GitHub上のコード変更を利用したプログラミング演習問題の実用性の評価

松井 秀憲[1,a)]　矢谷 浩司[1,b)]

概要：学校や教科書で用いられるプログラミングの演習問題と，実際のソフトウェア開発業務に関する知識やスキルの間に隔たりがあることが知られている．先行研究では，GitHub のコード変更データをプログラミング演習問題に転用するシステムである RealCode が開発され，既存の学習環境にはない独自の演習問題を提供可能であることが明らかになった．しかし，RealCode が演習問題を生成する条件に検討の余地が残されていることに加え，演習問題から学べる内容と難易度についての評価・予測は行われていない．そこで，Python または JavaScript で開発を行った経験のある 15 人の学生，開発者により，RealCode が作成した演習問題の妥当性，学習できる内容，難易度についての評価を実施した．本論文では，集めたデータの詳細を報告するとともに演習問題の妥当性，難易度，学習内容の予測に寄与する可能性のある特徴量について議論する．

1. Introduction

ソフトウェア開発者の就業人口は 2018 年から 2028 年にかけて 21%増加すると予測されており [8]，全職種の平均と比べて遥かに早い速度で増加している．ソフトウェア開発者に必要なスキルを高めるための手段として，プログラミングの演習問題の重要性は認知されている [2]．しかしながら，学校や教科書で用いられるプログラミング演習問題は基礎的な文法やアルゴリズムの領域にとどまっており，実際のソフトウェア開発業務に関する知識やスキルとは隔たりがあることが知られている [1]．

先行研究では，GitHub のコード変更データをプログラミング演習問題に転用するシステムである RealCode が開発され，既存の学習環境にはない独自の演習問題を提供可能であることが明らかになった [4]．しかし，RealCode が演習問題を生成する条件に検討の余地が残されていることに加え，演習問題から学べる内容と難易度についての評価・予測は行われていない．

そこで，Python または JavaScript で開発の経験のある 15 人の学生，および開発者により，RealCode が作成した演習問題の妥当性，学習できる内容，難易度についての評価を実施した．さらに得られたデータセットを用いて，演習問題の妥当性，難易度，学習できる内容の考察を行った．本論文ではその実験結果と考察に関して報告する．

2. GitHub と既存研究 RealCode の概要

ソースコードのホスティングサービスである GitHub は，特定の組織内でのソースコード・プロジェクト管理に利用されるほか，オープンソースプロジェクト (OSP) のプラットフォームとして頻繁に利用されている．GitHub は 2018 年 10 月時点で 3,100 万人以上の開発者に利用され，プロジェクトが管理される単位であるリポジトリを 1 億件以上保有している[*1]．

GitHub での作業に重要な役割を果たすのが Issue と呼ばれる課題管理機能であり，リポジトリの拡張アイディアやソフトウェアバグの報告などの課題ごとに Issue が作成される．OSP 開発を行うモチベーションとしては，OSP へ貢献して自分のスキルを高めたい，普段使用している OSP を改良したい，など様々なものがある．ここではあるユーザが特定の OSP の Issue を解決したいと考えた場合を仮定し，それが解決されるまでのフローを考える（表 1）．この際，Pull request のコメントに Issue 番号を書くと Pull request と Issue がリンクされ，Pull request が受け入れられた時はリンクされた Issue が自動的に閉じられる仕組みになっている．そのため，閉じられた Issue とそれに紐づく Pull request 内の commit を見ることで，OSP で過去に

1 東京大学大学院 学際情報学府
Interactive Intelligent Systems Laboratory,
Graduate School of Interdisciplinary Information Studies、
The University of Tokyo
7-3-1 Hongo、Bunkyo-ku、Tokyo、Japan
a) matsui@iis-lab.org
b) koji@iis-lab.org

*1 https://github.blog/jp/2018-11-09-state-of-the-octoverse/

© 2020 Information Processing Society of Japan

存在していた課題とそれを解決したソースコード変更について学ぶことができる．

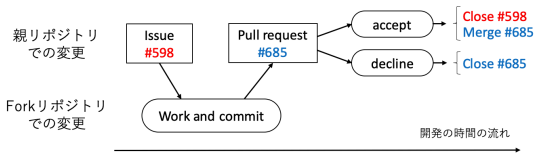

図 1: GitHub の OSP 開発のフローを表した図．Issue と Pull request がリンクされているため，OSP で存在していた課題とそれを解決したソースコード変更のデータを取得することができる．

Issue には作業の優先度や課題の種類などを示すラベルがつけられており，開発を効率的に進めるために役立っている．例えば，“bug” というラベルは Issue が予想外の問題や意図しない振る舞いを扱っていることを示し，“enhancement” は Issue が新しい機能のリクエストを扱っていることを示している[*2]．また，“good first issue” というラベルは OSP への最初のコントリビューションに適した Issue に付与され，外部の開発者が OSP へのコントリビューションを増やし，ソフトウェア開発者として成長するための手がかりとなる．

このように GitHub には，現実のソフトウェア開発における課題とそれを解決するソースコード変更のデータが豊富に含まれている．これに着目し，GitHub の公開リポジトリ上の Issue と，それに紐付く Pull request 内のコード変更を転用し、プログラミングを学ぶための演習問題を生成するシステムである RealCode が考案された [3, 4]. RealCode は，GitHub 上のリポジトリから取得した Issue の説明文を問題文，Issue がクローズされた際にマージされた Pull request に含まれるファイル変更（コード差分）を問題文に対する解答コード，Pull request の説明文を解答コードの説明文として転用する．先行研究においては JavaScript（336 件），Python（116 件）などのプログラミング言語の演習問題が生成された．

