

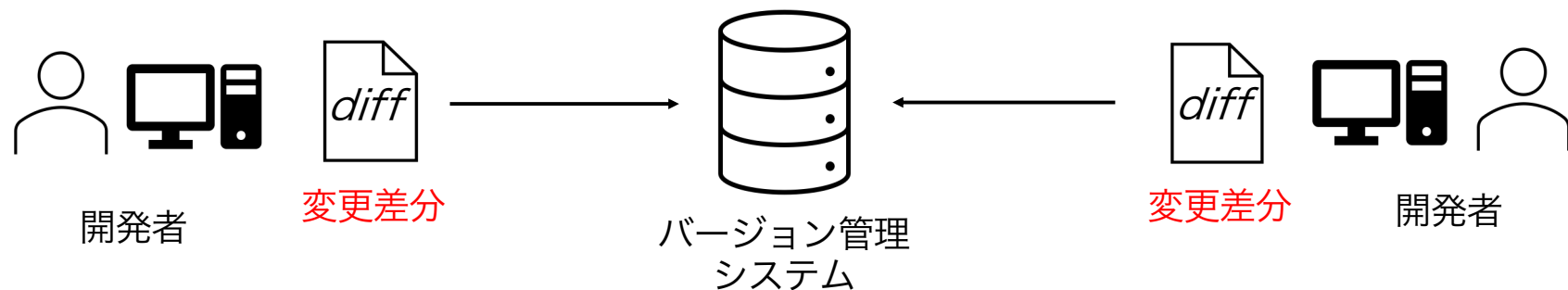
# 大規模言語モデルを用いた 変更内容と変更理由を含む コミットメッセージの自動分類

---

オープンソースソフトウェア工学研究室  
S2420142 山村慎太郎

# コミット

Gitにおいて、ファイル・ソースコードの**変更**を、バージョン管理システムに**登録**すること



コミットには通常、コミットメッセージが付与される



# コミットメッセージの現状と課題

メッセージには、**変更内容**と**変更理由**の両方含まれるのが開発者にとって好ましい[3]

変更部分 @Bean  
- @Primary  
public Integ

変更差分<sup>1</sup>

Remove @Primary from IntegrationMBeanExporter

Commit [3ea84f9](#) has wrongly introduced a `@Primary` marker on `IntegrationMBeanExporter` so any use of both Spring's JMX support and Spring Integration's JMX support leads to an exception. This commit makes sure to remove the unnecessary `@Primary`

コミットメッセージ<sup>1</sup>

変更理由

- 必要性：メンテナンス作業の効率化[3]，欠陥発生率の低減[5]

一方，これらの情報が記述されないケースも存在[3]

- 変更内容や理由が欠如しているメッセージ
- 何も記述されず，空白となっているメッセージ

➤ 原因：開発者がメッセージを書くための時間や動機の不足

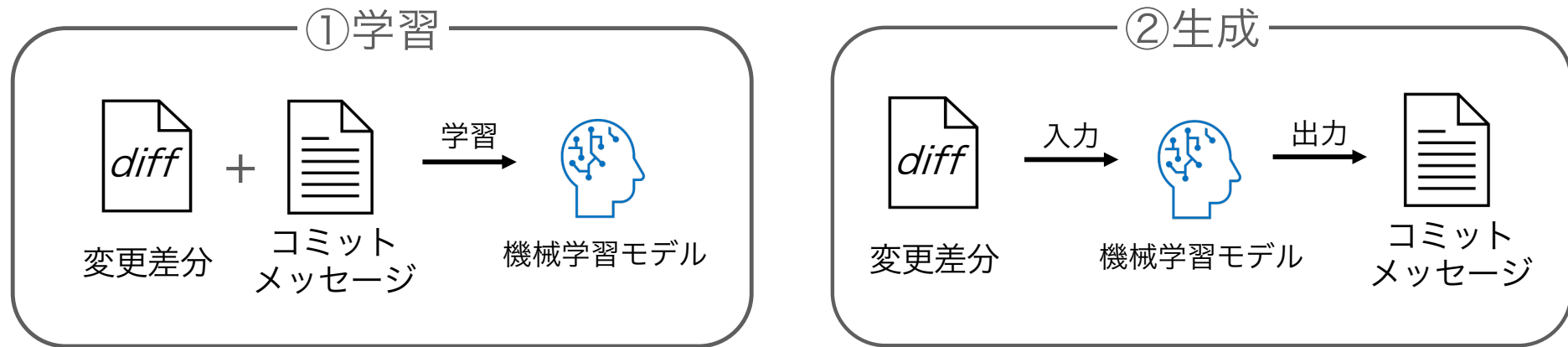
1. <https://github.com/junit-team/junit4/commit/078e45bd31a9920e00633ad6ce658e22529304f2>

