用したコードクローン検出と，その評価ベンチマークに関する関連研究を述べる．

3.1 LLM を用いたコードクローン検出

LLM を用いたコードクローン検出に関する先行研究として，Almatrafi らの研究 [19] を紹介する．Almatrafi らの研究では，GPT-4 と GPT-3.5-turbo に対して，コードクローン検出の精度を向上させるための few-shot instruction tuning を行った．訓練には BigCloneBench [20] から抽出した 100 個のデータを使用し，評価には同じく抽出した 2000 個のデータを使用した．

結果，LLM は Type-4 クローン検出において F1 スコア 0.93 を達成し，これは NiCad や CodeBERT などの既存ツールが Type-4 クローンで示す精度を大きく上回った．また，Type1〜4 を含む全体評価においても，GPT-4 が F1 スコア 0.87 を記録しており，既存のツールと比較して高い精度を示した．

3.2 ファインチューニング前後のクローン検出精度の評価

我々は以前の研究で，LLM を用いたコードクローン検出の精度向上を目的とし，FEMPDataset によるファインチューニングを試みた [14]．対象とした LLM は gpt-3.5-turbo，Llama2-Chat-7B，CodeLlama-7B-Instruct の 3 つである．実験の結果，全てのモデルでファインチューニングによる精度向上が確認され，特にコード処理に特化した CodeLlama-7B-Instruct で大きく精度が向上した．

3.3 Type-4 クローン検出の評価ベンチマーク

BigCloneBench [20] は Svajlenko らによって作成された Type-1 から Type-4 までのコードクローンを含むコードクローン検出のベンチマークである．ファイルのコピーやバブルソートといった 45 の機能ごとに，その機能を持つメソッドを抽出し，クローンペアを作成している．しかし，データの偏りやラベリングの正確性に異議を唱える声があり，機械学習の学習データとしては不適切であるとの指摘もある [21].

SemanticCloneBench [22] は，質問回答 Web サイトである StackOverflow から作成された Type-4 クローンのデータセットである．正解クローンを 2 名の審査員によって手動検証し，検証結果を元にデータセットを作成している．

GPTCloneBench [23] は，SemanticCloneBench のデータセットをもとに GPT-3 を用いて新たに作成された Type-4 クローンのデータセットである．生成されたメソッドペアは最終的に審査員 6 名によって検証され，検証結果を元にデータセットを作成している．

これらのデータセットは，人間の目視判断にもとづいてラベル付けされており，実際の動作とメソッドペアの等価性を保証するものではない．対照的に，本研究で用いる FEMPDataset は，テストケースの相互実行と 3 名による目視確認を通じて，メソッドの機能的な等価性を担保している．この機能等価性の観点から，本研究では FEMPDataset を評価に用いることとした．

4. 本研究のアプローチ

本研究では，LLM によるコードクローン検出で，識別子名が精度に与える影響を分析する．LLM がコードの構造よりも識別子名に依存する度合いを定量的に把握し，誤検出の抑制や効果的なプロンプト設計の知見を得る．本研究では以下のリサーチクエスチョン（RQ）を設定する．

RQ1　識別子名のマスクは，LLM によるコードクローン検出の性能にどのような影響を与えるか？

RQ2　メソッド名，引数名，変数名のうち，どの識別子が LLM の判定に最も大きな影響を与えるか？

これらの RQ に答えるため，本研究では識別子名をマスクしたデータセットを用いて実験を行う．具体的には, FEMPDataset に含まれる Java のメソッドペアを対象に，(1) メソッド名，(2) メソッド引数名，(3) 変数名の 3 種類の識別子を，意味を持たない文字列に置換する．以降，識別子名を意味をもたない名前に置換する処理をマスク処理と呼ぶ．複数のマスクパターンを適用して作成したデータセットを用いて LLM におけるコードクローン検出の性能を評価する．性能評価には Recall, Precision, Accuracy を用い，マスクの有無やマスクパターンの違いによる性能変化を比較分析する．

