

未知語の特徴が 技術的負債の自動分類性能 に与える影響の分析

オープンソースソフトウェア工学研究室

学籍番号：60256288

氏名：山村慎太郎



技術的負債(Technical Debt : TD)

| ソフトウェア開発において、納期やコストなどの制約が原因で、品質を妥協した設計や実装^[1]

例：スパゲッティコード

—構造が複雑に入り組んでいるプログラム

| TDが蓄積すると、

- ・後の保守作業の効率低下
- ・放置するほどコスト増大

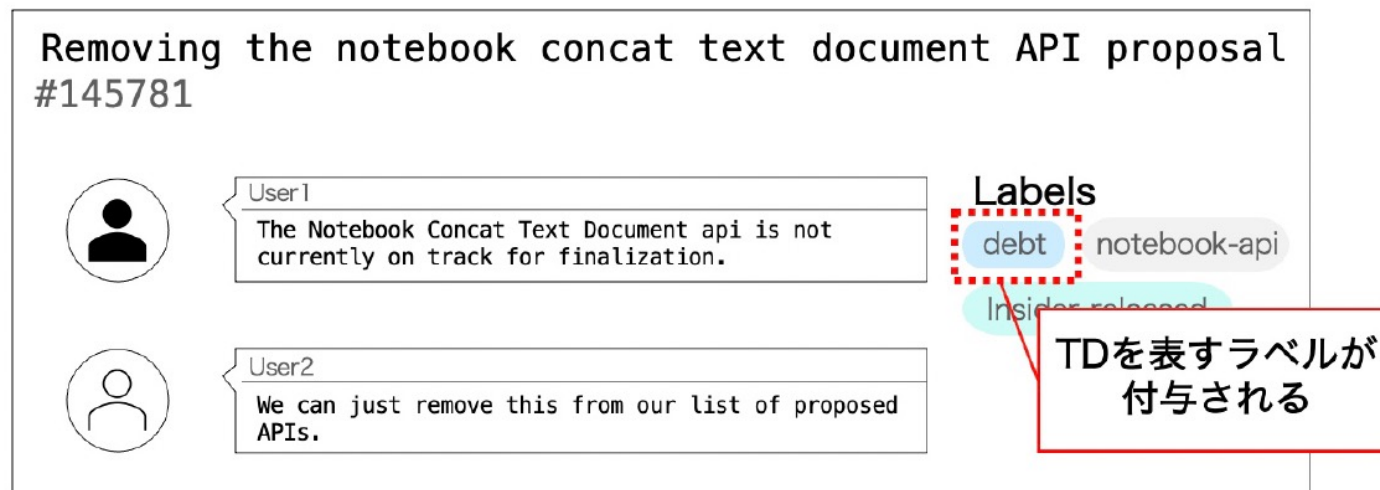
を引き起こすため、早期の除去が望ましい

⇒早期除去には、TDの把握・管理が不可欠

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

TD-Issue

GitHubにおいて管理される、**TD**に関する内容であることが明示されているIssue



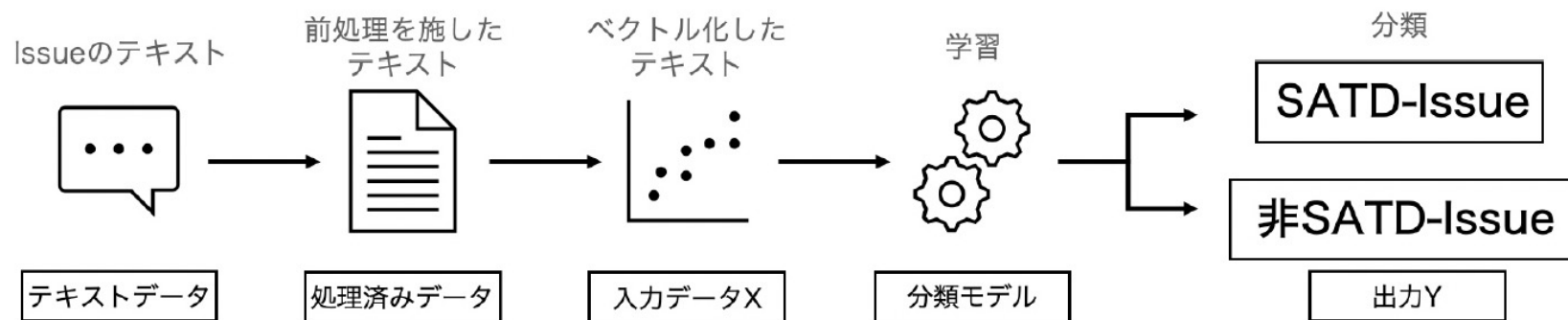
Issueが報告された段階で、そのIssueがTD-Issueであると示されない場合がある

⇒ **TD**と通常の不具合とでは対応が異なるため、報告時点で**TD**であるかどうか分類できているのが望ましい[2]

[2] 田口舞奈, 木村祐太, 大平雅雄. 技術的負債に関する課題票の単語分散表現を用いたテキスト分類. ソフトウェア・シンポジウム2023 in 仙台, No. 38, pp. 28–37

先行研究

IssueがGitHubに報告された時点で，TD-Issueであるかどうかを判別する分類モデルを構築^[2]



