修士学位論文

題目

# アプリレビュー分類におけるフォーラムを用いた分類モデルの構築

指導教員

楠本 真二 教授

報告者

市川 直人

令和 4 年 2 月 2 日

大阪大学 大学院情報科学研究科

コンピュータサイエンス専攻

令和 3 年度 修士学位論文

アプリレビュー分類におけるフォーラムを用いた分類モデルの構築

市川 直人

## 内容梗概

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よく取得するために，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，すで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 時間程度であり，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，提案手法の AUC は 0.8 以上であり，十分な精度でレビューを分類できることを示した．

### 主な用語

フォーラム，アプリケーションレビュー，機械学習，自然言語処理，BERT

# 目次


1 はじめに 1

2 準備 3

2.1 既存手法の流れ . . . . . 3

2.2 課題 . . . . . 5

2.3 フォーラム . . . . . 5

2.4 Steam . . . . . 8

3 提案手法 9

4 Research Question 11

5 実験手順 12

5.1 データの用意 . . . . . 12

5.2 自然言語処理 . . . . . 15

5.3 分類モデル構築 . . . . . 17

5.4 評価 . . . . . 18

6 実験結果 19

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？ . . . . . 22

6.2 RQ2: フォーラムを持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？ . . . . . 23

7 考察 24

8 妥当性の脅威 26

9 おわりに 27

謝辞 28

参考文献 29

# 図目次


1 既存手法の流れ . . . . . 4
2 フォーラムのカテゴリーの例 . . . . . 6
3 フォーラムのトピックの例 . . . . . 7
4 提案手法の流れ . . . . . 10
5 自然言語処理の流れ . . . . . 16
6 $E(R_{\mathrm{Cities}} \Rightarrow R_{\mathrm{Cities}})$ の実験結果 . . . . . 19
7 $E(F_{\mathrm{Cities}} \Rightarrow R_{\mathrm{Cities}})$ の実験結果 . . . . . 19
8 $E(F_{\mathrm{Euro}} \Rightarrow R_{\mathrm{Cities}})$ の実験結果 . . . . . 19
9 $E(R_{\mathrm{Euro}} \Rightarrow R_{\mathrm{Euro}})$ の実験結果 . . . . . 20
10 $E(F_{\mathrm{Euro}} \Rightarrow R_{\mathrm{Euro}})$ の実験結果 . . . . . 20
11 $E(F_{\mathrm{Cities}} \Rightarrow R_{\mathrm{Euro}})$ の実験結果 . . . . . 20
12 ROC 曲線 . . . . . 21
13 誤分類された文章量の多いレビューの例 . . . . . 25

## 表目次

1 実験データ . . . . . 14
2 ストップワード . . . . . 15
3 ハイパーパラメータ . . . . . 17

## 1　はじめに

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]．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グなどが挙げられる．

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持つ目的に従う内容ではあるが，開発の改善には繋がらない．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よく取得することを目的とした研究が多く行われている [2–16]．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 [5] は 4,400 件，Zhang ら [7] 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，すでにカテゴライズさ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[*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，すでにカテゴリー毎に分類された自然言語のデータであると見なすことができる．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，

[*1] https://devforum.zoom.us

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 上のフォーラムを持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，十分な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 [2–16]．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，アプリストアからいく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 [3] 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 [5] 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．さらに Zhang ら [7] 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，曖昧な性質を持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 [3] [4] [5] や，深層学習を利用した手法 [14] 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．