5. 実　　験

本研究では，GPT-4.1, Phi-4, Gemini-2.5-pro, Gemma-3-27b-it, CodeLlama-7B-Instruct, Codegemma-7b-it の 6 つの LLM を使用する．実験は，(1)FEMPDataset の識別子名をマスク加工，(2) 加工データで LLM に推論させ回答を取得，(3) 回答を集計し性能を評価，という手順で行う．

5.1 データセットの加工

実験では以下の識別子名を対象にマスク処理を行う．

- メソッド名："METHOD"に置換する．
- メソッド引数名："METHODPARAMETERX"に置換する．
- 変数名："VARIABLEX"に置換する．

ただし，マスク対象はメソッド内で定義された識別子のみとし，外部ライブラリのメソッド名や変数名はマスクしない．これは，外部ライブラリの識別子をマスクすることでコードが解読困難になることを避けるためである．マスク処理の例を図 1 に示す．

対象の識別子を組み合わせて，表 1 に記載したマスクパターンごとにデータセットを作成する．

オリジナルはマスク処理を行わないデータセットであり，マ

表 1　識別子名マスクパターン

	メソッド名	メソッド引数名	変数名
オリジナル	×	×	×
マスクパターン 1	○	○	×
マスクパターン 2	○	×	○
マスクパターン 3	×	○	○
マスクパターン 4	○	○	○

○：マスク処理を行う，×：マスク処理を行わない

マスク処理前

```
String getErrorMessage(Throwable e) {
        Throwable throwable;
        while ((throwable = e.getCause()) != null) {
                e = throwable;
        }
        return e.getMessage();
}
```

マスク処理後

```
String METHOD (Throwable METHODPARAMETER0) {
        Throwable VARIABLE0;
        while ((VARIABLE0 = METHODPARAMETER0.getCause()) != null) {
                METHODPARAMETER0 = VARIABLE0;
        }
        return METHODPARAMETER0.getMessage();
}
```

図 1　識別子名に対するマスク処理の例

スクパターン 1 はメソッド名とメソッド引数名をマスク処理したデータセット，マスクパターン 2 はメソッド名と変数名をマスク処理したデータセット，マスクパターン 3 はメソッド引数名と変数名をマスク処理したデータセット，マスクパターン 4 は全ての識別子名をマスク処理したデータセットである．

推論には 4 台の NVIDIA RTX A6000 を使用した．

5.2　LLM の推論

マスク処理を行ったデータから，「提示した 2 つのメソッドがクローンペアであるか」を"Yes"または"No"で回答するよう指示するプロンプトを作成し，LLM の回答を得る．

5.3　性 能 評 価

LLM から得た回答を集計し，FEMPDataset に含まれる全 2,194 ペアを対象に，Recall，Precision，Accuracy を計算して性能を評価する．

5.4　結　　果

マスクパターンごとに作成したデータセットを用いて LLM の性能評価を行った結果を，表 2 に示す．

6.　RQ に対する回答

本章では，5.4 に基づき RQ への回答を示す．

6.1　RQ1：識別子名のマスク処理による性能への影響

識別子名のマスク処理による性能への影響を分析するために，マスクなし（オリジナル）と全ての識別子名をマスクした場合（マスクパターン 4）の性能を比較する．

GPT-4.1 は Accuracy が 0.83 から 0.85 へ，Phi-4 は 0.80 から 0.81 へと，マスクによって僅かに向上した．一方で，Gemma-3-27b-it では Accuracy が 0.77 から 0.73 へと，性能が低下した．結果として，多くのモデルで，識別子情報がなくてもコードの構造的特徴からある程度のクローン検出が可能であることが示された．

6.2　RQ2：識別子名の種類による影響の比較

マスクパターンごとの結果を比較し，メソッド名，引数名，変数名のうち，どの識別子名が LLM の判定に最も大きな影響を与えるかを分析した．

全体的に性能が高い GPT-4.1 と Phi-4 では，メソッド名のみをマスクしない場合（マスクパターン 3）に，Accuracy が全マスクパターンの中で最も低い値を示した．この事実は，メソッド名という単一の識別子情報だけが残された場合，モデルがその情報に過度に依存して判断を誤る可能性を示唆するものであり，メソッド名が判定に与える影響の大きさを示している．