3. 本研究の方向性

3.1 本研究で構築した演習問題データセット

本研究は現実のソフトウェア開発業務に役立つプログラミングの知識の習得を支援するため，現在多く用いられているプログラミング言語を支援対象とする．2019 年 7 月から 9 月の間で，GitHub の Pull request で最も多く使用されているプログラミング言語は JavaScript (20.3%)，2 番目に多い言語は Python (17.1%) である[*3]ため，この 2 言語の演習問題により本研究のデータセットを構築する．

3.1.1 プログラミング学習に適した Issue の絞り込み条件

本研究ではより多くの問題からなるデータセットを構築するため，先行研究の Issue 収集に使用された Issue ラベルに追加して，新規の GitHub リポジトリを作成する際に提供されるデフォルトの Issue ラベル[*4]の中から “bug” と “enhancement” を追加した．その結果，Python588 件，JavaScript362 件からなる演習問題データセットを作成した．データセットで Issue を close した日が最も古い演習問題は，Python で 2014 年 1 月，JavaScript で 2015 年 9 月のものであった．

3.2 本研究の検討課題

先行研究においては，GitHub の Issue をプログラミングの演習問題として転用する可能性について議論が行われたが，演習問題から学べる内容や，演習問題の難易度についての評価は行われていない．そこで，本研究が目指す点は以下の 2 つである．なお，以降ではプログラミング学習に適する演習問題のことを妥当な問題と表記する．

1 妥当な問題である条件の議論の追究 RealCode においては，Issue に記載されているテキストや，Issue ラベルとして用いられているテキストなど，定性的な情報は妥当な問題の分類器に用いられていない．また定量的な情報については，コード変更行数や Issue 説明文の単語数が検討されているが，Issue に紐づくコメント数やリポジトリの Fork 数などの他の変数を検討する余地が残されている．本研究では，演習問題の定性的情報とより多種の定量的情報を用いて妥当性の予測を試みる．

2 学習者にとっての利便性向上の可能性の検討 RealCode においては，演習問題の独自性についての評価は行われているが，演習問題から学べる内容や，演習問題の難易度についての評価は行われていない．学習の目的によって問題が分類されていることと，平易な問題から順番に学べることは，この演習問題データセットを用いて学ぶ学習者の利便性に寄与すると考えられる．本研究では演習問題の学習内容と難易度の予測を試みる．

本研究では，1 のため改めて演習問題が妥当であるかどうかのデータを収集し，同時に 2 のために学習内容と難易度についてのデータも収集した．その詳細については次章で述べる．学習内容と難易度という 2 つのパラメータによって演習問題が整理されれば，目的に応じて無理なく学習を進められるようなインタフェースが実現可能となる．問題から学べる内容とその難易度を実務経験のある開発者がラベル付けすることで，これらを推定するモデルを構築することが本研究における目標である．

*2 https://help.github.com/ja/github/managing-your-work-on-github/about-labels
*3 https://madnight.github.io/githut/
*4 https://help.github.com/en/github/managing-your-work-on-github/about-labels

© 2020 Information Processing Society of Japan

4. プログラミング演習問題のラベル付け実験

前章までで，プログラミング演習問題計 950 件（Python 580 件，JavaScript 362 件）が作成された．本研究の目的は，Python と JavaScript の基礎的な文法は学習済みだが、まだソフトウェア開発の現場での使用経験はなく、ソフトウェア開発をする上で求められる実践的な知識を学びたい人を支援することである．そこで，ソフトウェア開発業務または研究において Python または JavaScript のプログラミング経験がある 15 名が評価することにより，各演習問題に対して妥当性，学習できる内容，難易度のデータを収集した．本章ではこれらの詳細について述べる．

4.1 本実験において収集したデータ

本実験で回答を収集した妥当性，学習内容，難易度についての質問項目を表 1 に示す．本節では，これらの質問項目の詳細について説明する．

表 1: プログラミング演習問題についての回答設問と，その回答形式または回答項目．

設問				回答形式・回答項目
Q1	この問題は，本システムの想定ユーザにとって有用だと思いますか？			プルダウンリスト
	上記設問で，全くそう思わない/そう思わないと答えた場合，その理由をすべて選択してください			
		QA-1	コード変更が断片的すぎて理解できない	チェックボックス
		QA-2	リポジトリ特有のライブラリ/変数などを使用しているため，コード変更が理解できない	チェックボックス
		QA-3	リファクタリングに関する知識である	チェックボックス
	他にあれば，自由に記述してください			自由記述
Q2	本システムの想定ユーザが，この問題から学ぶべき内容をすべて選択してください．			
	Web 開発者に求められる知識			
		QB-1	フロントエンドの知識	チェックボックス
		QB-2	バックエンド/サーバーサイドの知識	チェックボックス
	文法に関する知識			
		QC-1	エラー処理/例外処理に関する知識	チェックボックス，該当行数入力，具体記述
		QC-2	エラー処理/例外処理以外の文法に関する知識	チェックボックス，該当行数入力，具体記述
	その他のプログラミングに関する知識			
		QD-1	外部のライブラリ，フレームワーク，API の使用方法に関する知識	チェックボックス，該当行数入力，具体記述
		QD-2	自分が経験したことのあるバグに関する知識	チェックボックス，該当行数入力
		QD-3	ログ出力に関する知識	チェックボックス，該当行数入力，具体記述
		QD-4	データの取得・操作，データベースに関する知識	チェックボックス，該当行数入力，具体記述
		QD-5	アルゴリズムに関する知識	チェックボックス，該当行数入力，具体記述
	上記以外に学べる内容がある場合	QE		該当行数入力，具体記述
Q3	この問題のあなたにとっての難易度を教えてください．			プルダウンリスト
Q4	その他，この問題に対してのコメントが何かあれば記述してください．			自由記述