[3] Yingchen Tian, Yuxia Zhang, Klaas-Jan Stol, Lin Jiang, and Hui Liu. 2022. What makes a good commit message? In Proceedings of the 44th International Conference on Software Engineering (ICSE '22)

[4] Jiawei Li, Iftekhar Ahmed, Commit Message Matters: Investigating Impact and Evolution of Commit Message Quality, 45th International Conference on Software Engineering (ICSE2023)

# コミットメッセージの自動生成

例：学習ベース<sup>[5],[6]</sup>



① 変更差分とメッセージのペアを訓練データとしてモデルを学習

② 変更差分からメッセージを自動生成

自動で生成できるようにすることで、開発者の時間・  
動機不足に対処

[5] Zhongxin Liu, Xin Xia, Ahmed E. Hassan, David Lo, Zhenchang Xing, and Xinyu Wang. 2018. Neural-machine-translation-based commit message generation: how far are we?. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, Montpellier, France, September 3-7, 2018. ACM, 373-384.

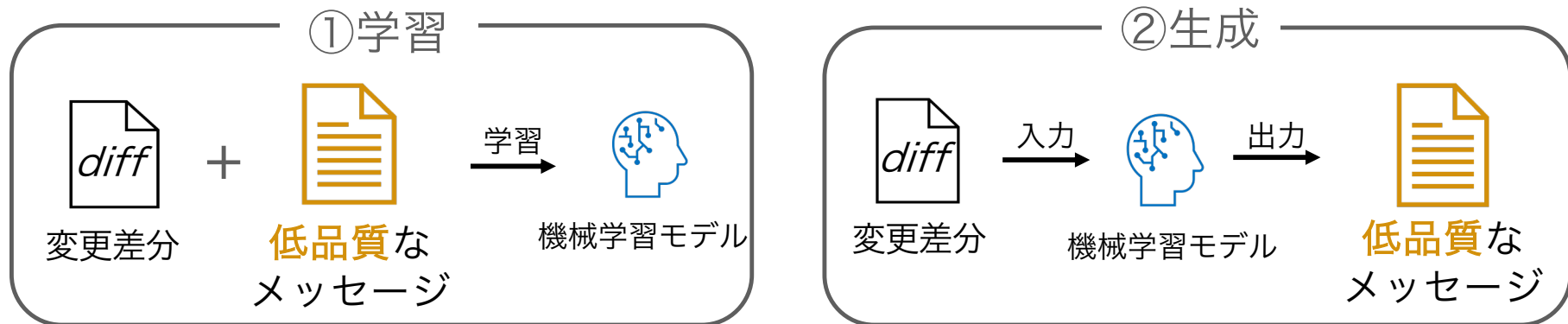
[6] Lun Yiu Nie, Cuiyun Gao, Zhicong Zhong, Wai Lam, Yang Liu, and Zenglin Xu. 2021. CoreGen: Contextualized Code Representation Learning for Commit Message Generation. Neurocomputing (2021).