課題：辞書外の単語(未知語)への対応ができていない

- ・ spaCy辞書外単語の「oov」への統一置換
- ・ FastTextのサブワード化非対応

※サブワード化：
単語をさらに分割して、
意味的に学習すること

⇒ 未知語の特徴を捉えずに，分類を行っていた

[2] 田口舞奈, 木村祐太, 大平雅雄. 技術的負債に関する課題票の単語分散表現を用いたテキスト分類. ソフトウェア・シンポジウム2023 in 仙台, No. 38, pp. 28–37

目的・アプローチ

目的

未知語の特徴を捉えて分類した結果，分類性能が向上するかを検証

アプローチ

未知語についての対応

- ・「oov」への置換を行わず実験
- ・FastTextのサブワード化の実装

⇒新たな特徴を得られることで，性能は変化？

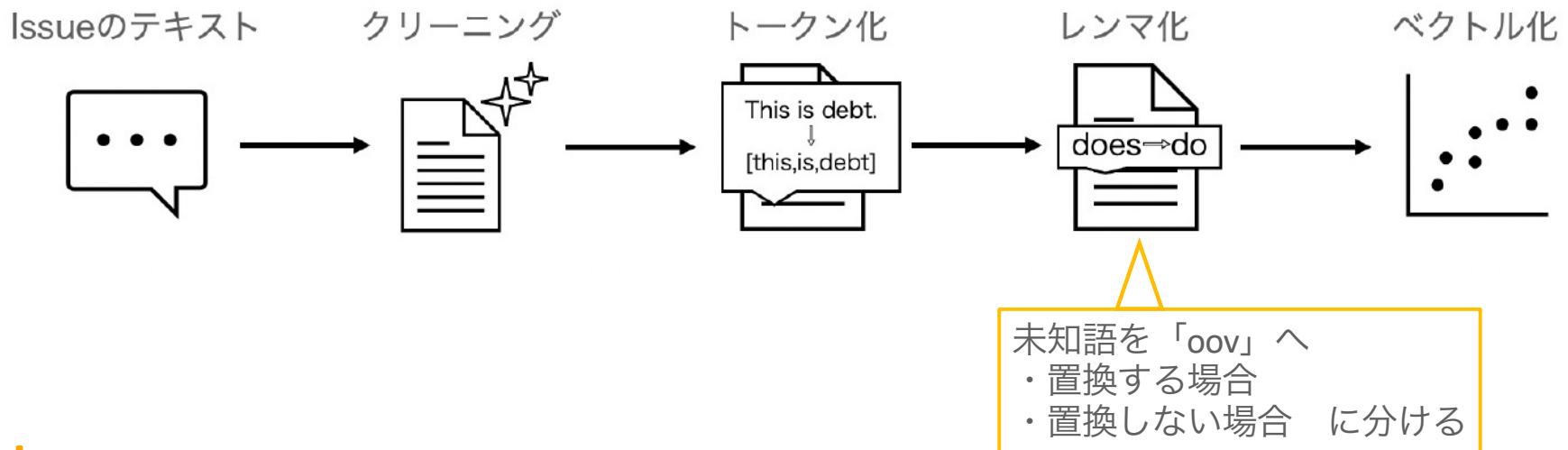
分類モデルに**ナイーブベイズ**を追加

- ・**ナイーブベイズ**：単語の出現頻度を元に分類を行う

⇒出現頻度の高い未知語の特徴を捉えることで，性能は変化？

評価実験 | 前処理・ベクトル化

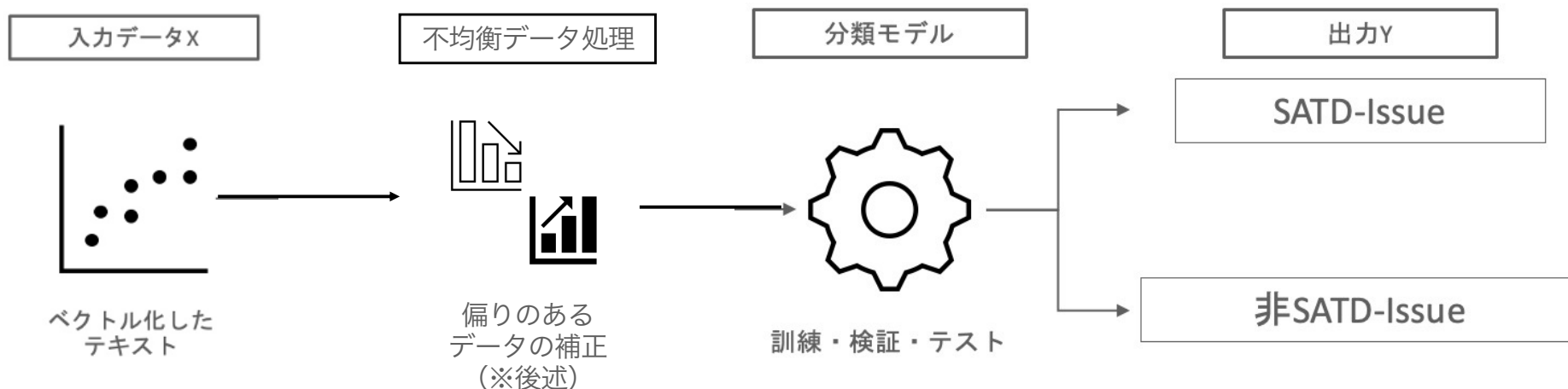
テキストデータの前処理



利用する単語のベクトル化手法

- BoW, TF*IDF, Word2Vec, **FastText**, Glove
- **FastText**については、サブワード化を行ったものを行わなかったものの両方を使用