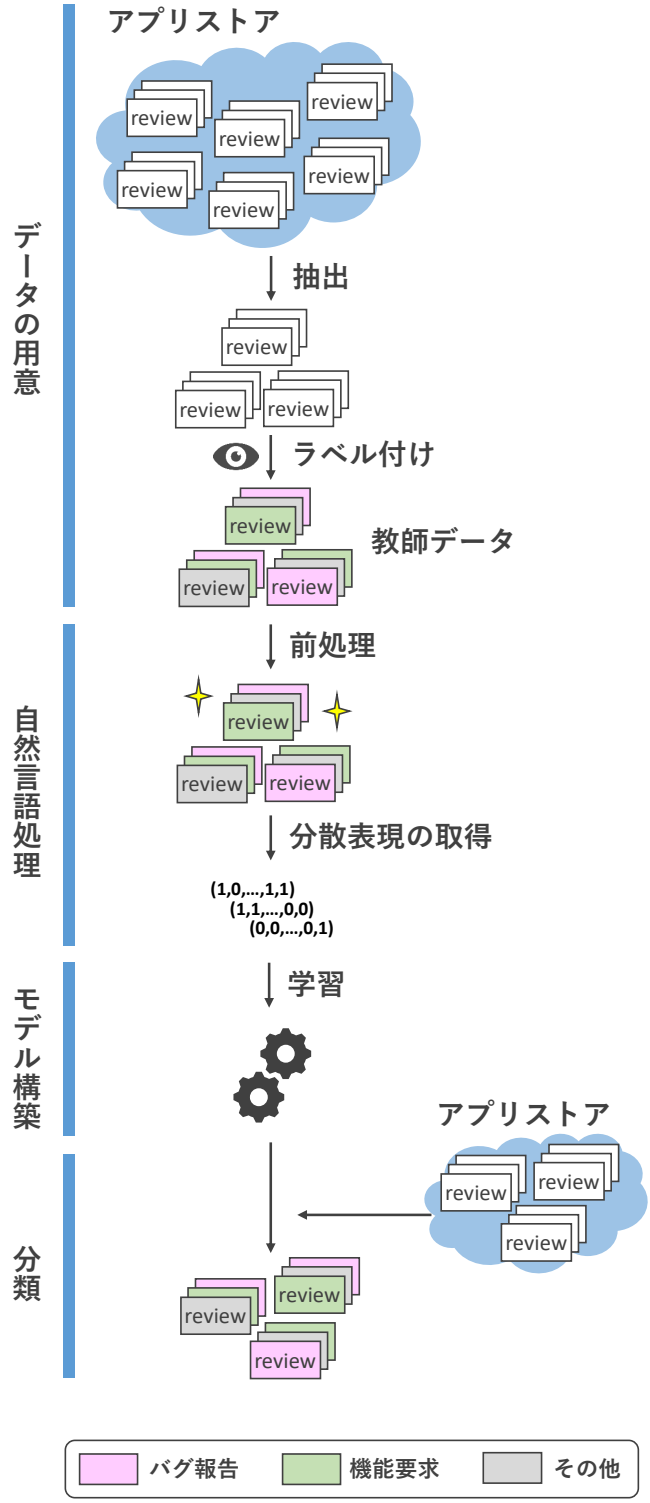

データの用意
自然言語処理
モデル構築
分類
アプリストア
review
抽出
ラベル付け
教師データ
前処理
分散表現の取得
(1,0,...,1,1)
(1,1,...,0,0)
(0,0,...,0,1)
学習
アプリストア
バグ報告
機能要求
その他

図 1　既存手法の流れ

### 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量のレビューを目視する必要がある．実際に Maalej ら [5] 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[*2]や Zoom Developer Forum のように，一般にフォーラムは 1 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）はその内容毎に分類されている．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，1 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，フォーラムを持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持つアプリが比較的多い傾向にある．

図 2 図 3 に，Euro Truck Simulator 2 というドライビングシミュレーションゲームのフォーラムを示す．図 2 は，このフォーラムで設定されているカテゴリーの一覧である．バグ報告や機能要求，一般議論に該当するカテゴリーが存在し，他にも複数のカテゴリーが確認できる．図 3 は，バグ報告として投稿されたトピックの一例である．トピックはタイトルと本文によって構成されている．また，ユーザや開発者はトピックに対して自由に返信を行うことができる．

