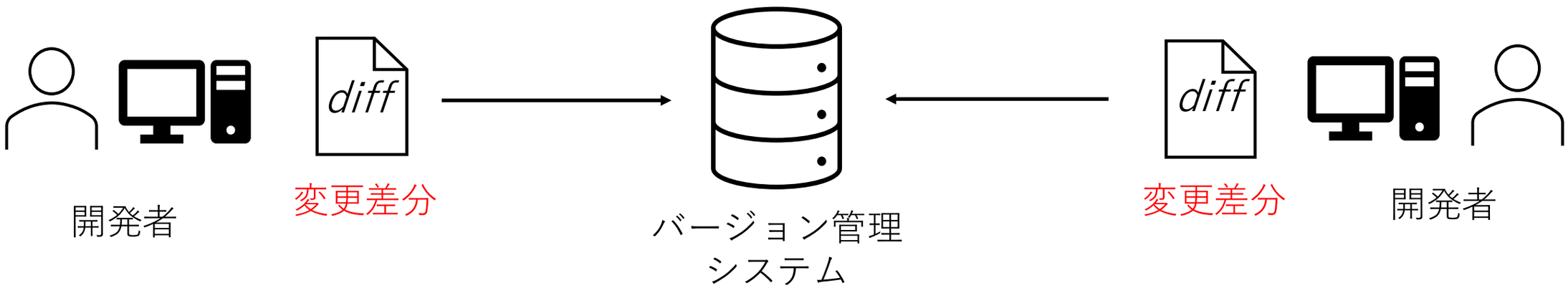
高品質なコミットメッセージの自動生成に向けたデータ収集手法の構築

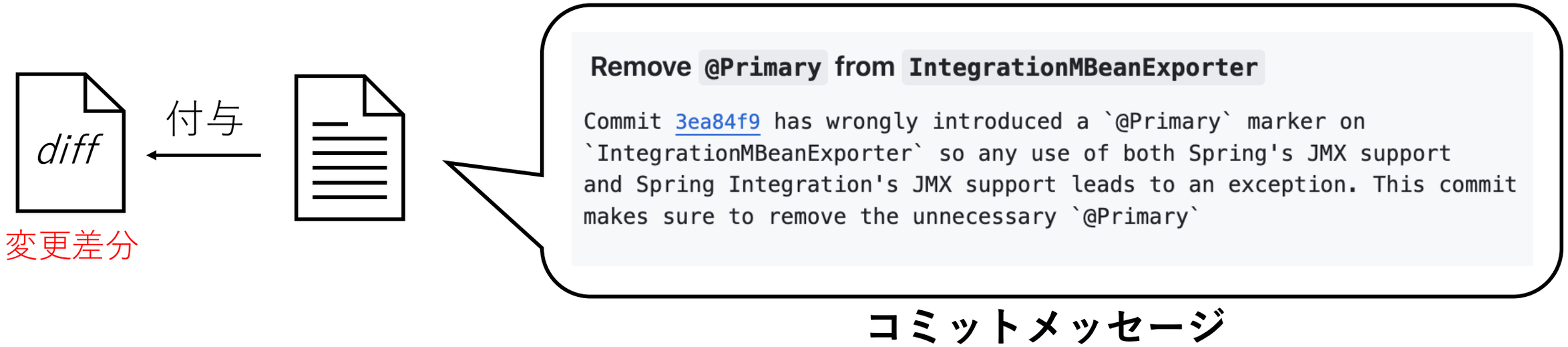
オープンソースソフトウェア工学研究室
S2420142 山村慎太郎

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[1]

変更内容

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`

変更理由

コミットメッセージ[1]