評価実験 | 分類モデル



機械学習アルゴリズム

- ・ ランダムフォレスト
- ・ ロジスティック回帰
- ・ サポートベクターマシン
- ・ ナイーブベイズ

「oov」置換あり・なし：2種類
×
ベクトル化手法：7種類
×
機械学習アルゴリズム：4種類
||
分類モデル：56種類
分類モデルの性能を比較

データセット

- GitHub上のスター数上位5000件のうち,
- ・ TDに関するIssueをよく扱う4つのプロジェクトの内,
- ・ 2回以上報告経験のある開発者の解決済みIssue

オーナー / リポジトリ名	SATD の label	非 SATD-Issue 数	SATD-Issue 数
influxdata / influxdb	kind/tech-debt	8,514	120
saleor / saleor	technical debt	2,230	108
nextcloud / server	technical debts	7,255	83
microsoft / vscode	debt	79,829	2,044
合計	-	97,828	2,355

正例データ数と負例データ数には大きな偏りがある

—少数派の特徴が学習できず、性能が悪化することが知られている

⇒不均衡データを補正する処理を行う

不均衡データへの対処

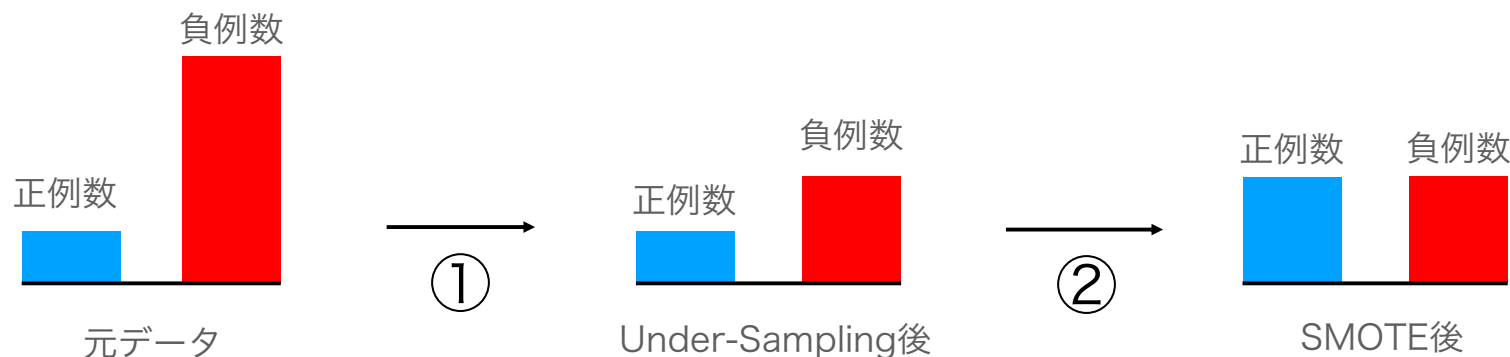
学習時，以下の2手法によりデータの偏りを補正

①アンダーサンプリング

多数派のデータを少数派のデータに合わせて削除する手法

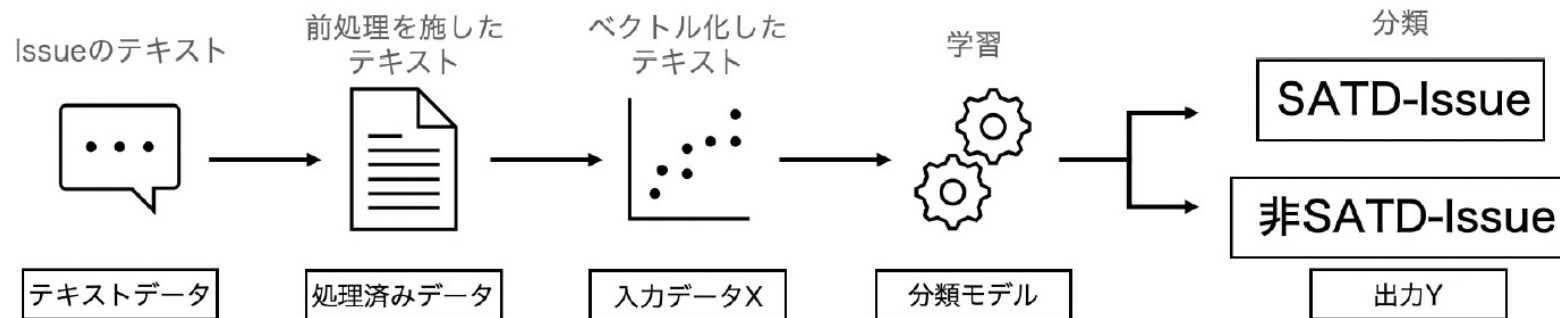
②SMOTE (Synthetic Minority Over-sampling Technique)

