論　文

フォーラムを教師データとしたアプリレビュー分類モデル

市川　直人[†a)]　　松本　真佑[†b)]　　楠本　真二[†c)]

Classification Model for Application Review Training with Forums

Naoto ICHIKAWA[†a)], Shinsuke MATSUMOTO[†b)], and Shinji KUSUMOTO[†c)]

あらまし　ソフトウェア開発において，アプリケーションレビューは開発者にとって豊富な情報源である．しかし，レビューは膨大な数が投稿されており，無意味なレビューも多い．これらの情報を開発者が効率良く取得するために，自然言語処理と機械学習を用いてレビューを自動分類する手法が数多く報告されている．これらの手法では主に教師あり学習を用いるため，大量のレビューを目視で確認し，バグ報告や機能要求といったラベルを付与することで教師データを生成する必要がある．またレビューとは別に，エンドユーザが情報を共有する場としてフォーラムがある．特にゲームの分野で盛んな文化で，フォーラムに投稿される各トピックはバグ報告や機能要求などのカテゴリーに分類されている．本研究では，既にカテゴライズされたフォーラムのトピックを教師データとして用いることで，教師データを用意するコストを削減したレビューの分類手法を提案する．実験では，デジタルゲーム配信プラットフォームである Steam 上のレビューを対象に，レビューをバグ報告，機能要求，その他の 3 種類に分類した．既存手法では教師データの用意に 50 時間以上費やしたが，提案手法では 2 時間程度で多くの教師データを収集でき，教師データを用意するコストが大幅に削減できた．また，分類精度は既存手法には劣るものの，提案手法は AUC 0.8 以上の精度でレビューを分類できることを示した．

キーワード　フォーラム，アプリケーションレビュー，機械学習，教師あり学習，自然言語処理，BERT

1. まえがき

ソフトウェア開発において，アプリケーションレビューは開発者にとって豊富な情報源である [1], [2]．有益な情報の例としては，バグの報告や新機能の提案，機能強化の要望，特定の機能に対するユーザのフィードバックなどが挙げられる．

これらのレビューは，アプリの開発やメンテナンスに活用できる．しかし，レビューの中には開発者ではなくユーザを対象とした内容も数多く存在しており，これらはソフトウェアの改善という目的にはあまり役に立たない．例えば，good や bad などの数単語のみで構成される内容や，インストールできないなど説明の不足した問題の報告，顔文字のみの内容などが存在する．これらは対象アプリの印象を共有する，というレビューそのものがもつ目的に従う内容ではあるが，開発の改善にはつながらない．人気のアプリに対しては 1 日に数百件のレビューが投稿されているため，開発者による手作業での確認には限界がある．

そこで，膨大な数のレビューから開発者が有用な情報を効率良く取得することを目的とした研究が多く行われている [3]〜[19]．これらの研究では自然言語処理と機械学習，主に教師あり学習を用いて，レビューをその内容ごとに自動で分類する．既存研究では主に，Apple App Store や Google Play のようなモバイルアプリ配信プラットフォームに投稿されたレビューを，バグ報告や機能要求といったカテゴリーに分類している．

既存研究の課題として，教師データを用意するコストの高さが挙げられる．機械学習において教師データの数と質は精度に大きく寄与するため，既存手法では大量のレビューを目視で精読し，バグ報告や機能要求といったラベルを付ける必要がある．実際に，Maalej ら [6] は 4,400 件，Zhang ら [8] は 3,092 件のレビューを目視で確認しており，大きな労力を費やしている．

そこで本研究では，アプリレビュー分類における教師データ作成の労力削減を目的として，レビューを目視でラベル付けする代わりに，既にカテゴライズされ

[†] 大阪大学，吹田市
Osaka University, Suita-shi, 565–0871 Japan
a) E-mail: n-itikaw@ist.osaka-u.ac.jp
b) E-mail: shinsuke@ist.osaka-u.ac.jp
c) E-mail: kusumoto@ist.osaka-u.ac.jp
DOI:10.14923/transinfj.2022JDP7014

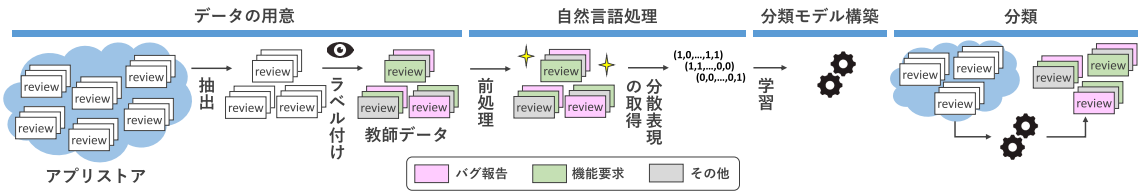


図 1　既存手法の流れ

たフォラーム上のトピックを教師データとして用いる手法を提案する．フォーラムとは，ユーザと開発者が自由に議論できるインターネットコミュニティの一形態であり，個々のソフトウェアに対して設置されることが多い．例えば Web 会議サービスである Zoom には，Zoom Developer Forum(注1)が設けられている．フォーラムはレビューとは異なり，バグ報告や機能要求，一般議論などのカテゴリーがフォーラム管理者によって事前に設けられている．よってフォーラムへの投稿（トピック）は，既にカテゴリーごとに分類された自然言語のデータであるとみなすことができる．このトピック群をレビュー分類の教師データとすることで，目視によるラベリングコストの大幅な削減が可能となる．

具体的な手法としては，アプリレビューをバグ報告，機能要求，その他の 3 種類に分類するために，フォーラムのバグ報告，機能要求，一般議論のカテゴリーに属するトピックをそれぞれに対応する教師データとして利用する．フォーラムが設けられているアプリはあまり多くないが，ゲームの分野ではフォーラムが設けられることが多い．そこで実験では，デジタルゲーム配信プラットフォームである Steam 上のフォーラムをもつゲームへのレビューを対象として，提案手法と既存手法の分類精度を比較した．その結果，提案手法は既存手法に比べて分類精度は低下するが，AUC 0.8 以上の精度でレビューを分類可能だと示した．