各モデルの挙動を詳細に見ると，以下の傾向が確認できる．

- **GPT-4.1**: メソッド名と変数名をマスクした場合（マスクパターン 2）に Recall が 0.90 と最も高かった．全ての識別子名をマスクした場合（マスクパターン 4）では Precision は 0.87，Accuracy は 0.85 と最高値を示し，識別子情報がない方が誤検出を抑制し性能が高かった．
- **Phi-4**: メソッド名とメソッド引数名をマスクした場合（マスクパターン 1）で Recall が 0.88，Accuracy が 0.83 と最高値を示した．GPT-4.1 と同様にマスクパターン 3 で性能が最も低下しており，メソッド名がコードクローン検出に大きな影響を与えていると考えられる．
- **Gemini-2.5-pro**: Recall が 0.96 と全てのマスクパターンで一貫して高い値を示した．マスク処理によって Precision と Accuracy が僅かに向上し，特にメソッド名と変数名をマスクしたマスクパターン 2 で Accuracy が 0.79 を示した．全体として，マスク処理によって僅かながら性能が向上した．

表 2　モデル別・マスクパターン別の性能評価結果

モデル	マスクパターン	Recall	Precision	Accuracy
GPT-4.1	オリジナル	0.88	0.84	0.83
	マスクパターン 1	0.88	0.87	0.85
	マスクパターン 2	**0.90**	0.85	0.84
	マスクパターン 3	0.88	0.83	0.82
	マスクパターン 4	0.88	**0.87**	**0.85**
Phi-4	オリジナル	0.82	**0.85**	0.80
	マスクパターン 1	**0.88**	0.84	**0.83**
	マスクパターン 2	0.85	0.84	0.81
	マスクパターン 3	0.80	0.83	0.77
	マスクパターン 4	0.84	**0.85**	0.81
Gemini-2.5-pro	オリジナル	0.96	0.74	0.77
	マスクパターン 1	**0.96**	0.75	0.78
	マスクパターン 2	0.96	**0.76**	**0.79**
	マスクパターン 3	0.96	0.75	0.78
	マスクパターン 4	0.96	0.76	0.78
Gemma-3	オリジナル	**0.79**	0.82	0.77
	マスクパターン 1	0.73	0.86	0.76
	マスクパターン 2	0.76	0.85	**0.77**
	マスクパターン 3	0.73	0.82	0.74
	マスクパターン 4	0.66	**0.88**	0.73
CodeLlama	オリジナル	0.66	**0.75**	0.66
	マスクパターン 1	**0.81**	0.71	**0.68**
	マスクパターン 2	0.69	0.73	0.65
	マスクパターン 3	0.67	0.73	0.65
	マスクパターン 4	0.77	0.71	0.66
Codegemma	オリジナル	0.08	**0.94**	0.43
	マスクパターン 1	0.13	0.94	0.46
	マスクパターン 2	0.11	0.92	0.45
	マスクパターン 3	0.12	0.94	0.45
	マスクパターン 4	**0.14**	0.92	**0.47**

- **Gemma-3-27b-it**: マスクなし（オリジナル）の Recall が最も高いものの，全ての識別子名をマスクした場合（マスクパターン 4）で Precision が 0.88 と大幅に向上しており，識別子情報が誤検出の一因であると考えられる．
- **CodeLlama-7B-Instruct**: メソッド名とメソッド引数名をマスクした場合（マスクパターン 1）で Recall が 0.81，Accuracy が 0.68 と最も高くなった．全体としてマスクパターンによる性能差は小さい．
- **Codegemma-7b-it**: 全体的に Recall が著しく低く，マスクによる性能変化も僅かである．

なお，CodeLlama-7B-Instruct と Codegemma-7b-it は，プログラムに特化したモデルであるが，発表時期が古く，近年の汎用モデルと比較して性能が低い．

以上の分析から，本研究で対象としたモデル，特に高性能な GPT-4.1 と Phi-4 においては，メソッド名がクローン判定に最も大きな影響を与える識別子であると考えられる．

