

生成AIを用いた 技術的負債の 詳細文生成手法の構築

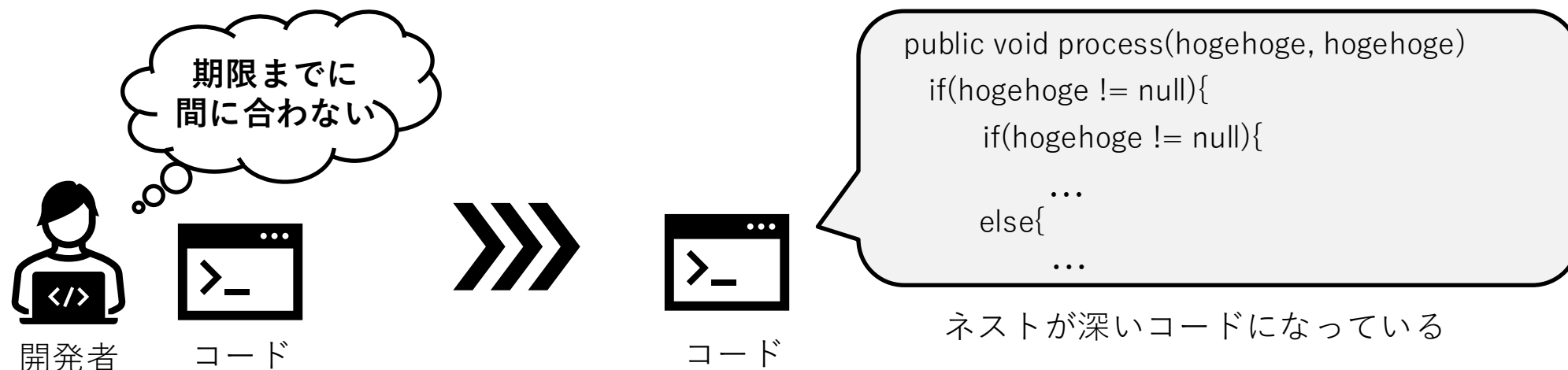
オープンソースソフトウェア工学研究室

学籍番号：S2420102

氏名：野口 優輝

技術的負債 (Technical Debt)

ソフトウェア開発において、納期やコストの制限によって導入される質の低い設計やソースコード^[1]



➤ 技術的負債が蓄積していくと起こること

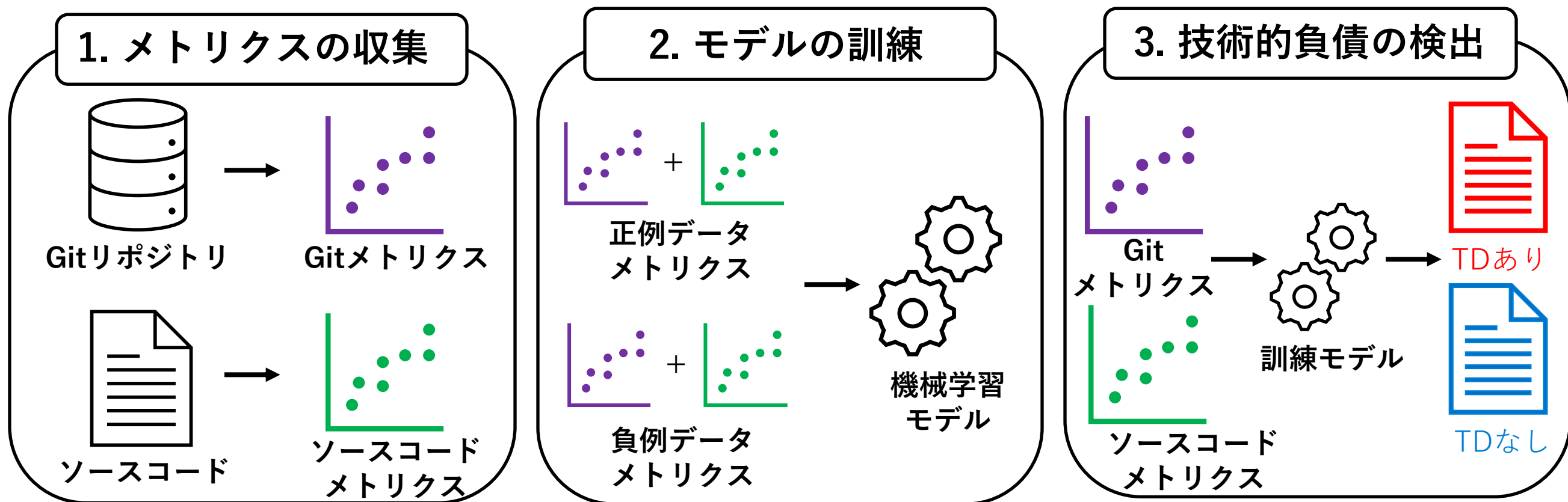
- 開発速度の低下, ソフトウェアの保守性低下, モチベーションの低下
→ かえってソフトウェア開発において非効率になってしまう

