Vol.2015-SE-187 No.16
2015/3/12

コーディングスタイルを基にした初心者プログラマ向けの著者解析

福本 大[1]　北川 裕基[1]　玉田 春昭[1]

概要：大学の講義において，他人からプログラムをコピーして提出する学生がいる．教員はコピーしたプログラムを提出してきた学生に対して，不正行為とみなし落第点をつける可能性がある．このとき，コピー元のオリジナルのプログラムを書いた学生も減点されるといった不益を被るため，好ましくない．本研究では，講義程度のプログラムを対象として，プログラムのオリジナルの開発者がどの学生なのか推定する著者解析を行う．大学の講義では事前にサンプルプログラムが提示されることが多いため，アルゴリズムを特徴として用いても個人間で差が出にくいと考えられる．そのため，本研究ではコーディングスタイルの特徴を用いる．また，効率的に開発者の推定を行うために，教師あり機械学習を用いる．事前に提示されたサンプルコードの影響を受けたために，平均正答率の最大は 36.6%であった．

1. はじめに

大学の講義課題において，自身で作成し提出すべきプログラムを他人からコピーして提出を行う学生がいる．このとき，教員はコピーと考えられるプログラムを提出してきた学生に対して，不正行為とみなし落第点をつける可能性が考えられる．この場合，そのオリジナルのプログラムを書いた学生は減点されるといった不利益を被る．この様な状況は，オリジナルのプログラムを書いた学生にとっては好ましくない．そこで本論文では大学の講義課題で提出されてきたプログラムを対象としてプログラムのオリジナルの開発者が一体どの学生なのかを推定する著者解析を行う．著者解析を行うことにより，教員はプログラムのオリジナルの学生を知ることができる．これにより，教員はコピーと考えられるプログラムを提出してきた学生の中から，コピー元のプログラムを書いたオリジナルの学生以外に対して落第点をつけることができる．そして，オリジナルの学生は減点されるといった不利益を被ることがなくなる．

また著者解析に使うメトリクスはプログラムの書き方によるコーディングスタイルの特徴量を用いる．この特徴量を用いる理由として，対象が大学の講義程度のプログラムであるために，事前にある程度のプログラムの挙動が決められている．そのため，プログラムに使用されているアルゴリズムを特徴として利用しても，学生毎に差が出にくいと考えられるためである．

本研究では，開発者を効率的に推定するために，機械学習を用いる．人間が学習する過程をコンピュータ上で再現したものを機械学習という．開発者の推定は開発者が判明しているプログラムからコーディングスタイルの特徴量を抽出し，それらを機械学習に入力し学習を行う．次に開発者が未知のプログラムから抽出した特徴量を機械学習に入力し予測を行う．その後，開発者が未知のプログラムを書いたと思われる開発者が出力される．この処理を各課題で行い，開発者の推定を行う著者解析の正答率を確認する．

2. 関連研究

永井らは機械学習を用いてプログラムを分類した [1]．またプログラムの分類には表層的な特徴であるコーディングスタイルを特徴量として用いている．また対象は C, C++ 言語で書かれたプログラムで，比較的大規模なプロジェクトで書かれたプログラムとしている．結果，永井らは表層的特徴であるコーディングスタイルの特徴量を基に，およそ 95%の精度で機械学習を用いてプログラムの分類に成功している．

武田らは初心者プログラマを対象とし，コーディングスタイルを用いてソースコードの盗用検出を行った [2]．実験において，本論文と同様に学生の演習課題を対象として盗用検出を行っている．コーディングスタイルは永井らが提案したインデントやコメント，演算子等の特徴を用いている．結果，武田らはコーディングスタイルを基に，演習課題の盗用検出に成功している．

[1] 京都産業大学コンピュータ理工学部, 603-8555 京都市北区上賀茂本山, Faculty of Computer Science and Engineering, Kyoto Sangyo University.

Eugene らは，実行コードとソースコードの断片から開発者の識別を行った [3]．対象の開発者は，コンピュータウィルスやトロイの木馬等を作成した開発者である．しかし，これらのソフトウェアを作る開発者は，初心者プログラマとは言い難い．

HaibiaoDing らは，Java 言語で書かれたソースコードを対象に開発者の識別を行った [4]．コーディングスタイルの特徴量を用いて，統計的に開発者の識別を行い成功している．しかし，この研究で用いられた特徴量は Java 言語のみを対象としている．また識別を統計的に行っているため，本研究が提案している内容と異なる．

これらの研究では，本研究と同じく，メトリクスにコーディングスタイルの特徴量を用いている．しかし，本研究での対象は，大学の講義での課題で書かれたプログラムである．すなわち，大規模なプロジェクトで書かれたプログラムと比較すると小規模なプログラムである．また，本研究は盗用発見を目的とするのではなく，当該プログラムの原著者を推定することが目的である..