[*2] https://forum.arduino.cc

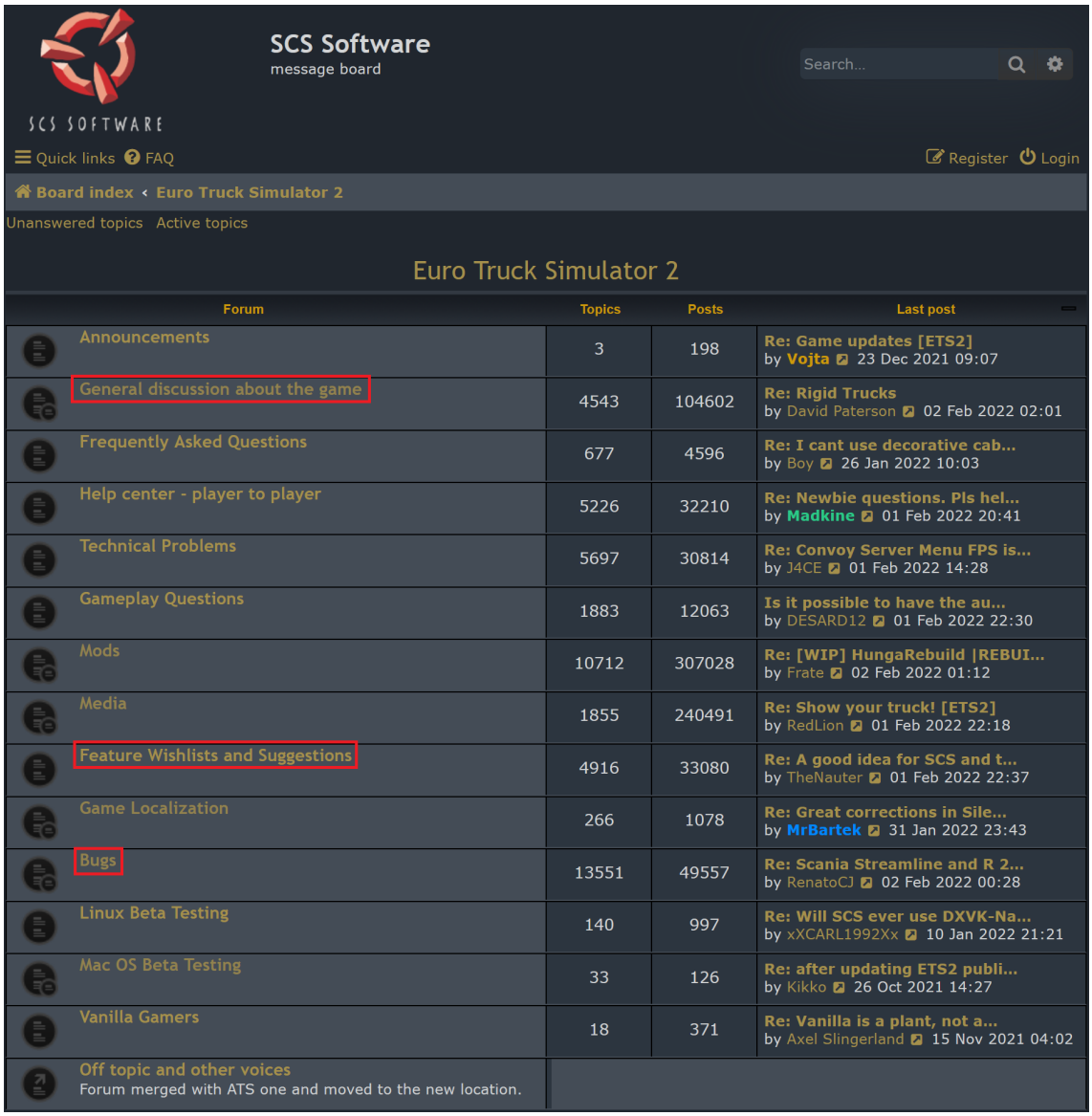



図 2　フォーラムのカテゴリーの例


図 3　フォーラムのトピックの例

### 2.4 Steam

Steam[*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，開発者がそのすべてに目を通すことは困難である [17]．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] https://store.steampowered.com/

## 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，すで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 4 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いく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，提案手法ではすで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 5.2 項，5.3 項にて述べる．

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．

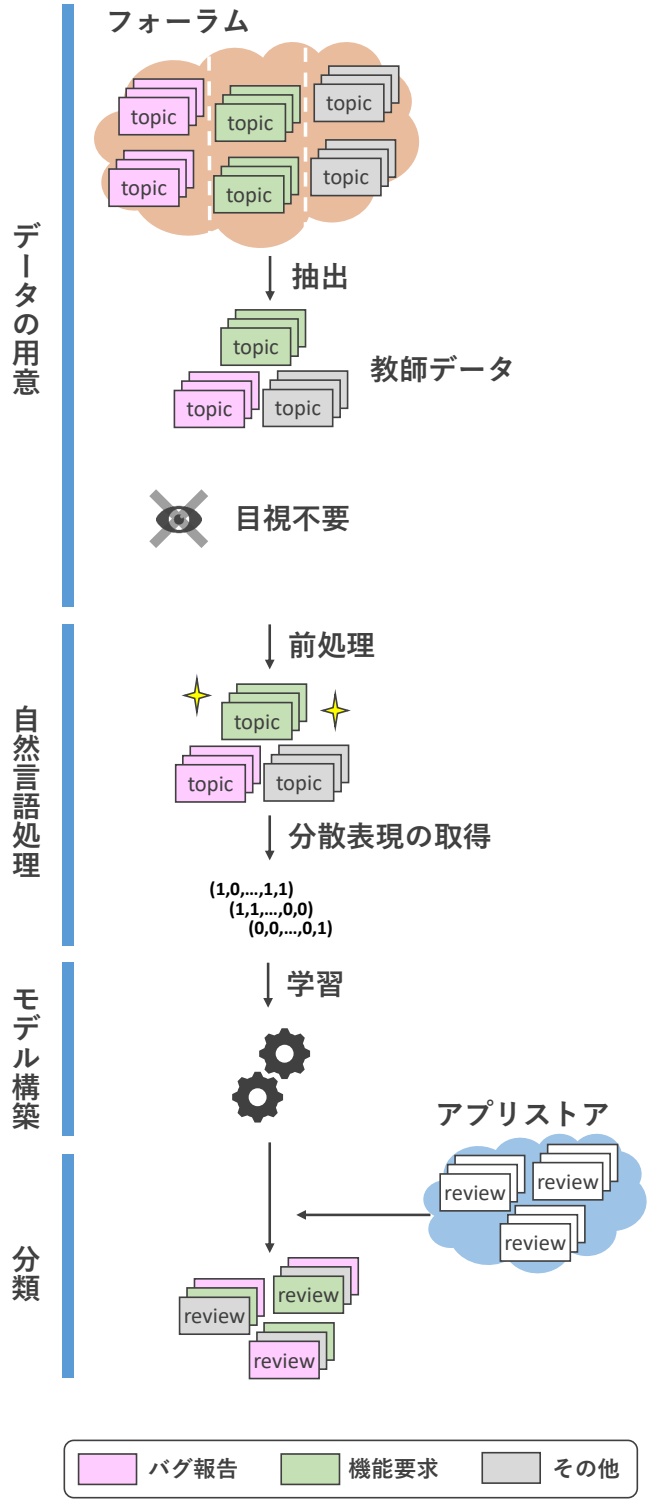

データの用意
フォーラム
topic
topic
topic
topic
topic
topic
抽出
topic
topic
topic
教師データ
目視不要
自然言語処理
前処理
topic
topic
topic
分散表現の取得
(1,0,...,1,1)
(1,1,...,0,0)
(0,0,...,0,1)
モデル構築
学習
分類
アプリストア
review
review
review
review
review
review
バグ報告
機能要求
その他