# 自動生成の課題

## 学習データの品質が保証されていない

- 学習には、OSSプロジェクトのメッセージが使用される
- 調査された5つのOSSプロジェクトのメッセージのうち、平均約44%のメッセージに改善の余地あり<sup>[3]</sup>

## 学習データの品質に問題があると、生成されるメッセージの品質にも影響



- 変更内容・理由を含まないメッセージが生成されてしまう
- 一方、品質が保証されたデータセットはほとんど存在しない

# 先行研究<sup>[3]</sup>

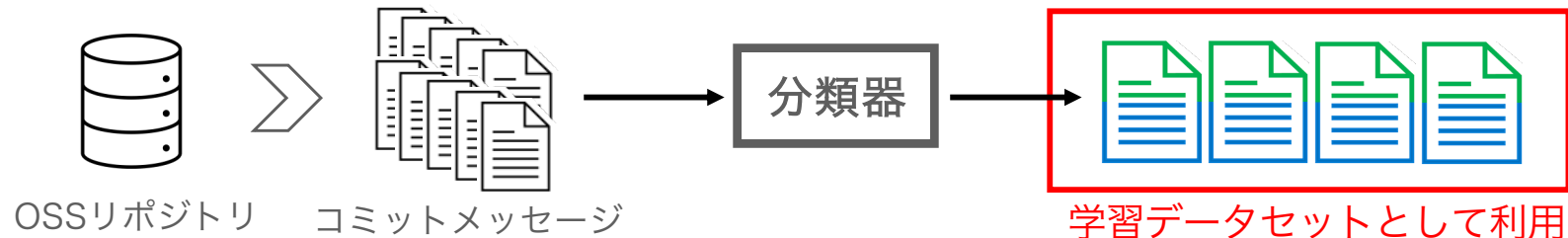
## 変更内容・理由を含むメッセージを自動で識別する分類器

- 変更内容・理由の有無を手動で分類した約1600件のデータで学習



## データセットの作成に活用可能

- 既存のOSSリポジトリから、変更内容・理由を含むメッセージを収集可
- 品質が保証された学習データセットの不足に対処



# 先行研究の課題点

先行研究の分類器（Bi-LSTM）の分類精度<sup>[3]</sup>

| 変更内容・理由 | Precision   | Recall | F1   |
|---------|-------------|--------|------|
| あり (正例) | <b>0.81</b> | 0.74   | 0.78 |
| なし (負例) | 0.70        | 0.78   | 0.74 |

- 正例予測のうち、実際に正例だった割合（Precision）は**約80%**
  - 約20%が誤分類（偽陽性）

データセットの作成に利用する場合、変更内容・理由が欠如したメッセージが20%程度混在

- 20%は学習において無視できない割合
- 大規模なデータセットを作成する場合、累積的に大きな影響

# 本研究の目的・アプローチ

コミットメッセージの自動生成に向けた、**変更内容**と**変更理由**を含む学習データセットの作成

- 高品質なデータセットを作成することで、生成メッセージの品質向上に貢献

アプローチ：事前学習済みLLMでメッセージを分類

- より高い精度での変更内容・理由の分類
- 特に，“Precision”を重視した分類を目指す
  - メッセージのデータは、既存のOSSリポジトリに大量に存在するため

# 大規模言語モデル（LLM）の利用

## 事前学習済みLLMの特徴

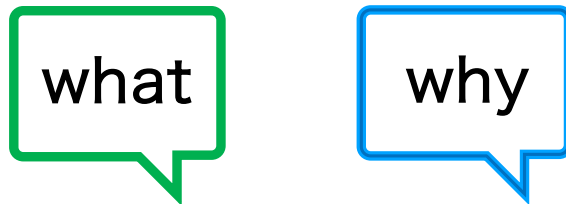
- 高度な文脈理解
  - 長いメッセージ，暗示的な内容にも対応可能
- 事前学習による知識の豊富さ
- プロンプトベースの柔軟な対応