3. 提案手法

本研究はプログラムの開発者の推定をする著者解析が目的である．開発者の推定には教師あり機械学習を用いて著者解析を行う．教師あり機械学習を用いた理由として，開発者の推定を効率的に行うためである．

3.1 著者解析

著者解析は，あるプログラムを作成した人物を推定することである．また解析に用いる情報は，プログラム内から得られる情報を使用する．本研究では，プログラムから得られる情報であるコーディングスタイルの特徴量を基に，このプログラムを作成した開発者を推定する．コーディングスタイルの特徴量を利用した理由として，対象が大学講義の課題であるために，プログラムの挙動がある程度事前に決められている．そのため，プログラムに用いられているアルゴリズムを比較しても，学生毎に差が出にくいと考えられるためである．

3.2 コーディングスタイルの特徴量

永井らは，永井ら自身で提案しているコーディングスタイルの特徴量 56 種類を用いて，C, C++言語で書かれた大規模なプロジェクトの分類に成功している [1]．また武田は，同じ特徴量を用いて大学講義程度でのソースコード盗用検出に成功している [2]．このことから，本研究でも永井らが提案しているコーディングスタイルの特徴量を用いる．特徴量は 56 種類存在する．それら 56 種類は次の 5 つの特徴量群に分類出来る．

インデント，ファイル　ファイル全体の見た目の特徴である，「プログラムの行の長さの平均・分散」,「インデントの平均・分散」等が含まれる．多くのプログラムはスコープレベルが増える毎にインデントの文字数が大きくなる．そこでこのスコープレベルの比を取り，どれだけ一貫したスコープレベルとインデントの文字数の関係を取る．

括弧　括弧はブロックを区別する要素で，重要であると考えられる．「(の前後に改行がある割合」,「) の前後に改行がある割合」等が含まれる．

コメント　利用の割合や形には様々なものがあるが，コメントの内容までは本研究では取り扱わない．利用の割合や形であるコメントのバイト数等を取る．

演算子　演算子の種類毎に全演算子に対する割合，書き手の開発者の特徴と考えられる演算子の前後の空白や平均を取る．またファイル全体での演算子の行あたりの出現頻度も取る．

定義された変数，関数，クラスの名前　変数や関数，クラスの名前には開発者の嗜好が強く出ると考えられる．そこで,「名前の長さの平均・分散」,「大文字のみ，もしくは小文字のみの割合」等を取る．

3.3 単純ベイズ分類器

単純ベイズ分類器は教師あり機械学習の 1 つである．単純ベイズ分類器は学習と予測の 2 つの過程がある．

本研究では，単純ベイズ分類器を用いる．単純ベイズ分類器を採用した理由として，学習が高速で実用的な精度を持ち合わせているためである．

3.3.1 学習過程

学習過程は，開発者とその開発者に対応する特徴量を示したデータを入力する．そのデータを基に，単純ベイズ分類器内に開発者に対応するクラスを生成する．

学習過程では，プログラムから抽出したコーディングスタイルの特徴量がどの開発者に対応しているかを示したデータを用意する．このデータを複数用意して，まとめたものを学習データと呼ぶ．

単純ベイズ分類器では，学習データを受け取り開発者毎にクラスを生成する．それぞれのクラスはその開発者に対応する特徴量を基にして生成される．

3.3.2 予測過程

予測過程は，どのクラスに属するのかが未知である特徴量を示したデータを入力する．そのデータを確率に基づいて，どのクラスに属するのかを予測する．

予測過程では，学習過程を既に行った単純ベイズ分類器を用いる．また，プログラムから抽出したコーディングスタイルの特徴量を示したデータを用意する．このデータをテストデータと呼ぶ．

