Rust 言語のための自動プログラム修復性能の学習データ依存性の評価

高橋颯ノ介[1]　森康祐[1]　廣津登志夫[1]

1. はじめに

大規模言語モデル (LLM) による自動プログラム修復 (APR) はプログラムコードを学習し，修正コードを予測し出力するもので，Rust 言語含む複数のプログラム言語に対応した LLM モデルが存在する．Rust 言語は Linux カーネルに正式に採用されるなど，利用場面が増加しているが，同分野における Rust 言語の学習用データセット，ベンチマークは少なく，Rust 言語の LLM-APR は少量である．関連研究 [1] では，fix，bug といった単語による変更差分のフィルタを行うが，真に学習に有用なデータ量を保証しない．このデータセットを用いた Silva らの研究の RepairLLaMA[2] は，Java 言語の関数内のコードの誤りを対象とした修正パッチの生成タスクについて，修正前のコードに存在するバグ箇所の指定と，指定部のみ修正後したコードを最適な学習ペアと示した．Rust 言語で RepairLLaMA の学習形式が機能するか，fix，bug といった単語以外に修正性能を向上させるフィルタが存在するか検証するため，ソースコードプラットフォームの GitHub からコミットデータを収集，コミットメッセージによるフィルタによってデータセットを作成，学習したモデルのベンチマークスコアを観察する．本論文の主な貢献は，RepairLLaMA の Rust 言語における性能の検証である．

2. 実験方法

2.1 データ収集と学習方法

8642 リポジトリから，RepairLLaMA と同等の単一関数修正データを Rust 言語で 20286 件，単一関数修正ファイルデータを約 33 万件入手した．学習データの抽出には，Python 言語の pydriller ライブラリを用いる．Meta 社の LLM である llama2 をプログラムコードで追加学習した CodeLlama-7B と LoRA 学習を中心的に扱い，RepairLLaMA の GitHub 公開実装を利用する．実験環境の制約として LoRA 学習オプションの勾配累積を 8，訓練バッチサイズは 2 と設定した．

[1] 法政大学
Hosei University

2.2 学習データの条件設定

学習データについて，コミットメッセージで検知する単語，コミット中の Rust ファイル数，データ件数，備考を表 1 に示した．条件 A では，言語間での性能の変化の観察のため，Java 言語と Rust 言語で，fix，bug，error，issue を含む，修正行数 40 行以下，1 つの Java，Rust ファイル内の単一関数修正コミットデータを用いる．条件 B，C，D では，有効的な学習データの構成を発見するため，コミットメッセージ制約とファイル数の制約を変更する．条件 B は，収集したデータ自体の性質を観察するため，単一関数修正ファイル約 33 万件のデータセットからランダムに学習データを選択する．条件 C は，コミットに 1 つのファイルしか含まない単一関数修正コミット約 57000 件を学習データに用いる．条件 D では，データセット中のコミットメッセージにおける単語の出現回数が，fix に次いで多い add と，その関連語として delete，modify の 3 つの単語を含むコミットを学習データに用いる．また，条件 A を除いて，過学習防止のため，出典重複リポジトリの除去を行い，学習件数とスコアの関係を観察するため，データセットを 1000 件，2000 件，4000 件，8000 件と段階的に増やす．

表 1　各条件の設定項目

	検知する単語	Rust ファイル数	データ件数	備考
条件 A	fix,bug,error,issue	1 つ	20286 件	Java 言語と比較
条件 B	指定無し	1 つ以上	331823 件	重複データ削除
条件 C	指定無し	1 つ	57205 件	重複データ削除
条件 D	add,delete,modify	1 つ以上	42709 件	重複データ削除

2.3 ベンチマーク選定

Rust 言語と Java 言語における推論性能の差を観察するため，単一関数修正タスクを含む HumanEvalFix と xCodeEval を採用した．HumanEval は LLM のコード生成機能を評価する多言語ベンチマークであり，HumanEvalFix は関数の修正タスクを専門とし，各言語に対して 164 問の問題が用意されている．xCodeaEval は競技プログラミングプラットフォームの Codeforces 由来のデータセットとベンチマークであり，本論文では Rust 言語で 31 問，Java 言語で 237 問を作成して評価に用いる．この 2 つ問題集は GitHub と独立して作成されたベンチのため，学習データを汚染しない．評価は LLM が推論した 10 個の修正候補の

© 2026 Information Processing Society of Japan

うち，少なくとも 1 つが全テストを通過した場合に正答と判定する方法を採用する．

3. 結果と考察

3.1 Rust 言語と Java 言語での性能比較

条件 A にて，Java 言語と Rust 言語でのスコアを表 2 に示す．

表 2 条件 A における生成 10 候補の正答率

	ベースモデル	1000 件学習	2000 件学習	4000 件学習	8000 件学習
HumanEvalFix-Java(164)	0.805	0.842	0.842	0.848	0.860
xCodeEval-Java(237)	0.116	0.228	0.240	0.259	0.263
HumanEvalFix-Rust(164)	0.592	0.811	0.762	0.793	0.750
xCodeEval-Rust(31)	0.250	0.438	0.438	0.438	0.469

