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

機能等価メソッドデータセットを利用した LLM によるコードクローン検出の精度向上

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　コードクローンとはソースコード中の一致または類似した部分を持つコード片で，バグの拡散の原因となるため，効率的な検出とリファクタリングが必要である．また，LLM（大規模言語モデル）を用いたコードクローン検出は，構文的な類似度の低いコードクローンに対して LLM を用いない既存ツールよりも高精度であるが改善の余地がある．そこで本研究では，FEMPDataset を用いたファインチューニングにより LLM の検出精度向上を試みた．性能評価には FEMPDataset および BigCloneBench を用いた．結果，FEMPDataset では精度向上が確認されたものの，BigCloneBench では精度の改善が見られなかった．

キーワード　コードクローン，LLM（大規模言語モデル），ファインチューニング，FEMPDataset，BigCloneBench

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る．この場合，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，コードクローンの生成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し，必要に応じてリファクタリングする必要がある．

コードクローンの検出を行うため，多くのツールが開発されてきた．既存のコードクローン検出ツールとして CCFinder [3]，NiCad [4]，NIL [5] などのツールが挙げられる．これらのツールは字句解析やメトリクスなどを用いて検出を行っている．しかし，既存のツールでは構文的な類似度の高いコードクローンの検出では高い精度を示すが，構文的な類似度の低いコードクローンの検出では精度が低いことが課題として挙げられる．

一方で LLM (大規模言語モデル) を用いたコードクローン検出では，構文的な類似度の低いコードクローンに対して既存のツールよりも高い精度での検出を実現している．LLM は主に自然言語処理の分野で高い成果を上げており，現在，多くの研究分野で注目されている．GPT といったモデルを Web サイトやアプリケーションを通して利用することができる ChatGPT は公開から約 2 か月でユーザー数が 1 億人を達するなど，多くのユーザーに利用されている．また，Meta が開発した Llama2 [6] は商用利用が可能なオープンソースとして注目されており，Llama2 の前身のモデルである Llama [7] をファインチューニングした Alpaca [8] や，Llama2 をファインチューニングした code-llama [9] などのモデルが開発されている．その他にも，Google が開発した Gemini [10] など，多くのモデルが日々開発されている [11]．

Dou らの先行研究では，複数の LLM を用いてコードクローンを検出し，その性能を LLM 以外の既存ツールと比較している [12]．NiCad や Oreo といった既存のコードクローン検出ツールでは構文的な類似度の低いコードクローンは，検出が難しく精度が悪い．一方で GPT-3.5-trubo や GPT-4，Llama2-Chat-7B といった LLM を用いたコードクローン検出では構文的な類似度の低いコードクローンに対して，LLM を用いない既存ツールよりも高い精度での検出を実現している．しかしながら，GPT-3.5-turbo や GPT-4 は構文的な類似度の低いコードクローンに対する LLM の検出精度は十分に高いとはいえず，改善の余地がある．また，Llama2 は，クローンペアとそうでないものの判断ができず，現状ほぼ全てのメソッドペアをコードクローンであると判断してしまっている．

そこで，本研究では LLM にファインチューニングを行い，構文的な類似度の低いコードクローンの検出の精度の向上を試みる．対象とする LLM は GPT-3.5-turbo と Llama2-Chat-7B，CodeLlama-7B-Instruct である．GPT-3.5-turbo のファインチューニングは OpenAI が提供する API を用いて行い，必要に応じてエポック数，バッチサイズなどのハイパーパラメータを変更する．また，LLM は推論やファインチューニングを行う際に多くの GPU メモリと時間を必要とする．よって実機での実行が必要となる Llama2-Chat-7B と CodeLlama-7B-Instruct のファインチューニングは，GPU の使用メモリ削減の理由から Lora や ZeRO といった技術を使用して行う．ファインチューニングでは，異構造で機能等価なコードクローンを集めた FEMPDataset を用いた．FEMPDataset は訓練データ，検証データ，テストデータの 3 つに分割し，学習と FEMPDataset に対する検出精度向上の評価を行う．また，評価用のデータセットとして，大規

