大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

トップエスイー ソフトウェア開発実践演習

大規模言語モデル (LLM) マルチエージェントの開発

エンジニアのための真のアシスタントを目指して

佐藤亨太†, 佐橋功一*, 高橋憲司‡, 髙橋雅裕§
†富士通株式会社, *株式会社 NTT データグループ, ‡株式会社 NTT データアイ, §株式会社デンソー

開発における問題点

ソフトウェア開発の現場へのLLM（大規模言語モデル）活用が進んでいる. しかし, コード生成タスクをLLMに依頼した際, **生成されたコードに正当性がない場合がある**

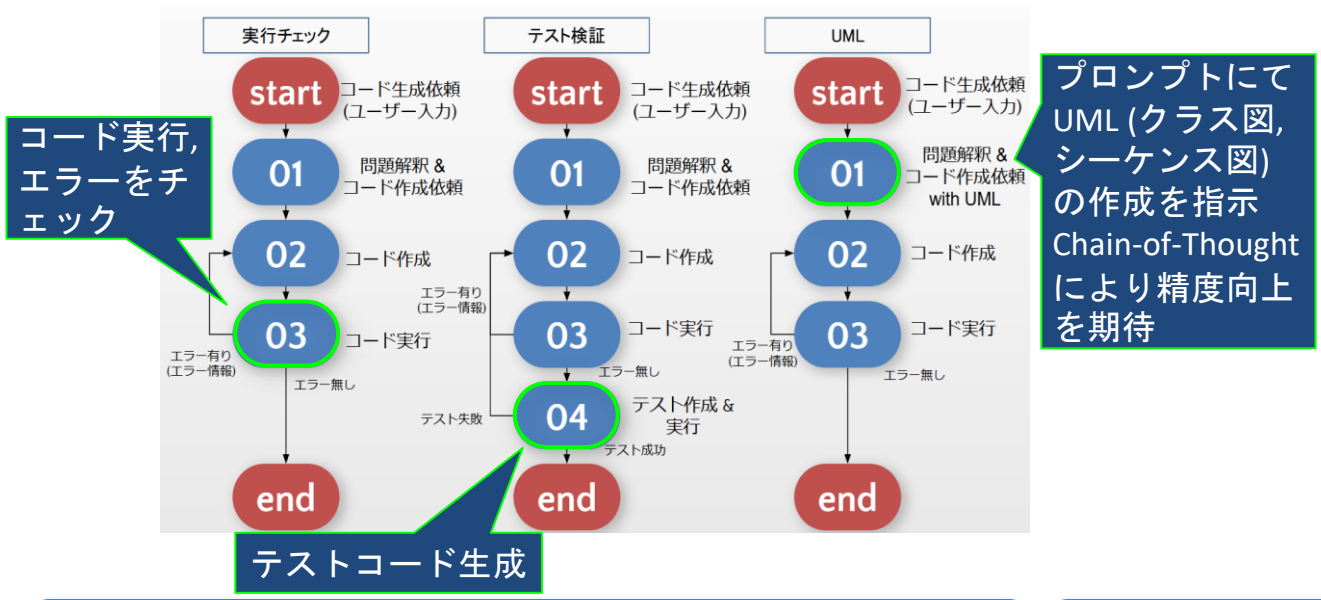
- **実行するとエラーが発生**
- **要件を満たしていない**

手法の適用による解決

- エラーが発生⇒生成されたコードを**実行チェック**する
- 要件が満たされていない⇒**テスト検証**を行い, 要件を満たすかチェックする
- 扱う問題が難しい⇒**モデリング言語**を使って問題を構造化する

提案アーキテクチャ

LangGraph フレームワークを用いて3種類の LLM マルチエージェントアプリケーションを作成(下図). 以下では特徴的なエージェントを ▭ で表示.



アーキテクチャ評価

AtCoderの問題をMarkdown形式で読み込み,LLMでコードを生成した.生成したコードをAtCoderに提出することで得られた採点結果から,以下の評価指標を算出した.

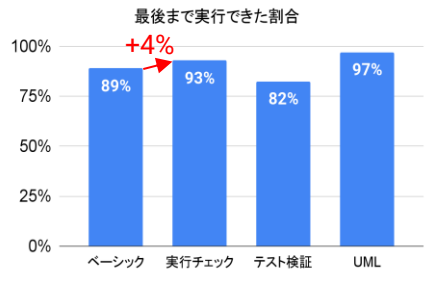
$$\text{最後まで実行できた割合} = 1 - \frac{RE}{AC + WA + TLE + RE}$$

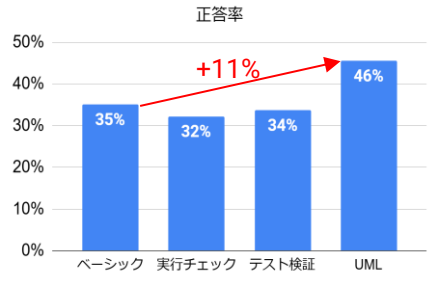
$$\text{正答率} = \frac{AC}{AC + WA + TLE + RE}$$

AC 正解　TLE 実行時間オーバー
WA 不正解　RE 実行時エラー

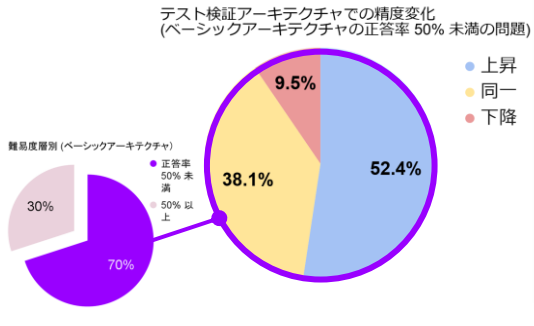
結果

- ベーシックアーキテクチャ(シンプルなLLMへの問い合わせ)を含む,4種類のアーキテクチャを評価.
- **実行チェックアーキテクチャ**は**最後まで実行できた割合**がベーシックアーキテクチャと比較して**4%向上**.
- **UMLアーキテクチャ**はベーシックアーキテクチャと比較して**正答率が11%向上**.





- テスト検証アーキテクチャでは全体の性能は低下したが, ベーシックアーキテクチャで**正答率の低い問題に対して精度向上**.



考察

UMLアプローチ No.1！ 正答率↑

最後まで実行できた割合と**正答率**が**大幅(約10%↑)に向上**.
問題を**構造化**し, コーディングしやすく**記号化**することで精度が向上したのではないか.

最後まで実行できた割合↑

実行チェックアプローチ　**テスト検証アプローチ**

最後まで実行できた割合 / 正答率が**下がる**結果が得られた.
プロンプトの微妙な差異によって**初期生成コードの品質に差**が生まれその結果が性能に影響していることが分かった.

今後の課題 / 展望

①正答率が減少した原因の分析と改善
②さらなる実行エラーの減少と正答率の向上
③中・大規模なアプリケーションの作成
⇒汎用的なソースコード生成ソリューションの構築
⇒エージェントアプリを使った自己再帰的な改善