2. 準　　備

2.1 既存手法の流れ

近年，膨大な量のアプリレビューの中から開発者にとって有益な情報を効率的に取得するために，機械学習を用いてアプリレビューを自動分類する研究が多く行われている [3]〜[19]．これらの研究は，図 1 に示したフローに従ってレビューの分類を行う．

まずはデータの用意を行う．教師あり学習に用いる教師データを作成するため，アプリストアから幾つかのレビューを抽出し，目視によって個々のレビューがどの種類に属するかをラベル付けする．既存研究では主に，Apple App Store や Google Play のようなモバイルアプリ配信プラットフォームに投稿されたレビューを対象に実験を行っている．分類の区分は研究によって様々である．Chen ら [4] は，レビューを開発者にとって有益かどうかで 2 種類に分類し，Maalej ら [6] は，バグ報告，機能要求，ユーザ体験，評価の 4 種類に分類している．更に Zhang ら [8] は，バグ報告や機能要求を含む計 17 種類のカテゴリーに分類している．

次に自然言語処理を行う．自然言語処理とは，曖昧な性質をもつ自然言語をコンピュータが扱えるように加工する処理である．具体的には，小文字への統一化や単語単位への分割などの前処理や，文章をその特徴を表す分散表現（ベクトル）に変換する処理を行う．

続いて分類モデルの構築を行う．分類モデルに対して自然言語処理を施した教師データを与えることで，正しく文章を分類できるように学習を行う．分類モデルには，ナイーブベイズやロジスティック回帰のような古典的な機械学習手法 [4]〜[6] や，深層学習を利用した手法 [15] など様々な手法が報告されている．

最後にレビューの分類を行う．学習済みモデルに自然言語処理を施した未分類状態のレビューを入力することで，各レビューがどの種類に属するかを判定する．

2.2 課　　題

既存研究の課題として，データの用意にかかるコストの高さが挙げられる．既存研究はそれぞれ，自然言語処理や分類モデルに独自のアプローチを施すことで分類精度の向上を目指している．一方で，データの用意については共通して，数千件のレビューを目視でラベル付けすることで教師データを生成している．機械学習には多くの教師データが必要であり，目視で教師データを作成する労力は非常に大きい．特にバグ報告や機能要求に該当するレビューは，レビュー全体に占める割合が少なく，十分な数を確保するためには大

(注1)：https://devforum.zoom.us

量のレビューを目視する必要がある．実際に Maalej ら [6] は，ラベリングを行うための GUI ツールを作成することで，Apple App Store と Google Play から取得した計 4,400 件のレビューを目視でラベル付けしている．このような目視で教師データを作成する労力は，アプリの開発者が，自身が開発したアプリに対するレビューを分類しようとした際に大きな障壁になると考えられる．

2.3 フォーラム

フォーラムとは，レビューと同様にエンドユーザが感想や質問を投稿し，開発者やユーザ同士で自由に議論や情報共有ができるインターネットコミュニティの一形態である．Arudino Forum(注2) や Zoom Developer Forum のように，一般にフォーラムは一つのアプリケーションやプロジェクト単位で設けられる．様々な内容の投稿が混在しているレビューとは異なり，フォーラムにはバグ報告，機能要求，一般議論のようなカテゴリーがフォーラムをの管理者によって事前に設けられており，フォーラムへの投稿（トピック）はその内容ごとに分類されている．これによって，開発者が開発に有益なトピックを確認しやすい仕組みが完成している．フォーラムはアプリの開発元が公式に運営していることもあれば，有志のエンドユーザが非公式に運営しているものもある．また，一つのアプリに複数のフォーラムが存在することもあり，その規模や設けられているカテゴリーは様々である．ただし，バグ報告や機能要求，一般議論に該当するカテゴリーは多くの場合で共通して設けられている．フォーラムは全てのアプリに設置されているわけではなく，フォーラムをもつアプリはあまり多くない．しかし，アプリの中でもゲームの分野ではフォーラムをもつアプリが比較的多い傾向にある．

2.4 Steam

Steam(注3) は，Valve Corporation が開発したデジタルゲーム配信プラットフォームである．30,000 以上のゲームが配信され，1 億 2,000 万人以上のアクティブユーザーがいるなど，PC ゲームのデジタル配信プラットフォームとしては最大級の規模を誇る．

Steam では，ユーザはプレイしたゲームのレビューを投稿することができ，1,000 万件以上のレビューが投稿されている．特に人気の高いゲームには 1 日に 500 件以上のレビューが投稿され，開発者がその全てに目を通すことは困難である [20]．これらのレビューには Apple App Store や Google Play に代表される一般的なモバイルアプリ配信プラットフォームに投稿されるレビューと同様に，雑多な感想やアスキーアートのような開発者にとってあまり役に立たないレビューと，バグ報告や機能要求のような開発者にとって有益なレビューが混在している．

3. 提案手法

本研究の目的は，アプリレビュー分類における教師データ作成の労力削減にある．そこで本研究では，目視でのラベル付けによって教師データを用意する代わりに，既にカテゴライズされたフォーラムのトピックを教師データとする．これにより，目視でのラベル付けによる教師データ作成のコストを大幅に削減したレビュー分類手法を提案する．具体的には，レビューをバグ報告，機能要求，その他の 3 種類に分類するために，フォーラムのバグ報告，機能要求，一般議論のカテゴリーに属するトピックをそれぞれに対応する教師データとして利用する．

図 2 に提案手法の流れを示す．既存手法との大きな違いは，データの用意のステップである．既存手法ではレビューを幾つか無作為に抽出して目視でラベル付けを行っているのに対して，提案手法では既にカテゴライズされたフォーラムのトピックを用いるため，目視によってラベル付けを行う必要がない．自然言語処理や分類モデルの構築，実際にレビューを分類するステップは既存手法と同様に行い，自然言語処理の技術や分類モデルは既存研究で用いられている技術の中でも特に分類精度の高かった技術を用いる．詳細は **5.3**，**5.4** にて述べる．

