

技術的負債の種類に関する課題票の 分類におけるGPT-4oとo1の比較

学部研究

オープンソースソフトウェア工学研究室
S2520048 陸井 優希

背景

技術的負債（TD : Technical Debt）

- 納期や予算の制約により導入される妥協された設計や実装
例）・スパゲッティコード
・コーディング規則に則って記述されていないソースコード
- デメリット
 - ・ 放置する時間が長引くほど，後の保守コストが増大する

➡ TDを管理し，早期に除去することが必要

ソースコード内のコメント

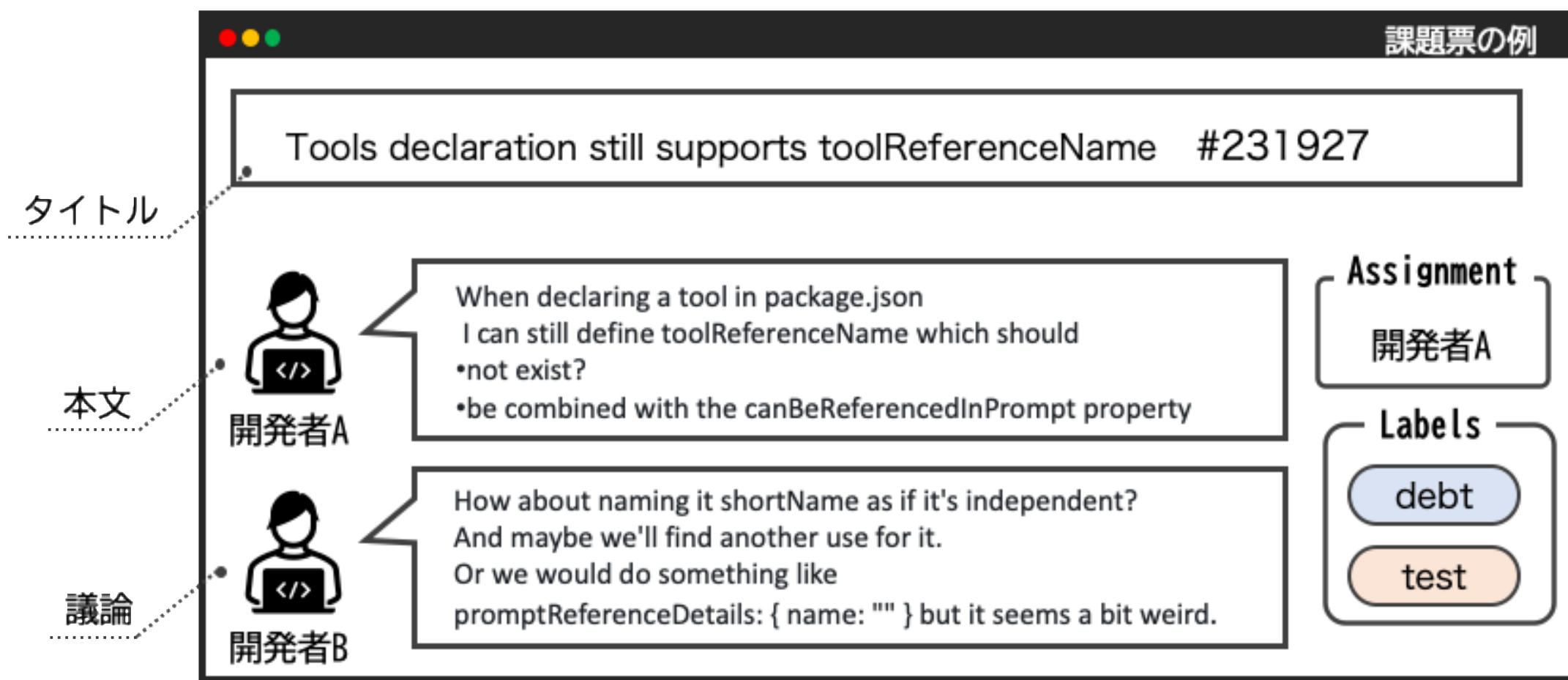
課題管理システムの課題票

」で管理される

技術的負債の管理

課題管理システムによるTD管理

- 課題票にラベルを付与することで管理する



技術的負債の管理の課題

課題票の解決までの流れ

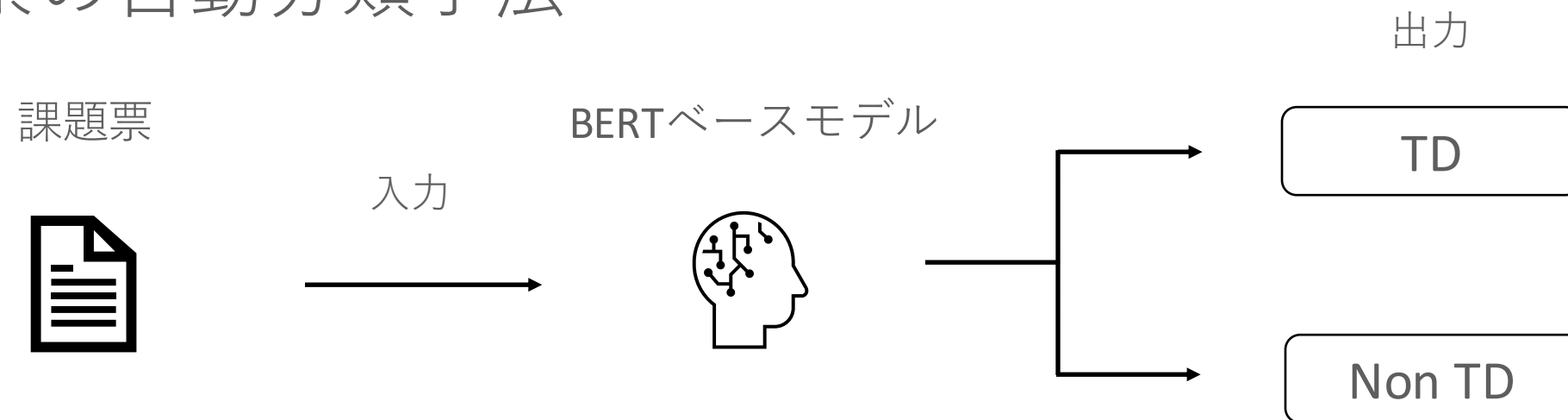


このいずれかのタイミングでラベルを付与しTDを管理する

大量の課題票に対して手動でラベル付けを行うのは労力がかかる

先行研究

課題票の自動分類手法^[1]



図：先行研究の手法の概要

表：提案手法の分類精度（モデル：DeBERTaV3）

	Precision	Recall	F1
TD	0.88	0.85	0.87
Non TD	0.88	0.91	0.90

高い精度で入力された課題票が技術的負債に関するか否か予測できた

[1]Daniel Skryseth, Karthik Shivashankar, IldikÅLo PilÅLan, and Antonio Martini. Technical debt classification in issue trackers using natural language processing based on transformers. In 2023 ACM/IEEE International Conference on Technical Debt (TechDebt), pp. 92–101, 2023.