テストデータを単純ベイズ分類器に入力する．単純ベイズ分類器では，単純ベイズ分類器内で生成されているクラスを基にテストデータに含まれる特徴量がそれぞれのクラ

スに属する確率が計算される．単純ベイズ分類器は，その確率が最も高いクラスを予測結果として出力する．

3.4 著者解析手法

本研究では著者解析に単純ベイズ分類器を用いる．単純ベイズ分類器への入力は永井らが提案しているプログラムのコーディングスタイルの特徴量を用いる [1]．コーディングスタイルの特徴量を基に学習を行うとそのプログラムの開発者 1 人につき対応するクラスが 1 つ作成される．その後，開発者が未知のプログラムのコーディングスタイルの特徴量を学習済みの単純ベイズ分類器に入力する．学習済みの単純ベイズ分類器では，既に作成されている全クラスに対して，そのコーディングスタイルの特徴量がそれぞれのクラスに属する確率が計算される．それぞれの確率が最も高いクラスに対応する開発者が出力される．

著者解析の流れは次の 3 つから構成される．

特徴量抽出フェイズ プログラムからコーディングスタイルの特徴量を抽出する．

学習フェイズ コーディングスタイルの特徴量を抽出してから単純ベイズ分類器に学習する．

著者解析フェイズ 学習済みの単純ベイズ分類器に開発者が未知のプログラムのコーディングスタイルの特徴量を入力して，開発者を推定する著者解析を行う．

これらを図 1，図 2，図 3 を用いて順に説明する．

3.4.1 特徴量抽出フェイズ

プログラムからコーディングスタイルの特徴量を抽出する過程を，図 1 を用いて説明する．

プログラム P から n 種類のコーディングスタイルの特徴量を抽出する．それらを特徴量 F_1 から特徴量 F_n とする．n 種類の特徴量を m 個のカテゴリに分類する．それらを m 個のカテゴリに分類してまとめたものを特徴量群 Q_1 から特徴量群 Q_m とする．

3.4.2 学習フェイズ

開発者が既知のプログラムからコーディングスタイルの特徴量を抽出する．その特徴量から単純ベイズ分類器に学習させる過程を，図 2 を用いて説明する．図 2 中の (a) では，開発者 H_A が作成したプログラム P_{A1} からコーディングスタイルの特徴量を抽出する．抽出した n 種類の特徴量を分類してまとめたものを特徴量群 $Q_{A1i}(1 \leq i \leq m)$ とする．次に，(b) において，m 個の特徴量群から任意の特徴量群を 1 つ選択する．図 2 に示す例では，特徴量群 Q_{A11} を選択する．選択した特徴量群 Q_{A11} を単純ベイズ分類器に入力する．そして (c) では，入力された特徴量群 Q_{A11} を基に学習を行い，プログラム P_{A1} の開発者 H_A に対応するクラス C_A が生成される．この処理を開発者 H_B，H_C が作成したプログラム P_{B1}，P_{C1} に対しても行う．そして開発者 H_B，H_C に対応するクラス C_B，C_C が生成される．この流れを学習という．

3.4.3 著者解析フェイズ

開発者が未知のプログラムのオリジナルの開発者を推定する著者解析を行う過程を，図 3 を用いて説明する．入力データにはプログラムのコーディングスタイルの特徴量を用いる．

また，ここでは開発者 H_A, H_B, H_C が作成したプログラムから抽出した特徴量を基に学習を行う．そして，開発者に対応するクラス C_A, C_B, C_C は既に生成されているものとする．(d) では，不明な開発者 H_X が作成したプログラム P_{X1} から，学習時と同様に n 種類の特徴量を抽出する．抽出した特徴量を分類してまとめたものを特徴量群 $Q_{X1i}(1 \leq i \leq m)$ とする．(e) において，m 個の特徴量群から任意の特徴量群を 1 つ選択する．図 3 に示す例では，特徴量群 Q_{X11} を選択する．その特徴量群 Q_{X11} を学習済みの単純ベイズ分類器に入力する．そして (f) では，入力された特