7. メソッド名が検出に与える影響の詳細な分析

本章では，6. 章で行った実験結果より特に性能の高かった GPT-4.1 と Phi-4 に注目し，メソッド名のマスク処理の有無による性能の変化について分析する．マスクパターン 3 とマスクパターン 4 の結果を比較し，メソッド名にマスク処理を行うことによる検出精度の変化を分析する．

メソッド名は，そのメソッドが持つ機能を要約したものであることから，メソッド名の類似度が検出精度に影響を与えると考えられる．そこで，メソッド名の類似度による検出精度への影響を検証する．また，メソッド名に使用される単語について，メソッド内での使用頻度と検出精度の影響の検証も行う．評価には Jaccard 係数，Cosine 類似度，単語出現頻度の 3 つの指標を使用する．

マスク処理の有無で判定結果がどう変化したかを 8 つのパターンに分類し，メソッド名の類似度と単語出現頻度を比較した．表 3 にその結果を示す．

7.1 分析結果のまとめ

分析結果から，GPT-4.1 と Phi-4 の挙動を「誤判定を引き起こす共通要因」と「モデル間の挙動の相違点」という 2 つの観点から分析する．

7.1.1 誤判定を引き起こす共通要因

両モデルに共通して，メソッド名の「高い類似度」と「単語の一般性」が誤判定の主要因となっている．

第一に，メソッド名の高い類似度は，コードの構造的な違いを無視させ，非クローンペアをクローンと誤判定させる最大の要因である．構造が異なるにもかかわらず一貫してクローンと誤判定されたペア（FP → FP）は，Jaccard 係数・Cosine 類似度ともに突出して高い値を示した（GPT-4.1: Jac 0.44, Cos 0.75; Phi-4: Jac 0.45, Cos 0.76）．

また，マスクによって判定が覆ったペアの分析は，メソッド名の情報が判断に影響を与えることを示している．マスクによりクローンと判定されなくなったペア（TP → FN）は，メソッド名の類似度が著しく低く（GPT-4.1: Cos 0.64, Jac 0.28; Phi-4: Cos 0.59, Jac 0.24），補助的な手がかりであった類似性の高いメソッド名の情報が失われたことで，正判定が覆ったと考えられる．逆に，マスクによって初めて正しく判定されたペア（FN → TP）も類似度が低く（GPT-4.1: Cos 0.65, Jac 0.33; Phi-4: Cos 0.66, Jac 0.32），こちらはメソッド名の不一致というノイズが除去されたことで，構造に基づいた判定が可能になったと推察される．

第二に，単語の一般性も判定に影響を与える．「get」「is」「to」のような頻出単語は判定の手がかりとなりにくく，両モデルで一貫してクローンと判定できなかったペア（FN → FN）で最も高い出現頻度（GPT-4.1: 129.56, Phi-4: 132.95）を記録した．逆に，珍しい単語は表面的な類似性を強調し，誤判定を引き起こす一因となっていた．

7.1.2 モデル間の挙動の相違点

誤判定の要因は共通しているが，単語の一般性に対する反応にはモデル間で明確な違いが見られた．

Phi-4 は，一般的な単語がノイズとなって生じたクローンペアの誤判定を，マスク処理によって訂正する能力に長けている．これは，メソッド名の類似度が低いことを非クローンと判断する強い手がかりとして用いており，マスクによって構造に基づいた正しい判定が可能になったためと考えられる．実際に，マスクを機に正しく判定されるようになったペア（FN → TP）の単語出現頻度は 119.85 と，GPT-4.1（102.24）よりも高い値を示した．しかしその一方で，特徴的な単語を手がかりとして利用する傾向があり，その情報が失われると正解を誤る依存性も見せた（TP → FN, Freq: 85.51）．

対照的に GPT-4.1 は，特徴的な単語が引き起こす非クローンペアの誤判定を修正する能力で Phi-4 を上回っていた（FP → TN, Freq: 67.05 vs 88.93）．