4.1.1 妥当性

コード変更のみを問題として演習問題が想定ユーザにとって有用だと思うかどうかを，5 段階の主観的評価（全くそう思わない – とてもそう思う）で実験参加者が選択する（Q1）．有用でない理由としては，コード変更がソースコード全体のごく一部となっており意味が理解できないことや，リポジトリ独自の変数を利用しており背景知識が必要な場合が考えられる．この場合，ファイル全体やリポジトリ全体の情報を探索するのは時間がかかるため，演習問題として効率的ではない．また，変数名やインデントの変更等のリファクタリングに関する知識も，これを支援するツールが存在するため相応しくない．例えば，Python のコーディング規約に従うようコードを整形するモジュールである autopep8[*5]などが存在する．実験参加者は，表 1 中 QA1 – QA3 の選択肢として提示された上記の理由に対し，当てはまる場合はチェックボックスを選択し，他にある場合は自由に記述する．

4.1.2 学習できる内容

ソフトウェア開発者の中の一分類として Web developers があり，これは専門性により Back-end web developers（バックエンド開発者）や Front-end web developers（フロントエンド開発者）に分類される．この 2 つの業務に求められる知識は異なるものである [8] ため，演習問題からこの 2 種類の業務に役立つ知識が学べると実験参加者が判断した場合，表 1 中 QB-1，QB-2 のチェックボックスを選択するようにした．

ソフトウェアを本番稼動させるためには，予め起こりうる例外への対策が大切であるが，一般的なプログラミング演習問題でこういった事柄はあまり扱われていない [10]．実験参加者が，本システムの問題からエラー処理または例外処理に関する知識が学べると判断した場合，QC-1 にチェックを入れるとともに，該当するコード変更の行数と，使用されているエラー型・例外型がある場合はその名称を含めて学べる内容を具体的に記述する．エラー処理・例外処理以外の文法に関する知識が学べると判断した場合は，QC-2 に同様の内容を記述する．

文法以外にもソフトウェア開発や研究の現場で求められるプログラミングの知識は多岐にわたるため，以下の 5 つの項目を用意した．実験参加者は，学習内容として当てはまるもの全てをチェックボックスにて選択した．

QD-1 プログラミング作業の能率を向上させるため，プログラミング言語そのものの機能とは別に，ライブラリ，フレームワーク，API などを活用する場面は多く存在する [6]．実験参加者は，当てはまる場合には該当する行数を入力するとともに，使用されているライブラリ，フレームワーク，API の名称を含めて学べる内容を具体的に記述する．

QD-2 多くの実験参加者が経験済みのバグを扱っている問題は，有用な問題である可能性が高い．実験参加者は，当てはまる場合には該当する行数を入力する．

QD-3 動作中のソフトウェアのログを取得しておくことは，本番稼働中のソフトウェア障害に迅速に対応する際に重要である [7]．実験参加者は，当てはまる場合

*5 https://pypi.org/project/autopep8/

には該当する行数を入力するとともに，使用されている関数，属性，クラス等の名称を含めて学べる内容を具体的に記述する．

QD-4 データサイエンスの訓練を受けた人材の需要が高まる一方で，データサイエンスを学べる機会は限られている [11]．データ構造やデータベースシステムの取り扱いについてのスキルは，データサイエンティストにとって重要なスキルである [5]．実験参加者は，当てはまる場合には該当する行数を入力するとともに，学べる内容を具体的に記述する．

QD-5 アルゴリズムに関する知識 アルゴリズムの選択と改良はプログラムデザインにおいて効果的であり [12]，アルゴリズムの知識はデータサイエンティストにとって重要なスキルの一つでもある [5]．実験参加者は，当てはまる場合には該当する行数を入力するとともに，学べる内容を具体的に記述する．

実験参加者は，QB-1 から QD-5 までの項目以外に学べる内容があると判断した場合，QE に学べる内容と該当する行数を記入する．

4.1.3 難易度やその他のコメント

問題の難易度を 5 段階評価（とても簡単 – とても難しい）で主観的に評価し（Q3），問題に関してその他のコメントがある場合は Q4 に記入する．

4.2 実験手順

4.2.1 収集用インタフェース

3.2 節で述べたように，本システムは RealCode を踏襲して，Issue のタイトルと説明文，Issue に紐づく Pull request 内のコード変更，Pull request のタイトルと説明文の 3 つをそれぞれ問題文，解答コード，解説文として提示する．本評価実験では，まずこれら 3 つの情報を図 2 のようにインタフェース上部に表示した．現在回答中の問題番号と，コード変更が含まれるファイルの名称も併せて表示した．ここでコード変更やイシュー説明文中のコードはコードブロックで囲ってあり，またコード変更中の追加行，削除行にはそれぞれ a3，d1 のように行番号を付けている．

インタフェース下部には表 1 に記載の設問に対して回答するフォームを設置した．

4.2.2 収集用インタフェースの実装

より多くの実験参加者を得るためには，時間的・物理的制約を極力排除して実験に参加しやすいインタフェースを用意することが望ましい．そこで，オンライン環境であればいつでも・どこからでも参加できるようなプログラミング演習問題の評価システムを構築した．具体的には，Amazon Web Services[*6] 上に Docker[*7] で仮想環境を構築し，プログラミング演習問題データベースと回答用 Web アプリケーションのデプロイを行った．プログラミング演習問題データと回答データの保管には MongoDB[*8]，Web アプリケーションの実装には React.js[*9] を使用した．