少数派のデータを多数派のデータに合わせて増加させる手法



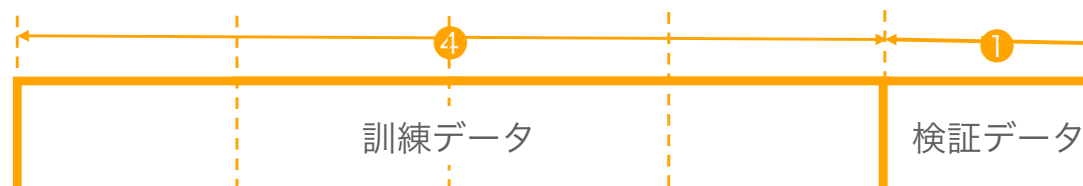
補正した結果, $\text{TD-Issue} : \text{非TD-Issue} = 1 : 1$

分類モデルの作成



データの正規化：
$$X' = \frac{X - x_{min}}{x_{max} - x_{min}}$$

層化5分割交差検証

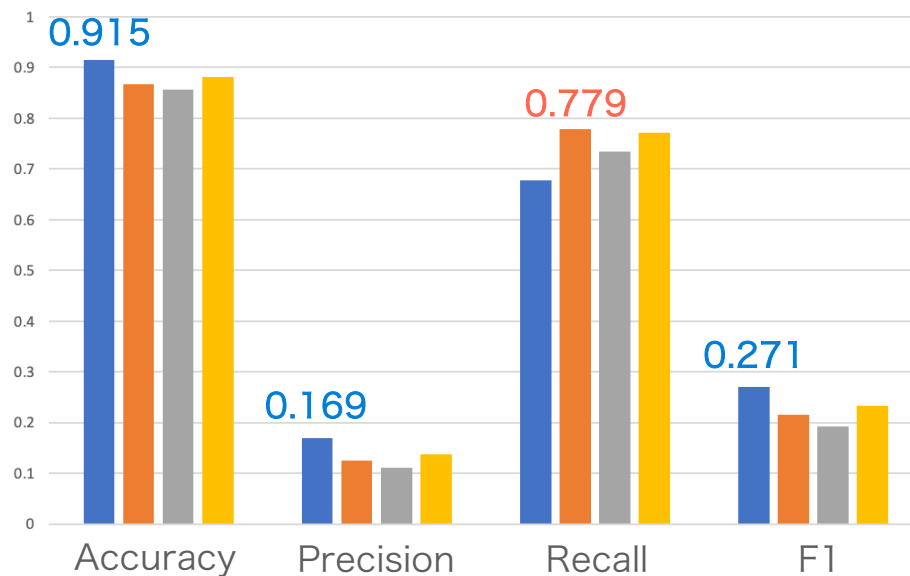


- ・ 検証データ交替5回×再分割して20回=合計100回学習

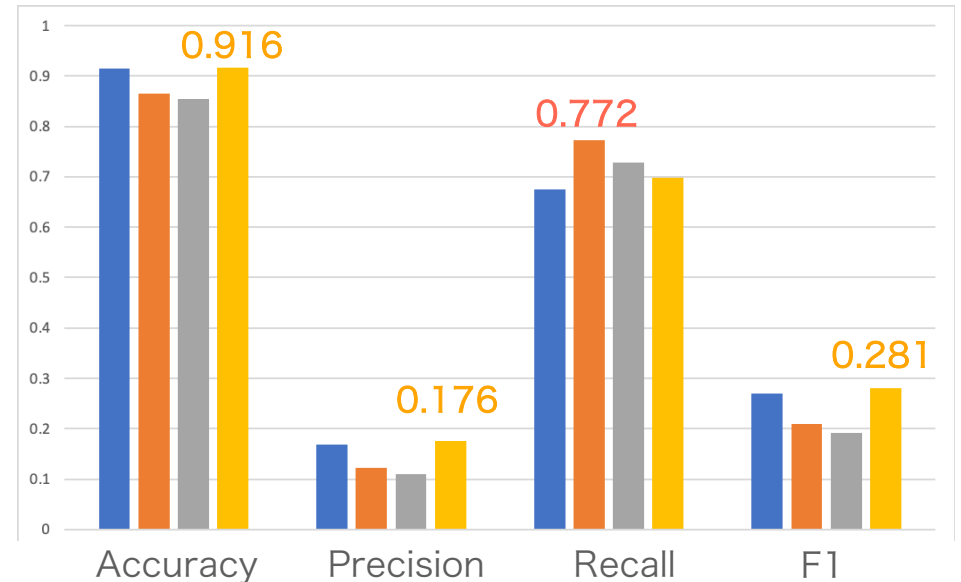
結果 | 分類性能

4種の機械学習アルゴリズムを用いたモデルのうち、F1値が最も高い組み合わせを抜粋して掲載

「oov」置換あり

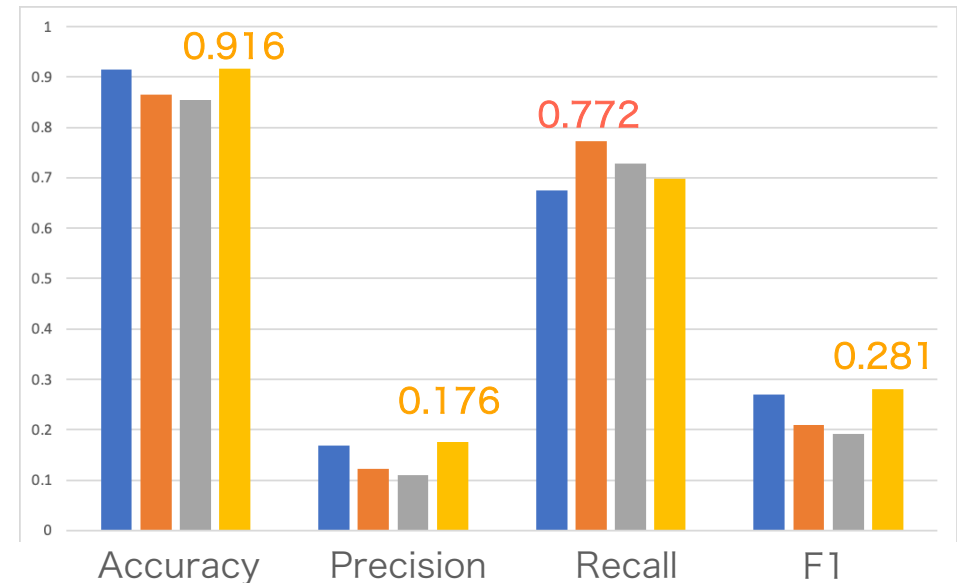
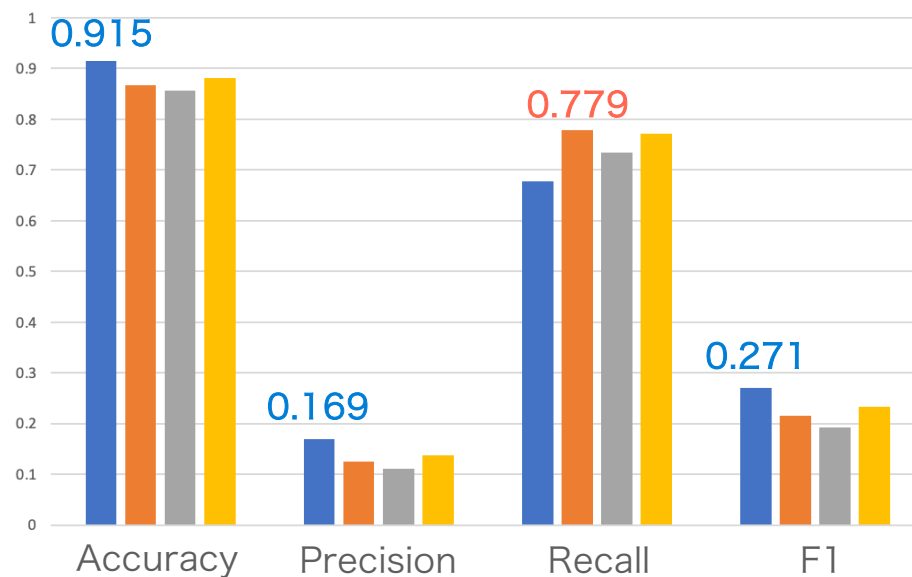


「oov」置換なし



■ ランダムフォレスト×BoW ■ ロジスティック回帰×TF*IDF ■ SVM×TF*IDF ■ ナイーブベイズ×TF*IDF

- 「oov」への置換を行わなかった際の
TF*IDF × ナイーブベイズ モデルの
Accuracy, Precision, F1が全体で最も高かった
- 一方で、未知語を特徴量に加えても、全体的に性能は大きく変化しなかった