提案手法の最大の目的は，教師データを用意する労力の削減であり，分類精度の向上ではない．レビューとフォーラムはどちらもエンドユーザが自身の意見を投稿することのできる場であるが，一般にレビューはライトユーザからの投稿が多く，フォーラムはヘビーユーザからの投稿が多い傾向にある．そのため，レビューとフォーラムに投稿される文章には少なからず性質の違いが生じる．これにより，提案手法の分類精度は既存手法よりも低下することが予想される．一方で，提案手法では大量の教師データを用意することができるため，機械学習において有利に働く可能性もある．

（注2）：https://forum.arduino.cc
（注3）：https://store.steampowered.com/

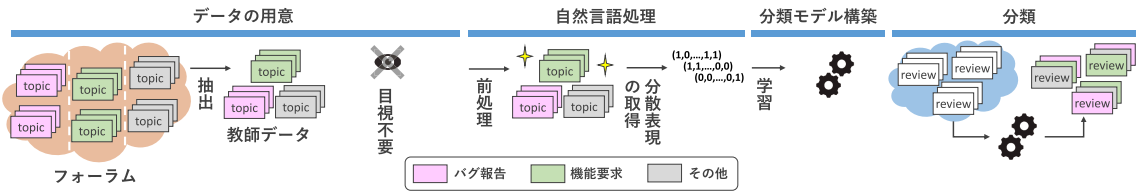


図 2　提案手法の流れ

4. Research Question

本研究では，提案手法の有効性を確認するために，以下の二つの Research Question（RQ）を設ける．

RQ1: 提案手法の分類精度は既存手法と比較してどの程度か？

RQ1 では，フォーラムのトピックが，レビューを分類する際の教師データとして機能するかどうかを確認する．ここで，あるアプリ app のフォーラムから得られたトピック群を F_{app}，アプリストアから得られたレビュー群を R_{app} と定義する．更に，F_{app} を教師データ，R_{app} をテストデータとする提案手法を，$F_{app} \Rightarrow R_{app}$ と表記する．既存手法は，教師データとテストデータの両方にレビューを用いるため，$R_{app} \Rightarrow R_{app}$ と表現できる．本 RQ では，これら二つの手法の分類精度を比較する．**3.** で述べたように，レビューとフォーラムの性質の違いによって提案手法の分類精度は既存手法に比べて低下することが予想される．しかし，提案手法でも一定の精度が確認できれば，フォーラムのトピックが教師データとして機能することが確認できる．

RQ2: フォーラムをもたないアプリへのレビューを分類する際にも提案手法を適用可能か？

2.3 で述べたように，フォーラムをもつアプリは多くはない．フォーラムをもたないアプリへのレビューを分類する際に提案手法を適用する場合，別アプリのフォーラムを教師データとして利用することが考えられる．異なるアプリのフォーラムを教師データとして用いる場合，そのアプリ固有の表現や文化が影響することで分類精度が低下する可能性がある．そこで，異なるアプリのフォーラムでも教師データとして機能するかを調査するために，あるアプリ app_A のフォーラムのトピックを教師データとして同じアプリへのレビューを分類する場合 $F_{app_A} \Rightarrow R_{app_A}$ と，別のアプリ app_B のフォーラムを教師データとしてあるアプリ app_A へのレビューを分類する場合 $F_{app_B} \Rightarrow R_{app_A}$ の分類精度を比較する．$F_{app_B} \Rightarrow R_{app_A}$ でも一定の精度が確認できれば，フォーラムをもたないアプリへのレビューを分類する場合でも提案手法が適用できることが確認できる．

5. 実 験 手 順

5.1 概　　要

本節では RQ を確かめるための実験手順，すなわち既存手法と提案手法による分類タスクの実現方法について説明する．図 1 と図 2 に示したとおり，既存手法と提案手法の本質的な違いは教師データの違いにある．よって，これら二つの手法は自然言語処理，分類モデルの構築，分類の 3 手順は共通となる．以降では図 1 と図 2 の流れに従い，データの用意（**5.2**），自然言語処理（**5.3**），分類モデル構築（**5.4**），評価（**5.5**）の順で説明する．

5.2 データの用意

実験を行うにあたって，分類対象になるレビューと教師データになるフォーラムのトピックを収集する．**2.3** で述べたように，ゲームにはフォーラムを設けられることが多い．そこで本実験では，ゲームに対するレビューを対象とし，Steam 上からレビューを収集する．これは **2.4** のとおり，Steam はゲーム配信プラットフォームとして最大級の規模を誇り，ゲームに対する多くのレビューが存在するためである．また，Steam 上の全てのゲームがフォーラムをもっているわけではないため，本研究で用いるデータセットとして，Steam の売り上げ上位のゲームタイトルからフォーラムをもつタイトルを選定する．売り上げ上位のタイトルを対象とするのは，より多くのレビューを収集するためである．なお，本実験では開発元が公式に運営しているフォーラムをもつタイトルのみ対象とした．これらの条件を満たすタイトルとして，Cities: Skylines(注4)（Cities）と Euro Truck Simulator 2(注5)（Euro）を本実験

（注4）：https://store.steampowered.com/app/255710/Cities_Skylines/
（注5）：https://store.steampowered.com/app/227300/Euro_Truck_Simulator_2/

の対象として選定した．これらのタイトルのレビューのうち英語で書かれたものと，これらのタイトルのフォーラム(注6)(注7)からバグ報告，機能要求，一般議論に該当するカテゴリーに分類されたトピックをスクレイピングによって収集した．

提案手法は，教師データを作成するために目視によるラベル付けを行わずにすむことがメリットであるが，本実験では，レビューを目視でラベル付けする作業を行う．これは，提案手法の分類精度の比較対象として既存手法を実施する必要があり，その既存手法の適用においてはレビューのラベル付けが必須であることが理由の一つである．更に，提案手法と既存手法のいずれにおいても，その分類精度を確認するためには分類の正解集合を知る必要があることも理由として挙げられる．

