

The Impact of Fine-tuning Large Language Models on Automated Program Repair

Roman Macháček*
University of Bern
Bern, Switzerland
roman.machacek@unibe.ch

Anastasiia Grishina
Simula Research Laboratory
Oslo, Norway
anastasiia@simula.no

Max Hort
Simula Research Laboratory
Oslo, Norway
maxh@simula.no

Leon Moonen
Simula Research Laboratory
Oslo, Norway
leon.moonen@computer.org

Proceedings of the 41th International Conference on Software Maintenance and Evolution (ICSME2025)

背景 | 自動プログラム修正 (APR)

ソフトウェア開発者が直面する課題

- コード内の意図しないバグの発生
 - 単一な1行のエラー, 複数ファイルの依存関係によるバグなど. . .
- 金銭的にも時間的にもコストのかかるプロセス^[1]

自動プログラム修正 (APR)

- 開発者が遭遇するバグを自動で修正
- バグの種類や複雑によって多くの手法が存在
- 近年はLLMを使用することに基づいている

[1] A. Arcuri. "On the Automation of Fixing Software Bugs." In: Companion of the 30th International Conference on Software Engineering. Association for Computing Machinery, 2008, pp. 1003–1006. DOI:10.1145/1370175.1370223.

LLMのファインチューニング

LLMを特定のタスクにファインチューニング

- LLM自体の性能をさらに向上させることができる
- 問題点^[11]
 - 大規模データセットとモデルに起因する計算上のオーバーヘッド
 - データセットへのモデルの過学習

ファインチューニングの有効戦略

- パラメータ効率良いファインチューニング技術の使用
 - パラメータの数を制限して訓練する
- モデルが能力を失い、性能が低下してしまう

効率向上と能力損失のトレードオフが発生

[11] J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu, and R. McHardy. Challenges and Applications of Large Language Models. 2023. DOI: 10.48550/arXiv.2307.10169. arXiv: 2307.10169 [cs].

目的とリサーチクエスチョン

目的

- LLMのAPRによるファインチューニング（FT）の影響を検証

リサーチクエスチョン（RQ）

- RQ1：FTを行わない場合のLLMの性能は？
- RQ2：LLMのFTすることでの影響は？
- RQ3：パラメータ効率の良いFT（PEFT）の影響は？
- RQ4：LoPAのハイパーパラメータでのFTの影響は？

データセットとモデル

ファインチューニング用データセット

- CLMデータセット
 - Github上のJavaプロジェクトから収集された約14万件のバグ修正ペア
- トレーニング：テストを8：2でファインチューニング

テスト用データセット

- Defect4J：219件のバグ
- QuixBugs：40件のプログラムで見つかったバグ
- HumanEval-Java：HumanEvalの163件のバグ