HumanEvalFix では Rust 言語ではベースモデルで性能が 59.2%，学習後のスコアは 15.8%〜21.9%程向上する．xCodeEval では，Rust 言語ではベースモデルで 25%を記録し，学習後の Rust 言語のスコアは Java 言語より約 20%高い．Rust 言語のスコアが Java 言語より高い点に関して，xCodeEval の Rust ベンチマーク数が 31 問と少なく，特定の環境下での定性的な傾向として位置付けることが妥当であると考えられ，ベンチマークの追加を検討する．

3.2 条件 A，B，C，D のスコア比較

各条件について，比較のためベンチマーク毎にスコアを表 3，表 4 に示す．

表 3 HumanEvalFix におけるベンチマークスコア

	1000 件学習	2000 件学習	4000 件学習	8000 件学習
条件 A	0.811	0.762	0.793	0.750
条件 B	0.799	0.780	0.732	0.750
条件 C	0.793	0.805	**0.799**	0.750
条件 D	**0.817**	**0.817**	0.750	**0.780**

表 4 xCodeEval におけるベンチマークスコア

	1000 件学習	2000 件学習	4000 件学習	8000 件学習
条件 A	0.438	0.438	0.438	0.469
条件 B	0.434	**0.484**	**0.516**	0.452
条件 C	0.412	0.471	0.382	0.412
条件 D	**0.548**	**0.484**	0.419	**0.484**

結果，両ベンチマークで条件 D が条件 A の性能を複数の学習件数 LoRA にて上回ることが観測された．全体の結果で，ベンチマークスコアと学習件数の間に単調な増加傾向が見られない点について，1000 件未満の件数でのスコアを検証するため，条件 D のデータ 300 件学習時と 600 件学習時のスコアを調査した．

表 5 条件 D のデータ 300 件，600 件で学習した LoRA のスコア

	ベースモデル	300 件学習	600 件学習	1000 件学習
HumanEvalFix(164)	0.592	0.640	0.774	0.817
xCodeEval(31)	0.250	0.322	0.452	0.548

学習データ 1000 件以下ではスコアが単調に増加し，1000 件以上ではスコアは不安定な横ばいとなった．次に，過学習の傾向の確認のため，8000 件学習時の学習曲線を示す．横軸はエポック数，縦軸は損失として，生の訓練損失の数値を Raw train，評価ステップまでの平均訓練損失を train mean，評価損失を Eval としてプロットする．

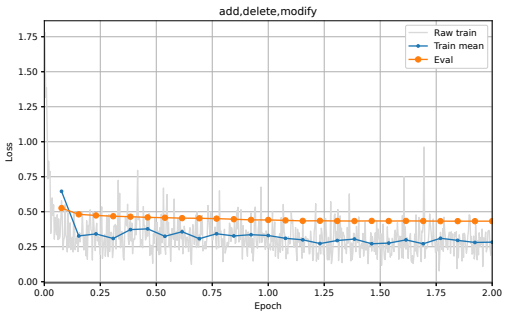


図 1 条件 D，8000 件学習時の学習曲線

一般に訓練損失と評価損失の低い値での収束が安定した学習とされ，直ちに過学習と判断できるようなグラフの振幅，値の変化はグラフから観測できないが，生の訓練損失値の振動が見られる．今後の課題として，条件 D における性能向上の要因と，スコアの不安定な遷移の原因解明が挙げられる．

4. 結論

Rust 言語の学習データを収集して RepairLLaMA のフレームワークを利用する場合，Java 言語同様に学習と推論が可能である．add，delete，modify の単語でコミットメッセージをフィルタした場合，4 種中 3 種の学習データ数の LoRA でベースラインよりスコアが高い．実験全体を通して，Java 言語に比べて Rust 言語では学習件数と性能の向上に不安定な関係が見られ，今後の調査課題として残る．

参考文献

[1] Monperrus, M., Martinez, M., Ye, H., Madeiral, F., Durieux, T. and Yu, Z.: Megadiff: A Dataset of 600k Java Source Code Changes Categorized by Diff Size (2021).
[2] Silva, A., Fang, S. and Monperrus, M.: RepairLLaMA: Efficient Representations and Fine-Tuned Adapters for Program Repair, *IEEE Transactions on Software Engineering*, Vol. 51, No. 8, p. 2366–2380 (online), DOI: 10.1109/tse.2025.3581062 (2025).

© 2026 Information Processing Society of Japan