収集したレビューからタイトルごとに約 1,000 件を無作為抽出し，目視によりバグ報告，機能要求，その他のラベルを付与した．しかし，バグ報告，機能要求に該当するレビューはそれぞれ 20 件以下であり，実験に十分な数が取得できなかった．そこでバグ報告によくみられる単語として`"bug"`，`"fix"`，`"crash"` [6] を含んだレビューの中から 800 件を無作為抽出し，目視によってラベル付けを行った．その中からバグ報告，機能要求のラベルが付与されたレビューを実験データに追加した．機能要求についても同様に，機能要求によくみられる単語として`"please"`，`"hope"`，`"improve"`，`"need"`，`"prefer"`，`"request"`，`"suggest"`，`"wish"` [6] を含んだレビューの中から 800 件を無作為抽出し，目視によってラベル付けを行った．その中からバグ報告，機能要求のラベルが付与されたレビューを実験データに追加した．

表 1 に収集したレビュー数とフォーラムのトピック数を示す．本研究ではレビューのラベル付け作業に 50 時間以上費やしたが，フォーラム上のトピックを収集するために費やした時間は 2 時間程度である．得られたレビュー数に対してフォーラムのトピック数は 10 倍以上であり，提案手法では少ないコストで教師データが大量に用意できることが分かる．

表 1　実験データ

	レビュー数			
	#バグ報告	#機能要求	#その他	合計
Cities	106 (17)	69 (15)	1,000	1,175
Euro	63 (5)	87 (6)	959	1,104

	フォーラムのトピック数			
	#バグ報告	#機能要求	#その他	合計
Cities	7,007	2,813	5,650	15,470
Euro	13,113	4,800	4,184	22,097

※括弧内の数値は，レビュー全体から無作為に抽出した約 1000 件に含まれるバグ報告，機能要求のラベルが付与されたレビューの数である．

5.3　自然言語処理

自然言語処理技術を用いてテキストに前処理を施すことによって，自然言語のもつ曖昧さを削減できる場合がある [6]．ここでは利用した BERT-Base-uncased と同様の前処理として，出現する全ての文字の小文字化処理のみを適用した．一般的な自然言語処理としては，ストップワード除去や Lemmatization などが広く知られているが，BERT ではトークナイズ処理において Byte Pair Encoding が採用されており，それらの前処理は不要であると指摘されている [21]．

5.4　分類モデル構築

本実験で用いる分類器は BERT（Bidirectional Encoder Representations from Transformers）[22] を用いて構築する．BERT は 2018 年に Google から発表された自然言語処理を行うための深層学習モデルであり，転移学習によりさまざまなタスクに対応でき，少ないデータを追加で学習することによってモデルの構築を行うことができるという特徴がある．アプリレビューの分類においても，ナイーブベイズやロジスティック回帰による分類モデルに比べて BERT による分類モデルが高い精度を示したことが報告されている [23]．

提案手法はテストデータとは別種のデータソースを教師データとして用いる性質上，学習データの語彙に存在しない未知語の存在が精度低下の要因になると考えられる．しかし，BERT におけるトークン化処理では WordPiece アルゴリズムが採用されており，未知語は既知の複数のサブワードに分解される．例えば`"playing"`という語が未知語であった場合，既知の`"play"`と`"##ing"`の二つの語に分解される．よって，BERT は提案手法と親和性の高いモデルであると考えられる．

本実験では，ラベル付けを行った Cities と Euro のレビューをそれぞれ 7:3 の比率で教師データとテストデータに分割し，R_{cities}，R_{euro}，F_{cities}，F_{euro} の 4 種類の教師データでそれぞれ学習した 4 種類の分類器を生成した．なお，本実験では BooksCorpus と Wikipedia によって事前学習された BERT-Base-uncased モデルをファインチューニングして使用した．今回の実験で設

(注6)：https://forum.paradoxplaza.com/forum/forums/cities-skylines.859/
(注7)：https://forum.scssoft.com/viewforum.php?f=3

表 2　ハイパーパラメタ

入力最大系列長	128
バッチサイズ	32
最大エポック数	10
損失関数	交差エントロピー
最適化手法	Adam [24]
学習率	3e-5

定したハイパーパラメタを表 2 に示す．分類モデルとしては，ファインチューニング後の最終層の CLS ベクトル（pooler レイヤ）から全結合層に直結する構造を構築した．最適化手法には Adam [24] を，損失関数には交叉エントロピーを用いた．過学習回避のために Early Stopping を有効としたが，それ以外の過学習回避の手段である Dropout 層は設置していない．実験の目的は学習データの違いによる効果の確認であり，ハイパーパラメタの調整やモデル構造の改善等は最小限とした．

5.5 評　　価

生成した分類器を用いて，R_{cities} と R_{euro} を分類し，その精度を調査する．まず，既存手法に倣い $R_{\text{cities}}\Rightarrow R_{\text{cities}}$，$R_{\text{euro}}\Rightarrow R_{\text{euro}}$ を実施する．次に，RQ1 を調査するために提案手法として $F_{\text{cities}}\Rightarrow R_{\text{cities}}$，$F_{\text{euro}}\Rightarrow R_{\text{euro}}$ を実施する．最後に，RQ2 を調査するために，提案手法を交差的に適用した $F_{\text{euro}}\Rightarrow R_{\text{cities}}$，$F_{\text{cities}}\Rightarrow R_{\text{euro}}$ を実施する．

分類精度は，Precision（精度），Recall（再現率），F1-score，ROC 曲線の AUC（Area under the curve）の四つの評価指標を用いて評価する．ROC 曲線は，縦軸を真の陽性率，横軸を偽陽性率として，様々なしきい値に対応する点をプロットした曲線で，AUC は ROC 曲線の下の面積で定義される．AUC は分類器の性能を特定のしきい値に拠らず評価する指標であり，0 から 1 の値を取る．AUC は分類器の性能が高いほど 1 に近づき，ランダムに分類を行う分類器に対しては 0.5 となる．