  - 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

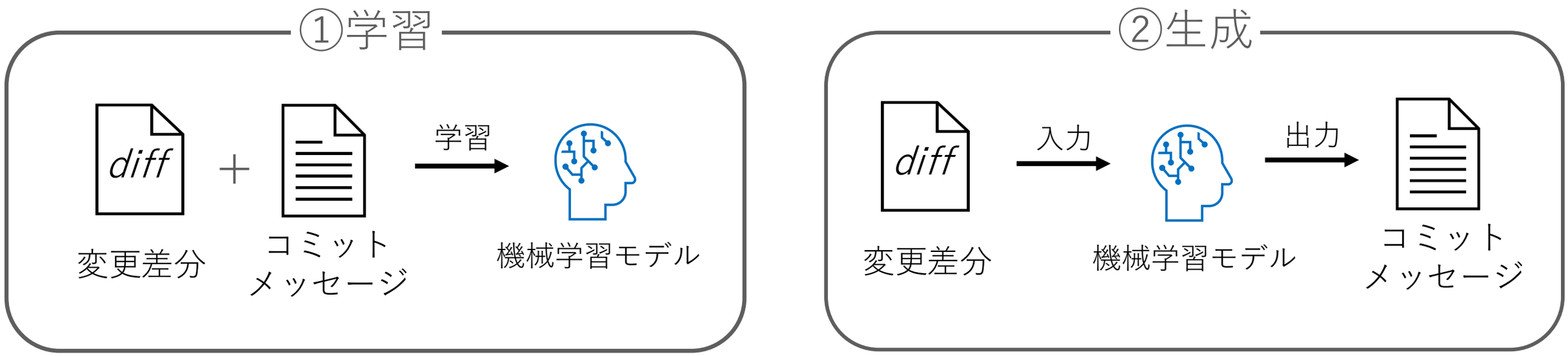
➢ 原因：開発者がメッセージを書くための時間や動機の不足

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[5],[6]



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

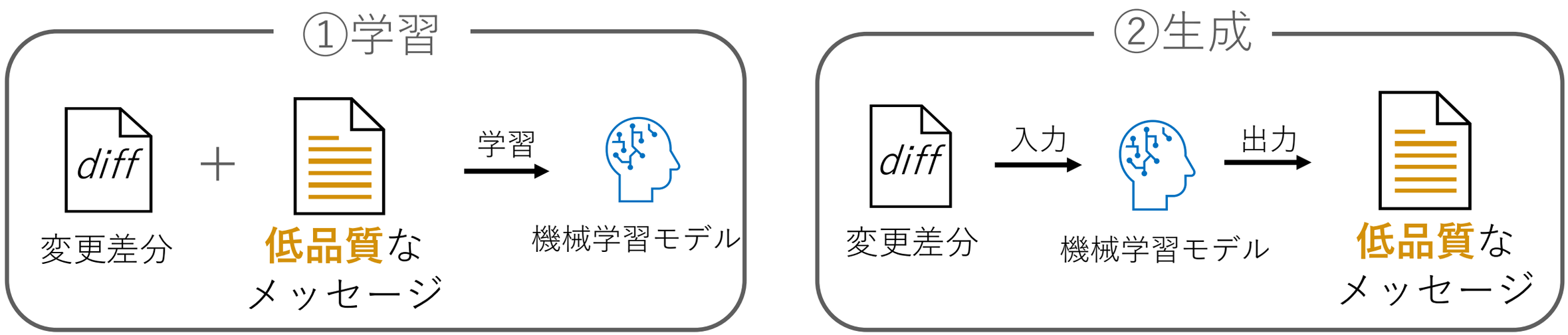
[5] Zhongxin Liu, Xin Xia, Ahmed E. Hassan, David Lo, Zhenchang Xing, and Xinyu Wang. 2018. Neural-machine-translation-based commit message generation: how far are we?. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, Montpellier, France, September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