## 変更内容・理由のより高精度な分類が行える可能性

- 高い文脈理解や事前学習による知識により，変更内容・理由を捉えやすい
- プロンプトに情報を追加することで，柔軟な分類を実現

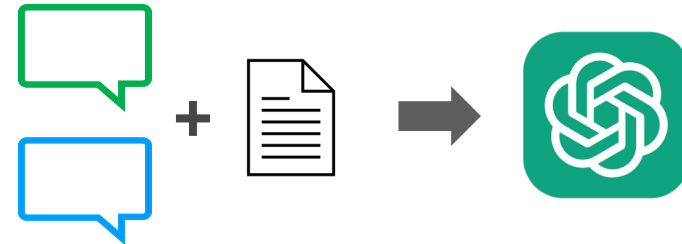
# 実験手順の概要

## 1. プロンプトの作成



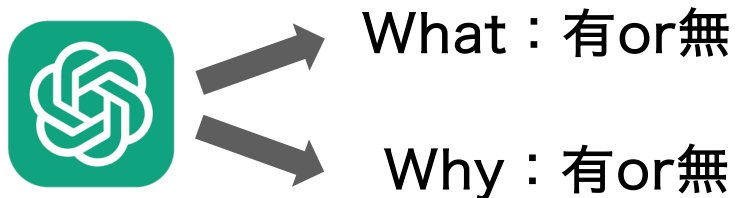
- ・ 変更内容の有無
  - ・ 変更理由の有無
- 分類用にそれぞれ作成

## 2. モデルへ入力



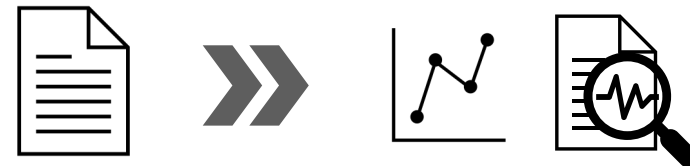
- ・ 使用モデル:gpt-4o
- ・ [3]でラベル付けされた1649件のメッセージを分類

## 3. 出力結果の評価



変更内容・理由それぞれについて  
Precision, Recall, F1

## 4. 結果の分析・改善



- ・ 偽陽性データの調査
- ・ 調査をもとに手法を改善

# 試験的な実験・結果 (10/3)

## プロンプト

- 変更内容or理由を含むかどうかを判断せよという文言
- 先行研究の論文から判断基準を抽出
- 偽陽性を減らせるような文言(理由の方のみ)

## 現状最もPrecisionが高かった際の分類精度

|      | Precision | Recall | F1    |
|------|-----------|--------|-------|
| 変更内容 | 0.912     | 0.705  | 0.795 |
| 変更理由 | 0.911     | 0.379  | 0.535 |

# 偽陽性の中身（変更理由）

プロンプトに組み込んだ分類判断基準の一つである, "ImPLY Necessity"（必要性の示唆）に引っかかってる？

- "Remove reference to obsolete mailing list"

→obsolete(廃止された)を"必要性の示唆"と判断？

- "Make serialization of Result compatible with 4.11"

→4.11との互換性を持たせるを"必要性の示唆"や"目的"と判断？

人によっては理由有りと判断しうる絶妙なラインのメッセージの偽陽性が多そう

- これらを正解ラベル通り識別させるのはまあまあ難しそう
- 判断の根拠も提示させるなど？

# 今後の方針

---

## 短期：分類精度の向上

- Precisionを中心に、できる限りの精度の向上を目指す
- プロンプト調整、偽陽性データの調査、分類後のフィルタリングなど
- 場合によっては取りこぼし(Recall)も考慮

## 中期：実際のOSSリポジトリへの応用可能性の検討

## 長期：収集したデータによる学習・生成・評価

# プロンプト

---

You are the assistant that identifies whether the incoming commit message contains the reason for the change.  
Return only 1 if the reason for the change is included, otherwise return only 0.

However, identify them based on the following criteria:

1. Describe Issue: The message should describe the error scenario, introduce the issue report, or describe shortcomings in the existing code.
2. Illustrate Requirement: It should illustrate usage needs, address out-of-date code, or adapt to changes in the runtime or development environment.
3. Describe Objective: The message should clearly state whether the goal is to fix defects or make improvements.
4. Missing Why (Common Sense): In some cases, the "why" might be common sense (e.g., fixing typos, adding test cases, or refactoring). While following these criteria, output as 1 only those that include the reason for the change as clearly as possible, since we want to reduce the number of false positives.

# データセットの内訳

## 変更内容・理由の分類における内訳

|      | 有    | 無   | 計    |
|------|------|-----|------|
| 変更内容 | 1397 | 252 | 1649 |
| 変更理由 | 1156 | 493 | 1649 |