技術的負債は**早期に検出し除去される**ことが望まれる

[1]Ward Cunningham. The wycash portfolio management system. SIGPLAN OOPS Mess., Vol. 4, No. 2, pp. 29–30, 1992.

技術的負債を検出する研究

機械学習を使って、ソースコード中のクラスに含まれている技術的負債を検出^[2]

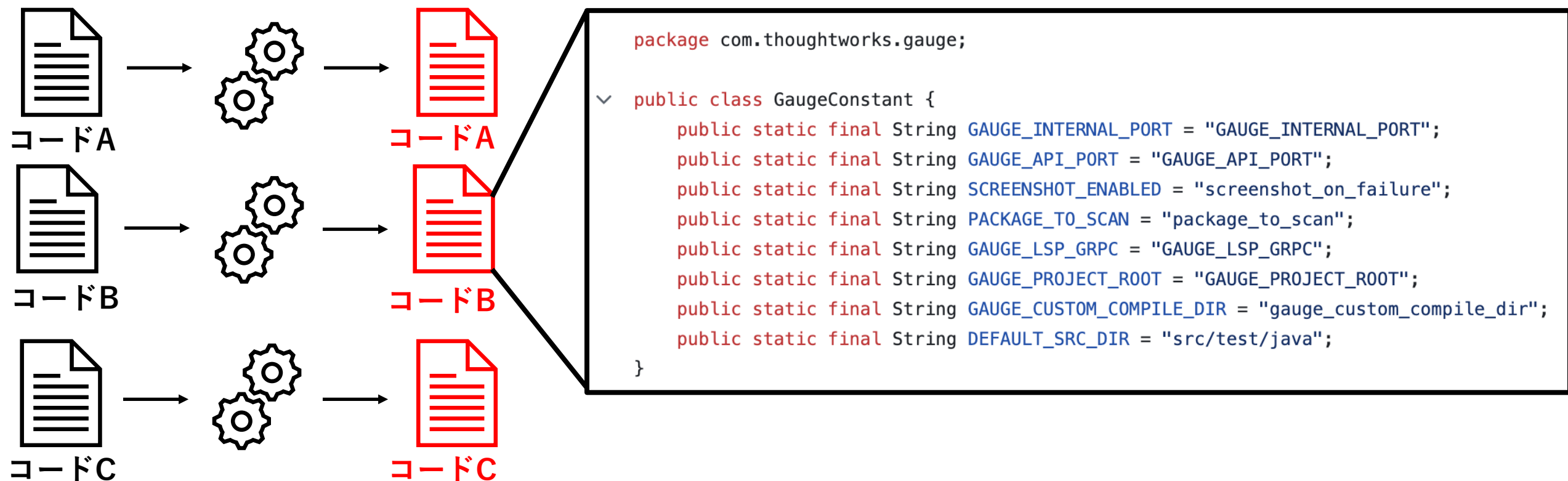


より高い精度で技術的負債を検出でき、技術的負債の管理が促進

[2] D. Tsoukalas, A. Chatzigeorgiou, A. Ampatzoglou, N. Mittas and D. Kehagias, "TD Classifier: Automatic Identification of Java Classes with High Technical Debt," 2022 IEEE/ACM International Conference on Technical Debt (TechDebt), Pittsburgh, PA, USA, 2022, pp. 76-80