模なコードクローン検出のベンチマークである BigCloneBench を用いて，構文的な類似度の低いコードクローンに対する検出精度を比較する．

以下，第 2 章では，本研究の準備として，コードクローンの定義や，LLM によるコードクローン検出に関する先行研究，実験で使用するデータセットについて述べる．第 3 章では，本研究の実験方法について述べる．第 4 章では，実験結果について述べる．第 5 章では，実験結果を踏まえた考察を行う．第 6 章では，本研究のまとめと今後の課題について述べる．

2. 準　　備

この章では，本研究の背景として，コードクローンの定義や，LLM によるコードクローン検出に関する先行研究，実験に使用するデータセットについて述べる．

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，主に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プログラム内に作られる [13]．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をもとに 4 つに分類し，定義した [14]．

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 BigCloneBench

BigCloneBench [15] は Svajlenko らによって作成されたコードクローン検出のベンチマークである．BigCloneBench では，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．

また，BigCloneBench では検出したクローンペアを Type-1 から Type-4 まで分類している．本研究では Type-4 のコードクローンを評価用データセットとして使用する．

2.3 FEMPDataset

FEMPDataset [16] 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．FEMPDataset はテストケースの相互実行を利用し機能等価なメソッドペアの候補を特定し，人が目視で対象の機能を持つか判定することで真に機能等価なメソッドを集めたデータセットを作成する．FEMPDataset は構文的な類似度が近いクローンペアとそうでないペアからなり，コードクローン検出ツールにおいてこれらは正しく分類，検出される必要がある．

先述の BigCloneBench と異なる点は，BigCloneBench はメソッド全体が機能等価なのではなくその一部に機能が同等と思われるコードを含んでいるのに対して，FEMPDataset はメソッド全体の機能等価を見ている点である．

2.4 LLM（大規模言語モデル）

LLM（大規模言語モデル）は，大規模なコーパスを用いて学習した言語モデルである．2017 年に Transformer と呼ばれるモデルが発表され [17]，以後モデルの大規模化が進み，学習するデータの量が増えている．

2.5 LLM のファインチューニング技術

ファインチューニングでは多くの GPU メモリが必要になる．ファインチューニングにおいて GPU メモリを削減するいくつかの技術について述べる．

2.5.1 Lora（Low-Rank Adaptation）

Lora（Low-Rank Adaptation）[18] は，ファインチューニング時に変更するパラメータ数を削減し，少ないリソースでファインチューニングを実現する技術である．

Lora では，ファインチューニング前後のパラメータの差分を低ランク行列で近似することで，パラメータ数を削減し効率的に学習する．また，ハイパーパラメータ r を指定し低ランク行列の大きさを決定する．r が小さいほどパラメータ数が削減される．

2.5.2 ZeRO（Zero Redundancy Optimizer）

ZeRO（Zero Redundancy Optimizer）[19] は複数の gpu を活用することで学習に必要な 1GPU あたりに必要な GPU メモリを削減する技術である．

ファインチューニング時には，学習率の動的変化の管理の情報，勾配，重みを GPU 上に保持し，計算する必要がある．そこで ZeRO では，各 GPU が LLM の特定の層に対してパラメータ変更を行い，その結果をまとめて LLM 全体のパラメータの変更とする．各 GPU は特定の層に対するパラメータ変更のみを担当することから，従来の手法よりも 1GPU あたりに必要な GPU メモリを削減することができる．

2.6 検出精度指標

コードクローン検出や LLM の性能指標には Recall，Precision，Accuracy が用いられる．これらのメトリクスの意味と計算式は以下の通りである．

Recall

実際にクローンであるメソッドペアのうち，コードクローンであると判定されたメソッドペアの割合．

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

Precision

コードクローンであると判定されたメソッドペアのうち，実際にコードクローンであるメソッドペアの割合．

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

Accuracy

正しい判定をしたメソッドペアの割合.

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

また，TP（True Positive），FP（False Positive），FN（False Negative），TN（True Negative）は以下を表す.

TP：実際にコードクローンであるメソッドのうち，コードクローンと判定されたメソッドペア数.

FP：実際にコードクローンでないメソッドのうち，コードクローンと判定されたメソッドペア数.

FN：実際にコードクローンでないメソッドのうち，コードクローンでない判定されたメソッドペア数.