■ ランダムフォレスト×BoW ■ ロジスティック回帰×TF*IDF ■ SVM×TF*IDF ■ ナイーブベイズ×TF*IDF

分類に寄与した単語の調査

TF*IDF×ナイーブベイズモデルの分類に最も影響を与えた単語を調査

—クラスごとの単語の出現対数確率の差（式-1）に基づいて調査
—差が大きいほど正例分類への寄与度が高い

$$P(w_i) = \log P_T(w_i) - \log P_F(w_i) \quad (\text{式-1})$$

結果（寄与度上位5単語）

未知語の「oov」への置換の有無	寄与度の高い単語				
置換あり	debt	breaker	smell	adopt	lift
置換なし	breaker	debt	<u>indexlist</u>	<u>resourcelist</u>	<u>clockface</u>

→TD分類に寄与する未知語は変数名・関数名

- ・ 未知語のうち，TD-Issue分類に最も寄与していた単語は，**変数名や関数名**
 - ・ **変数名や関数名**は，TDに関わらずIssueに出現するため，分類性能の向上には貢献しなかった
- 分類の際には，**変数名・関数名**の扱いを考慮する必要がある

未知語の「oov」への置換の有無	寄与度の高い単語				
置換あり	debt	breaker	smell	adopt	lift
置換なし	breaker	debt	<u>indexlist</u>	<u>resourcelist</u>	<u>clockface</u>