技術的負債の検出研究：課題点

技術的負債を検出するだけで、具体的な詳細がわからない



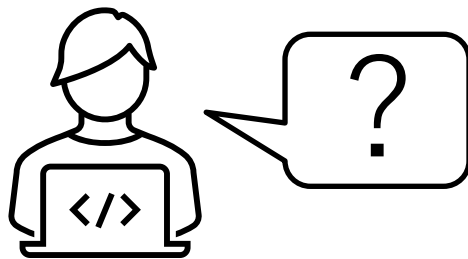
- 技術的負債を検出できても、開発者が理解できないといった状況が起こってしまう

→技術的負債の管理などに無駄なコストがかかってしまう可能性

技術的負債の検出研究：課題点

技術的負債を検出するだけで、具体的な詳細がわからない

負債がなぜ発生したのか
どの負債を修正すべきなのか
どの程度深刻な負債なのか



```
package com.thoughtworks.gauge;  
  
✓ public class GaugeConstant {  
    public static final String GAUGE_INTERNAL_PORT = "GAUGE_INTERNAL_PORT";  
    public static final String GAUGE_API_PORT = "GAUGE_API_PORT";  
    public static final String SCREENSHOT_ENABLED = "screenshot_on_failure";  
    public static final String PACKAGE_TO_SCAN = "package_to_scan";  
    public static final String GAUGE_LSP_GRPC = "GAUGE_LSP_GRPC";  
    public static final String GAUGE_PROJECT_ROOT = "GAUGE_PROJECT_ROOT";  
    public static final String GAUGE_CUSTOM_COMPILE_DIR = "gauge_custom_compile_dir";  
    public static final String DEFAULT_SRC_DIR = "src/test/java";  
}
```

プログラム実行時に、GAUGE_API_PORTが今後不要となる

- 技術的負債を検出できても、開発者が理解できないといった状況が起こってしまう

→技術的負債の管理などに無駄なコストがかかってしまう可能性

TD Issue

課題票（Issue）にTDラベルを割り当てて明示される

--- Issue ---

課題管理システムに記録される報告のこと

不具合や機能欠陥など様々な課題が管理されている

タイトル

techdebt: use of APNs binary format #26

New issue

本文

AndrewX192 opened on Apr 25, 2019

While setting up push notifications for the iOS application, I noticed that pushy (the library, not the service) is using a version from 2015 which pre-dates Apple's transition to HTTP/2. I'm not sure how much longer this will be supported but it seems like the new HTTP/2 format is easier to work with and the path forward. We should upgrade the pushy library and switch to HTTP/2.

Assignees

No one assigned

Labels

techdebt

Type

No type

techdebt

‘debt’などのTDに関するラベルが貼られる

AndrewX192 added techdebt on Oct 27, 2019

TD Issueの例

- TD Issueの本文には技術的負債について記載される
 - 技術的負債の問題点などを理解することができる

TD Issue：問題点

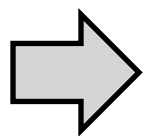
TD Issue本文での質が悪い

- 技術的負債を理解するために
 - どのような症状を持っているか
 - どのような影響をもたらすのか
 - 技術的負債を防ぐには，将来的にどのような設計にするべきか
- 実際のTD Issue本文

TD Issue本文例

例 1) **Add a test to NetworkIsServiceTest** witch cover changes from #2207

例 2) We will **have a file** that gets all the possible elements for well and wellbore and then will **pass that through the merge logic.**



将来行うべき内容のみの記載で，実際に技術的負債が
どんな症状・影響を与えているのかわからない

目的・アプローチ

目的

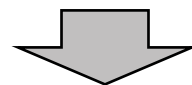
- 検出された技術的負債の詳細文（TD Issue本文）を自動生成する手法の構築

