能動的コンテキスト増強によるソースコードのバグ修正

清水 亮宏 [1]　福本 文代 [2]　鈴木 良弥 [2]
山梨大学　工学部 [1]　総合研究部工学域 [2]
{t22cs026, fukumoto, ysuzuki}@yamanashi.ac.jp

概要

本研究は，中規模オープンモデルによるリポジトリレベルのソースコード修正の実現を目的とする．既存の細粒度な構造提示手法は，中規模モデルにおいて情報過多による精度低下を招く恐れがある．この課題に対し，Agentless を基盤とし修正フェーズの提示情報を増強する「能動的コンテキスト増強」を提案する．本手法はモデル自身が必要な情報を取得することにより，ノイズを排除した文脈提供が可能となる. 増強アプローチとして，Agent 間の対話を用いる分散型手法と，自律探索を行う集中型手法を提案する．実験の結果，2 手法のいずれも Agentless よりバグ解決数が減少したが，RepoGraph の解決数より上回ることが確認できた.

1　はじめに

近年，生成 AI の進展に伴い，LLM を利用したプログラムコーディングの自動化が盛んに研究されており，Claude Code[1] や Codex[2] などのコーディング支援ツールに加え, リポジトリ全体を対象とした複雑なコード修正を LLM に行わせる手法が多く提案されている [3, 4, 5]．しかし高精度な修正精度が得られている既存手法の多くは，開発元が提供するクローズドかつ高性能な LLM に依存しており，再現性と透明性や運用コストなどの課題が指摘されていた [6].

2023 年に SWE-bench マークデータセット [7] が公開されて以来，リポジトリレベルのバグ修正能力を競う研究が活発に行われるようになった．これらは，SWE-agent[8] や OpenHands[9],Moatless Tools[10] に代表される Agent 型手法と Agentless [11] や RepoGraph [12] などの Pipeline 型に大別できる．Agent 型は Agent が自律的に情報の探索から修正までを行う手法であり，柔軟性が高い反面，推論の負担が大きい．一方，Pipeline 型は，修正箇所の特定（ローカライズ）や修正などの工程を段階的に実行する手法である．代表的な手法である Agentless は，「ローカライズ」.「修正」.「評価」の 3 段階で構成され, 修正フェーズに入力される情報がローカライズ箇所の周辺情報に限定されているため，バグ修正に必要な広範な文脈情報が不足するという問題が指摘されていた．この問題に対し，Ouyang らは RepoGraph を提案した [12]．RepoGraph は，リポジトリ全体をグラフで表現し，修正箇所に関連する部分グラフを提供することで精度の向上を目指した．しかし，外部から機械的に抽出された構造情報を付与する手法は，モデルのパラメータサイズに依存してその有効性が大きく変動するリスクがある．実際 Hongyuan らによると, リポジトリの構造的・意味的情報を LLM のアテンション機構に統合する CGM[13] における評価では，Qwen2.5-72B を用いた場合には 43.00%という高い修正率を達成している一方で，7B クラスの小規模モデルでは 4.00%にまで精度が急落することが報告されている.

本研究ではこの課題に対し, リポジトリを論理的単位（ファイル・クラス・関数）で構造化し, モデル自身が必要な情報を選択的に取得する「能動的コンテキスト増強」というアプローチを提案する. 本手法は推論能力に制約のあるモデルでも, ノイズを排除した最小限かつ高密度な情報に基づいた正確な修正が期待できる. 本稿では, このコンセプトに基づき, Agent 間の対話を通じた情報収集と, 単一モデルによる自律的なグラフ探索の 2 つのアプローチについて, それぞれの有効性と課題を検討する.

2　本手法

本研究では, Agentless のパイプラインを基盤とし, その「修正フェーズ」に提供されるコンテキストの質を能動的に向上させる仕組みを提案する. 具体的には. リポジトリ全体をファイル・クラス・関数という論理的単位でノード化した LocAgent[14] のグラフ構造を活用し, モデル自身が修正に必要な情報を収集・特定する能動的コンテキスト増強を導入する.

本研究では, このコンセプトを実現するためのアプローチとして, 分散型 Agent 間の対話を用いる手法 (以降，分散型と呼ぶ), および集中型自律グラフ探索を用いる手法 (以降，集中型と呼ぶ) を提案する.

2.1 共通の設計思想

分散型，及び集中型手法は, 中パラメータモデルの推論能力を最大限に引き出すため, 以下の 3 つの設計思想に基づいている.