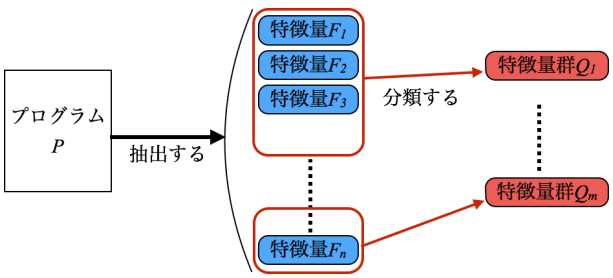

図 1 特徴量抽出フェイズのプロセス

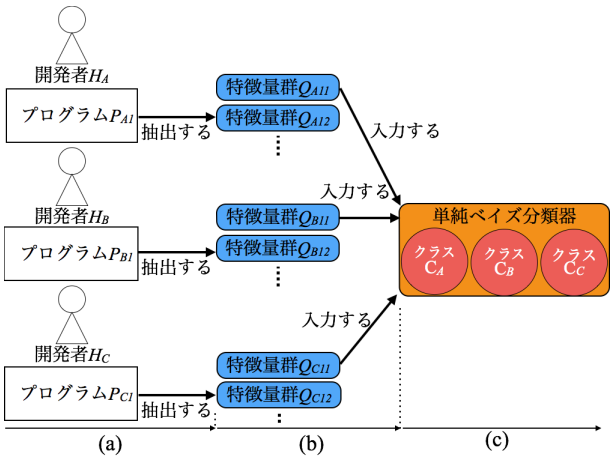

図 2 学習フェイズのプロセス

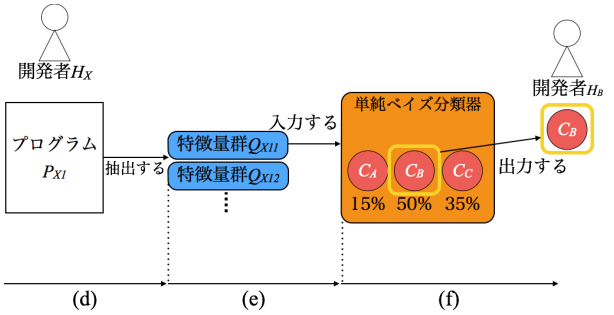

図 3 著者解析フェイズのプロセス

表 1 著者解析の結果

		学習データ		
		H_A	H_B	H_C
テストデータ	H_{X1}	1	0	0
	H_{X2}	0	0	1
	H_{X3}	0	0	1

徴量群 Q_{X11} を基に，単純ベイズ分類器内に生成されているそれぞれのクラス C_A, C_B, C_C に属する確率を計算する．

図 3 に示した例では，特徴量群 Q_{X11} がクラス C_B に属する確率が 50%と一番高かった．この結果から単純ベイズ分類器は，プログラム P_X の開発者をクラス C_B に対応する開発者 H_B と推定する．そして開発者 H_B として出力する．

3.5 適用例

本節では第 3.4 節で説明した 3 つのフェイズを実際のデータを用いて説明する．本節で用いたデータに含まれる開発者数は 3 名である．それぞれを開発者 H_A, H_B, H_C とする．これらの開発者はランダムに選択した．学習データは，それぞれの開発者が書いたプログラムの特徴量を用いる．そして，n 種類の特徴量を m 個の特徴量群に分類する．この適用例では m 個の特徴量群から 1 つを用いることとする．テストデータは，開発者 H_A, H_B, H_C が書いた課題 22 のプログラムの特徴量を用いる．そして，n 種類の特徴量を m 個の特徴量群に分類する．この適用例では m 個の特徴量群から 1 つを用いることとする．

これらの学習データとテストデータを使い，単純ベイズ分類器を用いて著者解析を行った結果を表 1 に示す．

表 1 の行はテストデータである．列は単純ベイズ分類器内に既に生成されている開発者に対応するクラスである．表 1 では，テストデータの開発者 H_{X1} を開発者 H_A として推定した回数が 1 回である．他のテストデータ H_{X2}, H_{X3} は，H_C として，それぞれ 1 回ずつ推定している．実際には，テストデータ H_{X1} は H_A である．これは，正しく推定できている．また，テストデータ H_{X3} は H_C である．これも正しく推定できている．しかし，テストデータ H_{X2} は実際には H_B であるが，単純ベイズ分類器は H_C として推定している．これは誤った推定である．

4. ケーススタディ

4.1 概要

本章では，実際に，実データを用いて，プログラムのコーディングスタイルの特徴量抽出の方法から著者の推定を試行した．対象とするプログラムの言語は，対象が Java の講義であるため，Java 言語とする．

オリジナルの開発者が判明している Java プログラム毎に特徴量を抽出する．それらを特徴量群に分類する．Java プログラムから抽出された特徴量群を学習データとする．また開発者が未知の Java プログラムの特徴量も同様に抽出し，これをテストデータとする．学習データを単純ベイズ分類器に入力し，開発者毎のクラスを生成する．学習済みの単純ベイズ分類器にテストデータを入力し，テストデータの特徴量群がそれぞれのクラスに属する確率を得る．一番高い確率が出力されたクラスに対応する開発者が，テストデータの特徴量群の抽出元であるプログラムを書いた開発者と言える．