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

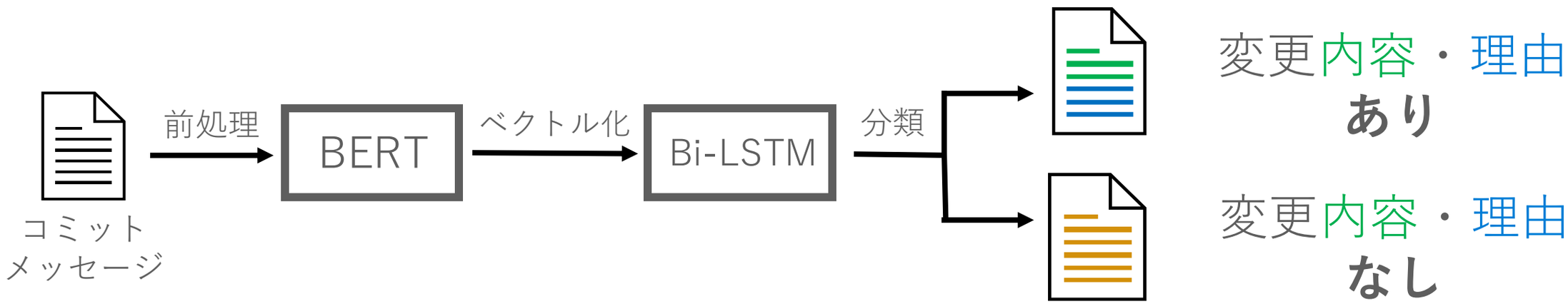


- 変更内容・理由を含まないメッセージが生成されてしまう
- 一方，品質が保証されたデータセットはほとんど存在しない

先行研究[3]

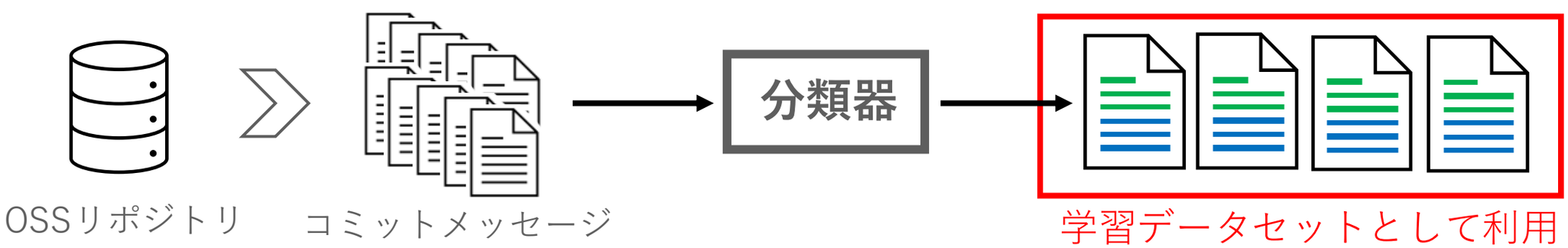
変更内容・理由を含むメッセージを自動で識別する分類器

- 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



[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)

先行研究の課題点

先行研究の分類器（Bi-LSTM）の分類精度[3]

変更内容・理由	Precision	Recall	F1
あり（正例）	0.81	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

[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)