6. 実 験 結 果

図 3 に，本実験で実施した各手法の結果をそれぞれ示す．図には混同行列と Precision，Recall，F1-score，AUC が示されており，黄色でハイライトされた箇所は正しく分類できているレビューの数を表している．また，図 4 に AUC を算出するための ROC 曲線を示す．この曲線で区切られた面積で AUC は定義される．

6.1 RQ1: 提案手法の分類精度は既存手法と比較してどの程度か？

RQ1 では，既存手法と提案手法の分類精度を比較する．まず，図 3(a) の R_{cities} に対して既存手法を適用した場合と，図 3(b) の提案手法を適用した場合の比較を行う．AUC に注目すると，既存手法は全てのカテゴリーで 0.9 を超えており，高い精度で分類できている．対して提案手法では，既存手法には劣るものの全てのカテゴリーで AUC は 0.8 以上であった．それぞれの混同行列に注目すると，提案手法では既存手法に比べて F 値は一貫して低下する結果となった．特に Cities の機能要求では，既存手法でも F 値が低かったが提案手法では適合率，再現率共に低下している．逆に Euro ではバグ報告の適合率と再現率が低下していた．提案手法はバグ報告，機能要求共にどちらのアプリに対しても AUC 0.8 以上の精度で分類できる一方で，F 値は低い結果となった．このことから，実際の利用シーンにおいては分類のしきい値設定に課題が残ると考えられる．

RQ1 の結論：提案手法は既存手法には劣るものの AUC 0.8 以上の精度で分類が可能である．バグ報告と機能要求の F 値は低いため分類のしきい値設定は改善が必要である．

6.2 RQ2: フォーラムをもたないアプリへのレビューを分類する際にも提案手法を適用可能か？

RQ2 では，フォーラムをもたないアプリのレビューを分類するシナリオを想定して，別アプリのフォーラムを教師データとして提案手法を適用した場合の分類精度を確認する．まず，図 3(b) の $F_{\text{cities}}\Rightarrow R_{\text{cities}}$ と図 3(c) の $F_{\text{euro}}\Rightarrow R_{\text{cities}}$ を比較する．AUC に注目すると，別アプリのフォーラムを用いることで分類精度は低下しているものの，AUC は全てのカテゴリーで 0.7 以上であった．次に，図 3(e)$F_{\text{euro}}\Rightarrow R_{\text{euro}}$ と図 3(f)$F_{\text{cities}}\Rightarrow R_{\text{euro}}$ を比較する．AUC を確認すると別アプリのフォーラムを用いた場合であっても，大きな精度の低下は見られなかった．AUC は全てのカテゴリーで 0.7 以上であった．これらの結果から，アプリ固有の表現や文化は分類精度にあまり影響を与えないことが分かる．

RQ2 の結論：別アプリのフォーラムを教師データとして提案手法を適用した場合でも AUC 0.7 以上の精度で分類が可能である．

		予測した分類			適合率	再現率	F値
		バグ	機能	他			
真の分類	バグ	20	1	5	0.69	0.77	0.73
	機能	4	8	10	0.53	0.36	0.43
	他	5	6	294	0.95	0.96	0.96

(a) $R_{\text{cities}} \Rightarrow R_{\text{cities}}$

		予測した分類			適合率	再現率	F値
		バグ	機能	他			
真の分類	バグ	23	0	3	0.52	0.88	0.66
	機能	1	5	16	0.38	0.23	0.29
	他	20	8	277	0.94	0.91	0.92

(b) $F_{\text{cities}} \Rightarrow R_{\text{cities}}$

		予測した分類			適合率	再現率	F値
		バグ	機能	他			
真の分類	バグ	13	1	12	0.43	0.50	0.46
	機能	1	11	10	0.24	0.50	0.32
	他	16	34	255	0.92	0.83	0.88

(c) $F_{\text{euro}} \Rightarrow R_{\text{cities}}$

		予測した分類			適合率	再現率	F値
		バグ	機能	他			
真の分類	バグ	17	2	4	0.77	0.74	0.76
	機能	2	14	11	0.70	0.52	0.60
	他	3	4	275	0.95	0.98	0.96

(d) $R_{\text{euro}} \Rightarrow R_{\text{euro}}$

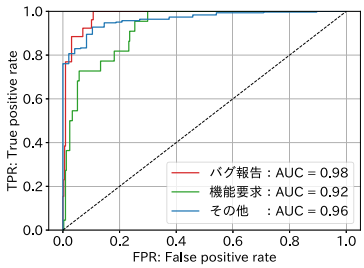
		予測した分類			適合率	再現率	F値
		バグ	機能	他			
真の分類	バグ	13	1	9	0.33	0.57	0.41
	機能	0	15	12	0.60	0.56	0.58
	他	27	9	246	0.92	0.87	0.90

(e) $F_{\text{euro}} \Rightarrow R_{\text{euro}}$

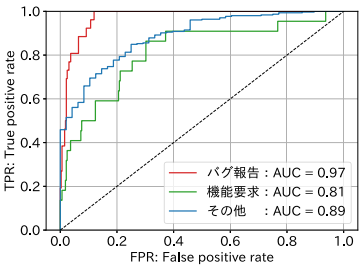
		予測した分類			適合率	再現率	F値
		バグ	機能	他			
真の分類	バグ	14	3	6	0.93	0.61	0.74
	機能	0	16	11	0.46	0.59	0.52
	他	1	16	265	0.94	0.94	0.94

(f) $F_{\text{cities}} \Rightarrow R_{\text{euro}}$

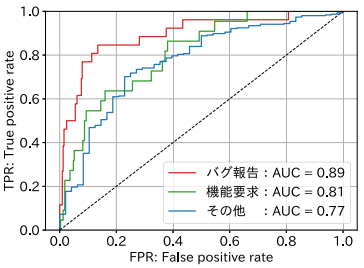
図 3　実験結果：混同行列



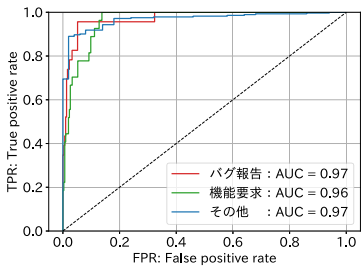
(a) $R_{\text{cities}} \Rightarrow R_{\text{cities}}$