図 4　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 2 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．さらに，$F_{app}$ を教師データ，$R_{app}$ をテストデータとする提案手法を，$E(F_{app} \Rightarrow R_{app})$ 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，$E(R_{app} \Rightarrow R_{app})$ と表現できる．本 RQ では，これら 2 つの手法の分類精度を比較する．3 項で述べたように，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: フォーラムを持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 項で述べたように，フォーラムを持つアプリは多くはない．フォーラムを持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 $E(F_{app_A} \Rightarrow R_{app_A})$ 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 $E(F_{app_B} \Rightarrow R_{app_A})$ の分類精度を比較する．$E(F_{app_B} \Rightarrow R_{app_A})$ でも一定の精度が確認できれば，フォーラムを持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　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 項で述べたように，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 項のとおり，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持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持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持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[*4](Cities) と Euro Truck Simulator 2[*5](Euro) を本実験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[*6][*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．さらに，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” [5] 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” [5] 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．

[*4] https://store.steampowered.com/app/255710/Cities_Skylines/

[*5] https://store.steampowered.com/app/227300/Euro_Truck_Simulator_2/

[*6] https://forum.paradoxplaza.com/forum/forums/cities-skylines.859/

[*7] https://forum.scssoft.com/viewforum.php?f=3

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わかる．

表 1　実験データ

| | レビュー数 | | | | フォーラムのトピック数 | | | |
|---|---|---|---|---|---|---|---|---|
| | #バグ報告 | #機能要求 | #その他 | 合計 | #バグ報告 | #機能要求 | #一般議論 | 合計 |
| Cities: Skylines | 106(17) | 69(15) | 1,000 | 1,175 | 7,007 | 2,813 | 5,650 | 15,470 |
| Euro Truck Simulator 2 | 63(5) | 87(6) | 959 | 1,104 | 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．

### 5.2 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，自然言語の持つ曖昧さを削減し，分類精度が向上することが報告されている [5]．収集したレビュー，フォーラムに対して一般的な自然言語処理として，小文字化，Lemmatization，ストップワード除去を行った．これらの処理を施す例を図 5 に示す．“I bought this game yesterday and it is full of bugs!” という文章に対して一連の自然言語処理を施すことで，“buy game yesterday full of bug !” という文章が得られる．

#### 小文字化

大文字小文字の違いによる曖昧さを無くすために，文章全体を小文字に統一する．

#### Lemmatization

Lemmatization とは，同一の語彙素をレンマ（その語彙素を代表する基本形）に統一する処理である．例えば，“fixing”, “fixed”, “fixes” は全て “fix” に統一され，複数形は単数形に統一される．

#### ストップワード除去

“the”, “am”, “their” のような，文法的な機能を持ち，文の意味にあまり影響を与えない一般的な単語をストップワードとして除去する．表 2 に，本研究で用いたストップワードを示す．これらは Maalej ら [5] によって定義されたものである．

表 2　ストップワード

| |
|---|
| i, me, up, my, myself, we, our, ours, ourselves, you, your, yours, yourself, yourselves, he, him, his, himself, she, her, hers, herself, it, its, itself, this, they, them, their, theirs, themselves, am, is, are, a, an, the, and, in, out, on, up, down, s, t |

