

技術的負債に関する課題票 のナイーブベイズを用いた テキスト分類(仮)

発表：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、設計や実装の質を妥協してしまうこと

- ・スパゲッティコード
- ・コーディング規約に反するソースコード など

| 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

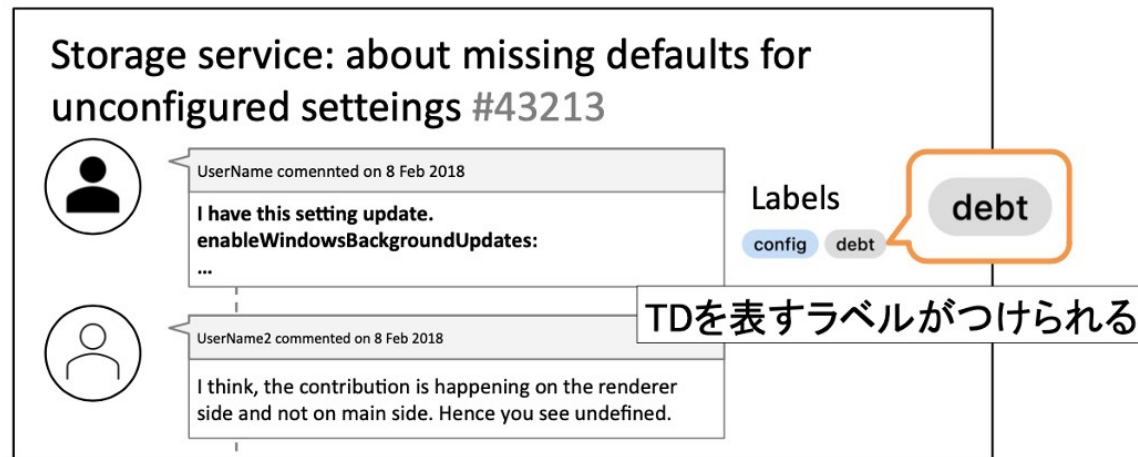
を引き起こすため、早期の除去が望ましい

⇒早期除去には、TDの管理が不可欠

SATD-Issue (Self-Admitted TD)

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**

※図の差し替え間に合いませんでした



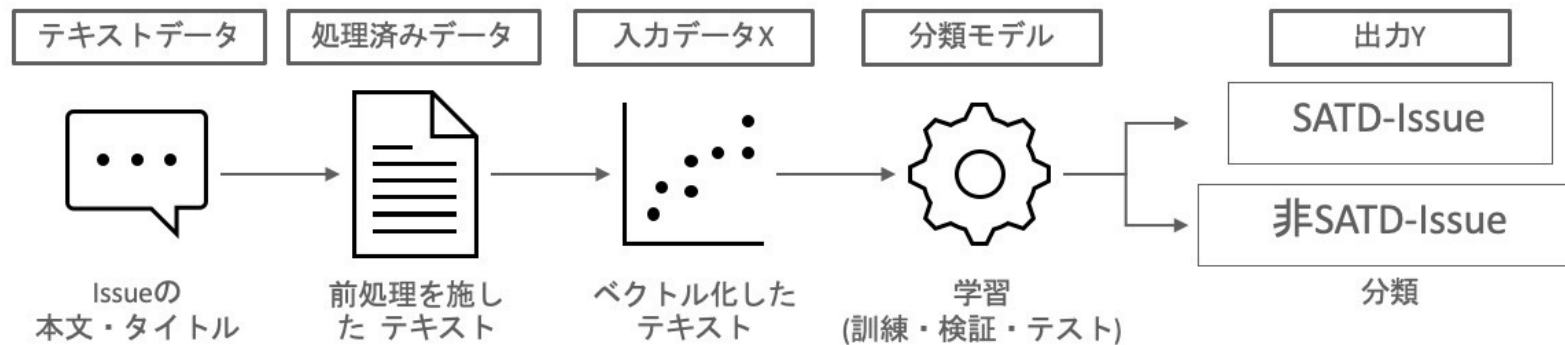
Issueが報告された段階で、その**Issue**がSATD-Issueであると示されない場合がある