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

研究の計画

（P1）変更内容・理由を含むメッセージの分類

- Precisionの高い分類手法の確立

（P2）変更内容・理由を含むメッセージの収集

- 既存のOSSリポジトリのデータに対し分類・収集

（P3）コミットメッセージの自動生成

- 収集したメッセージで学習・生成

研究の計画

（P1）変更内容・理由を含むメッセージの分類

- Precisionの高い分類手法の確立

今回

（P2）変更内容・理由を含むメッセージの収集

- 既存のOSSリポジトリのデータに対し分類・収集

進行中

（P3）コミットメッセージの自動生成

- 収集したメッセージで学習・生成

プロンプトエンジニアリングの活用

追加の学習(ファインチューニング)を必要としない

- 学習コストがかからず，**より大規模なモデル**を利用可能
- ➢ 変更内容・理由をより捉えやすい可能性

より柔軟な分類実験が行える

- プロンプトに情報を追加することで，目的に応じて**柔軟に対応**
- ➢ Precisionを重視した分類に貢献

➢ 使用モデル：GPT-4o

- 大規模なモデルで，柔軟なプロンプト設計が可能
- OpenAI APIを利用

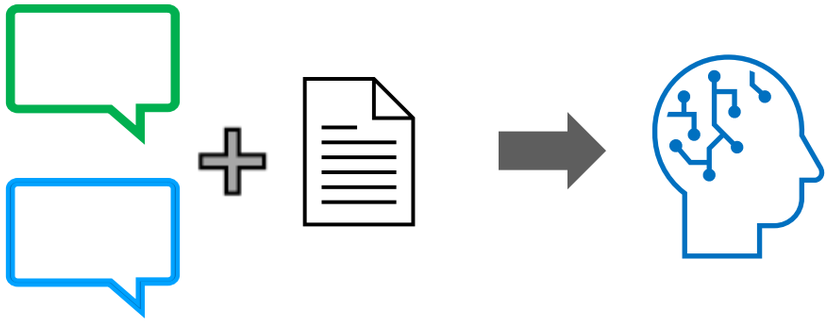
実験手順の概要

1.プロンプトの作成

変更内容分類用　　**変更理由分類用**

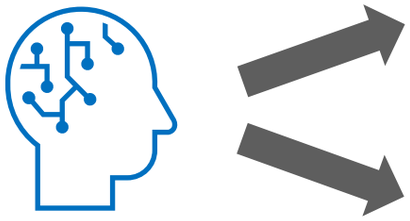
・変更内容の有無
・変更理由の有無
分類用にそれぞれ作成

2.モデルへ入力



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

3.分類結果の評価

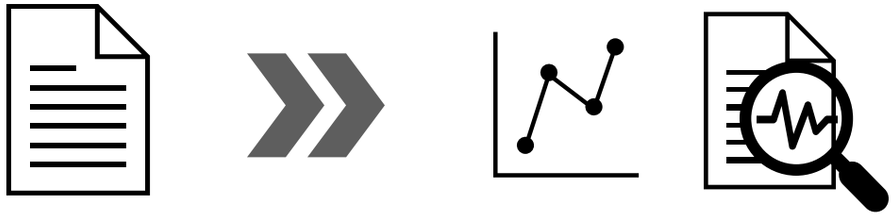


内容：有or無

理由：有or無

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
・調査をもとにプロンプトを改善

プロンプト作成手順①｜基本プロンプト

①基本プロンプトの作成・分類の試行

- メッセージを分類するための基本的な指示を記述（英語）
- Zero-shotでの分類

◆ 変更内容用 基本プロンプトの主な内容

・「変更**内容**が含まれているかどうかを判断せよ」
・「含まれていれば'1'を，含まれていなければ'0'のみを出力せよ」

◆ 変更理由用 基本プロンプトの主な内容

・「変更**理由**が含まれているかどうかを判断せよ」
・「含まれていれば'1'を，含まれていなければ'0'のみを出力せよ」

プロンプト作成手順②｜偽陽性の分析

偽陽性データの分析

- 基本プロンプト分類で，偽陽性となったデータの内容を調査

✓ 内容分類で，偽陽性となったメッセージの分析例

内容分類 偽陽性メッセージの例

例1）fixed issue #89 (wrong JavaDoc of)...

例2）Fix bug where request body isn't ...

"fix","fixed"の文言が散見

具体的な修正対象が無くとも，変更内容有りと誤認？

✓ 理由分類で，偽陽性となったメッセージの分析例

理由分類 偽陽性メッセージの例

例1）... This lets custom CallAdapter ...

例2）... This allows the regular mechanism of...

"This allows(lets)~" が散見

単なる変更の**結果や影響**を変更理由と誤認？

プロンプト作成手順③｜プロンプト改善

分析をもとに，プロンプトを改善

- 偽陽性を減少させられるような文を追加

◆ 変更内容用 追加した主な内容

・「“fix”や”fixed”から始まるメッセージには要注意」
・具体的な修正対象を含まないメッセージを弾くため

◆ 変更理由用 追加した主な内容

・「”This allows …”のような，結果や影響は理由と見做さない」
・変更理由との混同を防ぐため

試行・改善を繰り返し，最終的なプロンプトを作成

変更内容分類｜使用プロンプト

プロンプトに含めた主な内容（改善後）※1

- 基本プロンプト
  - “Determine if the commit message contains ... what was changed.”
  - “Return only ‘1’ if ... , otherwise return only’0’. ”
- 「“fix”や”fixed”から始まるメッセージには要注意」
  - ”Pay special attention ... starting with ‘fix’ or ‘fixed’ ... ”
- 「判断に悩むメッセージは，より厳格に判断せよ」
  - “When it doubt, ... more strictly and judge ...”