1. 情報の「網羅性」から「密度」への変更

既存手法である RepoGraph[12] のように, 行 (Line) レベルの依存関係を機械的に網羅し提示することは, 情報の完全性は確保できるものの, 中パラメータモデルにとっては注意力の分散を招くノイズとなる. Liu ら [15] は, 関連文書を網羅的に入力しても情報の位置や量に対するバイアスにより, かえって推論精度が低下する場合があることを中規模モデルを用いた実験で示している. 本研究では, 情報を網羅することよりも, モデルの思考を阻害しない密度の高い情報のみを増強することが中パラメータモデルの推論精度を上昇させると期待する.

2. LocAgent による論理的粒度の採用

本研究では, LocAgent のグラフ構造を採用する. RepoGraph[12] が行レベルという極めて微細な粒度で依存関係を定義しているのに対し，LocAgent[14] はファイル・クラス・関数という抽象度の高い粒度でノードを構成している. 中パラメータモデルが複雑なリポジトリ構造を扱う際，行レベルの膨大なグラフは全体像の把握を困難にし, 推論の迷走を招く. 一方， LocAgent の粒度であれば， モデルは「どの論理単位の情報を扱うべきか」という, 人間がコードを理解する際と同様の階層的な視点で, 効率的に意思決定を行うことが可能となる. この意味のある単位での情報のやり取りを基盤とすることで, モデルは広範な依存関係を失うことなく, 必要な文脈のみの特定が可能であると期待できる.

3. 収集情報の Agentless への統合

本研究で提案する分散型，及び集中型手法は, Agentless における「修正フェーズ」の入力を高度化することを共通の目的としている.Agentless のローカライズフェーズにおいて特定された修正候補関数を起点とし, どちらかの手法を用いてリポジトリ補足情報を収集する. これらは Agentless が標準的に取得する局所的な周辺コード（前後 10 行）を補完する背景知識として統合される. これにより, 中パラメータモデルであっても, 修正対象の周辺情報に留まらない, パッチ作成プロセスを実現する.

2.2 分散型 Agent 間対話による情報収集

分散型アプローチは，情報収集の方法として, リポジトリ内の各関数に独立した Agent を仮想的に割り当て, Agent 間の対話を通じて情報を増強する

Agent の配置と対話メカニズム

LocAgent により構築された異種混合グラフの関数ノードに対し, 自身のソースコードと周辺の依存関係を保持する専用 Agent を配置する. Agentless のローカライズフェーズで特定された修正候補関数を担当する Agent を起点とし, 依存関係（呼び出し元・呼び出し先等）にある隣接 Agent に対して問い合わせを行う. この際, モデルには隣接 Agent への質問を生成する `question_other_agent_tool` を提供する.

対話による情報の絞り込みと集約

各 Agent は, 自身が保持する関数の内容だけでは不明瞭な情報を quesion_other_agent_tool を利用し隣接 Agent に質問することで得る. この処理により, 数万行に及ぶリポジトリ全体を直接入力することなく, 修正に必要な文脈のみを抽出することが可能となる. 最終的に, 起点となった Agent は, 隣接ノードから得られた情報を「増強された背景知識」として集約し, 後続の Agentless へと出力する. 本手法は, 単一のモデルに膨大な文脈の理解を強いるのではなく, 論理単位ごとの局所的な理解を積み上げることで, 中パラメータモデルの注意力維持を図っている.

2.3 集中型グラフ探索による情報収集

集中型アプローチは，情報収集の方法として, 単一のモデルがリポジトリの構造を自律的に巡回し, バグの原因に関する仮説の立案を繰り返すことで情報を増強する手法である．本手法の設計は，推論と行動を密接に連携させる ReAct[16] の概念を基礎としており, モデル自身の内省に基づいた探索プロセスを実現している.

仮説検証に基づく動的探索メカニズム

モデルに「根本原因の仮説」と「不足している情報」を常に保持・更新させる. Agentless のローカライズ結果から得られた関数を起点とし, モデルは現在の仮説を検証するために次に調査すべき関数を LocAgent のグラフ構造から動的に選択する. これにより, モデル自身の思考プロセスに基づいた一貫性のある情報収集が可能となる.

探索プロセスの制御と文脈の増強

モデルは各ステップで「現在の情報で修正方針を決定できるか」を判断し, 情報が不足している場合にのみ探索を継続する. このプロセスは, 特定の探索回数に達するか, あるいは修正に必要な情報が揃ったとモデルが判断した時点で終了する. この巡回で現在の情報で修正方針を決定できると判断された情報を後続の Agentless への入力とする. 本手法はモデルが必要と判断した経路のみを辿るため, コンテキストの肥大化を抑制しつつ, バグの根本原因に辿ることが可能となる.

