

Is Agent Code Less Maintainable Than Human Code?

Shaswat Patel¹, Betty Li Hou¹, Arun Purohit¹, Kai Xu¹, Jane Pan¹, He He¹, Valerie Chen²

¹New York University

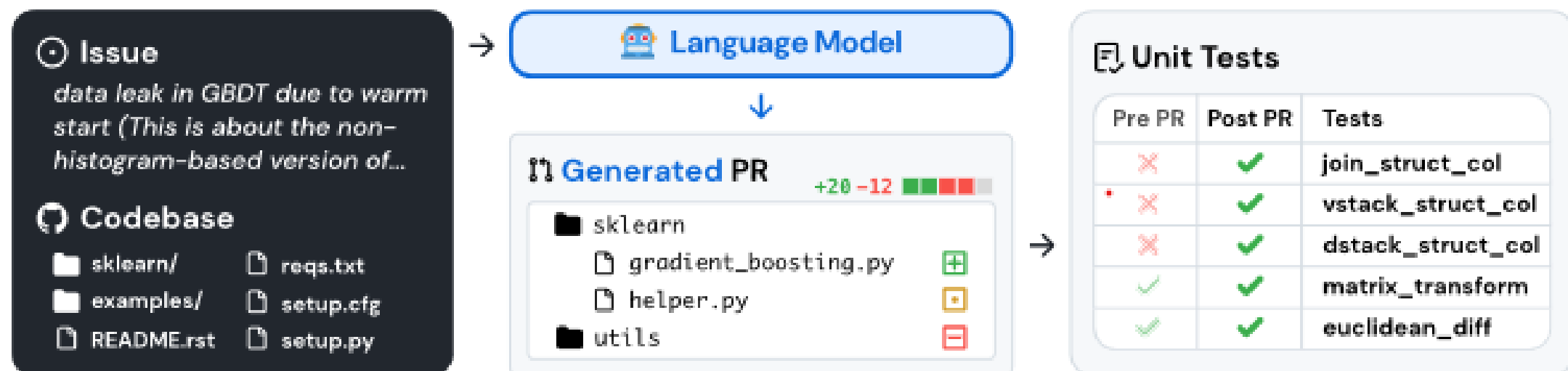
²Carnegie Mellon University

EMNLP 2026 (Oral)

背景：SWE-Bench

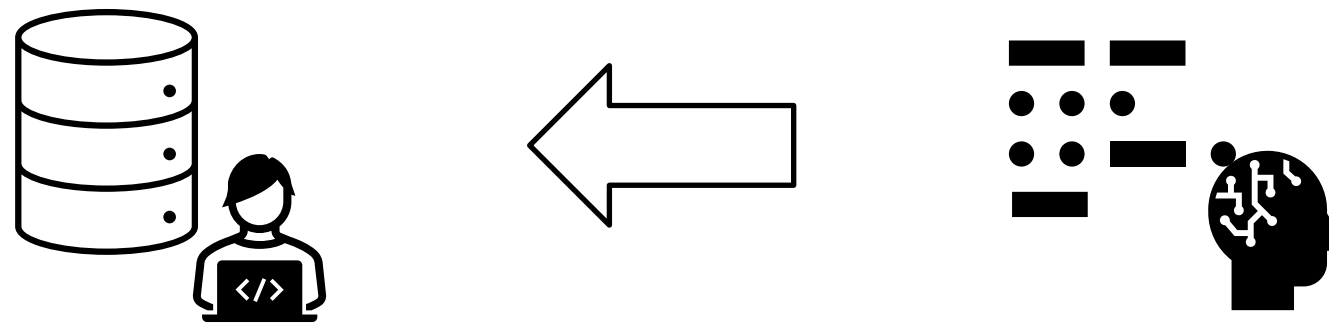
SWE-Bench:

- LLMの問題解決能力を評価するためのベンチマークの1つ
- Codebase：Issueを解く対象のリポジトリ（ソースコード一式）
- 能力評価プロセス：
 - 与えたIssueとCodebaseに対するPull Request(PR)をLLMに作成させる
 - 作ったPRに含まれるコードがUnit Testに通過するかどうかを検証
 - UnitTestに通過するPRを作成できたIssueがどれだけあるかで能力を評価