TN：実際にコードクローンであるメソッドのうち，コードクローンでないと判定されたメソッドペア数.

2.7 BigCloneBench と LLM に関する先行研究

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，Shihan Dou らの研究 [12] を紹介する．Dou らの研究では，単一のプロンプトを入力とし，LLM を用いない既存のツールと LLM を用いた場合の性能を BigCloneBench を用いて比較している.

結果を見ると，構文的な類似度の高いコードクローンは既存のツールの方が高い Recall を示しているが，構文的な類似度の低いコードクローンは LLM の方が高い Recall を示している.

また，GPT-3.5-turbo と GPT-4 は構文的類似度が低い Type-4 コードクローンの検出において既存ツールより高い精度を示すが，十分に高いとはいえず，改善の余地がある．Llama2-chat-7B は高い Recall を示すが Precision が低く，ほぼ全てのメソッドペアをクローンペアと認識してしまっている．そこで，本研究では LLM のファインチューニングによる精度向上を目指す.

3. 実　　験

この章では，本研究の具体的な実験方法について述べる.

実験対象とする LLM は以下の通りである.

- GPT-3.5-turbo
- Llama2-Chat-7B
- CodeLlama-7B-Instruct

また，実験では，ファインチューニングを用いて構文的な類似度の低いコードクローンに対する検出精度の向上を試みる．実験は以下の手順で行う.

STEP1：ファインチューニング

FEMPDataset を用いて，LLM のファインチューニングを行う.

STEP2：LLM の実行

ファインチューニング前後の LLM に対して，FEMPDataset のテストデータ，BigCloneBench のメソッドペアを与え，回答を"Yes"または"No"で得る.

STEP3：性能評価

ファインチューニング前後の LLM の回答を集計し，性能の比較を行う.

3.1 ファインチューニング

ファインチューニングには FEMPDataset を用いる．FEMPDataset は異構造で機能等価なメソッドペアを集めたデータセットである．ファインチューニングのため，データセットは訓練データ，検証データ，テストデータの 3 つに分割する．ファインチューニングは訓練データと検証データを，性能評価はテストデータを用いて行う．各データのメソッドペア数の内訳は表 1 の通りである.

GPT-3.5-turbo のファインチューニングは，OpenAI API を用いて行う．Llama2-Chat-7B と CodeLlama-7B-Instruct ファインチューニングは，GPU メモリ削減のため，Lora と ZeRO を利用して実行した．2.5 節で説明した Lora のハイパーパラメータ r は 2 とした．学習には Quadro RTX 8000 と Tesla V100 の 2 つの GPU を用いた.

3.2 LLM の実行

ファインチューニング前とファインチューニング後の LLM の性能評価は以下の手順で行う.

（1） **BigCloneBench のメソッドペアを抽出**

BigCloneBench から T4 に分類されるクローンペアと，クローンではないペアを抽出する．この 2 種類のメソッドペアを正しく分類することができるかを評価する.

（2） **プロンプトの作成**

抽出したメソッドペアはクローンペアであるかどうかを質問するプロンプトに変換し，LLM に入力する形式にする.

（3） **プロンプトの入力，回答の取得**

ファインチューニング前の LLM とファインチューニング後の LLM に対して，作成したプロンプトを入力し，"Yes"または"No"の回答を得る.

3.2.1 評価用データセット

性能評価は BigCloneBench を用いて行う．BigCloneBench から構文的な類似度の低い T4 に分類されるクローンペアと，クローンではないペアを用いて性能評価を行う.

BigCloneBench に含まれる T4 のクローンペア数は 7,729,291 と膨大である．このため，データセットからランダムに 2,000

表 1　分割後の各データのメソッドペア数の内訳

	訓練データ	検証データ	テストデータ
メソッドペア数	1,755	220	219
			計 2,194

個のメソッドペアを抽出し，実験に用いることとした．また，クローンではないペアも同様に 2,000 個抽出し，実験に用いる．したがって，実験に用いるデータセットのメソッドペア数の内訳は表 2 の通りである．