3 実験

3.1 実験設定

能動的コンテキスト増強の有効性を検証するため, ベースラインである Agentless[11], 及び関連研究である RepoGraph[12] との比較実験を行った.

1. 利用モデル

実験に先立ち, より軽量な Qwen3:8b を用いた予備実験を実施した. しかし, エージェント間対話やグラフ探索のプロセスにおいてハルシネーションが多発し, 情報の動的収集プロセス自体が破綻する傾向が確認された. この結果から LLM による能動的な情報収集の遂行には一定水準以上の推論能力が必要であると判断し, 20B クラスのモデルを採用した. 具体的には, コード修正能力を測定するベンチマークである Aider LLM Leaderboard [17] を参考に, かつ中規模レベルであることを考慮にいれ, 最終的に gpt-oss20b を採用した.

本研究は「中パラメータモデルによるバグ修正精度の向上」にあるため, パッチ作成を行う「修正フェーズ」および手法の核心である「動的探索フェーズ」にのみ gpt-oss:20b を使用した. その他の工程には, Agentless が標準的に採用し実績のある GPT-4o を利用した. これにより, モデル全体の性能差による影響を最小限に抑え, 提案手法による情報収集が修正精度に与える寄与をより厳密に評価することが可能となる. なお, 既存の Agentless ではエンベディングとして text-embedding-3-small が利用されているが, 本実験環境での再現性を考慮し, エンベディングを使用しない構成へと調整を行った.

2. ベンチマーク

提案手法の評価指標として SWE-bench Lite [7] を採用した. ベンチマークには, Django や Astropy といった GitHub 上の実プロジェクトで生じたバグ修正問題が含まれており, リポジトリレベルの修正能力を測定する代表的な指標である. 先行研究の RepoGraph を実行した際, Sympy リポジトリの問題において網羅的なグラフ構造からプロンプトを生成する際にメモリ不足が発生した. そこで, 全手法を同一条件下で公平に比較するため, Sympy リポジトリを除く 223 件の問題を対象として実験を実施した.

表 1 SWE-bench Lite における手法別性能比較

手法	提出数 (s)	解決数 (r)	r/s
Agentless[11]	**202**	**69**	**34.16**
分散型	198	64	32.32
集中型	196	64	32.65
RepoGraph[12]	165	50	30.30

3.2 実験結果

実験結果を表 1 に示す. 表 1 より, 提案手法は Agentless(69) と比較し, 集中型と分散型のバグ解決数は 64 件と減少している. 関連研究である RepoGraph(50) と比較すると, 本論文で提案した分散型, 集中型共に大きく上回る結果が得られた.

RepoGraph との比較

RepoGraph は, 提出数（Submitted）が 165 件と少ないだけでなく, 提出したパッチのうち実際にバグを解決できた割合（r/s）が最低の値 (30.30%) であった. これは, RepoGraph が提供する網羅的な情報が, 中規模モデルにおいてはパッチの書き出しを困難にさせるだけでなく, 仮にパッチとして成立し評価対象となった場合においても, 適切な修正を導けずに精度を低下させていることを示している. すなわち, RepoGraph は大規模なモデルには有効であるものの, 中規模なモデルでは有効ではないことを示している. 一方で本手法は, 情報の能動的な増強により追加情報を必要最小限に抑えることができ, パッチ提出数と r/s の低下を抑えることに成功した. このことは, モデルが必要な情報を獲得するアプローチが, 中規模モデルにおける「生成の安定性」と「修正の妥当性」の両面において, 網羅的な情報付与よりも優れていることを示している.

Agentless との比較

本手法は RepoGraph と比較すると高い精度が得られる反面, Agentless との精度差は僅かに劣っていた. そこで本手法の核心である, モデルが必要と判断した際に追加情報を付与する点について, 本手法であ

表 2 情報を追加した問題における解決数の比較

手法 (情報追加件数)	解決数	
集中型 (121)	44	–
分散型 (186)	–	56
Agentless[11]	45	60

る集中型，及び分散型と Agentless との詳細な比較を実施した．具体的には，本手法が情報を取得できずに終了した問題を除外し, 実際にプロンプトへ情報が追加された問題に限定し比較を行った．結果を表 2 に示す. 表 2 より，情報を追加したサブセットにおいても集中型の解決数 (44) は Agentless (45) を 1 事例下回り, 分散型 (56) は Agentless(60) を 4 事例下回る結果となった.