—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であるかどうか区別できているのが望ましい

先行研究

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

※以下同じ



| ベクトル化手法として BoW, TF*IDF, Word2Vec, FastText, Glove

| 分類には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を使用

先行研究における課題点

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

精度に基づいた分類モデルの実用性

先行研究で作成された2種のモデルはそれぞれ，

- ・ Precisionは高いが， Recallが低いモデル
- ・ Recallは高いが， Precisionが低いモデル

⇒ モデルを使い分ける必要があり，実用性に欠ける

目的・アプローチ

目的

TDに関するIssueの分類において，先行研究の課題点を改善した結果，分類精度が向上するかを調査

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，精度はどう変化？

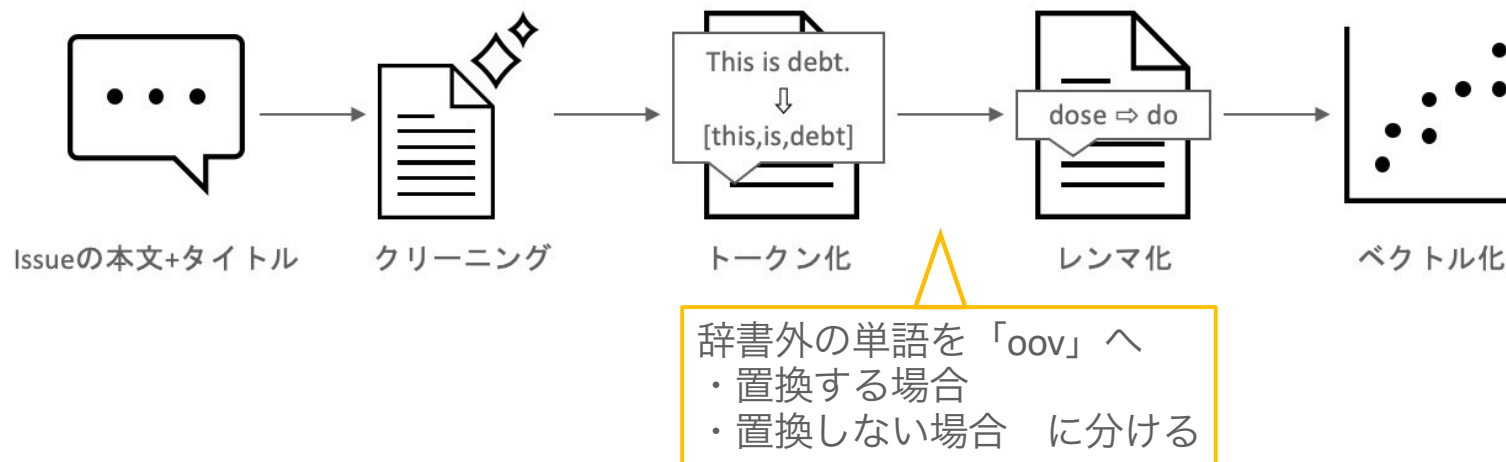
分類モデルにナイーブベイズを追加

- ・**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，精度は変化？

評価実験 | 前処理・ベクトル化

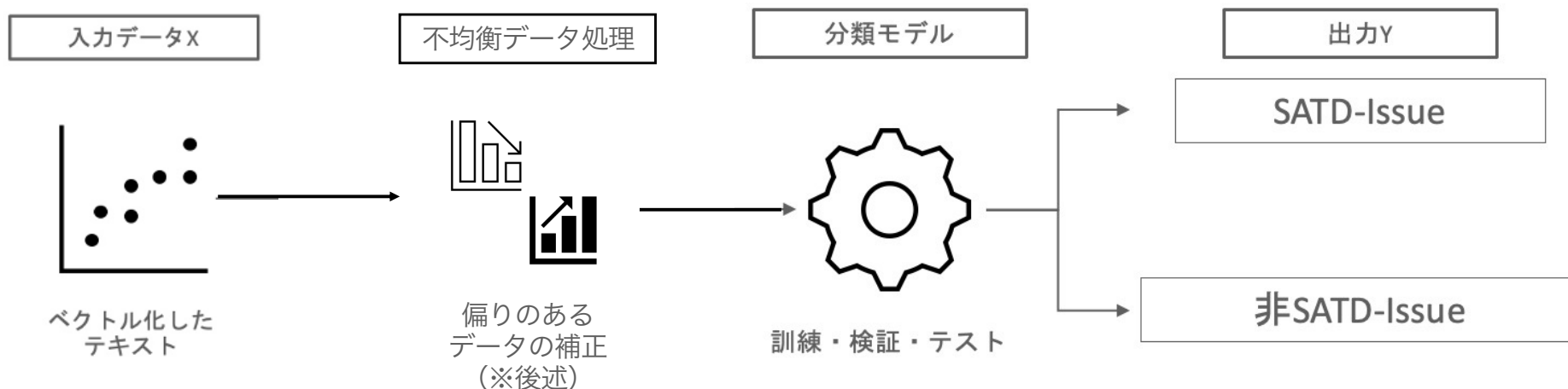
テキストデータの前処理



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で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

どの分類モデルの性能が良いかを調査

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
 - SATD-Issueの特徴を学習できず, 著しく精度が悪化する
- ⇒不均衡データを補正する処理を行う