モデル

- CodeGen, CodeT5, StarCoder, DeepSeekCoder, Bloom, CodeLlama2

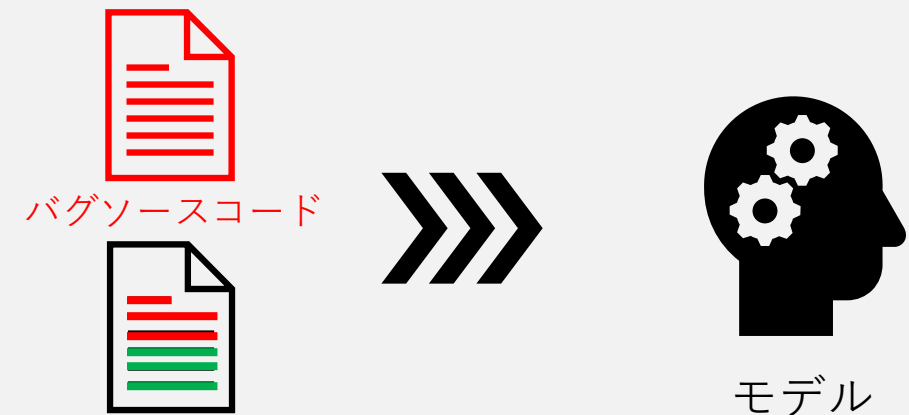
実験の詳細①

1. CLSデータセットを収集



Github上のJavaプロジェクトから約14万件の
バグ修正ペア（ソースコードとパッチ）を収集

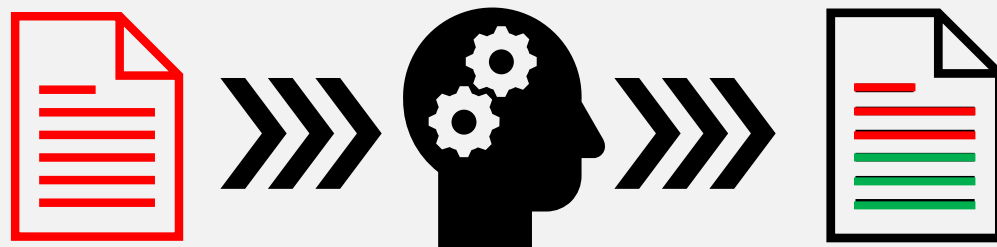
2. モデルをファインチューニング



パッチ

1で収集したデータセットを使って
モデルをファインチューニング

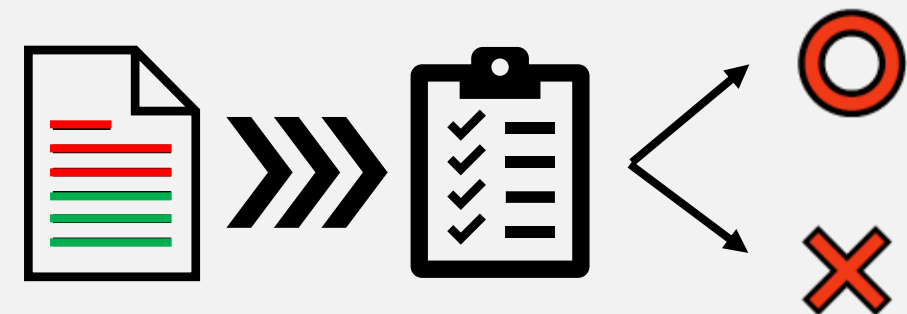
3. 実世界のバグを使ってパッチを生成



FTモデルを使って、Defect4Jなどの
実世界のバグからパッチを生成

△RQ1ではFTなしモデルを使用

4. パッチがテストに通りかを確認



テストケース

生成されたパッチがテストに通りかどうかを検証

実験の詳細②

2つのファインチューニングシナリオを考慮

1. バグのある行を含む
2. バグのある行を含まない

```
// CodeGEN:
public static int
    bitcount(int n) {
    int count = 0;
    while (n != 0) {
        // bug start:
        n = (n ^ (n - 1)); ←2では削除
        // bug end
        count++;
    }
    return count;
}
// fix:
n = (n & (n - 1));
```

- ファインチューニングする際に、バグのある行は必要なのかを検証 (RQ1)

実験評価

ベンチマーク問題に対して10個のパッチを生成し、単体テストで評価

妥当 (P)	コンパイル, 全てのテストに正常に合格
タイムアウト (T)	コンパイルの実行が時間制限を超過した
コンパイル不能 (U)	コンパイル・実行できなかった
間違い (W)	正常にコンパイルできたが, テストに合格しなかった
不明 (UNK)	空のパッチが原因で生じる固有のエラー

正解のパッチと生成されたパッチの類似度を評価

- CodeBLEU^[64]を使用

[64] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. "Bleu: A Method for Automatic Evaluation of Machine Translation." In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, 2002, pp. 311–318. DOI: 10.3115/1073083.1073135.

RQ1：ファインチューニングなし性能

- 各モデルが達成した妥当なパッチの数を記載

Model	Without buggy line			With buggy line		
	QB	HE	D4J	QB	HE	D4J
Bloom-560m	1	10	8	4	5	6
Bloom-1b7	1	15	10	1	12	8
Bloom-7b1	6	24	10	4	22	14
CodeGen-350M	6(7)	25(30)	12(3)	4	16	7
CodeGen-2B	13(15)	44(49)	20(4)	12	34	20
CodeGen-6B	16(16)	42(46)	18(8)	16	42	18
CodeLlama2-7b	33	95	83	28	96	93
CodeT5-small	0(3)	4(3)	10(2)	0	1	4
CodeT5-base	0(0)	4(5)	8(4)	0	0	2
CodeT5-large	2(3)	2(6)	4(1)	1	3	4
DeepSeekCoder-1.3b	33	94	72	31	97	86
DeepSeekCoder-6.7b	33	107	89	30	105	103
StarCoder-1b	22	69	62	22	70	74
StarCoder-3b	32	94	63	31	85	87
StarCoder-7b	33	91	82	31	91	96

- パラメータ数が多いほど性能は向上
- 最も高いモデル：DeepSeekCoder6.7B
QuixBugs：82.5% (33/40)
HumanEval：65.6% (107/163)
Defect4J：40.6% (89/219)

- バグのある行を含めることの効果
 - 29/45で性能が向上しなかった
 - モデルの事前学習フェーズで問題がある??