ただし，メソッドペアの 2 つを Llama のトークナイザーでトークナイズした際のトークン数の合計が 10,000 トークンを超えるメソッドペアは，メソッドペア抽出の際に対象から除外した．これは，Llama を実行する際に 10,000 トークンを超えるプロンプトを入力すると，GPU サーバーがメモリ不足で動かないためである．メソッドの抽出はランダム性を確保するために BigCloneBench のデータセットに格納されているデータからランダムにメソッドを順に取り出し，10,000 トークン以内のメソッドペアであれば，評価用のメソッドペアの一つとして採用した．これを 2,000 ペア取り出すまで繰り返した．クローンペアは 2,068 ペアを抽出し 2,000 ペアを，クローンではないペアは 2,037 ペアを抽出し 2,000 ペアを最終的に取得した．

3.2.2 プロンプト

LLM に入力するプロンプトは"system"と"user"，2 つのロールで構成される．system では，"Yes"または"No"のどちらかで回答するように回答方法を指定する．user では，提示した 2 つのメソッドが，クローンペアであるかを判定をするように指示を記述する．

3.3 性能評価

LLM から得た回答を集計し，Recall，Precision，Accuracy の 3 値を計算し性能を評価する．評価に使用するデータセットは，FEMPDataset のテストデータ計 219 ペアと BigCloneBench のメソッドペア計 4,000 ペアの 2 つである．

4. 実験結果

この章では，実験の結果について述べる．実験結果を FEMPDataset と BigCloneBench の 2 つのデータセットを用いて実行する．

4.1 FEMPDataset によるファインチューニングの評価

GPT-3.5-turbo と Llama2-Chat-7B，CodeLlama-7B-Instruct について，ファインチューニング前後の評価比較を FEMPDataset のテストデータを用いて行った．

4.1.1 GPT-3.5-turbo の FEMPDataset による評価

ファインチューニング前の GPT-3.5-turbo，ファインチューニング後の GPT-3.5-turbo，GPT-4-turbo の Recall，Precision，Accuracy は表 3 の通りである．また，以上の結果をグラフにまとめたものを図 1 に示す．

ファインチューニング後の GPT-3.5-turbo とファインチューニング以前の GPT-3.5-turbo を比較すると，Precision が大きく向上している．また，Recall はほぼ横ばいとなっている．このことから，ファインチューニングによってクローンでないペアを正しく判定するようになったことがわかる．

表 2　評価用データセットのメソッドペア数の内訳

	クローンペア	クローンではないペア
メソッドペア数	2,000	2,000
		計 4,000

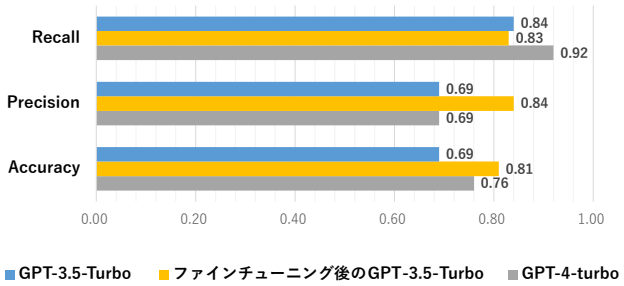

図 1　FEMPDataset による GPT-3.5-turbo と GPT-4-turbo の評価比較

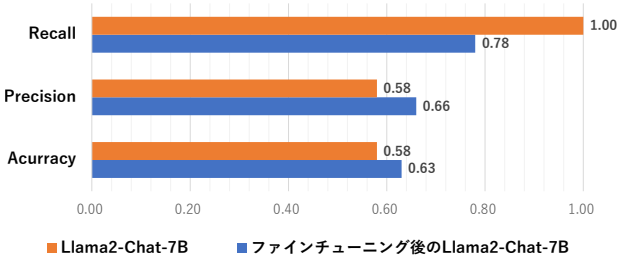

図 2　FEMPDataset による Llama2-Chat-7B の評価比較

4.1.2 Llama2-Chat-7B の FEMPDataset による評価

ファインチューニング前とファインチューニング後の Llama2-Chat-7B の Recall，Precision，Accuracy は表 4 の通りである．また，以上の結果をグラフにまとめたものを図 2 に示す．