4.2 実験データ

本研究では大学 2 年次生を対象とした Java プログラム演習の出題課題で提出されたプログラムを用いる．出題課題数は 22 個，提出されたプログラム数は 2,266 ファイルであった．また全課題のうち 1 課題以上でプログラムを提出した学生は 178 名いた．これらは Java 言語で書かれたものであり，プログラムの平均行数は約 30 行と比較的小規模なものである．

4.3 実験

収集された実験データを用いて，次の 4 種類の実験を行う．そして，それぞれの正答率を確認する．正答率はテストデータの開発者と単純ベイズ分類器が出力した開発者が一致したときを正解とし，その合計をテストデータ数で割った結果とする．正答率の計算をテストデータ毎に行い，この平均を平均正答率とする．

(1) 全学生を対象にした著者解析
  (a) 特徴量群でクラスを作成
  (b) 特徴量でクラスを作成
(2) 提出率 80%以上の学生を対象にした著者解析
  (a) 特徴量群でクラスを作成
  (b) 特徴量でクラスを作成
(3) 全学生を対象に特徴量を絞った著者解析
(4) 提出率 80%以上の学生を対象に特徴量を絞った著者解析

最初に，全学生を対象として著者解析を行い，正答率を確認する．その際，56 種類全ての特徴量を用いてクラスを作った場合と，5 つの特徴量群を対象にクラスを作った場合で結果に差が出るかを確認する．

一方，全学生の中には提出数が少ない学生も含まれている．これらのデータがノイズとなり，正答率を下げる要因になると考えられる．そこで全学生の中で，課題の提出率 80%以上の学生に絞ったときの著者解析を行う．提出率が高い学生に絞ることにより，学生 1 人あたりの課題毎のデータが充実する．また提出率が高い学生はプログラムの書き方に差が表れると考えられる．そのため，正答率が上がることが期待される．なお，この実験も (1) と同じく，全ての特徴量を用いた場合と，特徴量群を用いた場合の 2 種類の実験を行う．

また，武田らは次の 3 つの特徴量において，ソースコー

ド整形ツールによる改竄の有無に関わらず盗用関係にあるソースコードと盗用関係にないソースコードで差があることを確認している [2].

- スコープレベルの平均とインデントの差の平均
- 全演算子に対する二項代入演算子の出現割合
- 演算子の行あたりの出現回数

武田らは盗用発見が目的であるため，著者解析を目的とする本稿の目的とは異なる．しかし武田が確認した 3 つの特徴量を特徴量群とし，著者解析を行ったとき，どのような結果が出力されるのかを確認する．

学習データは出題課題 22 個の内，21 個の課題で提出されたプログラムから抽出した特徴量を用いる．テストデータは学習データで用いていない残りの 1 個の課題で提出されたプログラムから抽出した特徴量を用いる．テストデータとして用いる課題が課題 1 から課題 22 で 1 回ずつ用いられるように 22 回試行する．

4.3.1 全学生を対象にした著者解析

この実験では全学生を対象に著者解析を行う．そのため，テストデータのプログラムを書いた本来の開発者と別の開発者が同じ確率で出力される可能性がある．それぞれがどの程度の正答率になるか確認する．特徴量を 5 つの特徴量群に分けて著者解析を行ったときの正答率を図 4 に，特徴量を 5 つの特徴量群に分けずに著者解析を行ったときの正答率を図 5 に示す．

この 2 つの実験では，共に平均正答率は 5%未満であった．これは学習データが多いため過学習が起き，学習データに偏った分類器になったと考えられる．そのため，平均正答率は 5%未満であり著者解析には不適である．