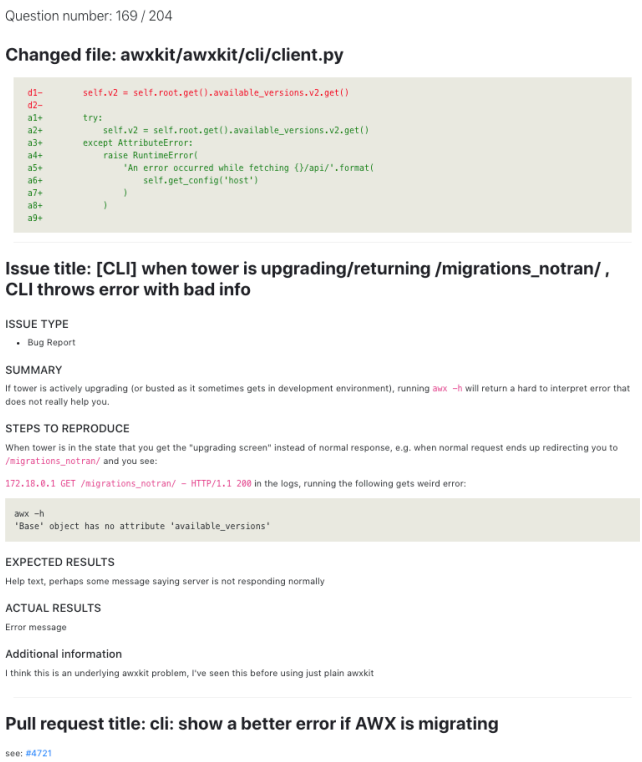

図 2: プログラミング演習問題の評価実験に用いられた Web アプリケーションのインタフェースの前半部分．現在回答中の問題番号，変更ファイルの名称・コード変更，Issue タイトル・説明文，Pull request タイトル・説明文からなる．

4.2.3 ラベル付け実験の参加者

Python または JavaScript をソフトウェア開発業務やデータ分析業務の用途で使用した経験のある参加者 15 名を SNS 上での募集により集めた．本実験では，データの質を担保するため演習問題 1 題につき 3 名が評価することとした．実験参加者には 3 週間の回答期間が与えられ，インターネット環境のある自由な場所で回答を行った．

表 2: 評価実験参加者の属性，回答したプログラミング演習問題の言語，その言語における研究・開発の経験について．

	学年・職業	回答言語	回答言語における経験
P1	修士課程 2 年	Python	コンパイラの作成，純 Python ライブラリの拡張機能の作成
P2	修士課程 1 年	Python	Web API の作成，データ分析，機械学習，Web スクレイピング
P3	修士課程 2 年	Python	実験データの解析，数値計算，機械学習
P4	修士課程 1 年	Python	商用エンターテイメントシステムのバックエンド開発
P5	学部 3 年生	Python	SNS 利用ユーザ間のネットワーク分析，Web アプリケーション開発
P6	修士課程 1 年	Python	センサ時系列データの分析
P7	修士課程 1 年	Python	脳波センサの制御と脳波データの処理，家賃予測モデルの構築
P8	学部 4 年生	Python	画像処理を活用した実験データ作成とデータ分析
P9	修士課程 1 年	Python	チャットボット，地理情報活用システム，予約管理システムの開発
P10	Web エンジニア	JavaScript	Web アプリケーション開発，社内業務ツール作成
P11	学部 3 年生	JavaScript	Node.js を用いたバックエンド開発
P12	Web エンジニア	JavaScript	フロントエンド，サーバーサイド，マイクロサービスの開発
P13	Web エンジニア	JavaScript	動的ウェブサイト作成
P14	Web エンジニア	JavaScript	Web アプリケーションの開発
P15	修士課程 1 年	JavaScript	趣味のゲーム開発，大学の課題のための簡単な AI 作成

*6 https://aws.amazon.com/
*7 https://www.docker.com/
*8 https://www.mongodb.com/
*9 https://reactjs.org/

© 2020 Information Processing Society of Japan

4.3 データ収集結果

1 つの演習問題について 3 名がラベル付けを行っており，3 名が付けた妥当性の平均値は図 3 のような分布になっている．ここでは全く有用だと思わないを 1，とても有用だと思うを 5 のように計算しており，平均値が 3.0 より大きい演習問題を妥当な演習問題であるとみなした．妥当な演習問題は，Python で 181 件（31.2%），JavaScript で 93 件（25.7%）存在した．妥当な問題のうち，演習問題の難易度の平均値の分布と，1 名でも該当するとラベル付けした課題内容の頻度分布は，それぞれ図 4，5 のようになっている．