技術的負債の種類

技術的負債は開発過程の様々な場面で発生するため
いくつかの種類に分けられる[2]

種類	説明
Design	内部メソッド設計とハイレベルアーキテクチャで使用される技術的なショートカット
UI	UI要素の負債を指す
Performance	時間やメモリ使用の面でシステムパフォーマンスに影響を与える負債
Infrastructure	サードパーティツール, 古い技術, または非推奨APIに関する負債
Documentation	不完全, または古くなったドキュメントを指す
Code Style	コードスタイル違反を指す
Build	ビルドをより複雑または遅くするスクリプトを使用する場合など, ビルドシステムの負債を指す
Security	システムデータを公開したり, ユーザ権限アクセスを危うくするショートカットを指す
Requirements	実装の問題に繋がる要件仕様の負債を指す
Test	テストの欠如, または既存コードをテストするための一時的な対応に関する負債

[2]Zengyang Li, Paris Avgeriou, and Peng Liang. A systematic mapping study on technical debt and its management. J. Syst. Softw., Vol. 101, No. C, p. 193–220, March 2015.

先行研究の問題点

問題点

先行研究：技術的負債に関するか否かの2値分類
種類を考慮した管理になっていない

とるべき対応の差

- 種類によって問題の性質が異なる

例)UI負債 : UIコンポーネントに関する問題

Security負債 : 脆弱性として悪用される可能性のある問題

種類	除去の優先度
UI	低
Security	高

種類ごとに除去する優先度が異なる



技術的負債は種類ごとに分類して管理すべき

種類別自動分類

考えられるアプローチ

- BERTなどをはじめとした最新NLP技術の利用
ファインチューニングに大量のデータが必要
- 問題点

技術的負債の種類によっては頻繁に見られる種類と見られない種類がある^[3]