アプローチ

- 生成AI（GPT5-nano）を用いて詳細文を生成

----- 生成AIを用いる利点 -----

- ・ファインチューニングを必要としない
- ・柔軟な文章を生成することができる



学習にかかるコストを削減でき、
プロンプトの最適化から目的に応じた詳細文を生成できる

アプローチ：概要

1. データセットの作成



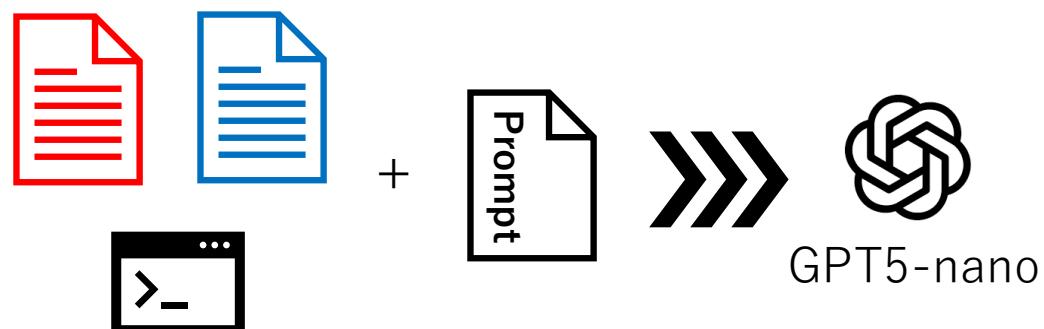
リポジトリからTD混入前後の
ファイルとTDコード片を取得

2. プロンプトの作成



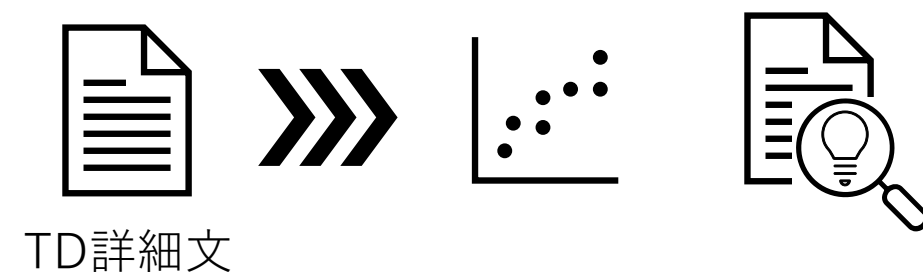
TDコード片が混入することで
与える影響・将来の改善案を
説明するようにプロンプトを作成