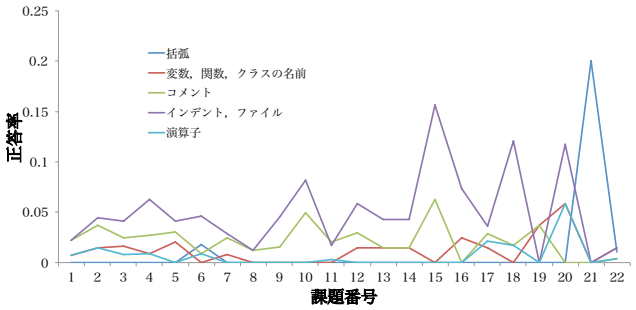

図 4 特徴量を 5 つの特徴量群に分けて著者解析を行ったときの正答率

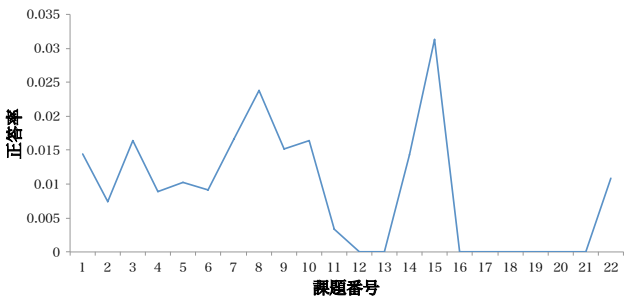

図 5 特徴量を 5 つの特徴量群に分けずに著者解析を行ったときの正答率

4.3.2 提出率 80%以上の学生を対象にした著者解析

この実験では，課題提出率が 80%以上の学生が書いたプログラムを対象とする．また特徴量を 5 つの特徴量群に分けたときと分けなかったときの結果を確認する．提出率が 80%以上の学生は全学生 178 名中で 11 名であった．11 名が書いたプログラムの平均行数は 40 行であった．この実験では，学生 1 人 1 人の学習データが充実している．また提出率が高い学生が書いたプログラムは，書き方に差が出ると考えられる．そのため，全学生 178 名の特徴量を学習させたときと比べると，本来の開発者が推定される可能性が高まり，正答率が高くなることが期待される．

特徴量を 5 つの特徴量群に分けて著者解析を行ったときの正答率を図 6 に，特徴量を 5 つの特徴量群に分けずに著者解析を行ったときの正答率を図 7 に示す．

この実験では，学生数を絞って実験を行った．平均正答率は全学生を対象にした実験と比べると上がっている．平均正答率が最も高かった特徴量群は，インデント，ファイルに関する特徴量であり，約 37%であった．それに次いで，コメントに関する特徴量が約 33%であった．これはインデント，ファイルに関する特徴量では，ファイルの行の長さの平均等を抽出している．またコメントに関する特徴量では，コメントのバイト数や行数の平均を抽出している．学

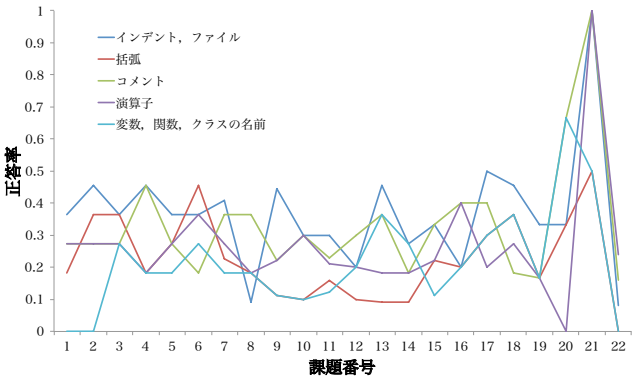

図 6 提出率 80%以上のプログラムの特徴量を 5 つに分けたときの正答率

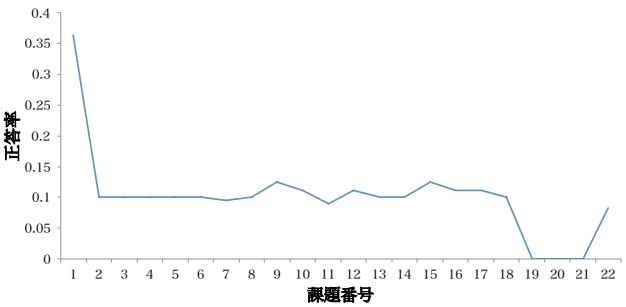

図 7 提出率 80%以上のプログラムの特徴量を 5 つに分けなかったときの正答率

生の中には，メソッドやメソッド内の処理毎にコメントを書く学生と全く書かない学生がいる．コメントを書く学生はファイルの行の長さも多くなり，コメントを書かない学生は短くなると考えられる．そのため，この 2 つの特徴量群は残りの 3 つの特徴量群よりも平均正答率が高くなったと考えられる．