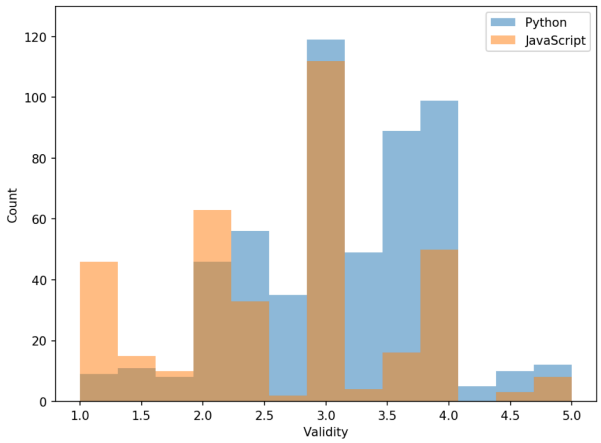

図 3: 3 名の評価者がつけた演習問題の妥当性の平均値の分布．

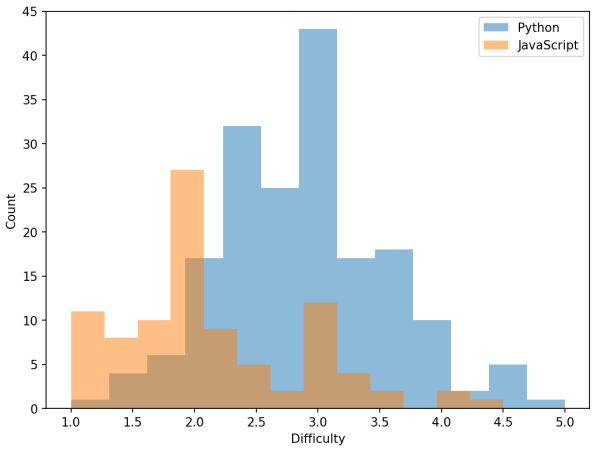

図 4: 妥当だと判断された演習問題のうち，3 名の評価者がつけた難易度の平均値の分布．

5. プログラミング演習問題の妥当性・難易度・課題内容の予測可能性検討

前章で収集したデータセットを用いて，演習問題の妥当性・難易度・学習内容の予測に有益となる可能性がある特徴量の検討を行った．本章では，検討した特徴量と予測結果の詳細について述べる．

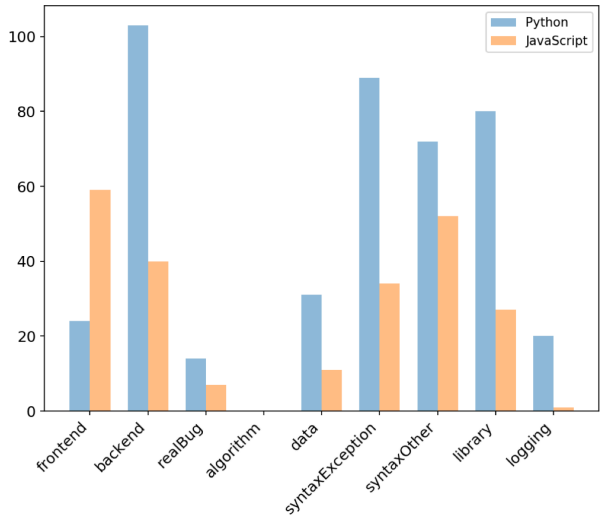

図 5: 妥当だと判断された演習問題のうち，1 名でも該当するとラベル付けした課題内容の頻度分布．

5.1 特徴量

本予測では，コード変更や Issue 本文等の回答インタフェース上に問題として表示されていた情報（問題情報）と合わせて，表示はされていないが問題が作成された Issue 等に付与されている属性情報を利用する（表 3）．また，単語数，ラベル数，コメント数といった定量的な情報の他に，テキストやコード片に含まれる定性的な情報も用いて予測を行う．定性的な情報は，該当する文章に出現する単語の TF-IDF 値 [13] によって表現する．

コード変更が存在するファイル名は，例えば “src/python/WMCore/WMLogging.py” のようにスラッシュにより区切られており，ディレクトリの深さの情報も含まれる．スラッシュの区切りを単語とみなし，この単語数を特徴量として採用する．コード変更やテキストにおいては，アルファベットまたはアンダーバーからなる単位を単語とみなした．また，Issue や Pull request に紐づくコメントは該当プロジェクトに貢献する [9] ため，これらの数はどの程度 OSS 開発が活発であったかを示す指標となる．リポジトリ全体の OSS 活動の活発さの指標となるリポジトリの Fork 数と合わせて，活発に開発が行われている Issue，Pull request，リポジトリから生成された問題であることと，問題の評価に関係があるのかを検証する必要がある．以下，妥当性，難易度，学習内容それぞれについて，特徴量がどの程度貢献する可能性があるかを述べる．

5.2 妥当性の予測可能性の検討

各特徴量に対して，2 標本の等分散性を仮定せず平均値の差の検定を行う Welch の t 検定を用いて，分類における寄与度が高い可能性がある特徴量の抽出を行った．

表 4 にコードに関する特徴量として検討した特徴量を示す．Python または JavaScript の予約語として決められて

表 3: 演習問題の妥当性，難易度，学習内容を予測するための特徴量の候補.

			定量情報	定性情報
問題情報	コード変更	ファイル名	単語数	出現単語の TF-IDF 値
		コード	追加行の単語数 追加行に出現する予約語の数　等	追加行の出現単語の TF-IDF 値 追加行に出現する予約語の TF-IDF 値　等
	Issue	タイトル	単語数	出現単語の TF-IDF 値
		本文	コード片とエラーメッセージの単語数 テキスト部分の単語数	コード片とエラーメッセージの単語の TF-IDF 値 テキスト部分の出現単語の TF-IDF 値
	Pull request	タイトル	単語数	出現単語の TF-IDF 値
		コメント	単語数	出現単語の TF-IDF 値
属性情報	Issue	ラベル	付与されている数	出現単語の TF-IDF 値
		その他	Issue が開いていた時間 Issue が閉じた日時 Issue へのコメントの数	
	Pull request		Pull request へのコメントの数	
	リポジトリ	トピック	付与されている数	出現単語の TF-IDF 値
		その他	Fork 数 Issue ラベル総数 使用プログラミング言語数	

いる単語は，文法上重要な意味を持つため，全単語数とは分けて語数を算出し特徴量とした．また，追加行と削除行を合わせた全変更行についての情報や，追加行と削除行の差に関する情報も検証した．

表 4: コード変更に関する特徴量．チェックマークが付いている特徴量は，妥当性があると判断された問題群とそうでない問題群との間に有意差があったことを示す．チェックマーク 1 つの特徴量は，JavaScript の演習問題のみで有意差があると判断されたことを表し，チェックマーク 2 つの特徴量は Python と JavaScript の両方で有意差があると判断されたことを表している．