例：技術的負債に関する課題票のうち Design負債 : 59.1% Security負債 : 1.1%

➡ 件数の少ない種類はファインチューニングに必要なデータ数が足りない

ファインチューニングが必要ない手法をとるべき

[3]Laerte Xavier, Fabio Ferreira, Rodrigo Brito, and Marco Tulio Valente. Beyond the code: Mining self-admitted technical debt in issue tracker systems. In Proceedings of the 17th International Conference on Mining Software Repositories, MSR '20, p.137–146, New York, NY, USA, 2020. Association for Computing Machinery

プロンプトエンジニアリング

プロンプトエンジニアリング

- 大量のデータなしに大規模言語モデルにタスクを解かせる手法

➡ 件数が少ない種類でも分類ができる

- モデルの選択

SEタスクではGPTシリーズがよく用いられる^[5]

新たなアプローチで作られたo1は研究事例が少ない

└ 推論過程を学習するように事前学習されている

[5]Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, KailongWang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang.
Large language models for software engineering:A systematic literature review. ACM Trans. Softw. Eng. Methodol., Vol. 33, No. 8,December 2024.

目的・アプローチ

目的

最新のGPTシリーズであるGPT-4oとo1のSEタスクにおける
性能を比較する

アプローチ

技術的負債の種類に関する課題票の分類において
結果にどのような差があるのか調査する

データセット

- ・ 収集対象プロジェクト（2022年8月31日まで）

GitHub上で言語ごとにスター数上位1,000件 “debt” ラベルを含む

- ・ ラベル付与

プログラミング経験3年以上の大学院生2名1つまたは2つラベルを選び、不一致の場合は議論により統一

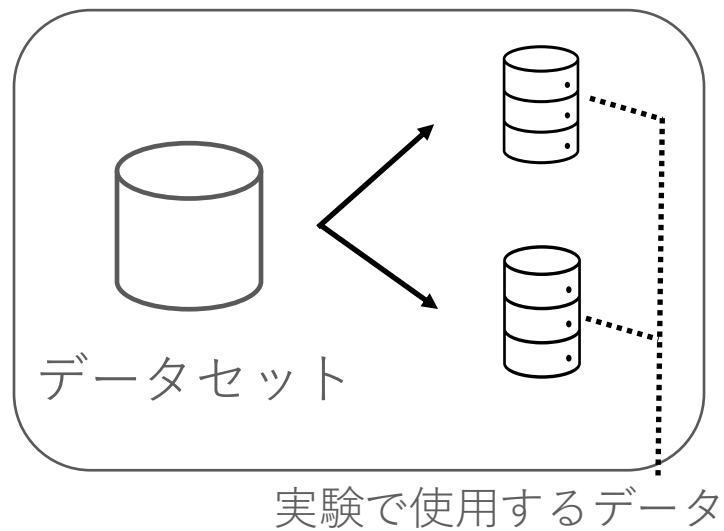
表：使用するデータセット^[4]

種類	件数
Design	1,093
Requirement	33
Test	371
UI	67
Performance	54
Security	14
Document	52
Infrastructure	132
Build	60
Codestyle	19

[4]田口舞奈(2025). 技術的負債に関する課題票の種別分類におけるテキストデータ拡張の効果検証, 和歌山大学大学院システム工学研究科修士論文

実験手順

1. データの整形



データセットから

- ・ Test負債が正例
- ・ Security負債が正例

の2つのデータを作成

種類	正例	負例
Test	371	1,481
Security	14	1,838

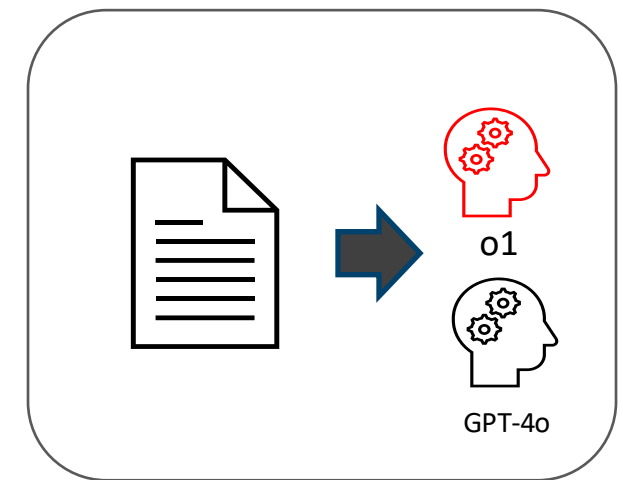
2. プロンプト作成



次の内容を含むプロンプトを作成

- ・ 入力のフォーマット
- ・ 行うタスクの内容
- ・ 分類するTDの種類の説明
- ・ 入力する課題票の内容
- ・ 出力するフォーマットの指定

3. 予測



作成したプロンプトを

- ・ GPT-4o
- ・ o1

に入力し予測を行う

作成したプロンプト

Test負債の分類のためのプロンプト

入力フォーマット

You are provided with a issue in this format:
Text: " the title and body of the issue "

タスクの説明

You have to assign it a label.
Possible labels are; "Test" or "Other"

ラベルの説明

The Test label indicates Test TD.