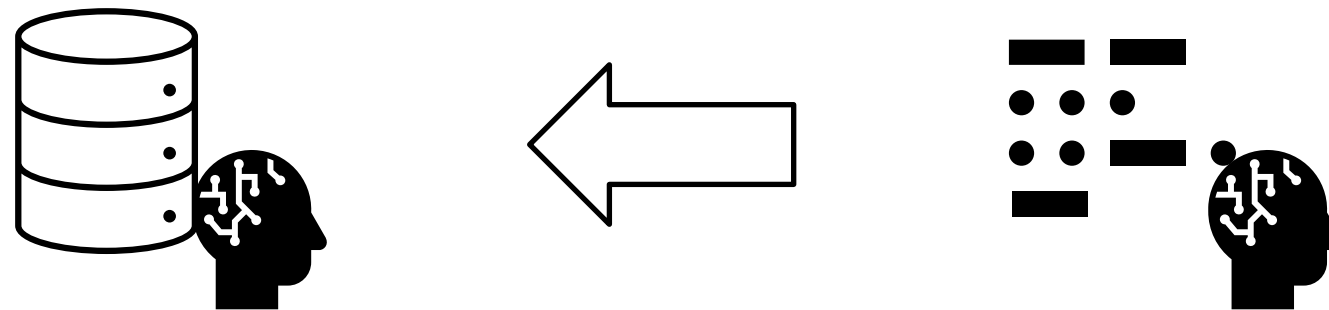


課題：既存ベンチマークの前提条件

多くの既存ベンチマークは、人間のコードのみのCodebase上でのタスクの実行を前提としている



- 人間のコードのみのCodebaseでの実行は想定されているが

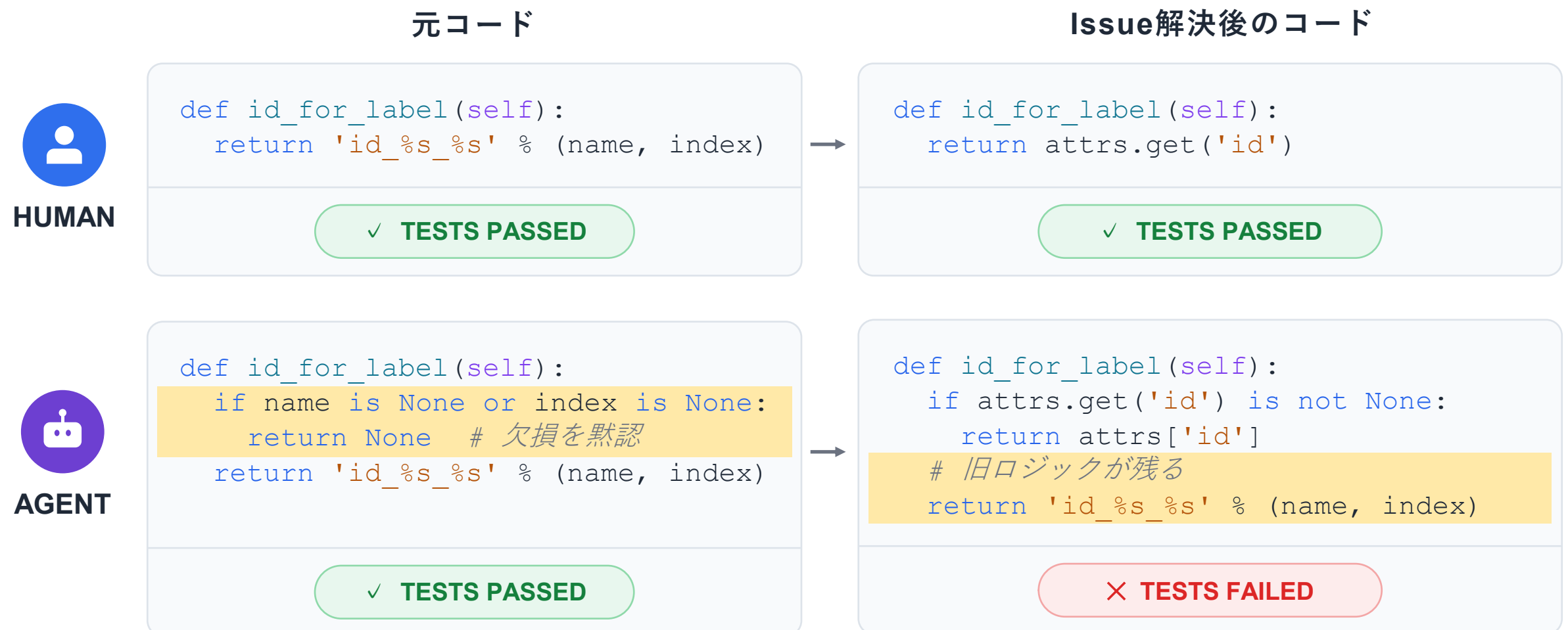


- エージェントのコードを含むCodebaseでの実行は想定されていない

課題：エージェントのコードを含むCodebase上での失敗

機能的には同じCodebaseでも、エージェントのコードを含むCodebaseでのみ失敗する場合があります

- 下図：Agentが開発者のコードと比べて余分なコードを実装したことにより発生したと考えられる失敗



課題：テストでは見えない問題

前ページの例で起きていること：

- 実装時：エージェントが「値が欠けていたら None を返す」処理を追加（テストは通過）
- 変更時：後続のエージェントがその処理を残したまま変更し、テストに失敗

問題は正しさではなく、後からの変更のしやすさ：

- 既存ベンチマークは、テストに通るかだけを評価している
- 後から変更しやすいかは評価していない

➤ この「後からの変更のしやすさ」がコードの保守性

課題：コードの保守性

保守性（Maintainability）：

- コードを後からどれだけ修正・拡張しやすいか

従来の保守性の評価：

- 循環的複雑度・Halstead Volume・認知的複雑度などの静的指標で近似
- エージェントのコードを静的指標で評価した研究はあるが、人間のコードとは比較していない
- 静的指標はCodebaseの慣習や文脈に左右されやすい