ファインチューニング前の Llama2-Chat-7B では，すべてのメソッドペアをクローンペアと認識していた．しかし，ファインチューニング後の Llama2-Chat-7B では，一部のクローンでないペアを正しく判定することができるようになった．結果，Recall が低下したが，Precision が向上した分，Accuracy は向上しており，ファインチューニング前の Llama2-Chat-7B よりも性能が向上していることがわかる．

4.1.3 CodeLlama-7B-Instruct の FEMPDataset による評価

ファインチューニング前とファインチューニング後の CodeLlama-7B-Instruct の Recall，Precision，Accuracy は表 5 の通りである．また，以上の結果をグラフにまとめたものを図 3 に示す．

表 3　ファインチューニング前後の GPT-3.5-turbo と GPT-4-turbo の評価

モデル名	Recall	Precision	Accuracy
GPT-3.5-turbo	0.84	0.69	0.69
FT[1]後の GPT-3.5-turbo	0.83	0.84	0.81
GPT-4-turbo	0.92	0.69	0.76

表 4　ファインチューニング前後の Llama2-Chat-7B の評価

モデル名	Recall	Precision	Accuracy
Llama2-Chat-7B	1.00	0.58	0.58
FT 後の Llama2-Chat-7B	0.78	0.66	0.63

（注1）：FT はファインチューニングの略

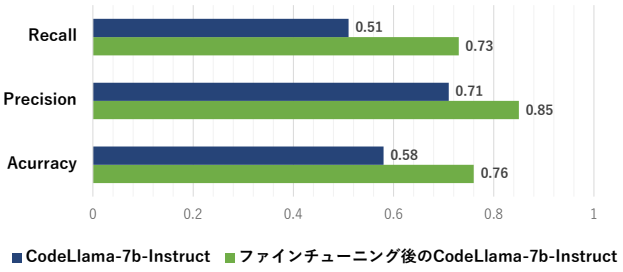

図 3　FEMPDataset による CodeLlama-7B-Instruct の評価比較

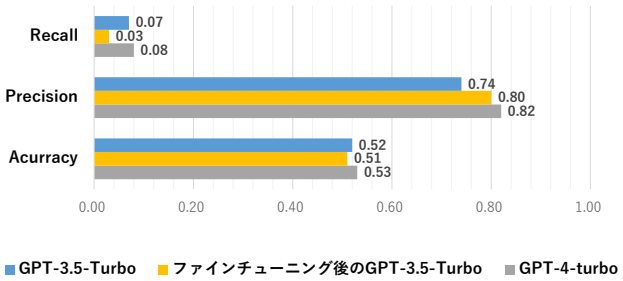

図 4　BigCloneBench による GPT-3.5-turbo と GPT-4-turbo の評価比較

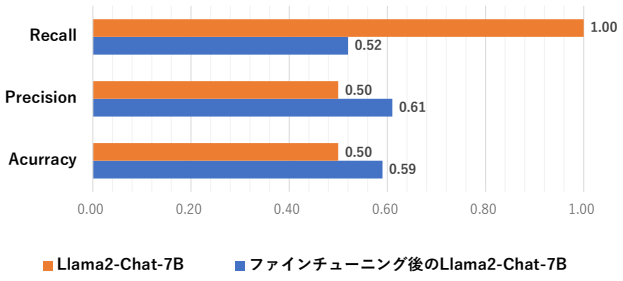

図 5　BigCloneBench による Llama2-Chat-7B の評価比較

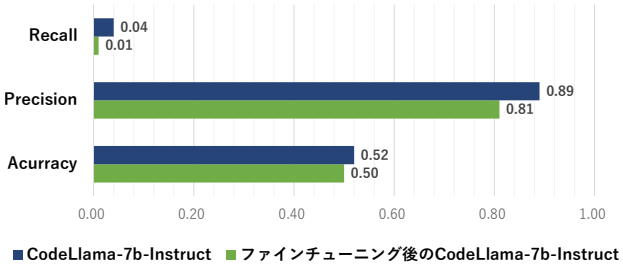

図 6　BigCloneBench による CodeLlama-7B-Instruct の評価比較

4.2　BigCloneBench によるファインチューニングの評価

GPT-3.5-turbo と Llama2-Chat-7B，CodeLlama-7B-Instruct について，ファインチューニング前後の評価比較を BigCloneBench を用いて行った．対象のデータは BigCloneBench の T4 に分類されるクローンペアと，クローンでないペア各 2,000 ペア，計 4,000 ペアである．