	全変更行	追加行	削除行	追加行内の数と削除行内の数の差	削除行になく追加行のみにある数
全単語					
予約語			✓	✓✓	✓

問題情報の特徴量について Welch の t 検定を行った結果を表 5 に示す．Python の演習問題では，コード追加行と削除行に現れる予約語の数の差のみが有意な差を示した．また JavaScript において同様に Welch の t 検定を行った結果，コード削除行に現れる予約後の数，コード追加行と削除行に現れる予約語の数の差，コード削除行になく追加行のみにある単語のうち予約語の数，Issue 本文のコード片またはエラーメッセージの単語数に有意差が認められた．

一方で，属性情報の特徴量について Welch の t 検定を行った結果，Python と JavaScript の演習問題の両者において有意差は認められなかった（表 6）．

5.3 難易度の予測

演習問題の難易度を予測するにあたり，まずは使用する特徴量の絞り込みを行った．演習問題の特徴量全 28 個のうち，1 個を用いて難易度を予測する単回帰モデルを構築し，予測性能の上位 10 個の特徴量を得た．その後，10 個の特徴量のうち説明変数として用いる特徴量の組み合わせを変えながら，難易度の予測性能を比較した．難易度予測性能比較は，回帰モデルとして Ridge regression を用い，妥当な演習問題群内での 5-fold cross validation により行った．評価指標には Mean squared error (MSE) を用いた．

Python の演習問題についての結果を表 7 に示す．上位

表 5: Python，JavaScript の妥当な問題群と妥当でない問題群それぞれにおける，問題情報の各特徴量の平均値と，Welch の t 検定による有意差の判定．*は p 値が 0.05 未満であることを示す．

	特徴量	Python 妥当群 平均	Python 非妥当群 平均	Python p 値	JavaScript 妥当群 平均	JavaScript 非妥当群 平均	JavaScript p 値
コード変更	変更ファイル名の単語数	3.02	2.99	0.7773	3.35	3.39	0.8571
	全コード変更行の単語数	51.92	50.74	0.7714	61.42	47.78	0.416
	全コード変更行に現れる予約語の数	7.54	6.49	0.1223	6.48	6.65	0.8386
	コード追加行の単語数	35.72	33.7	0.4505	41.95	32.52	0.3021
	コード追加行に現れる予約語の数	5.44	4.51	0.0532	4.87	4.37	0.4296
	コード削除行の単語数	16.2	17.04	0.6599	19.47	15.26	0.6052
	コード削除行に現れる予約語の数	2.1	1.98	0.6721	1.61	2.28	0.048*
	コード追加行と削除行の単語数の差	19.51	16.66	0.2055	22.47	17.26	0.2265
	コード追加行と削除行に現れる予約語の数の差	3.34	2.53	0.0446 *	3.26	2.09	0.0419*
	コード削除行になく追加行のみにある単語の数	19.15	18.35	0.6799	21.3	18.1	0.4328
	コード削除行になく追加行のみにある単語のうち予約語の数	3.03	2.66	0.2475	3.03	1.7	0.0123*
Issue	Issue タイトルの単語数	7.73	7.53	0.4574	8.45	7.96	0.2652
	Issue 本文のコード片/エラーメッセージの単語数	221.91	114.98	0.1483	20.94	48.8	0.0206*
	Issue 本文のテキストの単語数	84.03	96.07	0.0696	137.08	126.58	0.4636
PR	Pull request タイトルの単語数	7.2	6.94	0.3282	7.28	6.87	0.2538
	Pull request 本文の単語数	50.66	48.78	0.7674	56.44	48.59	0.5854

表 6: Python，JavaScript の妥当な問題群と妥当でない問題群それぞれにおける，属性情報の各特徴量の平均値と，Welch の t 検定による有意差の判定.

	特徴量	Python 妥当群-平均	Python 非妥当群-平均	Python p 値	JavaScript 妥当群-平均	JavaScript 非妥当群-平均	JavaScript p 値
Issue	Issue に付与されているラベルの数	1.87	1.87	0.9552	1.86	1.97	0.3992
	Issue へのコメント数	2.48	2.7	0.3905	2.23	2.68	0.1196
	Issue が開いていた時間	23:09:29	18:44:48	0.1811	11:59:39	13:19:53	0.3739
	Issue が閉じた日時	2019/4/20	2019/4/16	0.8639	2019/3/24	2019/4/16	0.3977
PR	Pull request へのコメント数	1.87	1.71	0.4818	1.23	1.1	0.5111
リポジトリ	リポジトリに付与されているトピック数	4.5	4.23	0.5505	4.39	4.24	0.8313
	リポジトリの Fork 数	336.27	365.06	0.7165	725.3	605.52	0.735
	リポジトリで使用されている Issue ラベルの総数	23.77	21.59	0.4158	20.66	20.58	0.98
	リポジトリで使用されているプログラミング言語の総数	4.33	4.42	0.8211	3.44	3.54	0.7705

10 個の特徴量は，左からそれぞれコード追加行の単語数，全コード変更行の単語数，コード削除行になく追加行のみにある単語の数，コード追加行と削除行の単語数の差，コード追加行の予約語数，全コード変更行の予約語数，Pull request タイトルの tf-idf，コード追加行と削除行の予約語数の差，Issue 本文のテキスト部分の単語数，Issue 本文のコード片・エラーメッセージの tf-idf である．このうち，コード追加行の単語数と Issue 本文のテキスト部分の単語数を組み合わせて用いた時が，最も良い予測性能が得られた．