3. 生成AIにプロンプトを入力



二つのコードとTDコード片を
プロンプトに組み込み詳細文を生成

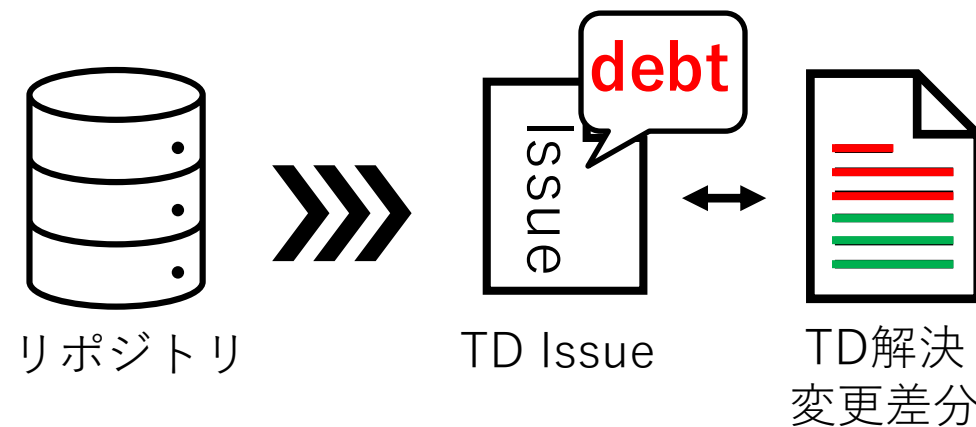
4. 生成された詳細文の評価・分析



2通りの方法で評価
・人間評価
・LLMを使って評価

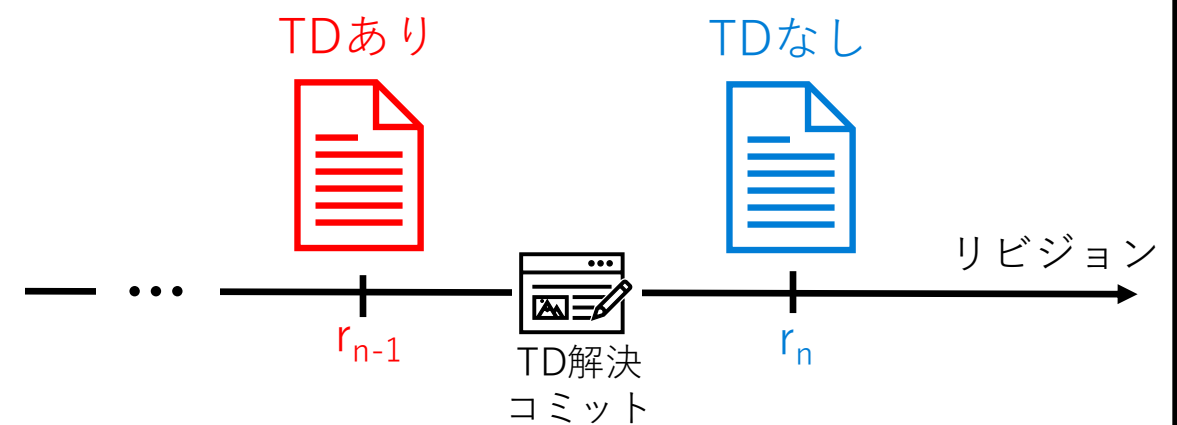
1. データセット：作成手順

1. TD Issueと紐づく変更差分を取得



リポジトリからTD Issueと
紐づくTD解決変更差分を取得

2. TD解決前後のファイルを取得



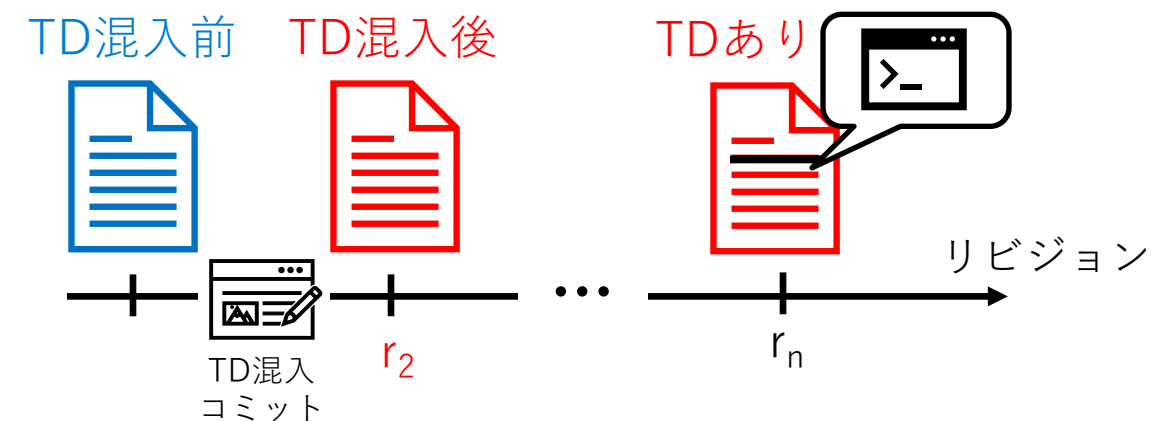
TDが解決されたコミットから
TD解決前後のファイルを取得

3. TDコード片を取得



TD解決変更差分から
削除行であるTDコード片を取得

4. TD混入前後のファイルを取得



TDコード片が初めて出現した
リビジョンからTD混入コミット
を特定し、前後のファイルを取得