RQ2：ファインチューニングあり性能

- LLMがAPRタスク向けにFTにより性能が向上するかを検証
 - エPOCH数ごとに記録し，損失，CodeBLEU，Exact Matchで性能を確認

	CodeBLEU (↑)					Loss (↓)					Exact Match (↑)				
	Validation Epoch					Validation Epoch					Validation Epoch				
	0	1	2	3	Train	0	1	2	3	Train	0	1	2	3	Train
Bloom-560m	0.45	0.49	0.54	0.57	0.73	1.21	0.88	0.74	0.76	0.25	0.00	0.06	0.13	0.19	0.53
Bloom-1b7	0.46	0.53	0.58	0.60	0.84	1.18	0.68	0.59	0.66	0.14	0.00	0.08	0.16	0.21	0.69
CodeGen-350M	0.41	0.57	0.61	0.62	0.86	1.06	0.49	0.46	0.55	0.13	0.00	0.11	0.18	0.22	0.70
CodeGen-2B	0.38	0.53	0.59	0.61	0.88	0.90	0.58	0.52	0.64	0.13	0.00	0.10	0.19	0.23	0.79
CodeT5-small	0.05	0.38	0.39	0.38	0.39	4.37	0.69	0.66	0.65	0.59	0.00	0.03	0.05	0.05	0.06
CodeT5-base	0.07	0.36	0.38	0.39	0.45	4.78	0.57	0.53	0.52	0.33	0.00	0.08	0.12	0.14	0.25
CodeT5-large	0.01	0.36	0.42	0.43	0.52	4.38	0.56	0.50	0.50	0.23	0.00	0.12	0.19	0.23	0.41
DeepSeekCoder-1.3b	0.45	0.60	0.64	0.65	0.89	0.75	0.44	0.44	0.56	0.10	0.00	0.15	0.23	0.26	0.82
StarCoder-1b	0.44	0.58	0.61	0.63	0.87	1.05	0.57	0.53	0.59	0.11	0.00	0.12	0.20	0.24	0.72
StarCoder-3b	0.46	0.52	0.60	0.62	0.86	0.89	0.64	0.58	0.66	0.13	0.00	0.11	0.19	0.24	0.75

- 3つのメトリクス全てで改善
 - DeepSeekCoder-1.3b：CodeBLEU「0.65」，Exact Match「0.26」
 - →コード理解はしているものの，小さなミスを犯し，Exact Matchの失敗につながっている

RQ2：ファインチューニングあり性能

- ファインチューニング後のAPR性能を示す

Model	Base			Epoch 1			Epoch 3		
	QB	HE	D4J	QB	HE	D4J	QB	HE	D4J
Bloom-560m	1	10	8	5	17	41	6	19	41
Bloom-1b7	1	15	10	8	10	43	11	23	53
CodeGen-350M	6	25	12	13	43	67	16	42	61
CodeGen-2B	13	44	20	8	25	66	11	36	64
CodeT5-small	0	4	10	14	44	53	13	39	60
CodeT5-base	0	4	8	14	45	68	17	39	75
CodeT5-large	2	2	4	9	36	62	16	47	72
DeepSeekCoder-1.3b	33	94	72	19	54	84	15	64	80
StarCoder-1b	22	69	62	12	41	72	13	49	71
StarCoder-3b	32	94	63	6	26	51	11	37	63

性能が向上

性能が低下

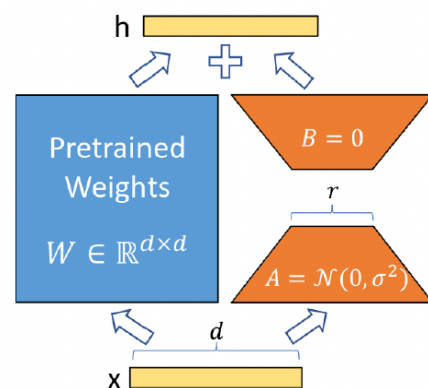
- DeepSeekCoder, StarCoderがFT後に性能が低下
 - データ分布の違いが原因であると考えられる
 - 大きなモデルは汎化的な修復能力を持っているが、FTにより逆に汎化的な能力が失われている