表 3　メソッド名のマスクパターンごとの性能評価

判定変化	GPT-4.1				Phi-4			
	Cos	Jac	N	Freq	Cos	Jac	N	Freq
クローンペア								
全体	0.73	0.46	1342	113.08	—	—	—	—
TP → TP	0.74	0.48	1142	112.01	0.75	0.50	1047	109.71
TP → FN	0.64	0.28	42	106.80	0.59	0.24	26	85.51
FN → FN	0.68	0.35	117	129.56	0.66	0.31	185	132.95
FN → TP	0.65	0.33	41	102.24	0.66	0.32	84	119.85
非クローンペア								
全体	0.65	0.31	852	96.68	—	—	—	—
TN → TN	0.61	0.27	583	102.87	0.61	0.27	583	99.23
TN → FP	0.66	0.30	34	81.35	0.68	0.31	42	105.32
FP → FP	0.75	0.44	141	94.54	0.76	0.45	165	88.38
FP → TN	0.71	0.34	94	67.05	0.70	0.34	62	88.93

注 Cos: Cosine 類似度，Jac: Jaccard 係数，
N: サンプル数，Freq: 単語出現頻度
判定変化は「マスク前→マスク後」の推移を表す

8. 妥当性への脅威

本研究における妥当性の脅威を，内的妥当性と外的妥当性に分けて述べる．

8.1 内的妥当性への脅威

本研究のマスク処理は，識別子を ‘METHOD‘のような特定の文字列に置換する一手法に過ぎない．ランダムな文字列へ置換するなど，他のマスク手法を用いた場合には異なる結果が得られる可能性がある．また，考察はメソッド名の類似度に限定しており，メソッド全体の構造や構文的な類似度が判定結果に影響を与える可能性がある．

評価に用いた FEMPDataset は，機能等価性をテストケースで検証した高品質なデータセットであるが，テストケースを作成可能な比較的単純なメソッドで構成されるており，データセットの特性が結果に影響した可能性がある．

8.2 外的妥当性への脅威

本研究は Java 言語で書かれた特定のデータセットを対象としており，他のプログラミング言語，異なるドメインや規模のデータセットに本研究の知見をそのまま適用できるかは不明である．また，評価対象とした LLM は 2025 年 6 月時点の 6 つのモデルに限られており，今後の技術進歩によって LLM の特性は変化しうる．加えて，用いたプロンプトも一例であり，異なる形式では結果が変わる可能性がある．

9. 結　　論

本研究では，LLM を用いたコードクローン検出において，識別子名が検出精度に与える影響を分析した．FEMPDataset の識別子をマスクしたデータセットを用いて 6 つの LLM の性能を評価した結果，多くのモデルで識別子をマスクしても性能が維持，あるいは向上した．特に高性能なモデル（GPT-4.1, Phi-4）では，メソッド名が判定に最も大きな影響を与えており，その高い字句的・意味的類似度が，構造的に異なる非クローンペアを誤判定する主要因であることが明らかになった．

また，メソッド名に含まれる単語の出現頻度の分析から，汎用的な単語は有効な手がかりとなりにくく，逆に珍しい単語は表面的な類似性による誤判定を助長することが判明した．

これらの知見は，LLM がコードの構造的特徴よりも表面的な識別子情報に依存する実態を定量的に示すものである．この結果は，識別子名の適切な前処理によるクローン検出システムの精度向上を示唆する．今後の課題として，選択的マスク処理の開発や，他言語への拡張などが挙げられる．

謝辞　本研究は JSPS 科研費 24H00692, 25K03102, 22H03567 の助成を受けた．

文　　献

[1] I.D. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier, "Clone detection using abstract syntax trees," Proceedings of the International Conference on Software Maintenance (ICSM), pp.368–377, 1998.

[2] M. Mondal, C. Roy, and K. Schneider, "A Fine-Grained Analysis on the Inconsistent Changes in Code Clones," Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME), pp.220–231, 2020.

[3] T. Kamiya, S. Kusumoto, and K. Inoue, "CCFinder: a multilinguistic token-based code clone detection system for large scale source code," IEEE Trans. Software Engineering, vol.28, no.7, pp.654–670, 2002.