→TD分類に寄与する未知語は**変数名・関数名**

総括

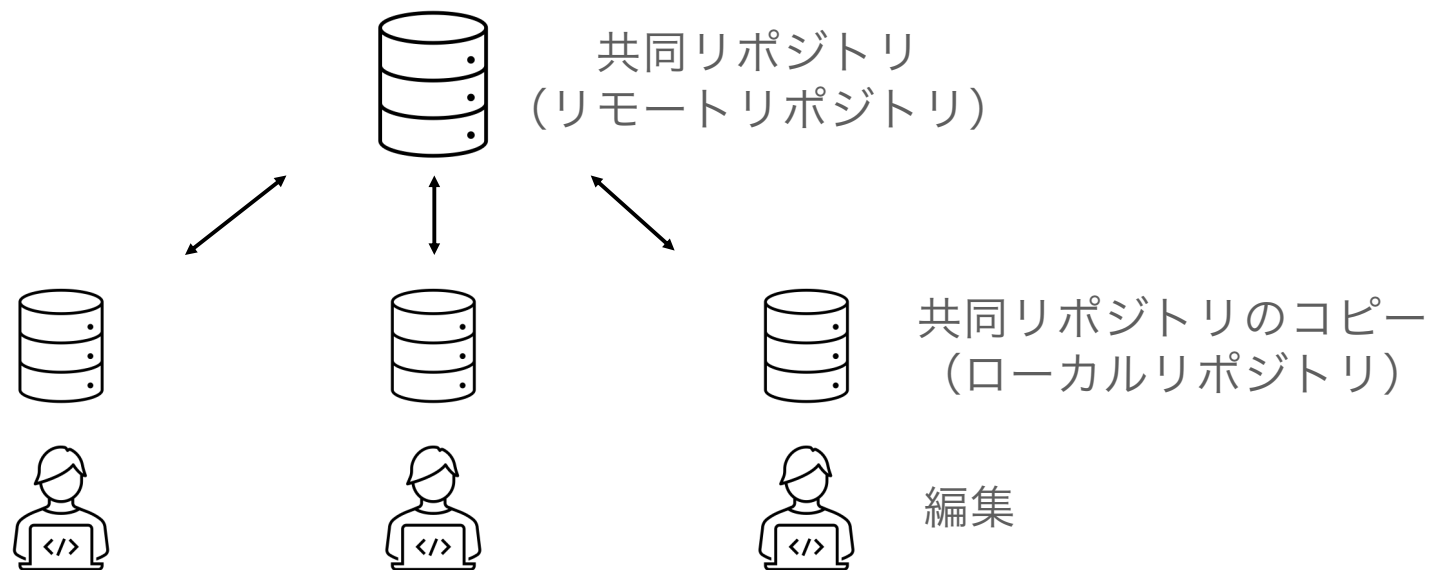
- | 先行研究では行えていなかった，未知語を特徴量として捉えたTD-Issueの分類を行った
 - | 未知語を特徴量として捉えても，分類性能に大きな変化は見られなかった
 - | TD-Issueの分類に寄与した未知語は，変数名や関数名が上位であった
- 分類の際には，変数名・関数名の扱いを考慮する必要がある

変更内容と変更理由を含む コミットメッセージの生成

オープンソースソフトウェア工学研究室
S2420142 山村慎太郎

Git

共同でのソフトウェア開発を効率的に行うための、分散型バージョン管理システム



チーム開発におけるソースコード管理のため、ソフトウェア開発の現場で広く利用されている^[3]

[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). Association for Computing Machinery, New York, NY, USA, 2389–2401.

コミット

Gitにおいて、ファイル・ソースコードの**変更**を、ローカルリポジトリに**記録**すること

手順

1. `git add ファイル名` でファイルの変更をステージングエリアに追加
2. `git commit -m “変更内容の説明”` で変更をコミット

コミットで変更を記録すると、

- ・ 変更履歴の確認
- ・ 過去のコミットの状態に戻す(ロールバック)

などが可能となる

コミットの際、通常、**”変更に関する説明文”**を付与する

コミットメッセージ

コミットを行う際に付与する変更に関する説明文

例^[4]

コード差分	コミットメッセージ
<pre>- import org.junit.internal.runners.JUnit4ClassRunner; + import org.junit.internal.runners.BlockJUnit4ClassRunner;</pre>	Brought in old JUnit4ClassRunner for compatibility. Renamed new JUnit4ClassRunner to BlockJUnit4ClassRunner.

コミットメッセージを書くことで、

- ・ 変更履歴の可読性の向上
- ・ 長期的な保守性の向上

をもたらし、チーム開発の効率化に貢献^[3]

[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). Association for Computing Machinery, New York, NY, USA, 2389–2401.