RQ3：パラメータ効率の良いFTの性能

パラメータ効率の良いFT（PEFT）

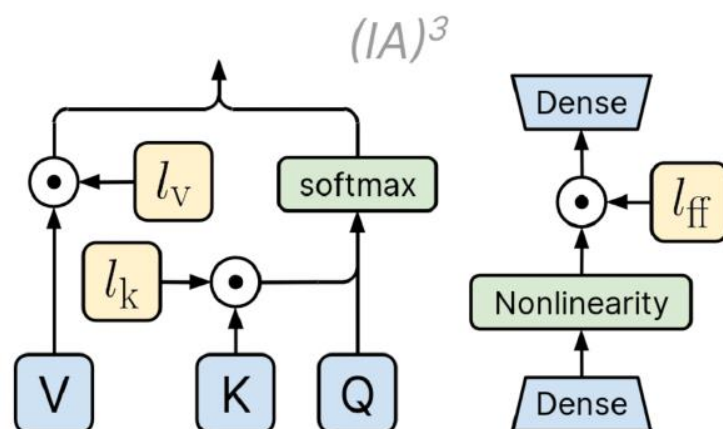
- モデルが能力を失い，性能が低下するのを改善
- LoRA[21]，IA3[22]が有望な結果を示す

- LoRA



大きな重み行列を小さな低ランク因子に分解
→トレーニング・推論時のモデルのメモリの削減し，
計算効率を向上させる

- IA3



キー行列とValue行列を再スケーリング
Liuらの実験によると，より少ない訓練可能な
パラメータで優れた性能を示す

RQ3：パラメータ効率の良いFTの性能

- LoRA, IA3によるファインチューニングの挙動

	CodeBLEU (↑)					Loss (↓)					Exact Match (↑)				
	Validation Epoch					Validation Epoch					Validation Epoch				
	0	1	2	3	Train	0	1	2	3	Train	0	1	2	3	Train
LoRA															
CodeGen-350M	0.39	0.58	0.58	0.58	0.60	1.12	0.47	0.47	0.46	0.44	0.00	0.06	0.07	0.08	0.09
CodeGen-2B	0.35	0.61	0.62	0.62	0.64	0.94	0.40	0.39	0.39	0.34	0.00	0.10	0.13	0.13	0.17
CodeGen-6B	0.42	0.61	0.62	0.62	0.65	0.93	0.38	0.38	0.37	0.33	0.00	0.11	0.14	0.14	0.20
CodeT5-small	0.04	0.33	0.35	0.34	0.34	4.39	0.84	0.82	0.81	0.79	0.00	0.02	0.03	0.03	0.03
CodeT5-base	0.08	0.34	0.35	0.34	0.35	4.80	0.69	0.67	0.67	0.65	0.00	0.04	0.05	0.05	0.05
CodeT5-large	0.01	0.33	0.35	0.35	0.36	4.37	0.57	0.56	0.55	0.53	0.00	0.06	0.07	0.07	0.08
DeepSeekCoder-1.3b	0.44	0.61	0.61	0.61	0.63	0.76	0.43	0.42	0.42	0.39	0.00	0.09	0.11	0.12	0.14
DeepSeekC.-6.7b	0.48	0.63	0.63	0.63	0.65	0.69	0.39	0.38	0.38	0.35	0.00	0.13	0.14	0.14	0.16
IA3															
CodeGen-350M	0.41	0.56	0.56	0.56	0.57	1.08	0.53	0.52	0.52	0.51	0.00	0.02	0.02	0.02	0.03
CodeGen-2B	0.38	0.59	0.59	0.59	0.60	0.93	0.46	0.45	0.45	0.44	0.00	0.04	0.04	0.04	0.05
CodeGen-6B	0.43	0.59	0.59	0.59	0.60	1.08	0.48	0.47	0.46	0.44	0.00	0.04	0.05	0.05	0.05
CodeT5-small	0.04	0.22	0.24	0.25	0.25	4.39	1.20	1.11	1.09	1.08	0.00	0.01	0.01	0.01	0.01
CodeT5-base	0.08	0.26	0.27	0.28	0.28	4.76	0.94	0.88	0.87	0.86	0.00	0.01	0.01	0.02	0.02
CodeT5-large	0.01	0.29	0.30	0.30	0.31	4.39	0.79	0.67	0.66	0.65	0.00	0.03	0.04	0.04	0.04
DeepSeekCoder-1.3b	0.46	0.60	0.60	0.60	0.61	0.83	0.49	0.48	0.47	0.47	0.00	0.06	0.06	0.06	0.06
DeepSeekC.-6.7b	0.50	0.62	0.62	0.62	0.62	0.75	0.43	0.42	0.42	0.42	0.00	0.08	0.08	0.09	0.09