それに対し，平均正答率が最も低かった特徴量群は，変数，関数，クラスの名前に関する特徴量で，約 22%であった．これは対象が大学講義程度で書かれたプログラムである．そのため，変数や関数，クラスの名前は，事前に提示されたサンプルプログラムの影響を大きく受け，それぞれの学生が同じような名前にする傾向があるためと考えられる．その結果，平均正答率が低くなったと考えられる．

課題 21 においては 5 つの特徴量群の内，3 つの特徴量群で正答率が 100%を示している．これは課題 21 のテストケースが 2 つしかなかったためである．そのため，課題 21 の結果は信頼出来るとは言い難い．しかし，提出数が多い学生はプログラムの書き方に差が出ると考えられる．それにより，正答率が高くなっていると考えられる．

また, 課題 17 から課題 21 は追加課題であり，課題 21 に進むにつれて提出数が少なくなっていった．それに伴い，特徴量を 5 つの特徴量群に分けたときの正答率は増加傾向にある．このことから，オリジナルのプログラムのコピーをテストデータとして用いる等，テストデータを絞り込んだとき，コーディングスタイルの特徴量から単純ベイズ分類器を用いて著者解析を行えると考えられる．

この実験では学生数を絞り実験を行った．その結果，全学生のデータを著者解析に用いたときの実験結果と比較すると，正答率が上がることが確認できた．そこで，更にデータを絞り込むことにより，正答率が上がると期待できる．

4.3.3 全学生を対象に特徴量を絞った著者解析

このときの正答率を図 8 に示す．

この実験では，平均正答率は 2%程であった．用いる特徴量数は先ほどの実験で用いた特徴量数よりも減ったが，学生数は変わらない．結果，この実験においても過学習が起き，学習データに偏った分類器となったと考えられる．そのため，平均正答率は 2%程であり著者解析には不適である．

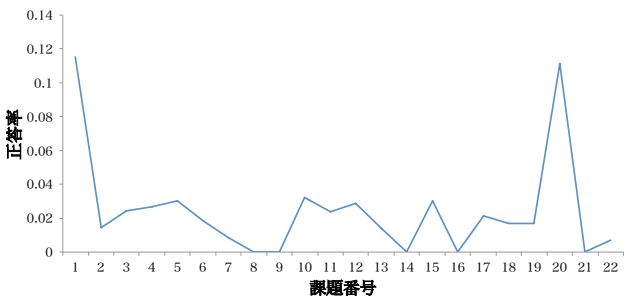

図 8 特徴量を絞って著者解析を行ったときの正答率

全学生を対象にして著者解析を行ったとき，学習データが多いために過学習が起きたと考えられる．これにより，用いる特徴量を絞っても学生数に変化がないために正答率は上がることはなかった．正答率を上げるために，学習データの絞り込みが必要であると考えられる．

4.3.4 提出率 80%以上の学生を対象に特徴量を絞った著者解析

この実験では，課題提出率が 80%以上の学生を対象とする．また用いる特徴量は，武田が確認した 3 つの特徴量に絞って実験を行う．この実験では，全学生を対象として武田が確認した特徴量を用いた実験よりも，データに含まれる学生数を絞っている．そのため，全学生のデータを用いた実験の結果よりも，高い正答率を得ることが期待される．実験結果を図 9 に示す．

この実験では，対象とする学生数と用いる特徴量を絞って実験を行った．課題 1 から課題 22 の平均正答率は約 22%と特に良い平均正答率ではなかった．ただし，追加課題である課題 17 から課題 21 においての平均正答率は約 32%であった．対して，図 8 の全学生を対象としたときの課題 17 から課題 21 の平均正答率は約 3%であった．このことからテストデータに用いるプログラム数と特徴量を絞ると，平均正答率が上がると期待される．これはテストデータを絞ったとき，一部のコーディングスタイルの特徴量から単純ベイズ分類器を用いて著者解析が行えると考えられる．

5. 議論

実験結果より，提出された全プログラムから学習データとテストデータを作成し学習・テストを行ったとき，全ての平均正答率は 5%未満であった．これは学習データが多いため過学習が起き，学習データに偏った分類器となり，テストデータを入力しても平均正答率は 5%未満と著者解析には不適である．また，単純ベイズ分類器はテストデータとして入力されたプログラムの特徴量が開発者のクラスに属する確率が全体的に低いときでも，その中で一番高い確率のクラスを出力する．そのため，誤認識が増え，正答率が下がったと考えられる．