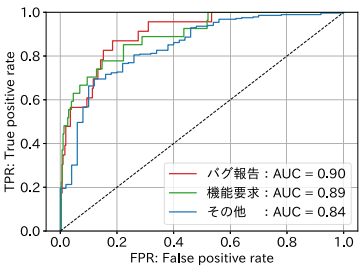
(b) $F_{\text{cities}} \Rightarrow R_{\text{cities}}$



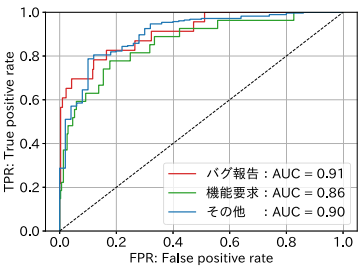
(c) $F_{\text{euro}} \Rightarrow R_{\text{cities}}$



(d) $R_{\text{euro}} \Rightarrow R_{\text{euro}}$



(e) $F_{\text{euro}} \Rightarrow R_{\text{euro}}$



(f) $F_{\text{cities}} \Rightarrow R_{\text{euro}}$

図 4　実験結果：ROC 曲線

7. 考　　察

本節では，実験結果の考察を行う．本実験の分類対象となったレビューの中から，正しく分類できなかったレビューを定性的に調査することで，その特徴を明らかにした．以下に，正しく分類できなかったレビューに見られた特徴を示し，その理由を考察する．

7.1　1 単語以下の情報量の少ないレビュー

提案手法では，1 単語以下の情報量の少ないレビューをバグ報告や機能要求に誤って分類してしまうケースが数件確認できた．例えば $F_{\text{euro}} \Rightarrow R_{\text{euro}}$ では，"a"という 1 文字だけのレビューをバグ報告に分類している．このレビューは，ストップワード除去によって"a"という単語が除去されるため，空の文章として分類器に入力される．F_{euro} を教師データとした場合，このように空の文章や 1 単語のみのレビューをバグ報告として分類するケースが数件確認できた．また，F_{cities} を教師データとした場合は，空の文章や 1 単語のみのレビューを機能要求に分類するケースが数件確認できた．

このような誤分類の原因として，フォーラムとレビューの性質の違いが考えられる．**3.** で述べたとおり，レビューはライトユーザが気軽に投稿する傾向にある一方，フォーラムはヘビーユーザが投稿する傾向にある．そのため，フォーラムにおける一般議論のカテゴリーには，1 単語のみのような情報量の少ないトピックがほとんど存在しないことが予想される．

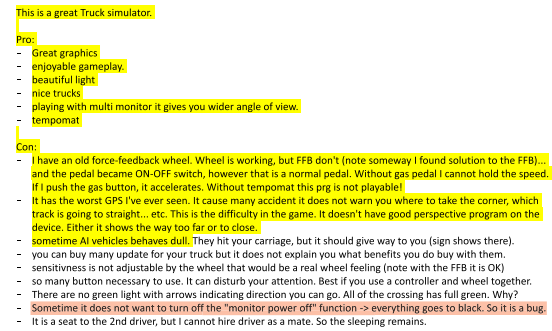
このような誤分類を防ぐための改善案として，レビューを分類する前に 1 単語以下のレビューを除外する手法が考えられる．今回の実験結果から 1 単語以下のレビューを除外したところ，図 3(b)$F_{\text{cities}} \Rightarrow R_{\text{cities}}$ において，実際にはその他であるが，機能要求と予測してしまった 8 件のうち 3 件が除外された．また，図 3(e)$F_{\text{euro}} \Rightarrow R_{\text{euro}}$ において，実際にはその他であるが，バグ報告と予測してしまった 27 件のうち 10 件が除外された．

7.2 文章量の多いレビュー

既存手法と提案手法のどちらにおいても，一定以上の文章量をもつレビューが正しく分類されていないケースが数件確認できた．これは，BERT における入力長の制限が原因であると考えられる．BERT は，文章を形態素単位に分割したリストを入力とするが，入力は固定長でなければならない．そこで，入力最大系列長に満たない文章は空いた部分を無意味なデータでパディングし，逆に入力最大系列長を超える文章は，入力最大系列長に合うようにトリミングされる．本実験では，文章の先頭から順に入力として扱い，入力最大系列長を超えた段階で残りの部分を無視する手法をとっている．表 2 のとおり，本実験では入力最大系列長を 128 としているため，これを超えるレビューについては，レビューの一部分が考慮されずに分類が行われる．そのため，文章量が多く文章の後半にバグ報告や機能要求に当たる記述があるレビューは，その部分が無視されてしまい，正しく分類ができなかったと考えられる．実際に文章量の多いレビューでは，初めに好意的な内容のレビューを行い，その後にバグ報告や改善点，不満点を述べる構成のレビューが多く見られた．

図 5 に，実際にはバグ報告であるが，誤ってその他と予測された文章量の多いレビューの例を示す．黄色でハイライトされた部分は入力として反映される箇所で，それ以降は入力最大系列長を超えて無視される文章である．また，赤でハイライトされた部分はバグを報告している箇所である．このレビューは，アプリの長所を述べたのちに短所を述べる構造だが，入力長の制限により短所を述べている途中で文章が打ち切られている．そのため，レビューの後半に書かれたバグを報告している箇所が無視されている．

このような誤分類を防ぐための改善案として，入力最大系列長をより大きく設定することが考えられる．入力最大系列長に比例して学習時に必要なメモリ容量


This is a great Truck simulator.