[4] <https://github.com/junit-team/junit4/commit/078e45bd31a9920e00633ad6ce658e22529304f2>

コミットメッセージの現状

コミットメッセージの記述には、**時間や労力**がかかる^[5]

原因：コミット内容が複雑、簡潔に要約するのが難しい など

その結果、

- ・メッセージ内容が完全に空白
- ・メッセージ内容が数語しか書かれてない（例^[6]）

コード差分	コミットメッセージ
<pre>- if (result.getFailureCount() == 0) - return "PASSED!";</pre>	Remove

のような、**品質の低いメッセージ**が書かれる場合がある

そこで、コミットメッセージの自動生成の研究が行われる

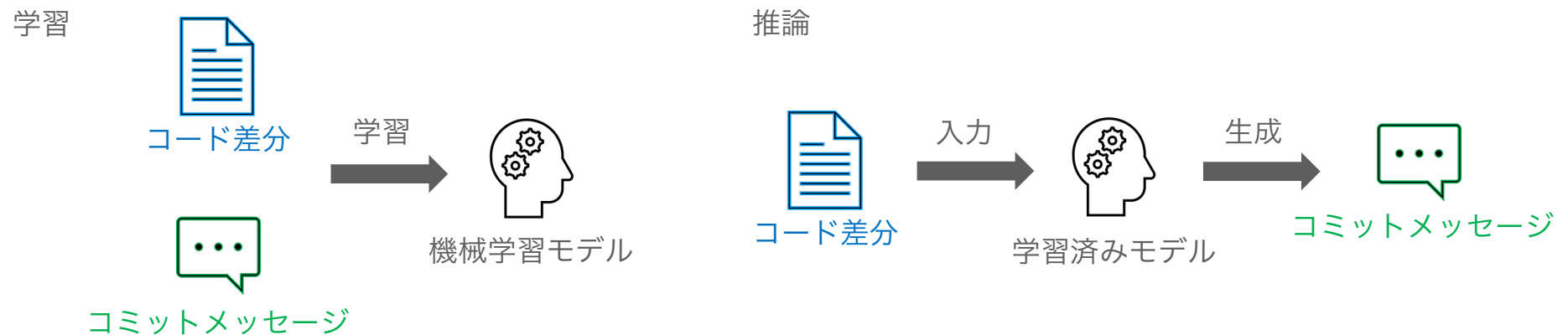
[5] L. Wang et al., "Delving into Commit-Issue Correlation to Enhance Commit Message Generation Models," 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), Luxembourg, Luxembourg, 2023, pp. 710-722

[6] <https://github.com/junit-team/junit4/commit/b52e4af4eb7b26619e1d80537952aed7276766cb>

コミットメッセージの自動生成

コミットメッセージを自動で生成することで、メッセージを書く手間が省ける

例：学習ベース



課題：内容が不十分なメッセージが生成されてしまう

原因：データセットについて、

- ・OSS projectの44%のメッセージは変更内容や変更理由を含まない^[3]
- ・メッセージの内容を考慮・区別して学習していない

[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). Association for Computing Machinery, New York, NY, USA, 2389–2401.

コミットメッセージ自動生成の課題

変更内容・理由両方を含むメッセージの生成が困難^[5]

含まない例：メッセージが変更内容のみ

コード差分	コミットメッセージ
- String expectation = ... + String expectation = ...	Converted tab to spaces

変更内容・理由が重要な理由^[5]

- ・ 共同開発者が変更内容を理解するため
- ・ 変更を行なった開発者がプロジェクトを去った後,
その変更の意図を知る唯一の情報源となりうるため

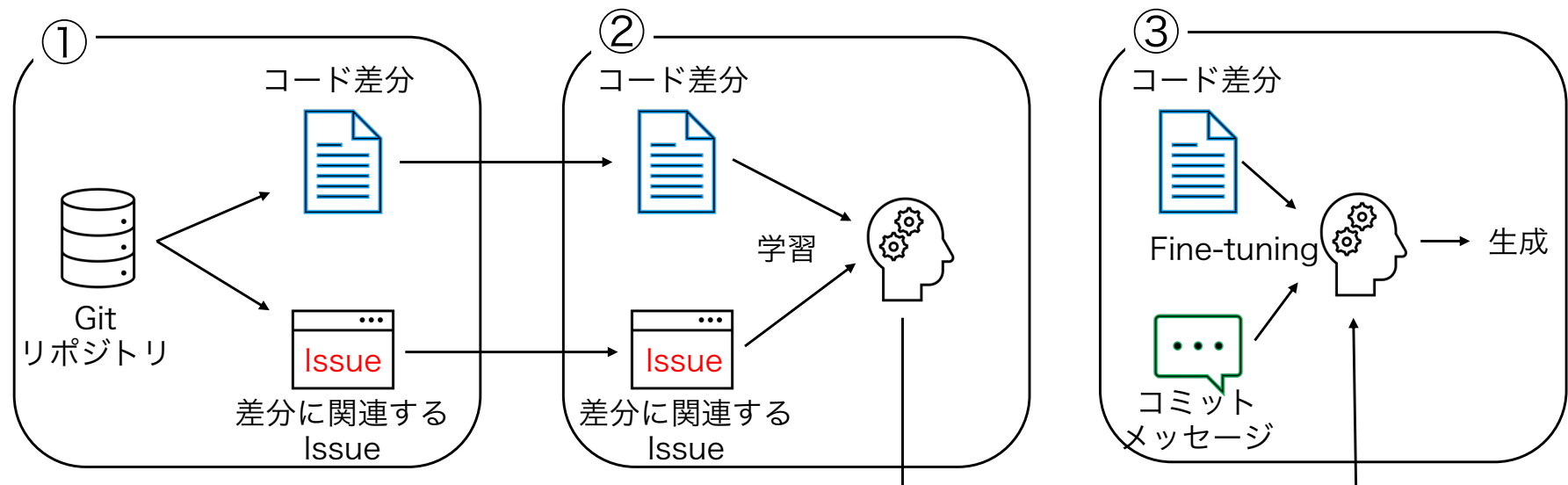
そこで変更内容・理由を生成する研究が行われる

[5] L. Wang et al., "Delving into Commit-Issue Correlation to Enhance Commit Message Generation Models," 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), Luxembourg, Luxembourg, 2023, pp. 710-722