➤ 後続のタスクを解けるかどうかで、保守性を直接測る必要がある

研究概要

目的：

- コーディングエージェントのコードを含むCodebaseの保守性の検証

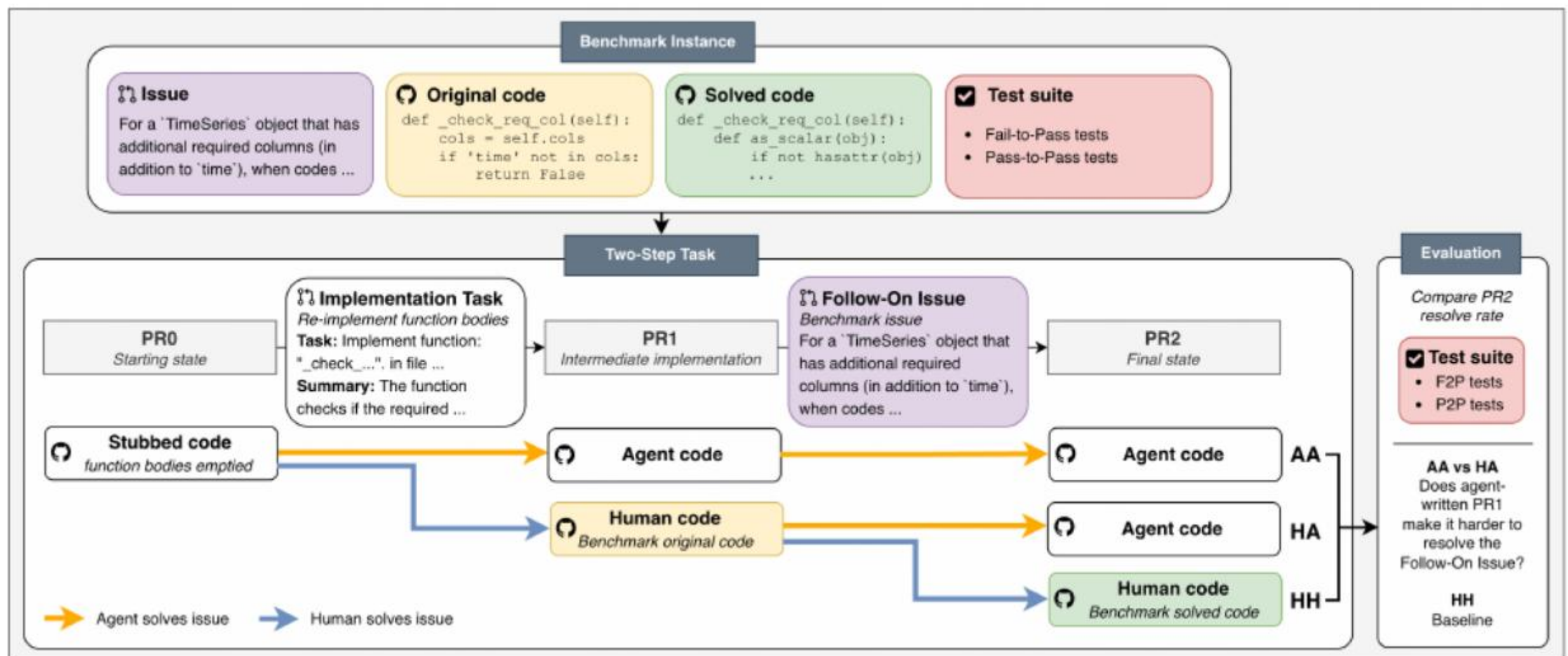
アプローチ：

- コーディングエージェントのコードを含むCodebase上でのIssueの解決率の評価
- 上記の評価を可能にするための評価フレームワーク CodeThread の提案

提案フレームワーク：CodeThread

簡単に言うと：

- ベンチマーク(SWE-bench)が用意してる「人間のコードのみのCodebase」の一部の関数をエージェントに書き直させることで「エージェントのコードを含むCodebase」を作る



実験設定

ベンチマーク :

- SWE-Bench
 - Verified (441)
 - Multilingual (171)
 - Pro (641)
- FeatBench (124)

タスクの種類

- バグ修正 (BF)
- 機能実装 (FI)
- リファクタリング (RF)

モデル :

- Claude 4.5 Sonnet
- GPT-5
- GLM 4.7
- MiniMax M2.5

エージェントフレームワーク

- SWE-Agent

略語

PR1（実装タスク）：

- Codebaseの一部の関数を実装する（人間またはエージェント）

PR2（後続Issue）：

- PR1の上で，元のSWE-BenchのIssueを解く

HA(Human PR1 followed by Agent PR2)：