※1 完全なプロンプトは補足資料参照

変更内容分類｜使用プロンプト

プロンプトに含めた主な内容（改善後）

- 基本プロンプト
  - “Determine if the commit message contains ... what was changed.”
  - “Return only ‘1’ if ... , otherwise return only’0’. ”

◆判断が難しいメッセージに対しては厳しく判断させる

・Precision向上のため
・明確に変更内容が含まれていると言えない場合は’0’を出力せよ

- 「判断に悩むメッセージは，より厳格に判断せよ」
  - “When it doubt, ... more strictly and judge ...”

変更理由分類｜使用プロンプト

プロンプトに含めた主な内容（改善後）※1

- 基本プロンプト
  - “Determine if the commit message ... reason for the change.”
  - “Return only ‘1’ if ... , otherwise return only’0’. ”

- 「結果や影響は変更理由とは見なさない」
  - ”The following are not considered change reasons ...”
  - “Explanation of the results or effects of the change ...”

- 「変更理由は，しばしば以下のような表現で示されます」
  - “Change reasons are often indicated ...”
  - “because...”, “since...”, “due to...”, ”in order to...”

※1 完全なプロンプトは補足資料参照

変更理由分類｜使用プロンプト

プロンプトに含めた主な内容（改善後）

- 基本プロンプト
  - “Determine if the commit message ... reason for the change.”
  - “Return only ‘1’ if ... , otherwise return only’0’. ”

◆ 理由を示す表現の提示

・明示的に理由を含むメッセージを拾いやすくするため
・データセットのメッセージに散見された表現を数個列挙

- 「変更理由は，しばしば以下のような表現で示されます」
  - “Change reasons are often indicated ...”
  - “because...”, “since...”, “due to...”, ”in order to...”

データセット

先行研究[3]で提供されているデータセット

- GitHub上の5つのJavaプロジェクトからサンプリングして得た1649件のコミットメッセージ
  - 対象プロジェクト：Spring-boot, Apache Dubbo, Okhttp, Junit4, Retrofit
- 2人の著者により，変更内容・理由の有無をラベル付け

表：データセットのラベル別の分布

	内容・理由 両方含む	内容のみ	理由のみ	内容・理由 いずれも含まない	計
件数	969件	428件	187件	65件	1649件
割合	58.8%	26.0%	11.3%	3.9%	100%

[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)

変更内容の分類精度

結果

- 比較対象として，先行研究，基本プロンプトでの精度を掲載
  - いずれも条件が異なるため，参考程度

変更内容分類	Precision	Recall	F1
先行研究 (10分割交差検証)	**0.782**	0.645	0.689
基本プロンプト (3回平均)	**0.914**	0.768	0.835
改善後プロンプト (10回平均)	**0.981**	0.456	0.622

高いPrecisionを実現できた

- Recallは低く，取りこぼしはやや多い結果となっている

変更理由の分類精度

結果

- 比較対象として，基本プロンプトのみ，先行研究の精度を掲載
  - いずれも条件が異なるため，参考程度

変更理由分類	Precision	Recall	F1
先行研究 (10分割交差検証)	0.765	0.709	0.731
基本プロンプト (3回平均)	0.893	0.400	0.552
改善後プロンプト (10回平均)	0.924	0.085	0.156