ほぼ全てのモデルで損失が停滞
→訓練可能なパラメータの数が大幅に削減したため

RQ2と比べて、Exact Matchは大きな低下が見られる

RQ3：パラメータ効率の良いFTの性能

- APRベンチマークでの性能評価

Model	Base			FMFT			LoRA			IA3		
	QB	HE	D4J	QB	HE	D4J	QB	HE	D4J	QB	HE	D4J
CodeGen-350M	6	25	12	16	42	61	16	64	82	13	52	68
CodeGen-2B	13	44	20	11	36	64	19	81	98	15	67	74
CodeGen-6B	16	42	18	X	X	X	24	81	104	14	70	83
CodeT5-small	0	4	10	13	39	60	10	36	50	5	18	42
CodeT5-base	0	4	8	17	39	75	12	50	67	8	32	52
CodeT5-large	2	2	4	16	47	72	17	60	72	10	47	61
DeepSeekCoder-1.3b	33	94	72	15	64	80	27	106	100	28	94	101
DeepSeekCoder-6.7b	33	107	89	X	X	X	31	108	108	30	100	114

- ベースラインモデルに比べて性能が向上した
 - 特にLoRAでのCodeGen, DeepSeekCoderで改善が見られた
- LoRaがIA3よりも良い性能を示した
 - 24ケース中21ケースでIA3よりも優れている

LoRAはLLMのファインチューニングに対する実行可能なアプローチ

RQ4：LoRAパラメータの影響

LoRAのモデルサイズと性能に影響を与えるのは、
ランク(r)とスケールリング(α)

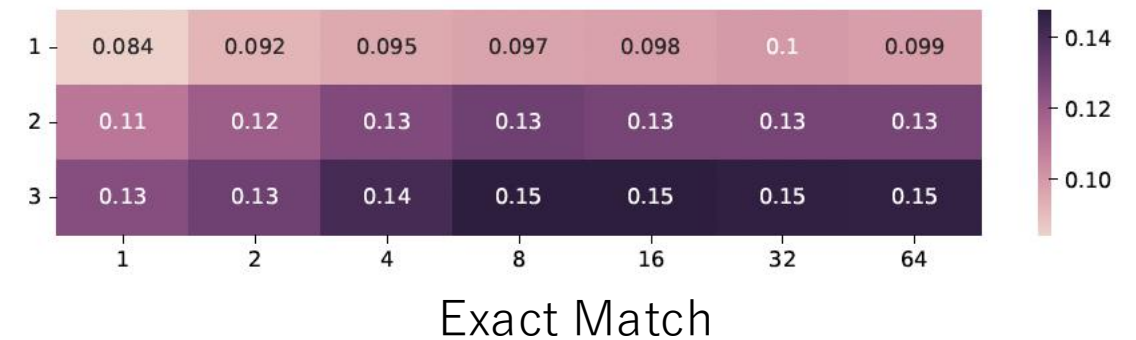
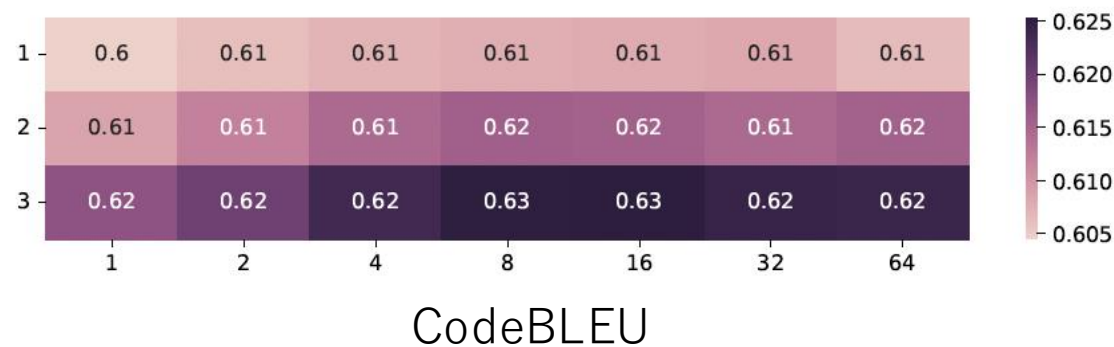
- ランク：訓練可能なパラメータの量に影響
- スケールリング：パラメータ更新の大きさを制御