- 人間のコードのみのCodebase上でエージェントがコードを作成した

AA (Agent PR1 followed by Agent PR2)：

- エージェントのコードを含むCodebase上でエージェントがコードを作成した

実験結果

表：ベンチマーク別のHAとAAのIssue解決率

Benchmark	Claude			GPT			GLM			MiniMax		
	HA	AA	Δ	HA	AA	Δ	HA	AA	Δ	HA	AA	Δ
SWE-Bench Verified	71.7	66.7	-5.1	62.6	61.6	-1.0	65.7	62.6	-3.0	74.7	68.7	-6.1
SWE-Bench Multilingual	53.1	50.0	-3.1	46.9	50.0	+3.1	62.5	59.4	-3.1	65.6	62.5	-3.1
SWE-Bench Pro	46.3	38.6	-7.7	39.0	39.8	+0.8	45.2	32.0	-13.1	43.6	36.7	-6.9
FeatBench	50.0	56.2	+6.2	56.2	43.8	-12.5	62.5	56.2	-6.2	56.2	56.2	0.0

全体的にAAの解決率はHAの解決率を下回る

- 解決率は最大13.1%低下 (GLM 4.7 × SWE-Bench Pro)

実験結果：組み合わせ別の傾向

低下が大きかった組み合わせ：

- GLM 4.7 × Pro (−13.1)：長く複数ファイルにまたがる編集
- GPT-5 × FeatBench (−12.5)：機能実装タスクのみのベンチマーク