[4] C. Roy and J. Cordy, "NICAD: Accurate Detection of Near-Miss Intentional Clones Using Flexible Pretty-Printing and Code Normalization," Proceedings of the 16th IEEE International Conference on Program Comprehension (ICPC), pp.172–181, 2008.

[5] J. Zhang, X. Wang, H. Zhang, H. Sun, K. Wang, et al., "A Novel Neural Source Code Representation Based on Abstract Syntax Tree," Proceedings of the 41st International Conference on Software Engineering (ICSE), pp.783–794, 2019.

[6] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, et al., "CodeBERT: A Pre-Trained Model for Programming and Natural Languages," Findings of the Association for Computational Linguistics: EMNLP 2020, pp.1536–1547, 2020.

[7] S.N. Pinku, D. Mondal, and C.K. Roy, "On the Use of Deep Learning Models for Semantic Clone Detection," Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME), pp.512–524, 2024.

[8] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, et al., "Large Language Models for Software Engineering: A Systematic Literature Review," ACM Trans. Softw. Eng. Methodol., vol.33, no.8, pp.1–79, Dec. 2024. https://doi.org/10.1145/3695988

[9] Gemini Team, "Gemini: A Family of Highly Capable Multimodal Models," 2023.

[10] Gemma Team, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, et al., "Gemma 3 Technical Report," 2025. https://arxiv.org/abs/2503.19786

[11] OpenAI, "GPT-4 Technical Report," 2023.

[12] M. Abdin, J. Aneja, H. Behl, S. Bubeck, R. Eldan, et al., "Phi-4 Technical Report," 2024. https://arxiv.org/abs/2412.08905

[13] A.A. Almatrafi, F.A. Eassa, and S.A. Sharaf, "Code Clone Detection Techniques Based on Large Language Models," IEEE Access, vol.13, pp.46136–46146, 2025.

[14] R. Inoue and Y. Higo, "Improving Accuracy of LLM-based Code Clone Detection Using Functionally Equivalent Methods," Proceedings of the 22nd IEEE/ACIS International Conference on Software Engineering Research, Management and Applications (SERA), pp.24–27, 2024.

[15] C. Roy and J. Cordy, "A Survey on Software Clone Detection Research," School of Computing TR 2007-541, pp.3–7, 01 2007.

[16] C. Roy, J. Cordy, and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," Science of Computer Programming, vol.74, no.7, pp.470–495, 2009.

[17] Y. Higo, "Dataset of Functionally Equivalent Java Methods and Its Application to Evaluating Clone Detection Tools," IEICE Trans. Inf. & Syst., vol.E107.D, no.6, pp.751–760, 2024.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, et al., "Attention is all you need," Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS), p.6000–6010, 2017.

[19] S. Dou, J. Shan, H. Jia, W. Deng, Z. Xi, et al., "Towards Understanding the Capability of Large Language Models on Code Clone Detection: A Survey," 2023. https://arxiv.org/abs/2308.01191

[20] J. Svajlenko, J. Islam, I. Keivanloo, C. Roy, and M. Mia, "Towards a Big Data Curated Benchmark of Inter-project Code Clones," Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME), pp.476–480, 09 2014.

[21] J. Krinke and C. Ragkhitwetsagul, "BigCloneBench Considered Harmful for Machine Learning," Proceedings of the 16th IEEE International Workshop on Software Clones (IWSC), pp.1–7, 2022.

[22] F. Al-Omari, C.K. Roy, and T. Chen, "SemanticCloneBench: A Semantic Code Clone Benchmark using Crowd-Source Knowledge," 2020 IEEE 14th International Workshop on Software Clones (IWSC), pp.57–63, 2020.

[23] A.I. Alam, P.R. Roy, F. Al-Omari, C.K. Roy, B. Roy, et al., "GPT-CloneBench: A comprehensive benchmark of semantic clones and cross-language clones using GPT-3 model and SemanticCloneBench," Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME), pp.1–13, 2023.