4.2.1　GPT-3.5-turbo の BigCloneBench による評価

ファインチューニング前の GPT-3.5-turbo，ファインチューニング後の GPT-3.5-turbo，GPT-4-turbo の Recall，Precision，Accuracy は表 6 の通りである．また，結果をグラフにまとめたものを図 4 に示す．

ファインチューニング後の GPT-3.5-turbo とファインチューニング以前の GPT-3.5-turbo を比較すると，Precision が少し向上しているが，あまり大きな変化は見られなかった．また，GPT-4-turbo と比較すると，Recall，Precision，Accuracy の全てで劣っている．

4.2.2　Llama2-Chat-7B の BigCloneBench による評価

ファインチューニング前の Llama2-Chat-7B，ファインチューニング後の Llama2-Chat-7B の Recall，Precision，Accuracy は表 7 の通りである．また，結果をグラフにまとめたものを図 5 に示す．

ファインチューニング前の Llama2-Chat-7B は BigCloneBench のすべてのメソッドペアをクローンペアであると認識していた．しかし，ファインチューニング後の Llama2-Chat-7B は一部のクローンでないペアを正しく判定することができるようになった．結果，FEMPDataset と同じように，Recall は低下したが Precision が向上し，Accuracy は向上が見られた．

4.2.3　CodeLlama-7B-Instruct の BigCloneBench による評価

ファインチューニング前の CodeLlama-7B-Instruct，ファインチューニング後の CodeLlama-7B-Instruct の Recall，Precision，Accuracy は表 8 の通りである．また，結果をグラフにまとめたものを図 6 に示す．

ファインチューニング前の CodeLlama-7B-Instruct は BigCloneBench のすべてのメソッドペアをクローンペアであると認識していた．しかし，ファインチューニング後の CodeLlama-7B-Instruct は一部のクローンでないペアを正しく判定することができるようになった．結果，FEMPDataset と同じように，Recall は低下したが Precision が向上し，Accuracy は向上が見られた．

表 5　ファインチューニング前後の CodeLlama-7B-Instruct の評価

モデル名	Recall	Precision	Accuracy
CodeLlama-7B-Instruct	0.51	0.71	0.58
FT 後の CodeLlama-7B-Instruct	0.73	0.85	0.76

表 6　ファインチューニング前後の GPT-3.5-turbo と GPT-4-turbo の評価比較

モデル名	Recall	Precision	Accuracy
GPT-3.5-turbo	0.07	0.74	0.52
FT 後の GPT-3.5-turbo	0.03	0.80	0.51
GPT-4-turbo	0.08	0.82	0.53

表 7　ファインチューニング前後の Llama2-Chat-7B の評価比較

モデル名	Recall	Precision	Accuracy
Llama2-Chat-7B	1.00	0.50	0.50
FT 後の Llama2-Chat-7B	0.52	0.61	0.59

表 8　ファインチューニング前後の CodeLlama-7B-Instruct の評価比較

モデル名	Recall	Precision	Accuracy
CodeLlama-7B-Instruct	0.04	0.89	0.52
FT 後の CodeLlama-7B-Instruct	0.01	0.81	0.50

5. 考　　察

この章では，実験結果について順に考察する．

5.1 GPT-3.5-turbo, CodeLlama-7B-Instruct で BigCloneBench に対して精度向上が見られなかった理由

GPT-3.5-turbo と CodeLlama-7B-Instruct を FEMPDataset を使用してファインチューニングを行った．しかし，BigCloneBench に対して精度向上が見られなかった．

これは，FEMPDataset と BigCloneBench のデータセットが異なるためであると考える．

BigCloneBench 内のデータセットのクローンペアは，等価な機能の持つメソッドペアを抽出しているため，メソッド全体として完全に機能的に等価であるとは限らない．一方で，ファインチューニングに使用した FEMPDataset は，異構造で機能等価なメソッドペアを集めたデータセットであり，メソッド全体で同じ結果を出力する完全に機能等価なメソッドペアを集めたデータセットである．