評価実験 | 不均衡データへの対処

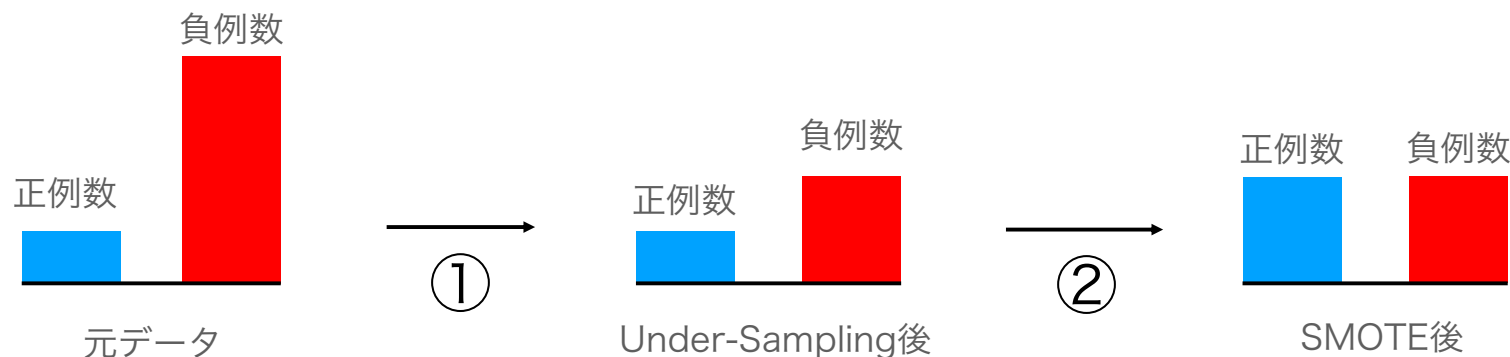
学習時, 以下の2手法によりデータの偏りを補正

①Under-sampling

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

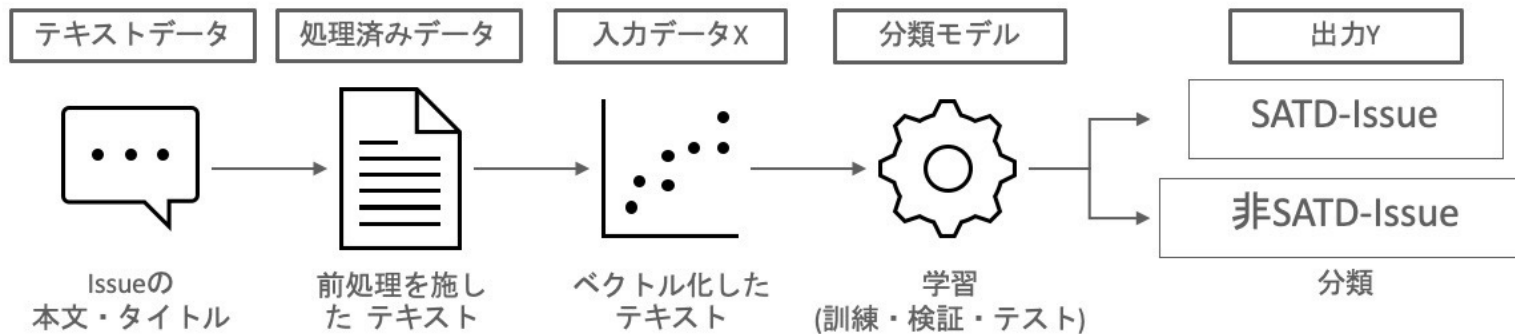
少数派のデータを多数派のデータに合わせて増加させる手法



補正後, SATD-Issue : 2355件→**2994件**
非SATD-Issue : 97828件→**2994件**

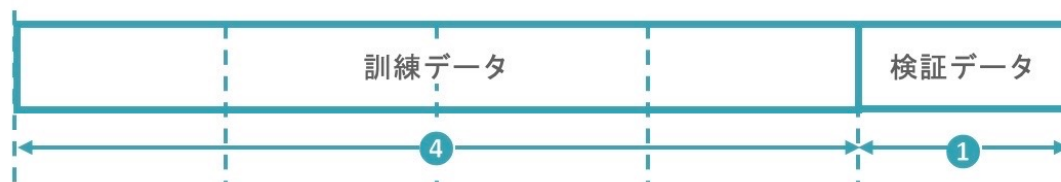
正例 : 負例 = 1 : 1
の学習データセット
を作成

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 | 一部抜粋

- 「oov」置換を行った, ランダムフォレストモデル

機械学習モデル	ベクトル化手法	Accuracy	Precision	Recall	F1
ランダムフォレスト	BoW	0.915	0.169	0.678	0.271
	TF*IDF	0.888	0.142	0.748	0.238
	SO_W2V	0.844	0.097	0.685	0.170
	Wiki_W2V	0.860	0.100	0.623	0.173
	FastText	0.850	0.094	0.623	0.163
	Sub_FastText	0.778	0.062	0.596	0.112
	Glove	0.852	0.095	0.624	0.165

- 「oov」置換を行わなかった, ナイーブベイズモデル

機械学習モデル	ベクトル化手法	Accuracy	Precision	Recall	F1
ナイーブベイズ	BoW	0.888	0.119	0.595	0.199