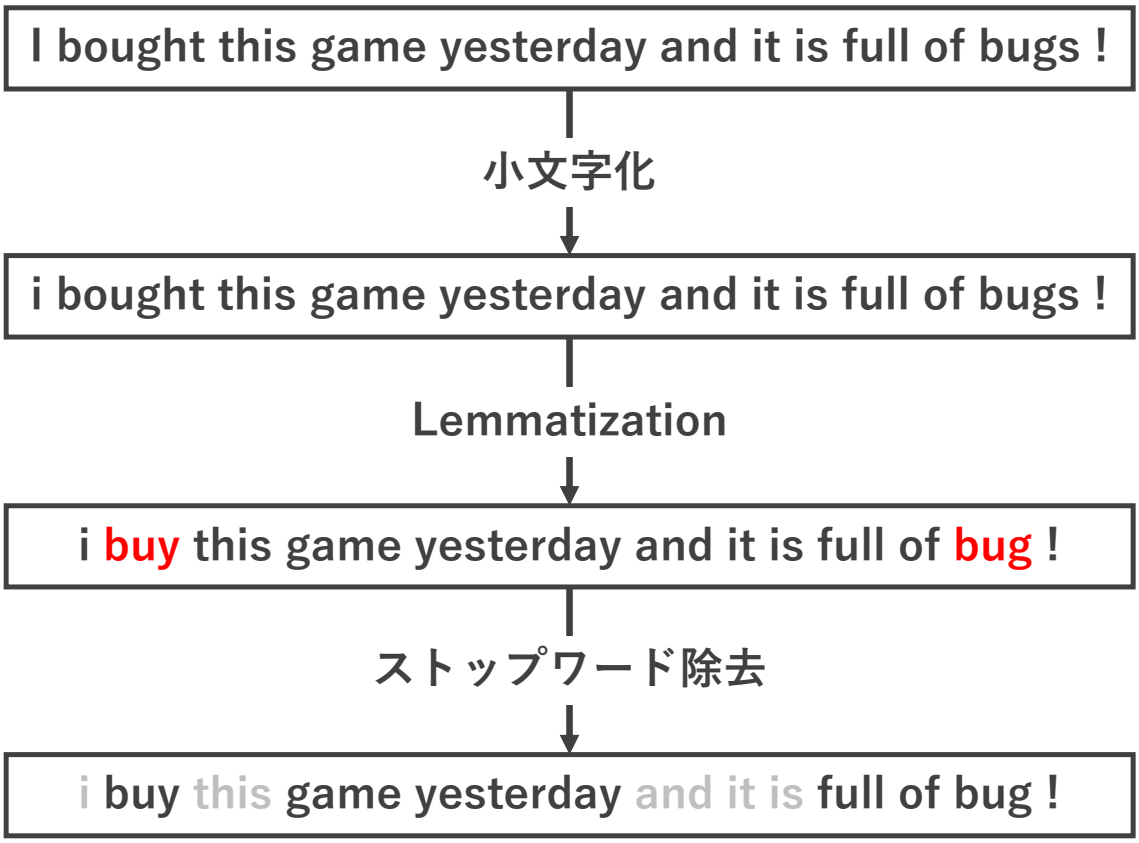



図 5　自然言語処理の流れ

### 5.3 分類モデル構築

本実験で用いる分類器は BERT (Bidirectional Encoder Representations from Transformers) [18] 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 [19]．

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，$R_{\mathrm{Cities}}$，$R_{\mathrm{Euro}}$，$F_{\mathrm{Cities}}$，$F_{\mathrm{Euro}}$ 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 モデルをファインチューニングして使用した．今回の実験で設定したハイパーパラメータを表 3 に示す．

表 3　ハイパーパラメータ

| | |
|---|---|
| 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 [20] |
| 学習率 | 3e-5 |

### 5.4 評価

生成した分類器を用いて，$R_{\text{Cities}}$ と $R_{\text{Euro}}$ を分類し，その精度を調査する．まず，既存手法に倣い $E(R_{\text{Cities}} \Rightarrow R_{\text{Cities}})$，$E(R_{\text{Euro}} \Rightarrow R_{\text{Euro}})$ を実施する．次に，RQ1 を調査するために提案手法として $E(F_{\text{Cities}} \Rightarrow R_{\text{Cities}})$，$E(F_{\text{Euro}} \Rightarrow R_{\text{Euro}})$ 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 $E(F_{\text{Euro}} \Rightarrow R_{\text{Cities}})$，$E(F_{\text{Cities}} \Rightarrow R_{\text{Euro}})$ 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 4 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，様々な閾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閾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．一般に，AUC が 1.0〜0.9 で高精度，0.9〜0.7 で中精度，0.7〜0.5 で低精度とされる [21].

## 6　実験結果

図 6-11 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 12 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．

| | | 予測した分類 | | | 適合率 precision | 再現率 recall | F値 F1-score | AUC |
|---|---|---|---|---|---|---|---|---|
| | | バグ報告 | 機能要求 | その他 | | | | |
| 真の分類 | バグ報告 | 23 | 1 | 2 | 0.74 | 0.88 | 0.81 | 0.99 |
| | 機能要求 | 2 | 10 | 10 | 0.45 | 0.45 | 0.45 | 0.94 |
| | その他 | 6 | 11 | 288 | 0.96 | 0.94 | 0.95 | 0.96 |

図 6　$E(R_{\mathrm{Cities}} \Rightarrow R_{\mathrm{Cities}})$ の実験結果

| | | 予測した分類 | | | 適合率 precision | 再現率 recall | F値 F1-score | AUC |
|---|---|---|---|---|---|---|---|---|
| | | バグ報告 | 機能要求 | その他 | | | | |
| 真の分類 | バグ報告 | 25 | 0 | 1 | 0.53 | 0.96 | 0.70 | 0.97 |
| | 機能要求 | 1 | 11 | 10 | 0.37 | 0.50 | 0.47 | 0.84 |
| | その他 | 21 | 19 | 265 | 0.96 | 0.87 | 0.91 | 0.91 |

図 7　$E(F_{\mathrm{Cities}} \Rightarrow R_{\mathrm{Cities}})$ の実験結果

| | | 予測した分類 | | | 適合率 precision | 再現率 recall | F値 F1-score | AUC |
|---|---|---|---|---|---|---|---|---|
| | | バグ報告 | 機能要求 | その他 | | | | |
| 真の分類 | バグ報告 | 14 | 1 | 11 | 0.50 | 0.54 | 0.52 | 0.89 |
| | 機能要求 | 1 | 15 | 6 | 0.26 | 0.68 | 0.38 | 0.83 |
| | その他 | 13 | 41 | 221 | 0.94 | 0.82 | 0.88 | 0.76 |

図 8　$E(F_{\mathrm{Euro}} \Rightarrow R_{\mathrm{Cities}})$ の実験結果

| | | 予測した分類 | | | 適合率 precision | 再現率 recall | F値 F1-score | AUC |
|---|---|---|---|---|---|---|---|---|
| | | バグ報告 | 機能要求 | その他 | | | | |
| 真の分類 | バグ報告 | 15 | 2 | 6 | 0.75 | 0.65 | 0.70 | 0.98 |
| | 機能要求 | 1 | 12 | 14 | 0.60 | 0.44 | 0.51 | 0.96 |
| | その他 | 4 | 6 | 272 | 0.93 | 0.96 | 0.95 | 0.97 |