この 2 つのデータセットの特徴の違いが，ファインチューニングの効果に影響を与えたと考えられる．

5.2 Llama2-Chat-7B で BigCloneBench に対して精度向上が見られた理由

Llama2-Chat-7B は学習以前，BigCloneBench に対して Yes を返すだけのモデルで，コードクローンに対しての理解に乏しかった．しかし，ファインチューニングを行うことで，BigCloneBench に対して精度向上が見られた．このことから，本研究のファインチューニングによって Llama2-Chat-7B がコードクローンに対しての理解を向上させたことがわかる．

6. まとめと今後の課題

本研究では，FEMPDataset を用いて GPT-3.5-turbo にファインチューニングを行い，BigCloneBench に対するコードクローン検出性能の向上を試みた．GPT-3.5-turbo に対してファインチューニングを行った結果，訓練に使用した FEMPDataset に対してモデルが適合したことが確認できた．BigCloneBench を用いた性能評価においては，Llama2-Chat-7B では精度向上が見られたが，GPT-3.5-turbo では精度向上が見られなかった．

今後の課題として，以下の 3 点が挙げられる．

調査対象の追加
: 本研究で扱わなかったモデルに対しても同様の実験を行うことで，モデル間の性能比較を行うことができると考える．

BigCloneBench 以外のベンチマークに対しての性能評価
: 本研究で使用した BigCloneBench 以外のベンチマークに対しても同様の実験を行うことで，ファインチューニングの性能評価をより詳細に行うことができると考える．

プロンプトの工夫
: 本研究では，プロンプトの工夫は行わなかった．今後，Chain of thought [20] などの技術を用いてプロンプトを改善することで，モデルの性能向上が見込めると考える．

謝辞　本研究は JSPS 科研費（JP20H04166，JP21K18302，JP21K11829，JP21H04877，JP22H03567，JP22K11985）の助成を得て行われた．

文　　献

[1] 井上克郎, 神谷年洋, 楠本真二. コードクローン検出法. コンピュータソフトウェア, Vol. 18, No. 5, pp. 529–536, 2001.

[2] M. Mondal, C. Roy, and K. Schneider. A Fine-Grained Analysis on the Inconsistent Changes in Code Clones. In *2020 IEEE 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. In *2008 16th IEEE International Conference on Program Comprehension*, pp. 172–181, 2008.

[5] T. Nakagawa, Y. Higo, and S. Kusumoto. NIL: large-scale detection of large-variance clones. ESEC/FSE 2021, p. 830 − 841, New York, NY, USA, 2021. Association for Computing Machinery.

[6] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, and ⋯ T. Scialom. Llama 2: Open Foundation and Fine-Tuned Chat Models, 2023.

[7] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. A. Lachaux, B. Rozière ⋯, and G. Lample. Llama: Open and efficient foundation language models, 2023.

[8] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, and T. ⋯ Hashimoto. Alpaca: A strong, replicable instruction-following model. Stanford Center for Research on Foundation Models, 2023.

[9] B. Rozière, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, and ⋯ G. Synnaeve. Code Llama: Open Foundation Models for Code, 2023.

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

[11] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, and ⋯ J. R. Wen. A Survey of Large Language Models, 2023.

[12] S. Dou, J. Shan, H. Jia, W. Deng, Z. Xi, W. He, and ⋯ X. Huang. Towards Understanding the Capability of Large Language Models on Code Clone Detection: A Survey. 2023.

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

[14] 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.

[15] J. Svajlenko, J. Islam, I. Keivanloo, C. Roy, and M. Mia. Towards a Big Data Curated Benchmark of Inter-project Code Clones. pp. 476–480, 09 2014.

[16] 肥後芳樹. 自動テスト生成技術を利用した機能等価メソッドデータセットの構築. ソフトウェアエンジニアリングシンポジウム 2023 論文集, Vol. 2023, pp. 30–38, 08 2023.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, and ⋯ I. Polosukhin. Attention Is All You Need, 2023.

[18] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-Rank Adaptation of Large Language Models, 2021.

[19] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He. ZeRO: Memory optimizations Toward Training Trillion Parameter Models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16, 2020.

[20] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, and ⋯ D. Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, 2023.