AAがHA以上だった例外：

- GPT-5 (Multilingual ▪ Pro)：HAの解決率がもともと低い
- Claude × FeatBench (+6.2)：共通の問題が16件と少ない

実験結果

表：タスク種別のHAとAAのIssue解決率

- FI + RF：機能実装とリファクタリングが同時に行われている

Task	N	Claude			GPT			GLM			MiniMax			Avg. Δ
		HA	AA	Δ	HA	AA	Δ	HA	AA	Δ	HA	AA	Δ	
BF	210	59.05	51.43	-7.62	49.52	50.00	+0.48	56.19	52.38	-3.81	61.43	56.19	-5.24	-4.05
FI	104	44.23	41.35	-2.88	41.35	38.46	-2.88	46.15	30.77	-15.38	42.31	38.46	-3.85	-6.25
RF	70	57.14	50.00	-7.14	48.57	51.43	+2.86	55.71	40.00	-15.71	54.29	41.43	-12.86	-8.21
FI + RF	22	27.27	22.73	-4.55	27.27	27.27	0.00	31.82	13.64	-18.18	27.27	22.73	-4.55	-6.82

3つのタスクすべてでモデル平均が低下

- リファクタリング(RF)が、最も大きく低下：8.21%

要因分析：なぜAAはHAを下回るのか？

- | これまでの分析でAAがほぼ全体的にHAを下回ることが分かった
- 下回った要因を分析するためにロジスティック回帰を適用する

ロジスティック回帰とは

各要因が結果を「HAだけ成功」「AAだけ成功」のどちらに傾けるかを調べる手法

要因分析：分析設定

使用データ：HAとAAで結果が分かれた405件

- HAだけ解決：約260件
- AAだけ解決：約140件

調べた要因：

- ① 複雑さ・コード量（従来の保守性指標）
- ② 編集した場所（HAとAAで同じ箇所を編集したか）
- ③ 入力・エラー処理のずれ（LLMが人間版とエージェント版を比べて判定）
- ④ 問題の難易度（他のモデルの正解率）

要因分析：結果

① 複雑さ・コード量：×

- PR2の書き足し量のみ○
- AAが多く書き足すほど，AAだけ失敗しやすい

② 編集した場所：×

③ 入力・エラー処理のずれ：○

- 入力・例外の扱いが人間版と違うと，AAだけ失敗しやすい
- 例：p4の「None を返す」処理

④ 問題の難易度

- 難易度をそろえても，上記の要因は効いていた

考察

保守性のコストは大規模な編集・機能実装で顕在化

- GLM × Pro -13.1, GPT-5 × FeatBench -12.5

リファクタリングで低下が最大：先行研究と一致

原因は複雑さではなく，入力・エラー処理のずれ

タスク解決だけでなく保守性でも評価すべき

まとめ

コーディングエージェントが作成したコードの保守性を検証

CodeThread : Codebaseの一部の関数の作者（人間 / エージェント）だけを変え、同じ後続Issueの解決率を比較

エージェントは、タスクを解けるかだけでなく保守性でも評価すべき

所感

- | なんとなく「エージェントは基本的なコードの実装に関しては完璧」と思い込みがちだが、そうではない
- | 逆にエージェントのコード生成能力が上がりすぎたが故に、エージェント自身が後から編集しづらいような大量の処理を抱えたコードを作ってしまった
- | 人間が介在すればどうにかなる問題ではある

付録：要因分析の特徴量一覧

表：特徴量一覧

カテゴリ	特徴量	数	内容
静的な保守性指標	CC, CogC, HV, LLOC の AA-HA の差	8	4指標 × PR1とPR2の2時点。構造の複雑さ（CC, CogC）と冗長さ（HV, LLOC）
パッチの編集箇所	ファイル単位と関数単位のJaccard係数	2	HAのPR2とAAのPR2が、どれだけ同じ箇所を編集したか
振る舞いのずれ	IEC（0か1）	1	エージェントのPR1が、入力の扱いやエラー処理を人間版から変えているか
難易度の統制	LOOのインスタンス解決率	1	同じインスタンスに対する、他の3モデルの平均解決率
固定効果	ベンチマーク, モデル	—	ベンチマークやモデルごとの違いを吸収する