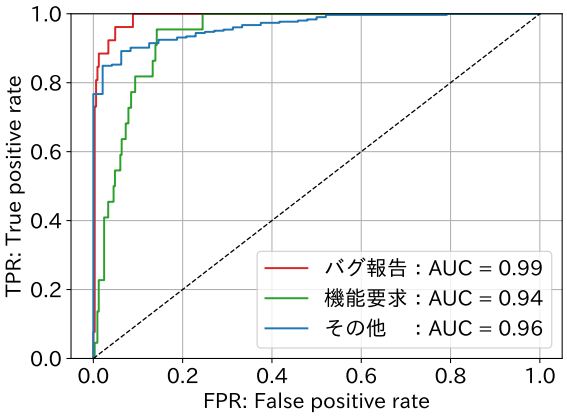
図 9　$E(R_{\mathrm{Euro}} \Rightarrow R_{\mathrm{Euro}})$ の実験結果

| | | 予測した分類 | | | 適合率 precision | 再現率 recall | F値 F1-score | AUC |
|---|---|---|---|---|---|---|---|---|
| | | バグ報告 | 機能要求 | その他 | | | | |
| 真の分類 | バグ報告 | 13 | 4 | 6 | 0.27 | 0.57 | 0.37 | 0.81 |
| | 機能要求 | 0 | 20 | 7 | 0.42 | 0.74 | 0.53 | 0.93 |
| | その他 | 35 | 24 | 223 | 0.94 | 0.79 | 0.86 | 0.83 |

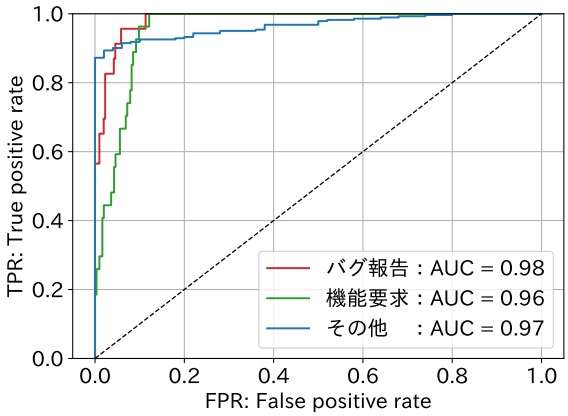
図 10　$E(F_{\mathrm{Euro}} \Rightarrow R_{\mathrm{Euro}})$ の実験結果

| | | 予測した分類 | | | 適合率 precision | 再現率 recall | F値 F1-score | AUC |
|---|---|---|---|---|---|---|---|---|
| | | バグ報告 | 機能要求 | その他 | | | | |
| 真の分類 | バグ報告 | 17 | 1 | 5 | 0.65 | 0.74 | 0.69 | 0.93 |
| | 機能要求 | 1 | 16 | 10 | 0.62 | 0.59 | 0.60 | 0.79 |
| | その他 | 8 | 9 | 265 | 0.95 | 0.94 | 0.94 | 0.85 |

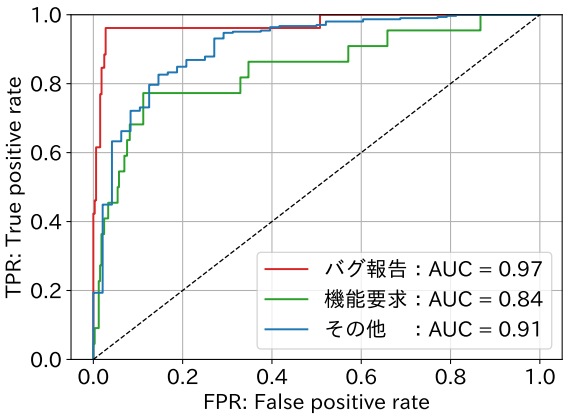
図 11　$E(F_{\mathrm{Cities}} \Rightarrow R_{\mathrm{Euro}})$ の実験結果