Pro:
- Great graphics
- enjoyable gameplay.
- beautiful light
- nice trucks
- playing with multi monitor it gives you wider angle of view.
- tempomat

Con:
- I have an old force-feedback wheel. Wheel is working, but FFB don't (note someway I found solution to the FFB)... and the pedal became ON-OFF switch, however that is a normal pedal. Without gas pedal I cannot hold the speed. If I push the gas button, it accelerates. Without tempomat this prg is not playable!
- It has the worst GPS I've ever seen. It cause many accident it does not warn you where to take the corner, which track is going to straight... etc. This is the difficulty in the game. It doesn't have good perspective program on the device. Either it shows the way too far or to close.
- sometime AI vehicles behaves dull. They hit your carriage, but it should give way to you (sign shows there).
- you can buy many update for your truck but it does not explain you what benefits you do buy with them.
- sensitivness is not adjustable by the wheel that would be a real wheel feeling (note with the FFB it is OK)
- so many button necessary to use. It can disturb your attention. Best if you use a controller and wheel together.
- There are no green light with arrows indicating direction you can go. All of the crossing has full green. Why?
- Sometime it does not want to turn off the "monitor power off" function -> everything goes to black. So it is a bug.
- It is a seat to the 2nd driver, but I cannot hire driver as a mate. So the sleeping remains.

図 5　誤分類された文章量の多いレビューの例

が大きくなるため，本研究の実験環境では 128 が入力最大系列長の限界であった．しかし，実験環境によっては入力最大系列長をより大きくすることで，文章量の多いレビューに対応することができる．また，入力最大系列長を超えた文章をトリミングするのではなく，分割する手法も考えられる．一つの長いレビューを分割して二つ以上のレビューとみなして分類を行い，分割した各レビューのうち一つでもバグ報告や機能要求に分類されれば，分割元のレビューをバグ報告や機能要求とみなすことで，情報を損なわずに文章量の多いレビューを分類できる可能性がある．

8. 制　　約

8.1 フォーラム構造への強依存

提案手法が抱える一つの強い制約として，学習データとなるフォーラム構造への強依存が挙げられる．提案手法はフォーラムの構造を学習データのラベルとして直接利用するため，フォーラム上に設置されていないトピックの文章を分類することはできない．

Pagano と Maalej らの調査 [25] によると，アプリへのフィードバックは以下 4 種類に大別できる．

- コミュニティ内での情報の共有．他アプリの推薦やアプリ自体の使い方の共有など．
- アプリ自体の改善につながる情報．機能要求やバグ報告など．
- アプリの評定．称賛や非難など．
- アプリ使用の体験．機能や UI の説明など．

本論文では上記 4 種類のうち，アプリ自体の改善につながる情報に着目し，バグ報告と機能要求の自動分類を試みた．バグ報告と機能要求は多くのフォーラムで独立したトピックとして設置されることが多く，提案手法による分類が適するユースケースであるといえる．

他方，それ以外の 3 種類を自動分類するようなユースケースに適用できるかはフォーラムに強く依存する．また機能要求をより詳細なサブカテゴリ，例えば機能と非機能へ分類するようなケース [26], [27] への適用も困難である．このようなユースケースの場合，提案手法を用いて自動分類を試みた後に人手によるラベリングと併用することで，低コストかつ詳細な分類ができると考えられる．

8.2 妥当性の脅威

データセットのラベル付けが分類器の性能に影響している可能性がある．本実験ではまず初めに，著者ら 2 名で数十件のレビューを確認し，バグ報告，機能要求の基準を定めたのちに，著者が 1 人でラベル付けを行った．そのため主観の影響が完全には排除できていない．

本研究では，Cities: Skylines と Euro Truck Simulator 2 の二つのゲームのフォーラムを対象として実験を行った．しかし，フォーラムによってその規模やカテゴリーの区分は様々である．本実験で対象としたフォーラムとは別のフォーラムを教師データとした場合に，本実験と同程度の分類精度を達成できるかどうかは明らかでない．

6.1 で述べたとおり，提案手法は AUC 自体は高いものの F 値は低い傾向にあった．実際の利用シーンにおいては分類結果そのものが重要となるため，分類時のしきい値設定には課題が残るといえる．この F 値の低さの理由の一つは，実験に用いたデータの不均衡にあると考えられる．その他にカテゴライズされる文章は，バグ報告や機能要求と比べて 10 倍以上の数があり典型的な不均衡データである．本論文では損失関数として BERT と同じ交叉エントロピーを用いたが，不均衡データを考慮した Focal Loss [28] や Dice Loss [29] 等の指標が提案されており，これらを利用することで結果が変化する可能性がある．

9. むすび

本研究では，フォーラムを用いることで，教師データを用意するコストを削減したアプリレビュー分類手法を提案した．実験では，デジタルゲーム配信プラットフォームである Steam 上のレビューを対象に，レビューをバグ報告，機能要求，その他の 3 種類に分類した．その結果，提案手法は既存手法と比べて精度は下がるものの，AUC 0.8 以上の精度で分類ができることを示した．またフォーラムがないアプリについても，別のアプリのフォーラムを教師データとすることで提案手法が適用できることを示した．

今後の課題としては，対象とするゲームタイトルを増やし，一般化可能性を向上させる点が挙げられる．また今回はゲームドメインを対象に実験を行ったが，既存研究で主に対象とされているモバイルアプリでの実験も重要な課題である．

謝辞　本研究の一部は，JSPS 科研費（JP20H04166, JP21H04877）による助成を受けた．

文　献

[1] A. Holzer and J. Ondrus, “Mobile application market: A developer’s perspective,” Telematics and Informatics, vol.28, no.1, pp.22–31, 2011.

[2] J. Dąbrowski, E. Letier, A. Perini, and A. Susi, “Analysing app reviews for software engineering: A systematic literature review,” Empir. Softw. Eng., vol.27, no.2, p.63, 2022.

[3] C. Iacob and R. Harrison, “Retrieving and analyzing mobile apps feature requests from online reviews,” Proc. Working Conference on Mining Software Repositories, pp.41–44, 2013.

[4] N. Chen, J. Lin, S.C.H. Hoi, X. Xiao, and B. Zhang, “AR-Miner: Mining informative reviews for developers from mobile app marketplace,” Proc. Int. Conf. Softw. Eng., pp.767–778, 2014.

[5] E. Guzman, M. El-Haliby, and B. Bruegge, “Ensemble methods for app review classification: An approach for software evolution,” Proc. Int. Conf. Automated Softw. Eng., pp.771–776, 2015.