対象プロジェクト

140のプロジェクトに対象に分析を実施

- 開発言語：Java
 - 多くの大規模プロジェクトがあり，TDの研究が盛んに行われるため
 - [3]のデータセットをフィルタリングし，利用

フィルタリング

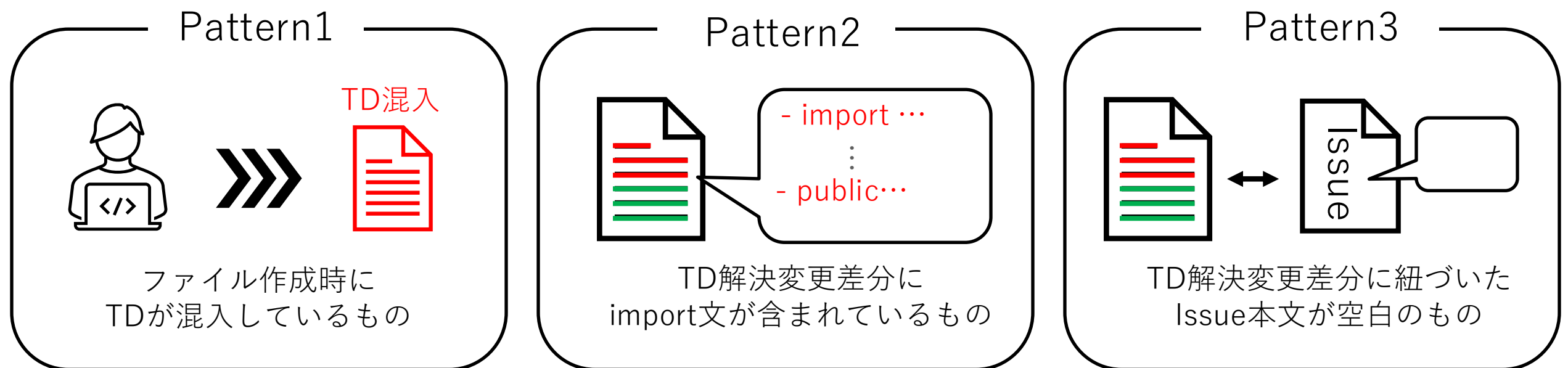
- 公開されているリポジトリであること
- 過去に技術的負債の除去（**コードの変更**）が行われていること

[3] D. Skryseth, K. Shivashankar, I. Pilán and A. Martini, "Technical Debt Classification in Issue Trackers using Natural Language Processing based on Transformers," In Proceedings of 6th International Conference on Technical Debt (TechDebt), pp. 92-101, 2023