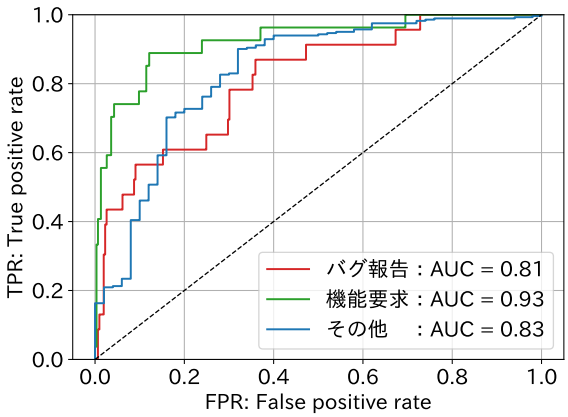

(a) $E(R_{\text{Cities}} \Rightarrow R_{\text{Cities}})$ の ROC 曲線



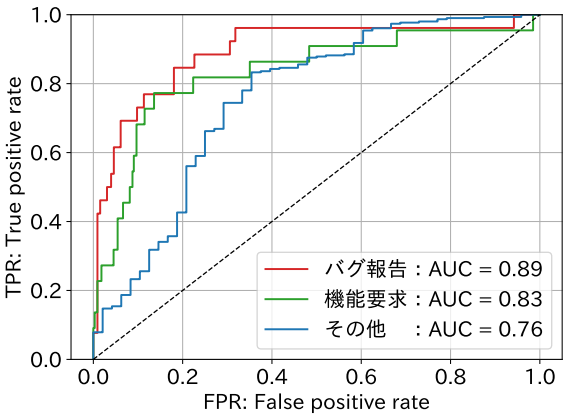

(d) $E(R_{\text{Euro}} \Rightarrow R_{\text{Euro}})$ の ROC 曲線



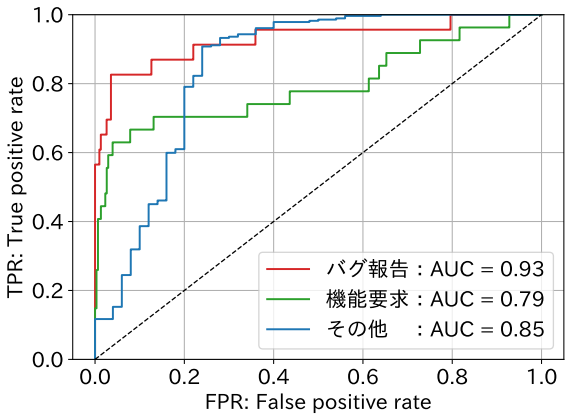

(b) $E(F_{\text{Cities}} \Rightarrow R_{\text{Cities}})$ の ROC 曲線




(e) $E(F_{\text{Euro}} \Rightarrow R_{\text{Euro}})$ の ROC 曲線




(c) $E(F_{\text{Euro}} \Rightarrow R_{\text{Cities}})$ の ROC 曲線




(f) $E(F_{\text{Cities}} \Rightarrow R_{\text{Cities}})$ の ROC 曲線

図 12　ROC 曲線

### 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，$R_{\text{Cities}}$ に対して既存手法を適用した場合 (図 6) と提案手法を適用した場合 (図 7) 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，非常に高い精度で分類ができている．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 以上であり，十分高い精度で分類ができているといえる．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，その他のレビューを誤ってバグ報告や機能要求に分類してしまった件数が多く，precision は既存手法に劣る結果となった．しかし，バグ報告と機能要求を正しく分類できた件数は既存手法よりも提案手法の方が多く，バグ報告と機能要求の recall は提案手法の方が優れている．レビューの自動分類では多くの場合，バグ報告や機能要求ではないレビューを誤ってバグ報告や機能要求であると分類してしまうよりも，バグ報告や機能要求に該当するレビューの見逃しを避ける方が重要である．そのため，precision よりも recall の方が重要であると考えられる [5]．また，$R_{\text{Euro}}$ に対して既存手法を適用した場合 (図 9) と提案手法を適用した場合 (図 10) についても同様の傾向が見られる．こちらは，既存手法では正しく分類できていたバグ報告を 2 件，誤って機能要求に分類してしまっているが，機能要求の recall は大きく向上している．これらの結果から，提案手法の有効性が確認できた．

**RQ1 の結論**：提案手法は既存手法には劣るものの高い精度で分類が可能である．特にバグ報告と機能要求の recall は既存手法と同等以上であった．

### 6.2 RQ2: フォーラムを持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，フォーラムを持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，$E(F_{\mathrm{Cities}} \Rightarrow R_{\mathrm{Cities}})$(図 7) と $E(F_{\mathrm{Euro}} \Rightarrow R_{\mathrm{Cities}})$(図 8) 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 以上であり，十分な精度で分類ができている．次に，$E(F_{\mathrm{Euro}} \Rightarrow R_{\mathrm{Euro}})$(図 10) と $E(F_{\mathrm{Cities}} \Rightarrow R_{\mathrm{Euro}})$(図 11) を比較する．AUC を見ると，別アプリのフォーラムを用いた場合，バグ報告の AUC は向上し，機能要求の AUC は低下している．AUC は全てのカテゴリーで 0.7 以上であり，十分な精度で分類ができている．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わかる．

**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十分な精度で分類が可能である．

## 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．

### 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 $E(F_{\mathrm{Euro}} \Rightarrow R_{\mathrm{Euro}})$ 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ている．$F_{\mathrm{Euro}}$ 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，$F_{\mathrm{Cities}}$ 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 節で述べた通り，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，フォーラムはヘー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，$E(F_{\mathrm{Cities}} \Rightarrow R_{\mathrm{Cities}})$(図 7) 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 19 件のうち 5 件が除外された．また，$E(F_{\mathrm{Euro}} \Rightarrow R_{\mathrm{Euro}})$(図 10) 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 35 件のうち 16 件が除外された．

### 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持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 3 の通り，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 13 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．

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 13　誤分類された文章量の多いレビューの例

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．1 つの長いレビューを分割して 2 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 1 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　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，著者と指導教官の 2 人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 2 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．

## 9　おわりに

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，十分な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ことが最優先の課題として挙げられる．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考えられる．