1. 集中型に関する考察

集中型のみが成功している事例では, 正確な情報の抽出とその情報に基づいた修正が行われていた. 集中型は,（scikit-learn__scikit-learn-14092) のバグの原因を「NumPy 型と Python 標準型の継承関係の断絶」と特定し, データの正規化といったエンジニアリングとして堅牢な修正を実現した. Agentless 単体ではエラーを場当たり的に回避するコードに留まっていた. 一方で集中型が誤った原因として以下の 2 点が挙げられる.

(1) 追加情報の誤りによる推論の誤誘導

集中型エージェントが提示したバグの根本原因自体に誤りがあり, それがパッチ作成を誤誘導したケースである. (sphinx-doc__sphinx-11445) ではバグの原因を「空白行の挿入」と判定した. しかし, 実際のバグは正規表現によるドキュメント抽出範囲の不備に起因していた. Agentless 単体ではコードの字面から修正を試みたのに対し, 集中型では提示された誤った根本原因箇所を削除するだけの不適切なパッチを生成したために, 解決に失敗した.

(2) 構造的修正による誤り

(pytest-dev__pytest-7220) ではバグの原因を「フィクスチャによるディレクトリ移動の影響により, パスの基準が変化することが原因である」と正確に特定できた. 同様に，情報なしのモデルは,「パスの基準を絶対パスに強制する」という最小限の変数更新で解決に成功した. 一方, 正確な情報を得たモデルは「原因であるパスの基準の変動を止めるべき」と判断し, パスの基準を動的な cwd から静的な root_dir に変更する構造的修正を試みた. しかし, これがセッション全体の整合性を破壊し, 結果として不適切なパッチとなった. すなわち, リポジトリ全体の構造が把握できていないにもかかわらず, 正確な情報が付与されたことで, モデルの修正戦略が「防御的な場当たり的修正」から「根本的解決」へ変更されてしまい, 解決に失敗した.

2. 分散型に関する考察

Agentless は (psf__requests-2674) において，周辺コードの命名規則に合わせ，結果的に誤った例外名を使用していた．一方で分散型は, 追加対象とする例外 (`TimeoutError`, `DecodeError`) がどのファイルで定義されているかを確認した上で, 各例外に対するハンドリング指示をコンテキストに追加し, 正しい修正を行うことに成功した. 一方で，分散型が誤った原因として以下が挙げられる.

複数の修正方法の提示

本来，分散型は修正対象の周辺コード以外の「背景知識」を取得することを目的としていた．しかし実際には，各関数のエージェントから独自に具体的な「修正方針」が提案される傾向が見られた．これらの修正方法は, バグを解決することはできても, 既存のコードを破壊するような修正が多い. 加えて, 背景知識の前提で, 複数の関数を起点にしたために, 追加情報に複数の修正方針が書かれるようになった. 例えば (django__django-14672) では,「`through_fields` を make_hashable 関数の帰り値として初期化する方法」と,「参照方法を変更し, f.name で重複チェックを行う」2 つの手法が提案された. 最終的に前者の手法が採用され, 他のコードを破壊するような修正結果となった.

4 おわりに

本研究では, 中パラメータモデルの推論能力を最大限に活用するため, 必要な情報を動的に増強する「能動的コンテキスト増強」を提案した. 実験の結果, 提案手法はいずれも解決数が関連研究である RepoGraph を上回ったことが確認できた一方で, Agentless よりも僅かに下回る結果となった．理由として (1) 分散型において追加された情報が既存のコードを破壊するような教示であった点，(2) 集中型の追加情報の付与によりモデルの修正戦略を誤った点が挙げられる．今後は，先行研究 [18, 19, 20] に基づき，モデルに対してバグの根本原因の把握や指示追従に特化した学習を施すことで, 本手法の課題である出力精度と情報抽出能力を改善する予定である.

謝辞

本研究の一部は, JKA(2024M-557), 及び科研費 24K15085 の助成を受けたものです.

参考文献

[1] Anthropic. Claude code | claude product page, 2025-12 閲覧. https://code.claude.com/docs/ja/overview.

[2] OpenAI. Introducing Codex, 2025-12 閲覧. https://openai.com/index/introducing-codex.