1. データセット作成

TD混入前後のファイル特定

- 以下は本研究のデータセット作成からは除く



最終的なデータセット

- **3,861件**のTDコード片とTD混入前後のファイルを取得

2,3. プロンプトの作成・生成AI入力

プロンプト作成（英語で作成）

- TDを説明するための基本的な指示を記述
- 出力を統一するために，フォーマットを指定

「TDコード片が混入することによる影響を説明せよ」

「TDを解消するために，今後の設計をどのようにするべきかを提案せよ」

出力フォーマット

TDコード片が混入したことによる影響の説明：

提案する今後の設計と説明：

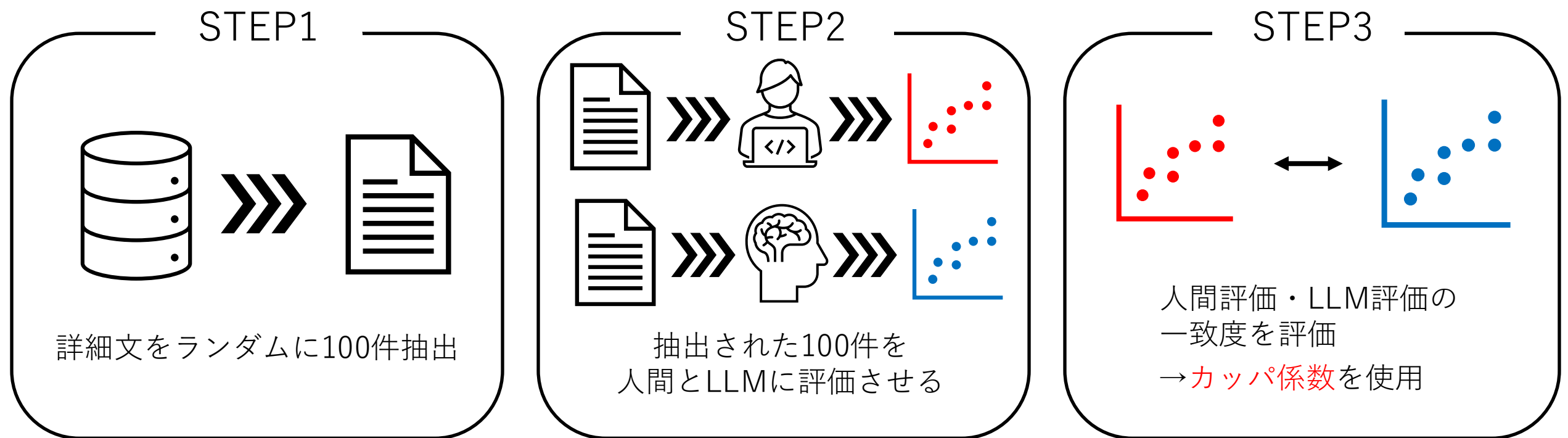
生成AI入力

- 使用モデル：GPT5-nano
 - 最速かつ，低価格なモデル

4. 予備実験：評価方法

人間とLLMで評価を実施

- 考えている実験手順



- 評価用LLMモデル：GPT5-nano（まだ未定）
- カッパ係数が比較的高い場合，すべての詳細文をLLMで評価

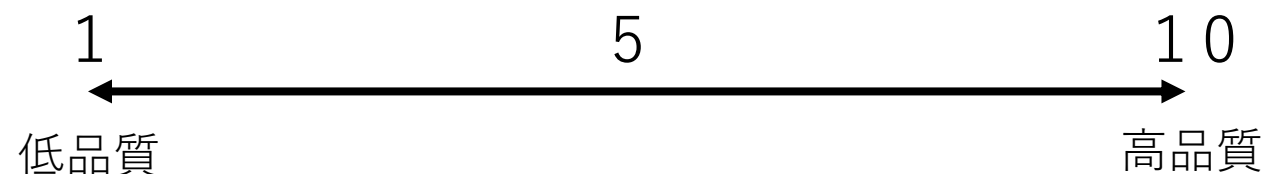
評価項目

5つの評価項目を用いて実施

- 明瞭さ：文章そのものの内容がわかりやすいか
- 簡潔さ：冗長でなく，しっかりまとまっているか
- 表現力：文法的に正確・流暢であるか
- 合理性：TDについて論理的な説明を提供しているか
- 網羅性：TDに関する重要な詳細をすべて含んでいるか