[7] <https://github.com/junit-team/junit4/commit/2246c5a6de14ae94f3d29a82b165653e67ca93a1>

先行研究

| **Issueの情報**を用いて，変更内容・理由を含むコミットメッセージを生成するモデルを構築^[5]



| Issueには，バグ報告，機能の提案，必要な対処などが記載される

→ **コード変更に関する情報**をIssueから抽出して学習に利用

[5] L. Wang et al., "Delving into Commit-Issue Correlation to Enhance Commit Message Generation Models," 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), Luxembourg, Luxembourg, 2023, pp. 710-722

先行研究の課題点

| **Issue**を用いることによる制約が存在

- ・ 全てのコミットが、関連するIssueを持っているとは限らない
→ 実用性・汎用性に制限
- ・ Issueデータの品質や一貫性のばらつき
→ モデルの学習に影響を与える可能性

| 本研究では、**差分・メッセージ以外の情報**を用いず、
変更内容・理由を含むコミットメッセージの生成を目指す

本研究の目的・アプローチ

目的

変更内容・理由の両方を含むコミットメッセージの自動生成

アプローチ

- ・データを以下のように区別して学習
 - ①変更内容・理由を含むコミットメッセージ
 - ②変更内容・理由を含まないコミットメッセージ
 - ①のみを学習データとして用いて学習
- ・変更内容・理由を含むメッセージを多く含む学習データセットの作成
 - ①のみでは，学習データ不足が予想される
 - そこで，学習データを増加させる方向性を考える

事前調査(未達)

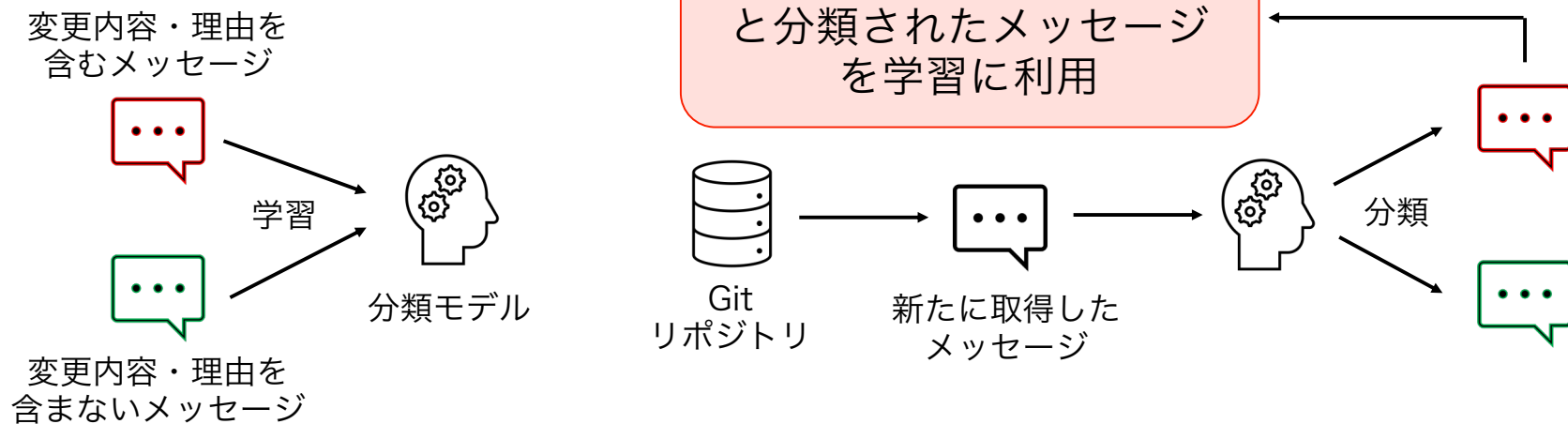
①変更内容・理由を含むコミットメッセージのみを学習データとして用いた際の実験結果(未)

書く内容：

- ・ 先行研究_[3]で手動分類された約950件の、変更内容・理由を含むデータを使用
- ・ 内容を区別しない数万件のデータでの学習時との比較
- ・ データ不足により、生成が困難(予定)

データ不足への対処案

変更内容・理由を含むメッセージかそうでないかを判別する分類器を構築



分類器の学習には, [3]で得た変更内容・理由を含むデータを使用

最終的に, 変更内容・理由を含むメッセージを明らかに多く含むデータセットを得て, 学習に使用

今後の予定

- | データ不足への対処に向けた、メッセージ内容を判別する分類器の構築

- | 分類器により得たデータセットによる、変更内容・変更理由を含むコミットメッセージの生成

- | その際生まれる新たな課題への対処