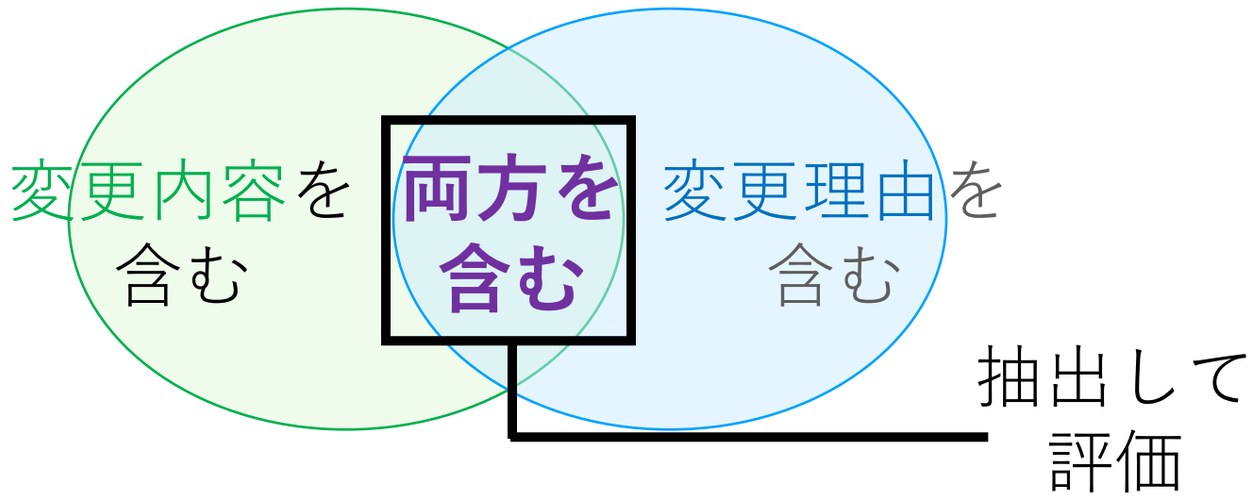
比較的高いPrecisionを実現できた

- Recallは著しく低く．取りこぼしは非常に多い結果となった

内容・理由両方を含むメッセージ

内容・理由両方を含むメッセージの分類性能の評価

- 内容/理由それぞれの分類で真となったものを最終的な真と判定



結果

	真陽性	偽陽性	Precision
平均(10回)	40.6件	1.2件	0.971
最高(Precision)	40件	0件	1.000
最低(Precision)	36件	2件	0.947

- 一回の試行で，内容・理由両方を含むメッセージを40件前後取得
  - データセット全体(1649件)の約2.5%程度

考察① 学習データとしての偽陽性の影響

10回の評価実験のうち，複数回偽陽性となった例

内容・理由分類 偽陽性のメッセージ例

5回：”Platform.getMtu is unused. **This was a failed experiment**”

4回：”… **AOSP moved OpenSSLSocketImpl to com.android.org.conscrypt.***
so add that as the first choice for unbundled apps. …”

いずれも先行研究では変更理由無しとラベル付け

- 変更理由が含まれているとも捉えられる
- ➢ 学習データとして用いる際の偽陽性の影響は小さいと言える

5回：https://github.com/square/okhttp/commit/3eda83ab14d334bf5a5cb900674df1552de74e73
4回：https://github.com/square/okhttp/commit/1192af61fd1f6a102a7d74422e83dfb1fc9a94b9

考察② 安定して真陽性となるメッセージ

10回の実験で、常に/時々 真陽性となったメッセージ
それぞれの特徴を分析

- データセット作成時に得られるメッセージの特徴を把握

✓ 安定して真陽性となっていたメッセージ例

- 理由を示す表現が使用されている

10回：”Delete .cvsignore files, because we're using Git now.”

- 長文が多く，問題や解決が明示的で，解釈の必要性が低い

10回：”...Notably, this would prevent LaunchedURLClassLoader from loading classes or resources on an interrupted thread. ...”

10回(上)：https://github.com/junit-team/junit4/commit/fa35d6d84a0b2c59254cb479633d0e20bc6e12e9
10回(下)： https://github.com/spring-projects/spring-boot/commit/85c0b44dcbbe4d201f8b7fd72a97c31c2d07a3ec

考察② 不安定に真陽性となるメッセージ

✓ **不安定に真陽性となっていたメッセージ例**

- 短文が多く，技術的な**背景知識や文脈理解の必要性**が高い

4回：”Reorganized the AssertTest, which had grown unwieldy.”