表 7: Python の演習問題の難易度予測結果．コード追加行の単語数と Issue 本文のテキスト部分の単語数を組み合わせて用いた時が，最も良い予測性能が得られた．

add_ keyword	add_ only_ word	add_ word	all_ keyword	all_ word	diff_ keyword	diff_ word	issue_ text	tfidf_ issue_ error	tfidf_ pullreq_ title	MSE
		✓					✓			0.447
		✓			✓		✓			0.448
✓		✓					✓			0.449
	✓			✓	✓		✓			0.45
	✓			✓			✓			0.45

JavaScript の演習問題についての結果は表 8 のように

なった．上位 10 個の特徴量は，左からそれぞれコード追加行の予約語数，全コード変更行の予約語数，コード削除行になく追加行のみにある単語の数，コード追加行と削除行の単語数の差，コード削除行になく追加行のみにある予約語の数，コード追加行と削除行の予約語数の差，ファイル名の tf-idf，コード削除行の予約語数，コード追加行の予約語の tf-idf，Pull request タイトルの tf-idf である．このうち，コード削除行になく追加行のみにある予約語数，コード削除行の予約語数，ファイル名の tf-idf，Pull reqeuest タイトルの tf-idf を組み合わせて用いた時が，最も良い予測性能が得られた．

表 8: JavaScript の演習問題の難易度予測結果．コード削除行になく追加行のみにある予約語数，コード削除行の予約語数，ファイル名の tf-idf，Pull reqeuest タイトルの tf-idf を組み合わせて用いた時が，最も良い予測性能が得られた．

add_ keyword	add_ only_ keyword	add_ word	all_ keyword	diff_ keyword	diff_ word	rem_ keyword	tfidf_ add_ keyword	tfidf_ file_ name	tfidf_ pullreq_ title	MSE
	✓					✓		✓	✓	0.3
✓	✓		✓			✓		✓	✓	0.302
✓	✓					✓		✓	✓	0.302
	✓		✓	✓		✓		✓	✓	0.302
	✓		✓	✓				✓	✓	0.302

5.4 課題内容の予測可能性の検討

前章までで，演習問題に対して 8 つの学習内容が学べるかどうかの評価を収集した．8 つの学習内容について，それが学べると分類された演習問題群と，学べないと分類された演習問題群を分割し，それぞれにつけられている Issue ラベルの単語のうち TF-IDF 値が高い上位 50 単語を抽出した．この 50 単語のうち，学べる群と，学べない群それぞれのみに出現した単語の上位 5 つを抜き出したものが表 9 である．データの取得・操作，データベースに関する知識（data）が学べる問題群には “ipython” が，フロントエンド開発に関する知識が学べる問題群には “ux” といった特徴的な Issue ラベルが利用されている．このように，Issue の目的や優先順位を整理するためにつけられる Issue ラベルに出現する単語により，演習問題の学習内容が分類できる可能性が示された．

6. 考察

6.1 実験結果の考察

6.1.1 妥当度に関する考察

コード変更においては，予約語の数に有意差が認められたが，追加行ではなく，削除行や，追加行と削除行の両方を比較した場合に有意差が認められた．Python の結果からは，予約語が含まれないコードを削除し予約語が含まれるコードに置き換えるようなコード変更が，妥当な問題と判断されていると考えられる．しかし JavaScript においては，コード削除行に含まれる予約語の数が妥当群より非妥当群の方で多くなっており，Python とは逆の傾向を示している．これは，冗長な記述を簡潔にするようなコード変更が多く含まれ，そのようなコード変更が妥当とみなされていると考えられる．

表 9: 各課題内容が学べると分類された演習問題と，学べないと分類された演習問題それぞれに含まれる Issue ラベルの単語のうち TF-IDF 値が高い上位 50 単語から，それぞれのみに出現する単語の上位 5 つを抜き出したもの．

課題内容	学べる問題の Issue ラベル	学べない問題の Issue ラベル
frontend	hal, branch, only, ux, low	critical, moonshot, up, 2019, wmagent
backend	critical, ipython, 2019, patch, low	friday, locked, moonshot, user, newcomer
realBug	ipython, hal, 2019, low, salt	moonshot, user, newcomer, pr, crash
data	ipython, friendly, 2019, patch, low	timers, newcomer, pr, crash, status
syntaxException	crash, 2019, patch, low, fix	moonshot, timers, user, newcomer, status
syntaxOther	user, crash, 2019, patch, low	moonshot, newcomer, important, windows, ui
library	user, crash, 2019, low, fixed	newcomer, status, important, windows, ui
logging	user, crash, 2019, low, fixed	error, moonshot, newcomer, important, windows

また，Issue 本文に含まれるコード片やエラーメッセージの単語数とテキストの単語数も，Python と JavaScript で大きく異なる傾向にあった．まず，JavaScript の Issue は，Python の Issue に比べると Issue 本文に含まれるコード片やエラーメッセージの割合が少なく，それ以外のテキストの割合が大きい．これは，Python では長いエラーメッセージやソースコードをそのまま Issue に記載している一方で，JavaScript では $< pre >$ タグで囲まれるエラーメッセージやソースコードを極力少なくし，自然言語による説明が重視されているといえる．そのため，JavaScript においては Issue 本文中のソースコードやエラーメッセージが少ないほど，理解しやすく妥当な問題になる傾向があると考えられる．

6.2 難易度に関する考察

難易度に関しては，予測精度向上に寄与する特徴量がわかった．Issue 本文のテキスト部分の単語数が長いものは，問題を説明するためにより詳細な説明が必要であると言えるため，変更の難易度が上がると想定される．またファイル名や Pull request のタイトルは，コード変更に関連する単語を含むため，難易度との関連性があると想定される．したがって，この結果はある程度の妥当性を持つと考えられる．

一方で，検討に使用したモデル選択はまだ限定的であり，今後は Ridge regression 以外のモデルを用いた予測結果を検討する必要があると考えられる．また，Python と JavaScript の間に生じた差異についての検討も今後の課題である．