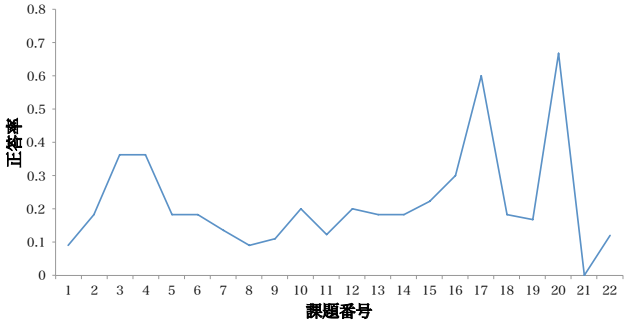

図 9 提出率 80%以上の学生を対象に特徴量を絞ったときの正答率

そこで，対象を提出率が 80%以上の学生が作成したプログラムの特徴量を用いて学習・テストを行った．特徴量を 5 つの特徴量群に分けて学習・テストを行った．

5.1 サンプルコードの影響を受けやすい特徴量

提出率が高い学生が書いたプログラムの特徴量群を用いて，学習・テストを行ったとき，平均正答率が最も低かった特徴量群は変数，関数，クラスの名前で約 22%となった．これは対象が大学講義程度で書かれたプログラムであるために，変数，関数，クラスの名前は，事前に提示されたサンプルプログラムの影響を大きく受け，それぞれの学生が同じような名前にする傾向があるため平均正答率が低くなったと考えられる．この特徴量群は，サンプルコードの影響を受けやすい特徴量であると考えられる．そのため，この特徴量群は，大学講義程度で書かれたプログラムを対象とした著者解析には不適であると考えられる．

5.2 サンプルコードの影響を受けにくい特徴量

第 5.1 節と同様に学習・テストを行ったとき，平均正答率が最も高かった特徴量群は「インデント，ファイルに関する特徴量」であった．この特徴量群の平均正答率は約 37%であった．それに次いで「コメントに関する特徴量」で約 33%であった．インデント，ファイルの特徴量では，ファイルの行の長さの平均等を抽出している．また，コメントの特徴量では，コメントのバイト数や行数の平均を抽出している．学生の中には，メソッドやメソッド内の処理毎にコメントを書く学生と全く書かない学生がいる．コメントを書く学生はファイルの行の長さも長くなり，コメントを書かない学生は短かくなるためと考えられる．それにより，この 2 つの特徴量群は残りの 3 つの特徴量群よりも平均正答率が高くなったと考えられる．この 2 つの特徴量群は残りの 3 つの特徴量群と比べると，サンプルコードの影響を受けにくいと考えられる．この 2 つの特徴量群は，大学講義程度で書かれたプログラムを対象とした著者解析には向いていると考えられる．

6. まとめ

本論文では，コーディングスタイルを基に初心者プログラマの著者解析を行った．また著者解析を効率的に行うために教師あり機械学習を用いた．また，教師あり機械学習には単純ベイズ分類器を用いた．単純ベイズ分類器を採用した理由として，実用的な精度を持ち合わせているためである．単純ベイズ分類器に入力する学習データとテストデータには，プログラムの表層的特徴であるコーディングスタイルの特徴量を用いた．結果，平均正答率は最大で約 37%であった．全学生と提出率が高い学生とを比較したとき，提出率が高い学生の方が，平均正答率は高い結果となった．

初心者プログラマはサンプルプログラムなどの影響を受けやすい．そのため，サンプルプログラムなどの影響を受けにくい特徴量の調査を行う必要がある．また，正答率を上げるために単純ベイズ分類器内に閾値を設けるといった対策が必要である．

参考文献

[1] 永井洋一，シムウォンボ，三輪　誠，近山　隆：機械学習を用いたプログラムの表層的特徴による分類，第 9 回プログラミングおよび応用のシステムに関するワークショップ (2010).
[2] 武田隆之，牛窓朋義，山内寛巳，門田暁人，松本健一：コーディングスタイルの特徴量とソースコード盗用との関係の分析，情報処理学会 (IPSJ SIG Technical Report 2010)(講演番号 2010-SE-167(8)) (2010).
[3] Spafford, E. H. and Weeber, S. A.: Software Forensics: Can We Track Code to its Authors?, *Purdue Technical Report CSD-TR 92-010* (1992).
[4] Ding, H. and Samadzadeh, M. H.: Extraction of Java program fingerprints for software authorship identification, *The Journal of Systems and Software* (2004).