ランクとスケールリング係数の両方を調査

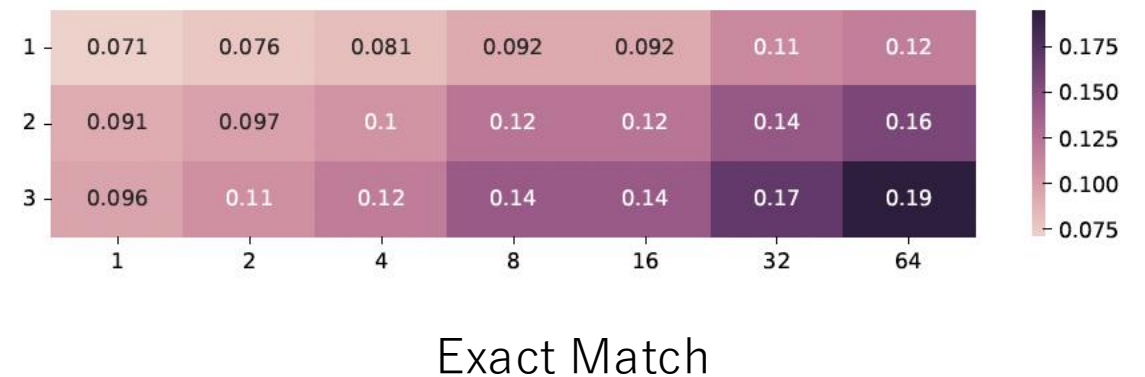
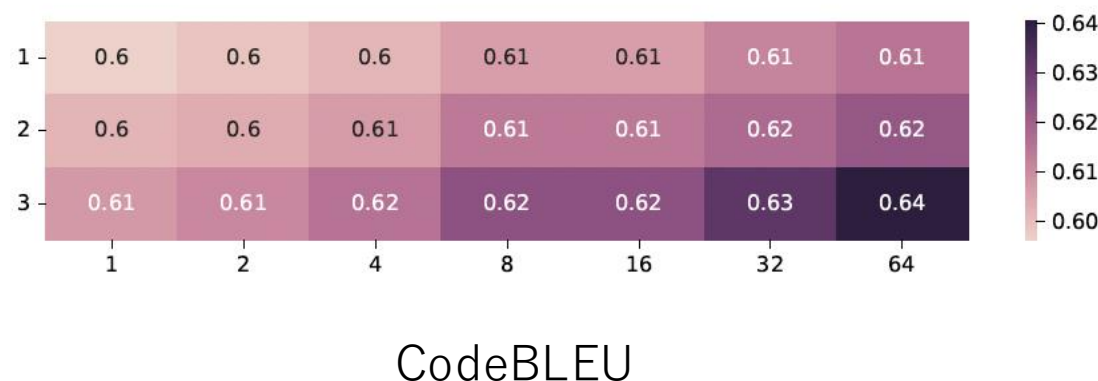
- 8つの値（1,2,4,8,16,32）で調査
 - LoRAのデフォルト値は $r=8, \alpha=16$
- 実験概要
 - 一方のハイパーパラメータのデフォルト値を固定し、もう一方を変更

RQ4：LoRAパラメータの影響

ランクの影響



スケーリング係数の影響



ランクもスケーリング係数も変化はほとんどなく、モデルの性能に大きく影響しない
→ハイパーパラメータの変更が有意な役割を満たさない

まとめ

実験

- LLMのAPR性能におけるファインチューニングの影響について実証研究を実施
 - ファインチューニングなし
 - ファインチューニングあり
 - パラメータ効率の良いファインチューニング（PEFT）

結果

- ファインチューニングは性能が低下するものもあった
 - 過学習やデータセットの多様性により汎化的な能力が失われた
- LoRA, IA3を調査
 - LoRAがファインチューニングの問題を解決し、良い結果を達成できた