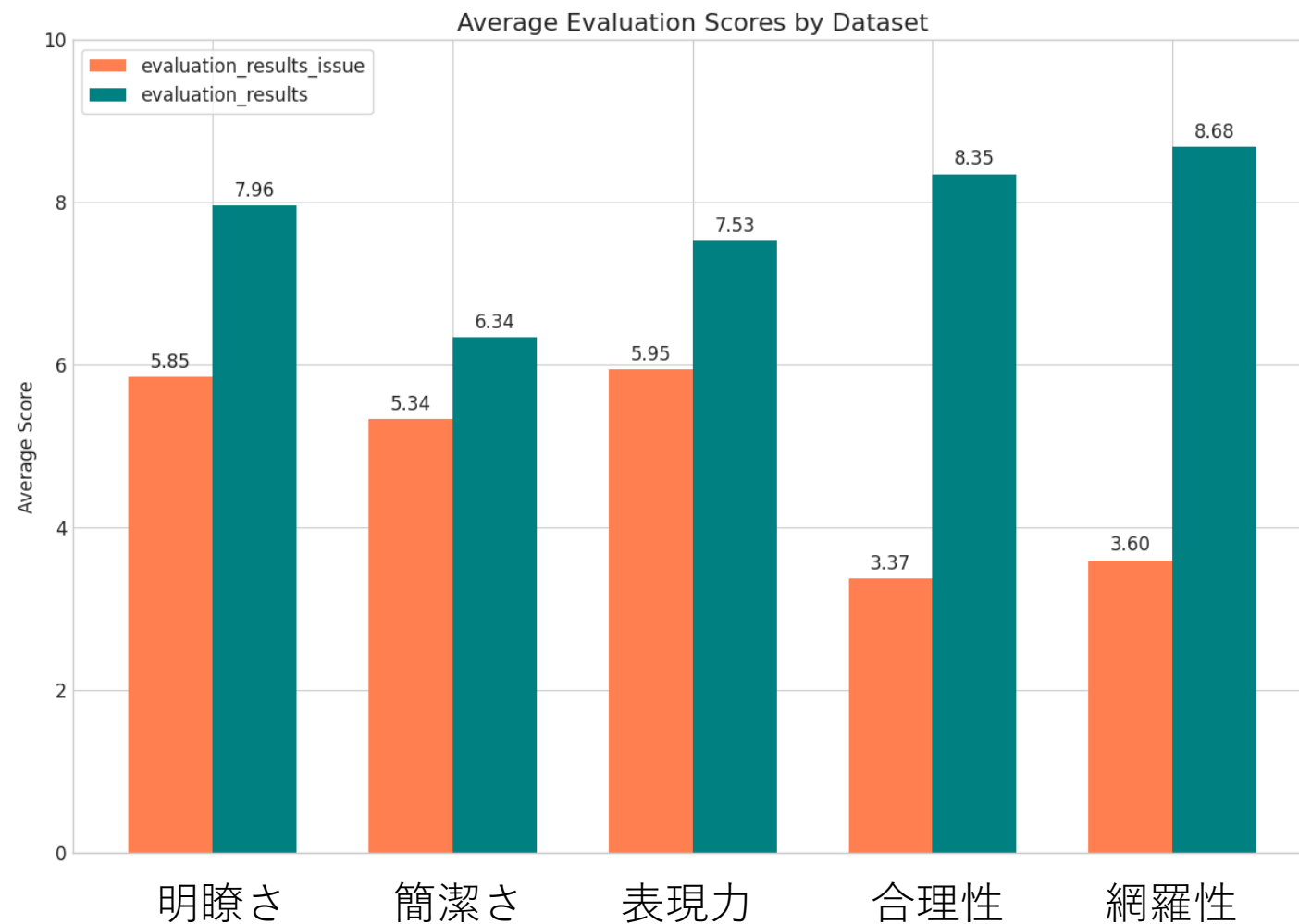
それぞれの評価項目をスコアリング

- 人間, LLMに1~10でスコアをつけてもらう



予備実験結果（LLM評価）

比較のため、Issue本文もTD詳細文と同様に評価



全ての指標で、Issue本文のスコアを上回った

- 質の高い詳細文が生成されていることがわかる

今後の予定

生成された詳細文と対応するIssue本文を人間が評価

- 2名ほど募って、評価してもらう
 - 意見の不一致がある場合は、議論
- LLM評価とのカッパ係数も算出

評価用LLMの選定

- TD詳細文生成と同様にGPT5-nanoを使うか？
 - LLMでテキストを評価している論文をサーベイ

補足：プロンプト

You are an AI assistant that analyzes two code snippets and reports the issues you find as a GitHub Issue.

The two code snippets to analyze are:

- Code without technical debt
- Code with technical debt

Technical debt refers to parts of the design or source code that are of low quality due to short-term delivery or cost constraints.

The code to compare represents the code before and after technical debt was introduced.

Please explain the impact of introducing the specific technical debt code snippet into the second code snippet.

Also, propose how future design should be approached to remove technical debt.

When doing so, use a standard diff format (e.g., a format generated by git diff or similar tools) to clearly show only the diff for the specific technical debt code snippet and its proposed fix.

Do not include any other code changes or related diffs.

The diff should indicate additions, deletions, and modifications.

Input Format:

Code without technical debt: {TD混入前のファイルを入力}

Code with technical debt: {TD混入後のファイルを入力}

The specific technical debt code snippet: {TDコード片を入力}

Output Format:

Explanation of the impact of adding {TDコード片を入力}:

Proposed future design and explanation: