

Can I Solve It? Identifying APIs Required to Complete OSS Tasks

Fabio Santos, Igor Wiese, Bianca Trinkenreich,
Igor Steinmacher, Anita Sarma, Marco A. Gerosa

花田

背景

- オープンソースソフトウェアのプロジェクトでは,投稿者がタスクを選択しやすいように,課題票にラベルを付ける.
- しかし,手作業で課題にラベルを付けるのは時間がかかり,ミスも起こりがち
- ラベルを作成するための現在の自動化されたアプローチは,ほとんどがバグ/非バグとして課題を分類すること限定.

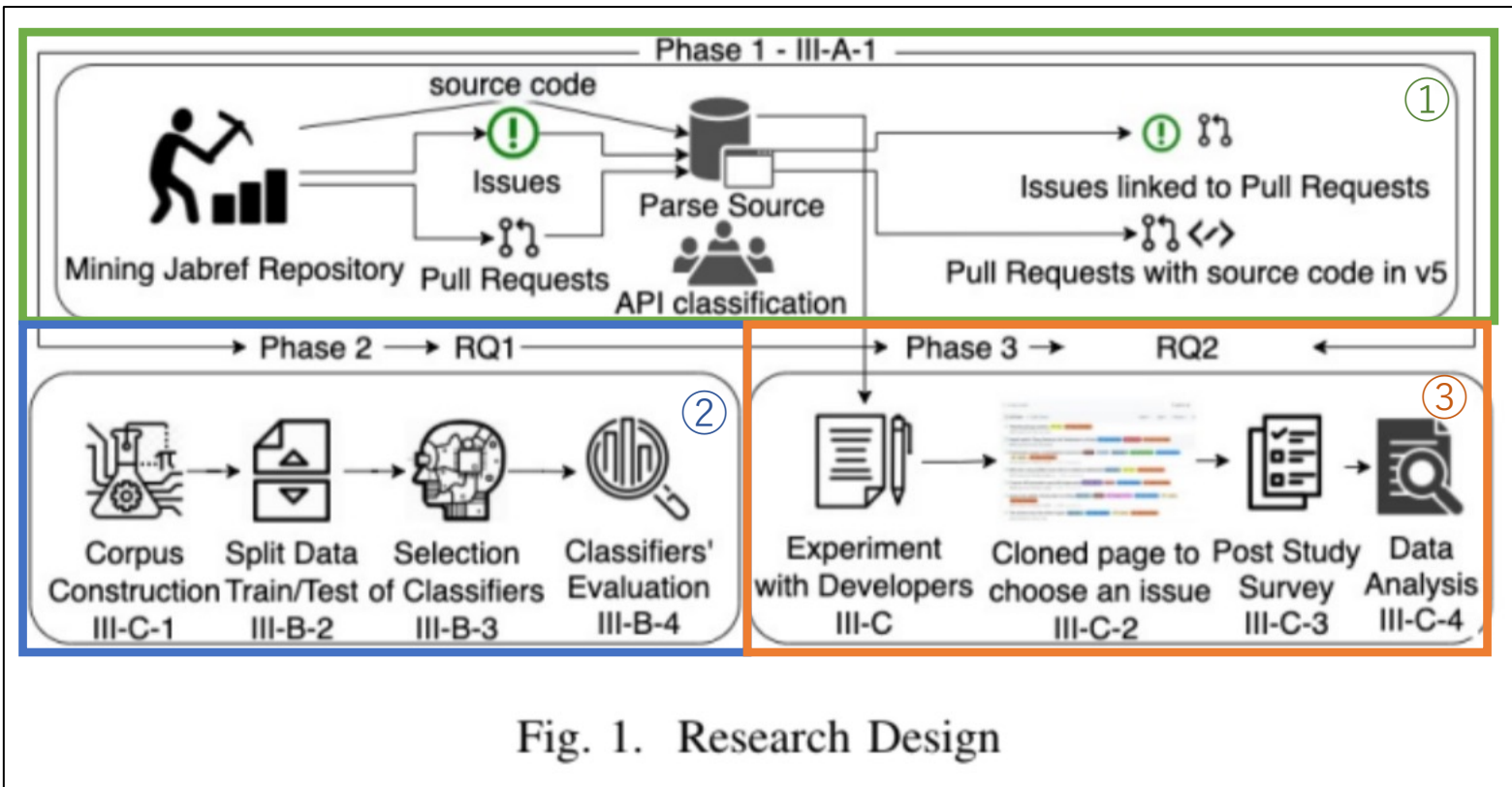
目的

- 本論文では,タスクを完了するために必要なAPIのドメインで課題をラベリングすることの実現可能性と関連性を調査.
- 課題の説明文とプロジェクトの履歴を活用して予測モデルを構築

RQ

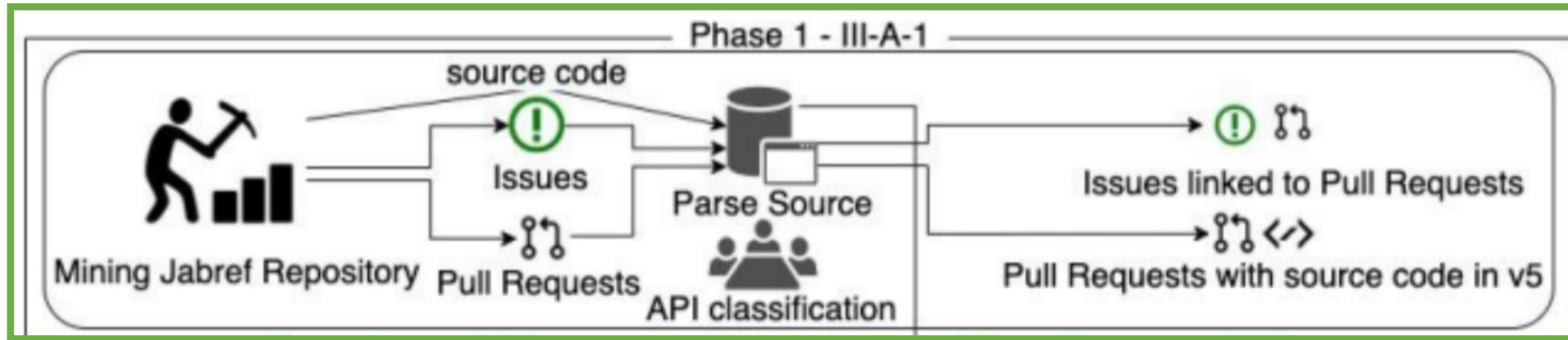
- RQ1:ソフトウェアの問題を修正するコードで 사용되는APIのドメインはどの程度予測可能か?
- RQ2.API-domain ラベルは新しい貢献者にとってどの程度適切か?

研究デザイン



- ①ソフトウェアリポジトリのマイニング
- ②分類器の構築
- ③開発者によるAPIドメインラベルの評価

研究デザイン phase1 マイニング



JabRefのデータ収集(GitHubAPIを使用)

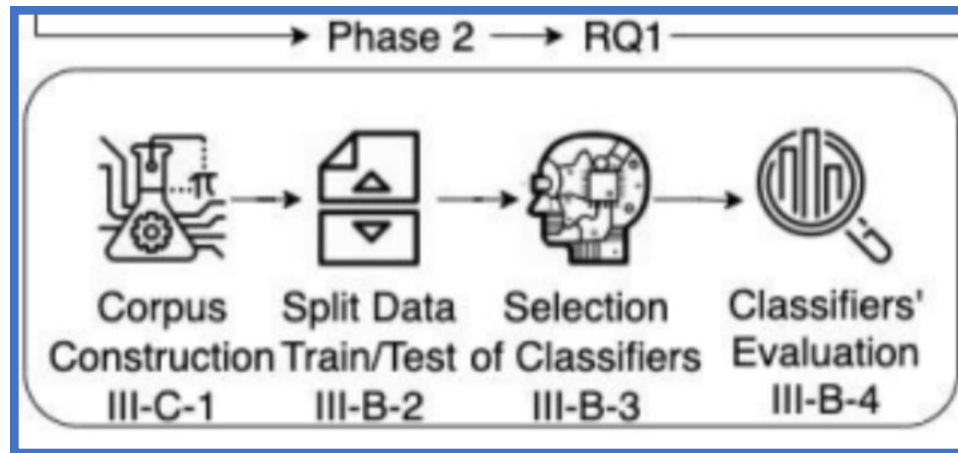
- 1976件(Issues,Pull requestsのタイトル,本文,コメント,投稿日)
- Pull requestsで変更されたファイル名と,各コミットに関連するコミットメッセージも収集

データを収集した後,未解決の課題と課題に明示的にリンクしていないPull requestsをフィルタリングして除外

最後に,インポートされたAPIを,APIのドメインに基づいて上位のカテゴリに手動で分類するカードソーティングのアプローチを採用.

Javaソースのimport文からのAPIを13種類に分類(13のラベル),705件の課題とPull requestsのデータセット

研究デザイン phase2 分類器の作成



コーパスの構築分類モデルを作成- 1

- 問題のタイトル,本文,コメントからコーパスを作成
- 各単語を小文字に変換し,URL,ソースコード,数字,句読点を削除
- Pythonのnltkパッケージを使用し,ストップワードを削除し,単語のステミング
- 課題およびプルリクエストをフィルタリング
- TF-IDFを適用し,各課題のベクトル化
- TF-IDFの計算に用いた単語の数に,データセットに含まれる13個のラベルを加える.
(各ラベルは対応するAPIドメインが課題に存在するかどうかを示す2値)

交差検証- 2

- ShuffleSplit法[40]を採用
705の課題を80%(564)の訓練セットと残りの20%(142)のテストセットにランダムに分割
- オーバーフィッティングを避けるため,10回交差検証を実行
- データセットのバランスを改善するため,マルチラベルアプローチのSMOTEアルゴリズム[41]を実行

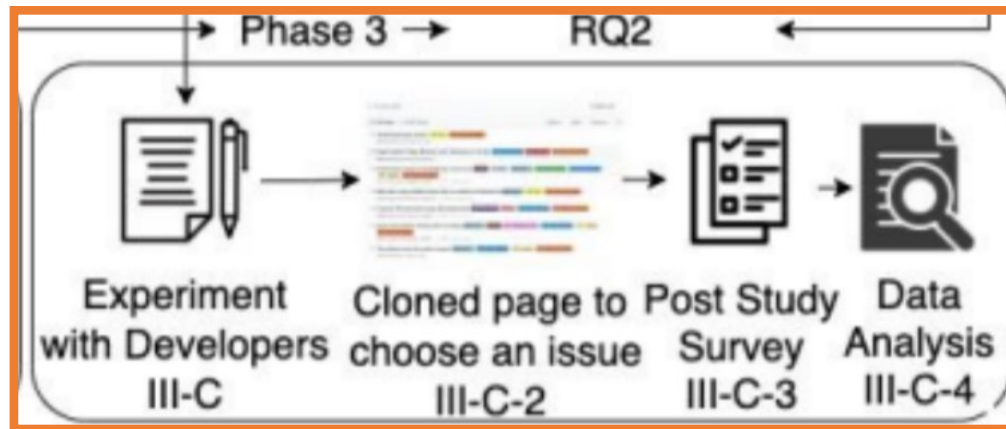
分類モデル作成-3

- マルチラベルアプローチで動作し,学習モデルを作成するために異なる5つの分類器を選択
 - 決定木
 - ランダムフォレスト
 - MPLC分器
 - MLkNN[40, 42]
 - ロジスティック回帰
- Python sklearn パッケージを用いて分類器を実行し,いくつかのパラメータをテスト

分類器モデル評価-4

- Mann-Whitney U 検定を用いて,分類器の評価指標を比較し,その後,Cliffのデルタ効果量検定を用いて,分類器の評価指標を比較

研究デザイン phase3 ラベル評価



デベロッパーによるAPI-DomainLabelsの評価

新規投稿者の視点からAPIドメインラベルの関連性を評価するため、74人の参加者を対象に実験研究を実施
JabRefの課題ページにラベルを貼ったものと貼らないものを作成し、参加者を2つのグループに分けて実験。
参加者には、投稿したい課題を3つ選んでランク付けしてもらい、その決定を支えた情報は何かというフォローアップ調査に回答

投稿したい課題

- ・ 課題ページで、どのような情報（地域）が参考になったか
- ・ 選択した情報は、なぜ関連性があるのか
- ・ 3つの課題を選択する際に、関連性があると考えたラベルの選択

API-domain ラベルの関連性についての認識を理解するため、

- 2グループの結果を比較
- バイオリンプロットを用いて分布を視覚的に比較し、効果量を測定
- クリフのデルタ検定で効果量を測定

RQ1

ソフトウェアの問題を修正するコードで使用するAPIのドメインはどの程度予測可能か？

- 各問題で変更されたファイルに含まれるAPIドメインを予測するためにベースラインをRandom Forestを使用し,作成
- コーパス構成の選択肢を評価し,入力情報を変化TITLEとBODY,TITLEとBODYとCOMMENTSの3つをもとに,各モデルを比較するためRandom Forestの構成を選択し,Mann-Whitney U 検定
- 表IIからわかるように,異なる入力をテストし,TITLEのみと比較した場合のみと比較した場合,TITLE+BODY+COMMENTS は, 精度, 再現率, F値において, 他のすべての手法よりも優れていた

TABLE II OVERALL METRICS (SECTION III-B-4) FROM MODELS CREATED TO EVALUATE THE CORPUS. HLA - HAMMING LOSS				
Model	Precision	Recall	F-measure	Hla
Title (T)	0.717	0.701	0.709	0.161
Body (B)	0.752	0.742	0.747	0.143
T, B	0.751	0.738	0.744	0.145
T, B, Comments	0.755	0.747	0.751	0.142

RQ1

ソフトウェアの問題を修正するコードで使用するAPIのドメインはどの程度予測可能か?

APIドメインラベルを予測することが可能であり,精度0.755,再現率0.747,Fラベルを精度 0.755,再現率 0.747,F-measure 0.751,ハミングロス 0.142 で予測することができる.
0.751,ハミングロス0.142を達成した.アルゴリズム,コーパスとしてTITLE, BODY, COMMENTS,そしてユニグラムを用いた.

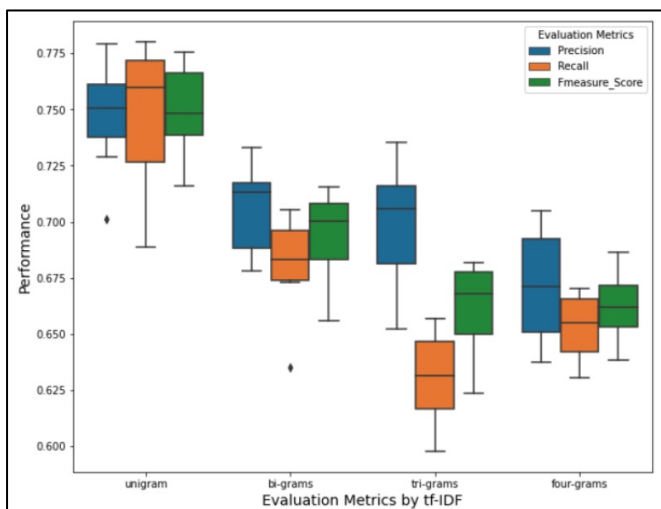


Fig. 4. Comparison between the unigram model and n-grams models

TABLE III
CLIFF'S DELTA FOR F-MEASURE AND PRECISION: COMPARISON OF CORPUS MODELS ALTERNATIVES - SECTION III-B-1. TITLE(T), BODY(B) AND COMMENTS (C).

Corpus Comparison	Cliff's delta			
	F-measure		Precision	
T versus B	-0.86	large***	-0.92	large***
T versus T+B	-0.8	large**	-0.88	large***
T versus T+B+C	-0.88	large**	-0.88	large***
B versus T+B	0.04	negligible	0.04	negligible
B versus T+B+C	-0.24	small	-0.12	negligible
T+B versus T+B+C	-0.3	small	-0.08	negligible

* $p \leq 0.05$; ** $p \leq 0.01$; *** $p \leq 0.001$

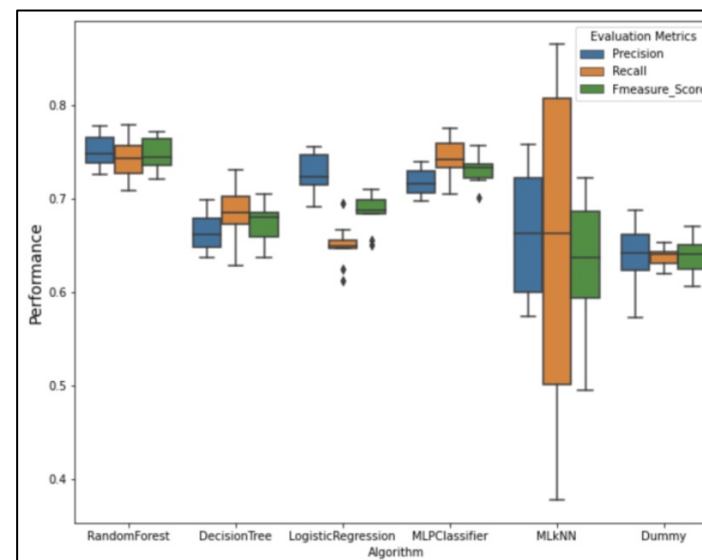


Fig. 5. Comparison between the baseline model and other machine learning algorithms

RQ2

RQ2.API-domainラベルは新しい貢献者にとってどの程度適切か？

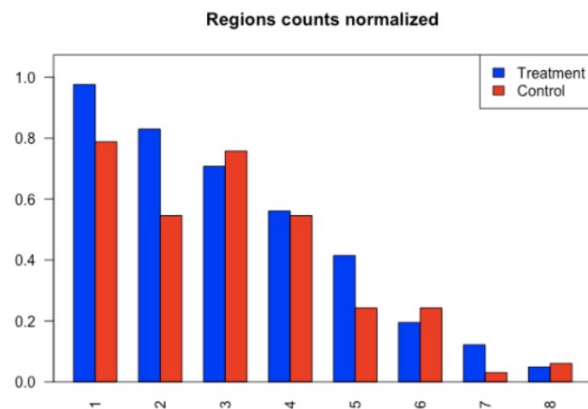


Fig. 6. The regions counts (normalized) of the issue's information page selected as most relevant by participants from Treatment and Control groups. 1-Title,2-Label,3-Body,4-Code,5-Comments,6-Author,7-Linked issues,8-Participants.

参加者が有用と考える領域

2つのグループの一つ (control)

「本文」 (75.7%) と「タイトル」 (78.7%) が上位を占め,次いで「ラベル」 (54.5%),「コード」 (54.5%)

一方のグループ (treatment)

「タイトル」 (97.5%),「ラベル」 (82.9%),「ボディ」 (70.7%),「コード」 (56.1%)

- プロジェクトで作成されたラベルはほとんど役に立たず,参加者はコードも見直す必要があったことを示唆
- タイトルとラベルの組み合わせが,Issueを総合的に判断できるようにしている

RQ2

RQ2.API-domainラベルは新しい貢献者にとってどの程度適切か？

TABLE VI
LABEL DISTRIBUTIONS AMONG THE CONTROL AND TREATMENT GROUPS

Type of Label	Control	C %	Treatment	T %
Issue Type	145	56.4	168	36.8
Components	112	43.6	94	20.6
API Domain	-	-	195	42.7

- Issue タイプ（プロジェクトに既に存在）タスクのタイプに関する情報：バグ,拡張,機能
- Components（プロジェクトに存在）JabRefの特定のコードコンポーネントに関する情報
- API-domain（新しいラベル）：分類器によって生成されたラベル（13種類のラベル）
※Treatmentグループのみが利用可能であった。

2 グループを比較するため

➤ Controlグループでは,
Issue タイプとComponents

➤ Treatmentグループでは,
API-domainを含めた3種類

どのラベルの関連性があるのかを調査

今回の結果から,ラベルは課題の選択において重要であることが示唆.
API-domainラベルは,関連性の認知を高めることができる.
API-domainラベルは、業界や経験豊富なコーダーにとって特別な関連性がある.

結論

- 本論文では,APIドメインラベルをどの程度予測できるかを検討
→jabRefプロジェクトの705の課題からデータを抽出し12個のAPI-domainラベルを予測.
- ランダムフォレストアルゴリズム,ユニグラム,課題本文のデータを用いて作成したモデルが最も良い結果を示した.
- APIドメインラベルが投稿者の役に立つかどうかを調査するため を提示する実験を行った.

その結果,業界の実務家や経験豊富なコーダーは APIドメインラベルを選択する頻度が,学生や初心者のコーダーよりも高いことがわかった.

参加者はプロジェクトで既に使用されていたコードコンポーネントラベルよりも,APIドメインラベルの方を好んだ.

- この研究は,新しい貢献者が各タスクとAPIスキルをマッチングさせ,より良く貢献するための一歩となる.

所感

- やっていることの理解が難しかった
- 検定や分散を比較するために使っていたものをそもそも知んなかった