1回：”Clean out Java 8 READMEs to further discourage use.”

安定/不安定 まとめ

- 内容・理由が明示的で，説明が詳細なメッセージを安定して取得できる
- 簡潔なメッセージも一部取得できる

4回：https://github.com/junit-team/junit4/commit/f41f48d8f0a71620b878617e5fffb4b48c872978
1回： https://github.com/square/retrofit/commit/ebc0128add0d416a846ff661888910d16751dd34

今後の予定

（P1）変更内容・理由を含むメッセージの分類

- Precisionの高い分類手法の確立

今回

（P2）変更内容・理由を含むメッセージの収集

- 既存のOSSリポジトリのデータに対し分類・収集

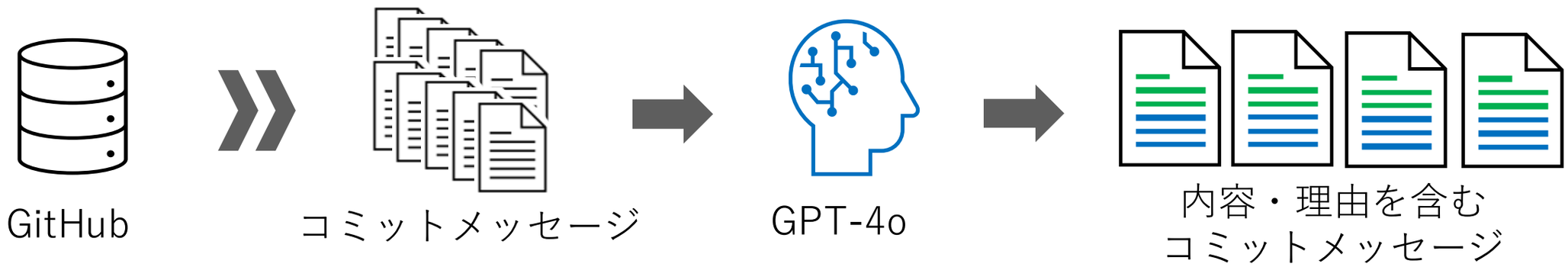
進行中

（P3）コミットメッセージの生成

- 収集したメッセージで学習・生成

次段階の構想｜学習データセットの構築

既存のOSSプロジェクトから新たにメッセージを収集



- 本手法により変更内容・理由を含むメッセージを収集し，学習データセットとして活用

データセット作成には，**大規模な収集・分類**が必要

- 分類実験時のRecallが低く，大量のコミットの収集が必要
- 収集にかかる時間や，他プロジェクトへの適用可能性が不明

現在の進捗①｜大規模データ収集

コミット情報の大規模な収集

- GitPythonにより収集
  - Gitの操作をPythonで行うためのライブラリ
  - リポジトリをローカルにクローンし，コミット情報を収集（制限無し）
- 100~200万件のコミット情報の収集を目標
  - 最終的な学習データとして，5~10万件程度を目安としているため
  - 収集後，GPT-4oで分類

現在の収集コミット数：約1,200,000件

- 対象プロジェクト数：418件
- かかった時間：約15時間

➢ 件数増加のため，フィルタリング条件の緩和も検討

現在の進捗②｜対処すべき課題

理由分類のRecallが低すぎる

- 大ゼミにて指摘
- 理由表現のキーワードベースによる分類精度と大差なし
  - “because”, “since” などを含むメッセージを真予測とした精度

変更理由分類	Precision	Recall	F1
gpt-4o（10回平均）	0.924	0.085	0.156
キーワードベース	0.917	0.048	0.090

➢ GPT-4oを利用する優位性を示せない

対処：プロンプトの再改善によるRecall向上

- 偽陰性を分析し，Precisionを維持しながらの向上を目指す

対外発表状況

第219回ソフトウェア工学研究発表会

- 「高品質なコミットメッセージの自動生成に向けたデータ収集手法の構築」
- 山村慎太郎，大平雅雄，木村祐太（**学生研究賞 受賞**）

補足1-A 基本プロンプト全文

変更内容 基本プロンプト（system）

```
Determine if the commit message contains
a description of what was changed.
Return only 1 if a change description is
included, otherwise return only 0.
```