In software development, there are various types of issues reported as Issues. While bug reports and feature requests are typical examples, one important classification is Technical Debt (TD). Technical Debt itself is categorized into multiple types, and one of these is Test Technical Debt (TestTD)
Test TD refers to issues found in testing activities which can affect the quality of testing activities.

課題票を入力

Here is the text of the issue

Text: {課題票のタイトルと本文}

出力フォーマット

Answer with Test or Other

結果|Test負債

混合行列（10回の平均）

o1		予測	
		陽性	陰性
正解	陽性	361.2	9.6
	陰性	174.1	1,302.8

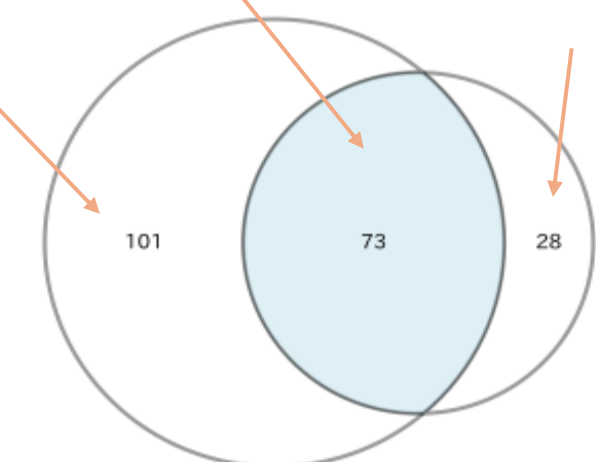
GPT-4o		予測	
		陽性	陰性
正解	陽性	351.1	13.6
	陰性	101.1	1,261.2

o1はGPT-4oに比べて偽陽性と予測した課題票が多かった

o1のみが偽陽性と分類
(101件)

共通して偽陽性と分類
(73件)

GPT-4oのみが偽陽性と分類
(28件)