[3] Lin Shi Song Wang Shoubin Li Qing Wang Fangwen Mu, Junjie Wang. Experepair: Dual-memory enhanced llm-based repository-level program repair. arXiv preprint arXiv:2506.10484, 2025.

[4] Shunfu Jin Yang Liu Feng Liu Bach Le Haoye Tian Boyang Yang, Jiadong Ren. Enhancing repository-level software repair via repository-aware knowledge graphs. arXiv preprint arXiv:2503.21710, 2025.

[5] Abhinav Japesh Zhijing Jin Bernhard Schölkopf Vaibhav Aggarwal, Ojasv Kamal. Dars: Dynamic action resampling to enhance coding agent performance by adaptive tree traversal. arXiv preprint arXiv:2503.14269, 2025.

[6] Chengxing Xie, Bowen Li, Chang Gao, He Du, Wai Lam, Difan Zou, and Kai Chen. Swe-fixer: Training open-source llms for effective and efficient github issue resolution. In Findings of the Association for Computational Linguistics: ACL 2025, 2025.

[7] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? In The Twelfth International Conference on Learning Representations (ICLR 2024), 2024.

[8] John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, Ofir PressJohn Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. In Advances in Neural Information Processing Systems 37, 2024.

[9] Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for ai software developers as generalist agents. The Twelfth International Conference on Learning Representations (ICLR 2025 年), 2025.

[10] Antonis Antoniades, Albert Örwall, Kexun Zhang, Yuxi Xie, Anirudh Goyal, and William Wang. Swe-search: Enhancing software agents with monte carlo tree search and iterative refinement. The Twelfth International Conference on Learning Representations (ICLR 2025 年), 2025.

[11] Soren Dunn Lingming Zhang Chunqiu Steven Xia, Yinlin Deng. Agentless: Demystifying llm-based software engineering agents. arXiv preprint arXiv:2407.01489, 2024.

[12] Siru Ouyang, Wenhao Yu, Kaixin Ma, Zilin Xiao, Zhihan Zhang, Mengzhao Jia, Jiawei Han, Hongming Zhang, and Dong Yu. Repograph: Enhancing ai software engineering with repository-level code graph. The Twelfth International Conference on Learning Representations (ICLR 2025 年), 2025.

[13] Zhenhao Tang Hongen Peng Xukun Zhu Bingchang Liu Yingguang Yang Ziyin Zhang Zhaogui Xu Haipeng Zhang Linchao Zhu Rui Wang Hang Yu Jianguo Li Peng Di Hongyuan Tao, Ying Zhang. Code graph model (cgm): A graph-integrated large language model for repository-level software engineering tasks. arXiv preprint arXiv:2505.16901, 2025.

[14] Zhaoling Chen, Xiangru Tang, Gangda Deng, Fang Wu, Jialong Wu, Zhiwei Jiang, Viktor Prasanna, Arman Cohan, and Xingyao Wang. Locagent: Graph-guided llm agents for code localization. Findings of the Association for Computational Linguistics: ACL 2025 年, 2025.

[15] John Hewitt Ashwin Paranjape Michele Bevilacqua Fabio Petroni Percy Liang Nelson F. Liu, Kevin Lin. Lost in the middle: How language models use long contexts. Transactions of the Association for Computational Linguistics, vol. 12, pp. 157–173, 2024.

[16] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. The Twelfth International Conference on Learning Representations (ICLR 2023 年), 2023.

[17] Paul Gauthier. Aider LLM leaderboards, 2025-12 閲覧. https://aider.chat/docs/leaderboards/.

[18] Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with swe-gym. The Twelfth International Conference on Learning Representations (ICLR 2025 年), 2025.

[19] Kelin Fu Wenyang He Weimin Xiong Yibo Liu Yibo Miao Bofei Gao Yejie Wang Yingwei Ma Yanhao Li Yue Liu Zhenxing Hu Kaitai Zhang Shuyi Wang Huarong Chen Flood Sung Yang Liu Yang Gao Zhilin Yang Tianyu Liu Zonghan Yang, Shengjie Wang. Kimi-dev: Agentless training as skill prior for swe-agents. arXiv preprint arXiv:2509.23045, 2025.

[20] Yuzhen Xiao Changshi Li Chris Yuhao Liu Rui Yan Tianwen Wei Jujie He Xuchen Song Yang Liu Yahui Zhou Liang Zeng, Yongcong Li. Skywork-swe: Unveiling data scaling laws for software engineering in llms. arXiv:2506.19290 arXiv:2506.19290, 2025.