理由内容 基本プロンプト（system）

```
Determine if the commit message contains
a reason for the change.
Return only 1 if the reason for the change
is included, otherwise return only 0.
```

補足1-B 変更内容プロンプト全文

変更内容 使用プロンプト（system）

```
Determine if the commit message contains an explicit
description of what was changed in the code or project.
Return only 1 if a clear change description is included,
otherwise return only 0.

Pay special attention to messages starting with "Fix" or
"fixed":
- These messages must include specific details about the code
changes made, not just the problem that was fixed.
- Messages that only describe the problem or the reason for a
fix, without specifying the actual changes, should be
classified as 0.

When in doubt, interpret more strictly and return 0 (not
included) unless there is an unambiguous statement of what was
modified, added, or removed.
```

補足1-C 変更理由プロンプト全文

変更理由 使用プロンプト（system）

```
Determine if the commit message explicitly contains a reason for the change.
A change reason refers to a clear statement of the direct necessity or motivation
for making the change.

The following are not considered change reasons:
1. Description of what was changed
2. Explanation of the results or effects of the change (e.g., sentences starting
with "This allows..." or "This lets...")
3. Description of what became possible due to the change

Change reasons are often indicated by expressions such as:
- "because..."
- "since..."
- "due to..."
- "in order to..."
- "to address..."

Return 1 if the message clearly includes a change reason, and 0 if it does not.
When in doubt, interpret more strictly and return 0 (not include) unless
there is a clear and direct statement of the reason.
```

補足2 内容・理由の表現カテゴリ

262件のメッセージを手動でサンプリング・ラベル付

Summarize Code Object Change	変更対象の要約	変更対象や変更の特徴，変更前後の対比	70.6%
Describe Implementation Principle	実装原理の説明	実装における技術的な振る舞い	2.4%
Illustrate Function	機能の説明	変更によって導入される新しい振る舞い	25.8%
Missing what	変更内容の欠如	タイポ修正，名称変更，スペースの追加・削除	7.5%

Describe Issue	問題の記述	エラーシナリオの説明，欠点の説明	27.8%
Illustrate Requirement	要件の説明	使用におけるニーズ，古くなった，環境の変化	18.3%
Describe Objective	目的の記述	欠陥の修正のため，改善のため	8.7%
Imply Necessity	必要性の暗示	改善と利点，実装機能との関係	28.6%
Missing Why	変更理由の欠如	タイポ修正，リファクタリング	19.4%

補足資料3 次実験の想定概要

①プロジェクトの選定

- 対象は未定．先行研究と同様にjava？それとも他言語も？

②メッセージの収集

- 件数は？

③GPT-4oによる分類

④真予測メッセージの内容確認

- 目視．一部をサンプリング？

⑤結果の分析・考察

補足資料5 フィルタリング条件 ※[7]を参考に設定

プロジェクト選定条件

- GitHub上のJavaプロジェクト
- 人気度
  - スター数上位1000プロジェクトから
- プロジェクト特性
  - チュートリアルプロジェクトでない
  - コミット数1000件以上

フィルタリング条件一覧

- 非英語コミットメッセージを除外
- コード変更条件
  - 大規模なコード変更（1500トークン超）を除外
- メッセージ長
  - 短すぎるメッセージ（3単語未満）を除外
  - 長すぎるメッセージ（330トークン超）を除外
- ファイル種別条件（収集段階）
  - Javaファイル変更のみ
- その他（収集段階）
  - bot・自動生成

[7]Wang, Guoqing and Sun, Zeyu and Dong, Jinhao and Zhang, Yuxia and Zhu, Mingxuan and Liang, Qingyuan and Hao, Dan
“Is It Hard to Generate Holistic Commit Message?”
ACM Transactions on Software Engineering and Methodology, Volume 34, Issue 2 Article No.: 36, Pages 1 28