	TF*IDF	0.916	0.176	0.698	0.281
	SO_W2V	0.606	0.047	0.825	0.089
	Wiki_W2V	0.731	0.059	0.707	0.109
	FastText	0.709	0.055	0.709	0.102
	Sub_FastText	0.490	0.038	0.861	0.073
	Glove	0.716	0.056	0.709	0.104

結果 | 一部抜粋

- 「oov」置換を行った, ランダムフォレストモデル

- 「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

- サブワード化を行って未知語への対応を行った**FastText**モデルは, 比較的 Accuracyが低く, Recallが高い傾向にあった

ナイーブベイズ	OO-W2V	0.666	0.047	0.829	0.059
	Wiki_W2V	0.731	0.059	0.707	0.109
	FastText	0.709	0.055	0.709	0.102
	Sub_FastText	0.490	0.038	0.861	0.073
	Glove	0.716	0.056	0.709	0.104

考察に書こうと思っていること

分類に寄与した単語(単語の出現確率)の調査

「oov」の中身は変数名・関数名がみられたことに触れる

”oovなし×NB×TF*IDF”がよかった要因について

SATDに頻出する変数名・関数名を拾ったため？

(分類に寄与した上位単語の出現Issue割合の算出, など追加実験)

FastTextモデルのAccuracyが低くRecallが高い要因

FastTextは似たような意味ベクトルの単語を含むissueをまとめて正例予測するため？

(BoWでは負例, FastTextでは正例に分類しているIssueを見てみる, など追加実験)