[6] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik, “On the automatic classification of app reviews,” Requirements Engineering, vol.21, no.3, pp.311–331, 2016.

[7] S. Panichella, A. Di Sorbo, E. Guzman, C.A. Visaggio, G. Canfora, and H.C. Gall, “ARdoc: App reviews development oriented classifier,” Proc. Int. Symp. Foundations of Softw. Eng., pp.1023–1027, 2016.

[8] L. Zhang, X.-Y. Huang, J. Jiang, and Y.-K. Hu, “CSLabel: An approach for labelling mobile app reviews,” J. Comput. Sci. Technol., vol.32, pp.1076–1089, 2017.

[9] P. Zhenlian, J. Wang, K. He, and M. Tang, “An approach of extracting feature requests from app reviews,” Proc. Collaborate Computing: Networking, Applications and Worksharing, pp.312–323, 2017.

[10] N. Jha and A. Mahmoud, “Using frame semantics for classifying and summarizing application store reviews,” Empir. Softw. Eng., vol.23, pp.3734–3767, 2018.

[11] M. Tavakoli, L. Zhao, A. Heydari, and G. Nenadić, “Extracting useful software development information from mobile application reviews: A survey of intelligent mining techniques and tools,” Expert Systems with Applications, vol.113, pp.186–199, 2018.

[12] C. Stanik, M. Häring, and W. Maalej, “Classifying multilingual user feedback using traditional machine learning and deep learning,” Proc. International Requirements Engineering Conference Workshops, pp.220–226, 2019.

[13] C. Wang, P. Liang, M. Daneva, and M. vanSinderen, “Augmenting app reviews with app changelogs: An approach for app reviews

classification," Proc. Int. Conf. Softw. Eng. Knowl. Eng., pp.398–403, 2019.
[14] M.B. Messaoud, I. Jenhani, N.B. Jemaa, and M.W. Mkaouer, "A multi-label active learning approach for mobile app user review classification," Proc. Knowledge Science, Engineering and Management, pp.805–816, 2019.
[15] N. Aslam, W.Y. Ramay, K. Xia, and N. Sarwar, "Convolutional neural network-based classification of app reviews," IEEE Access, vol.8, pp.185619–185628, 2020.
[16] F. Rustam, A. Mehmood, M. Ahmad, S. Ullah, D.M. Khan, and G.S. Choi, "Classification of shopify app user reviews using novel multi text features," IEEE Access, vol.8, pp.30234–30244, 2020.
[17] M. Häring, C. Stanik, and W. Maalej, "Automatically matching bug reports with related app reviews," Proc. Int. Conf. Softw. Eng., pp.970–981, 2021.
[18] S. Panichella, A. Di Sorbo, E. Guzman, C.A. Visaggio, G. Canfora, and H.C. Gall, "How can i improve my app? classifying user reviews for software maintenance and evolution," Proc. Int. Conf. Software Maintenance and Evolution, pp.281–290, 2015.
[19] K. Srisopha, C. Phonsom, M. Li, D. Link, and B. Boehm, "On building an automatic identification of country-specific feature requests in mobile app reviews: Possibilities and challenges," Proc. Int. Conf. Softw. Eng. Workshops, pp.494–498, 2020.
[20] D. Lin, C.-P. Bezemer, Y. Zou, and A.E. Hassan, "An empirical study of game reviews on the steam platform," Empir. Softw. Eng., vol.24, no.1, pp.170–207, 2019.
[21] F. Fernández-Martínez, C. Luna-Jiménez, R. Kleinlein, D. Griol, Z. Callejas, and J.M. Montero, "Fine-tuning BERT models for intent recognition using a frequency cut-off strategy for domain-specific vocabulary extension," Applied Sciences, vol.12, no.3, p.1610, 2022.
[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," Proc. Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp.4171–4186, 2019.
[23] 山田侑樹，櫨山淳雄，"BERT によるアプリケーションレビュー分類モデルの構築と評価，" 信学技報，KBSE2020-38, 2021.
[24] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," Int. Conf. Learning Representations, pp.1–15, 2014.
[25] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," Proc. International Requirements Engineering Conference, pp.125–134, 2013.
[26] T. Hey, J. Keim, A. Koziolek, and W.F. Tichy, "Norbert: Transfer learning for requirements classification," Proc. International Requirements Engineering Conference, pp.169–179, 2020.
[27] M. Lu and P. Liang, "Automatic classification of non-functional requirements from augmented app user reviews," Proc. Int. Conf. Evaluation and Assessment in Softw. Eng., pp.344–353, 2017.
[28] T. Lin, Goyal P., Girshick R., He K., and P. Dollar, "Focal loss for dense object detection," IEEE Trans. Pattern Anal. Mach. Intell., vol.42, no.2, pp.318–327, 2020.
[29] X. Li, X. Sun, Y. Meng, J. Liang, F. Wu, and J. Li, "Dice loss for data-imbalanced NLP tasks," Proc. Annual Meeting of the Association for Computational Linguistics, pp.465–476, 2020.

（2022 年 3 月 14 日受付，6 月 20 日再受付，
7 月 27 日早期公開）

市川　直人

2020 大阪大学基礎工学部情報科学科卒．同年より同大学大学院情報科学研究科コンピュータサイエンス専攻博士前期課程在学中．自然言語処理に関する研究に従事．

柗本　真佑　（正員）

2010 奈良先端科学技術大学院大学博士後期課程了．同年神戸大学大学院システム情報学研究科特命助教．2016 大阪大学大学院情報科学研究科助教．博士（工学）．エンピリカルソフトウェア工学の研究に従事．

楠本　真二　（正員）

1988 大阪大学基礎工卒．1991 同大学大学院博士課程中退．同年同大学基礎工学部助手．1996 同講師．1999 同助教授．2002 同大学大学院情報科学研究科助教授．2005 同教授．博士（工学）．ソフトウェアの生産性や品質の定量的評価に関する研究に従事．IEEE，IFPUG 各会員．