偽陽性と分類した課題票

結果|Security負債

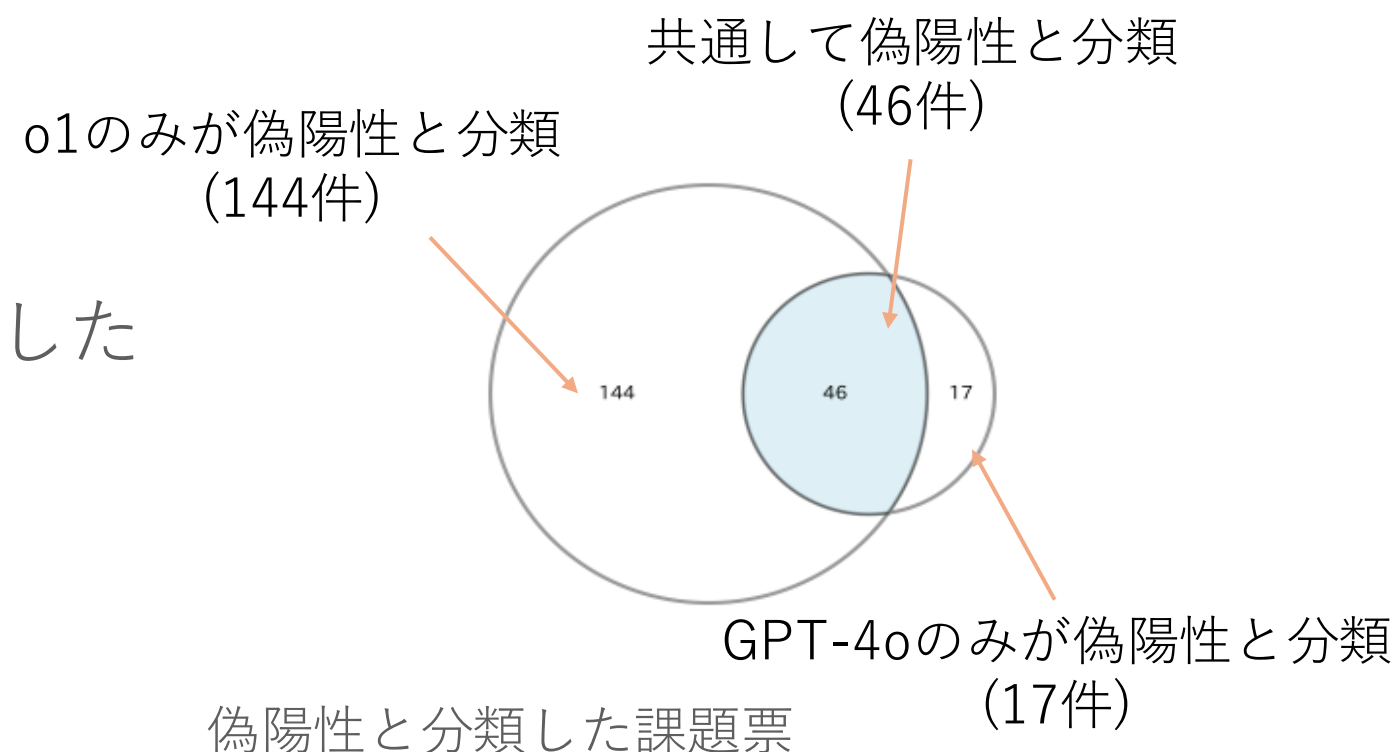
混合行列（10回の平均）

		予測	
		陽性	陰性
正解	陽性	12.2	1.8
	陰性	194.1	1,639.1

		予測	
		陽性	陰性
正解	陽性	12.0	1.9
	陰性	58.8	1,669.8

Test負債と同様に

o1はGPT-4oに比べて偽陽性と予測した課題票が多かった



課題票の分析

内容（日本語訳）

http-parser から llhttp への切り替えについて。http-parser は放棄されたソフトウェア（abandonware）となっており、node.js が llhttp への移行作業を開始したということです。

o1の思考過程（抜粋）

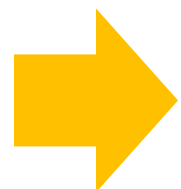
メンテナンスされていないモジュール(http-parser)を使い続けるのは脆弱性を引き起こす可能性がある

→ Security負債

GPT-4oの思考過程（抜粋）

ライブラリの依存関係の維持または更新が主な焦点である

→ その他



o1は課題票に明示されていない内容を出力できた

考察

o1とGPT-4oの違いにより結果に差が生まれた

- ・ GPT-4o

o1と比べて偽陽性と分類した件数が少ない

課題票に明示されていない内容は予測できない

- ・ o1

偽陽性がGPT-4oに比べると多い

課題票に明示されていない内容も予測できたが、その数は少ない

プロンプトについて

推論過程が異なるため、o1に適したプロンプトは異なる可能性

総括

目的

- 技術的負債の種類に関する課題票の分類におけるGPT-4oとo1の比較

結果

- o1にのみ正しく分類できる場合もあったが、件数は少なかった
- o1は比較的偽陽性として分類した件数が多かった。

開発者の意図を含んだ インラインコメント生成

大学院での研究

背景

コードコメント

役割：コードの理解を助ける説明文

- インラインコメント

コードの行内に記述される短いコメント
特定のコード行や処理の意図などを説明

```
// No guarantee that toString() returns a non-null value, despite what Spotbugs thinks.
```

- メソッドコメント

メソッド、関数全体の仕様を記述
形式的な記述

```
/**  
 * Checks if {@code [b <= a <= c]} or {@code...  
 *  
 * @param b the object to compare to ...  
 * @param c the object to compare to ...  
 * @return true if the base object is ...  
 */
```

コードコメント生成

コードコメントに関する問題点

- 手動でのコメント作成は時間がかかる
- コードコメントを自動生成する研究が行われている

既存のコードコメント生成

- 既存手法の多くはメソッドコメントに注目している^[1]
インラインコメントはあまり注目されていない

メソッドコメントとインラインコメントは同等の重要度^[1]

インラインコメントに注目した生成手法も必要

[1] A Comparative Study on Method Comment and Inline Comment
ACM Transactions on Software Engineering and Methodology, Vol. 32, No. 5, Article 126. Publication date: July 2023.

現状

方向性

先行研究^[2]で「開発者の意図」を考慮する必要性が取り上げられた
→インラインコメント生成でも”意図”を組み込む手法を提案する

懸念点

- データセット

“意図”を含んだインラインコメントを収集することが困難な可能性

- 評価

生成できたとして、どのように評価するか

- 研究の意義

“意図”を含めることができたことによる影響を測るのが困難

[2] Developer-Intent Driven Code Comment Generation

ICSE '23: Proceedings of the 45th International Conference on Software Engineering Pages 768 - 780