## 謝辞

本研究を行うにあたり，暖かく励まして頂きました，楠本 真二 教授に心より感謝申し上げます．

本研究に関して，的確で丁寧なご助言を頂きました，肥後 芳樹 准教授に深く感謝申し上げます．

本研究の全過程を通し，研究に対する方針や実現方法など，終始丁寧かつ熱心なご指導を賜りました，柗本 真佑 助教に心より感謝申し上げます．研究に行き詰まった際にも真摯に相談に乗っていただき，感謝の念に堪えません．本当にありがとうございました．

本研究を進めるにあたり，様々な形で励まし，協力を頂いたその他の楠本研究室の皆様に深く感謝申し上げます．

最後に，本研究に至るまでに講義等でお世話になりました大阪大学大学院情報科学研究科の諸先生方に，この場を借りて心より御礼申し上げます．

## 参考文献


[1] Adrian Holzer and Jan Ondrus. Mobile application market: A developer's perspective. *Telematics and Informatics*, Vol. 28, No. 1, pp. 22–31, 2011.

[2] Claudia Iacob and Rachel Harrison. Retrieving and analyzing mobile apps feature requests from online reviews. In *Proc. Working Conference on Mining Software Repositories*, pp. 41–44, 2013.

[3] Ning Chen, Jialiu Lin, Steven C. H. Hoi, Xiaokui Xiao, and Boshen Zhang. AR-Miner: Mining informative reviews for developers from mobile app marketplace. In *Proc. International Conference on Software Engineering*, pp. 767–778, 2014.

[4] Emitza Guzman, Muhammad El-Haliby, and Bernd Bruegge. Ensemble methods for app review classification: An approach for software evolution. In *Proc. International Conference on Automated Software Engineering*, pp. 771–776, 2015.

[5] Walid Maalej, Zijad Kurtanović, Hadeer Nabil, and Christoph Stanik. On the automatic classification of app reviews. *Requirements Engineering*, Vol. 21, No. 3, pp. 311–331, 2016.

[6] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado A. Visaggio, Gerardo Canfora, and Harald C. Gall. ARdoc: App reviews development oriented classifier. In *Proc. International Symposium on Foundations of Software Engineering*, pp. 1023–1027, 2016.

[7] Li Zhang, Xin-Yue Huang, Jing Jiang, and Ya-Kun Hu. CSLabel: An approach for labelling mobile app reviews. *Journal of Computer Science and Technology*, Vol. 32, pp. 1076–1089, 2017.

[8] Peng Zhenlian, Jian Wang, Keqing He, and Mingdong Tang. An approach of extracting feature requests from app reviews. In *Proc. Collaborate Computing: Networking, Applications and Worksharing*, pp. 312–323, 2017.

[9] Nishant Jha and Anas Mahmoud. Using frame semantics for classifying and summarizing application store reviews. *Empirical Software Engineering*, Vol. 23, pp. 3734–3767, 2018.

[10] Mohammadali Tavakoli, L. Zhao, Atefeh Heydari, and Goran Nenadic. Extracting useful software development information from mobile application reviews: A survey of intelligent mining techniques and tools. *Expert Systems with Applications*, Vol. 113, , 2018.

[11] Christoph Stanik, Marlo Häring, and W. Maalej. Classifying multilingual user feedback using traditional machine learning and deep learning. *International Requirements Engineering Conference Workshops*, pp. 220–226, 2019.

[12] Chong Wang, Peng Liang, Maya Daneva, and Marten van Sinderen. Augmenting app reviews with app changelogs: An approach for app reviews classification. In *Proc. International Conference on Software Engineering and Knowledge Engineering*, No. 31, pp. 398–403, 2019.

[13] Montassar Ben Messaoud, Ilyes Jenhani, Nermine Ben Jemaa, and Mohamed Wiem Mkaouer. A multi-label active learning approach for mobile app user review classification. In *Proc. Knowledge Science, Engineering and Management*, pp. 805–816, 2019.

[14] Naila Aslam, Waheed Yousuf Ramay, Kewen Xia, and Nadeem Sarwar. Convolutional neural network-based classification of app reviews. *IEEE Access*, Vol. 8, pp. 185619–185628, 2020.

[15] Furqan Rustam, Arif Mehmood, Muhammad Ahmad, Saleem Ullah, Dost Muhammad Khan, and Gyu Sang Choi. Classification of shopify app user reviews using novel multi text features. *IEEE Access*, Vol. 8, pp. 30234–30244, 2020.

[16] Marlo Häring, Christoph Stanik, and W. Maalej. Automatically matching bug reports with related app reviews. *International Conference on Software Engineering*, pp. 970–981, 2021.

[17] Dayi Lin, Cor-Paul Bezemer, Ying Zou, and Ahmed E. Hassan. An empirical study of game reviews on the steam platform. *Empirical Software Engineering*, Vol. 24, No. 1, pp. 170–207, 2019.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4171–4186, 2019.

[19] 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. 電子情報通信学会技術研究報告, Vol. 120, No. 423, pp. 25–30, 2021.

[20] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014.

[21] Joachim Fischer, Lucas Bachmann, and Roman Jaeschke. A readers' guide to the interpretation of diagnostic test properties: Clinical example of sepsis. *Intensive care medicine*, Vol. 29, pp. 1043–1051, 2003.