6.2.1 課題内容に関する考察

課題内容に関しては特徴的な Issue ラベルが存在することがわかった．特にフロントエンドに関係するものは，“ux” などという，直接関係する単語が出現していること

が確認された．一方で，複数の課題内容に出現するラベルもいくつかあり（例えば “low”），これらがどの程度学習内容を同定する上で有用かは今後の課題である．

また，“2019” といったラベルが出現していることも確認できた．これは 2019 年に発生したイシュー，あるいは 2019 年に関係するイシューを指すものと考えられ，一般的に課題内容を抽出するうえで役に立つラベルとは言い難い．今後の研究ではこのようなラベルをどのように自動的に排除し，課題内容の分類を行うかが重要となる．

6.3 本研究の結果・考察に関する注意点

本研究における結果，および考察において注意すべき点がいくつかある．まず，データのラベル付けが主に大学院生，大学生によって行われた点である．このため，プログラミングを職業とするプロフェッショナルによる評価と乖離が生じる可能性がある．今後の研究では，より高度な知識を持つ評価者によってデータの見直しを行うことで，本研究の結果の信頼性をより高めることができる．

また，プログラミング言語として Python と JavaScript のみしか本研究では扱っておらず，他言語においては本研究で検討されていない特徴量が重要な情報となり得ることがある．一方で，イシューに記載されている情報など，プログラミング言語に依存しない特徴量については，プログラミング言語が変わっても利用できる可能性があり，今後の研究でさらに検証されるべきである．

本研究では実際に分析された演習課題の教育的効果に関しては評価していない．既存研究においても RealCode が，一般的な教科書やチュートリアルとは違った演習課題を提供できる可能性を示しているが，本研究はそれを再確認するものではない．一方，本研究により，演習問題としての妥当性や難易度を定量的に推定できる可能性が示されたため，教育効果を検証する今後の研究において，本研究の知見が有益に働くことが期待される．

7. 結論

本研究では，GitHub 上に存在するコード変更を利用し，プログラミング課題に転用するための基礎技術の 1 つとして，コード変更がどの程度演習問題として適しているか，その難しさはどれくらいか，またどのような課題内容か，を推測する上で有用となりうる特徴量の検討を行った．本研究の結果を受けて，今後の研究では転用された演習内容を用いることでどのような学習効果が起きるかを検証する実験につながると考えられる．

謝辞

本研究の一部は，科研費若手研究「コードの理解と教育における GitHub のプルリクエスト情報の有効性の検証」によって支援されました．

参考文献

[1] Allen, E., Cartwright, R. and Reis, C.: Production Programming in the Classroom, *SIGCSE Bull.*, Vol. 35, No. 1, pp. 89–93 (online), DOI: 10.1145/792548.611940 (2003).

[2] Caiza, J. C. and Álamo Ramiro, J. M. d.: Programming Assignments Automatic Grading: Review of Tools and Implementations (2013).

[3] Daisuke, S.: GitHub 上のコード変更を利用したプログラミング演習問題システムの研究, Master’s thesis, 東京大学工学系研究科 (2019).

[4] Daisuke, S. and Koji, Y.: GitHub のプルリクエストを用いたプログラミング課題自動生成システムの実現可能性に関する検討, **第 *80* 回全国大会講演論文集**, Vol. 2018, No. 1, pp. 319–320 (2018).

[5] Dhar, V.: Data Science and Prediction, *NYU Working Paper*, No. No. 2451/31635 (online), DOI: https://ssrn.com/abstract=2071041 (2012).

[6] Guerrouj, L., Bourque, D. and Rigby, P. C.: Leveraging Informal Documentation to Summarize Classes and Methods in Context, *Proceedings of the 37th International Conference on Software Engineering - Volume 2*, ICSE ’15, Piscataway, NJ, USA, IEEE Press, pp. 639–642 (online), available from ⟨http://dl.acm.org/citation.cfm?id=2819009.2819125⟩ (2015).

[7] Kernighan, B. W. and Pike, R.: *The practice of programming*, Addison-Wesley Professional (1999).

[8] of Labor Statistics, U. B.: Computer and Information Technology Occupations : Occupational Outlook Handbook. U.S. Bureau of Labor Statistics, `https://www.bls.gov/ooh/computer-and-information-technology/home.htm` (2019). (Accessed on 01/08/2020).

[9] Onoue, S., Hata, H. and Matsumoto, K.-i.: A Study of the Characteristics of Developers’ Activities in GitHub, *2013 20th Asia-Pacific Software Engineering Conference (APSEC)*, Vol. 2, pp. 7–12 (online), DOI: 10.1109/APSEC.2013.104 (2013).

[10] Piteira, M. and Costa, C.: Learning Computer Programming: Study of Difficulties in Learning Programming, *Proceedings of the 2013 International Conference on Information Systems and Design of Communication*, ISDOC ’13, New York, NY, USA, ACM, pp. 75–80 (online), DOI: 10.1145/2503859.2503871 (2013).

[11] Strawn, G.: Data Scientist, *IT Professional*, Vol. 18, No. 3, pp. 55–57 (online), DOI: 10.1109/MITP.2016.41 (2016).

[12] Wirth, N.: Algorithms + Data Structures = Programs, *Prentice Hall, Englewood Cliffs*, Vol. 15, pp. 253–265 (1975).

[13] Wu, H. C., Luk, R. W. P., Wong, K. F. and Kwok, K. L.: Interpreting TF-IDF Term Weights as Making Relevance Decisions, *ACM Trans. Inf. Syst.*, Vol. 26, No. 3 (online), DOI: 10.1145/1361684.1361686 (2008).

© 2020 Information Processing Society of Japan