社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

ソースコードメトリクスを用いた プログラミングコンテストの類似解答群の検出

原口　公輔†　神田　哲也†　井上　克郎†

† 大阪大学大学院情報科学研究科　〒565–0871　吹田市山田丘 1–5
E-mail: †{k-haragt,t-kanda,inoue}@ist.osaka-u.ac.jp

あらまし　アルゴリズム学習やプログラミング技術向上のため，多くのユーザがプログラミングコンテストに参加している．プログラミング初級者の学習支援を目的に，提出された解答群を分類し類似解答を提示する研究では，分類に際して語彙情報と実行時間等を用いている．本研究では，既存研究に準拠した特徴量の他にソースコードメトリクスを併せて分類することで，構造的に類似した解答が分類できるかを調査した．
キーワード　プログラミングコンテスト，N-gram IDF，ソースコードメトリクス

Finding Groups of Similar Answers in Programming Contests with Source Code Metrics

Kosuke HARAGUCHI†, Tetsuya KANDA†, and Katsuro INOUE†

† Graduate School of Information Science and Technology　1-5 Yamadaoka, Suita, 565–0871 Japan
E-mail: †{k-haragt,t-kanda,inoue}@ist.osaka-u.ac.jp

Abstract　Many users participate in programming contests to learn algorithm and improve programming skills. In the existing study that classify submitted answers and provides similar answers for learning support for programming beginners, vocabulary information and meta information like execution time are used for classification. In this study, we investigate whether structurally similar answers can be classified by existing research features plus source code metrics.
Key words　programming contest, N-gram IDF, source code metrics

1. ま え が き

プログラミングコンテストは，アルゴリズムに関する問題をユーザが一斉に解き，ソースコードの提出の早さや，解答の正確さを競うコンテストである．プログラミングコンテストを Web サービスとして提供するプログラミングコンテストサイトでは，ユーザに問題を公開し，オンラインジャッジシステムと呼ばれる採点システムを利用してその解答ソースコードの採点を行い，またユーザのレーティングを自動で行っている．

多くのプログラミングコンテストサイトでは，過去に提出されたソースコードがリポジトリに保存されており，自由に閲覧することができる．プログラミングコンテストのユーザはこれらのソースコードを参考にすることで，アルゴリズムの学習や，自身のソースコード改善に活用する事ができる．ところが，同一の問題に対するプログラムの実装方法は使用言語や使用するライブラリ，アルゴリズムによって多様である．さらに，プログラミングコンテストの解答は，使用言語や判定結果，提出時間などのソースコード以外のメタ情報でしかソースコードを検索できない．そのため，検索結果のソースコードがユーザにとって高度で理解できなかったり，あるいはユーザのものより非効率なものであったりするなど，学習の参考になるソースコードを効率的に検索することが困難である．

この課題を解決するために藤原らは，プログラミング学習者向けの解答提示システム TAMBA [1] を開発した．TAMBA はまず，プログラミングコンテストのソースコードリポジトリを語彙的な特徴で複数のクラスタに分類し，そのクラスタごとにコードのサイズ，メモリ使用量，CPU 時間の品質メトリクスでさらに細分化することで，類似する解答が含まれるクラスタを生成する，その後，提出履歴などの情報で各クラスタを頂点とする有向グラフを作成する．ユーザがシステムに対してソースコードを新たに入力すると，それに類似するソースコードが含まれるクラスタと，隣接するクラスタに含まれる全ソースコードを提示する．検索結果に性能面で最も優れたソースコードが提示されるわけではなく，入力されたソースコードに語彙的か

表 1: 上級者において値が大きいメトリクス [2]

メトリクスの種類	Cohen's d の値
セミコロンで区切られた論理行数	0.245
関数の数	0.191
物理行数	0.109
クラス当たりメソッド数	0.084
クラス数	0.069

表 2: 初級者において値が大きいメトリクス [2]

メトリクスの種類	Cohen's d
各関数のネストの深さの平均値	-0.457
全体の論理行数に占める分岐文の割合	-0.351
各関数の循環的複雑度の平均値	-0.303
各関数の循環的複雑度の最大値	-0.214
各関数のネストの深さの最大値	-0.179
全体の物理行数に占めるコメントの割合	-0.059

つ性能的に類似した解答だけが提示されるされるので，ユーザはこれらのソースコードを参考に自身のソースコードを段階的に改善できる．

一方堤は，プログラミングコンテストの解答は，解答者のレーティングによってソースコードメトリクスが異なることを明らかにした [2]．TAMBA ではソースコードメトリクスはコードのサイズ以外は用いられていなかったが，堤の研究結果を踏まえると，ソースコードメトリクスによってユーザの学習レベルにより近い解答を検索できる可能性がある．

そこで本研究では，TAMBA で用いていた語彙的な特徴に加えて，ソースコードメトリクスを考慮したプログラミングコンテストの解答の分類を試みた．また，分類結果の可視化を行うことで，本研究と TAMBA との分類の違いを調査した．T

2. 関 連 研 究

2.1 プログラミングコンテスト初級者・上級者間におけるソースコード特徴量の比較

堤はプログラミングコンテストの一種である Codeforces [3] の提出済みソースコードのうち C/C++ で記述されたものを対象に，初級者・上級者間の予約語の使用頻度やソースコードメトリクストの値などを調査した [2]．ここで，初級者・上級者とはプログラミングコンテストの順位に応じて決まる「レーティング」によって全ユーザをソートしたときの下位 25% と上位 25% のユーザを指す．

ソースコードメトリクスの集計は，メトリクス計測ツールの一つである Source Monitor [4] が計測できる 11 種類のソースコードメトリクスについて行った．さらに，それぞれのメトリクスについて Cohen's d [5] を計算し，効果量を測定した．Cohen's d は，ある 2 群において，各群の標準偏差に対してどの程度平均が異なるかを示す．これに基づいて，初級者・上級者間で差が大きいメトリクスを調査した．

堤の研究 [2] における表 9 から一部抜粋する形で，初級者と比較して上級者の計測値が大きいメトリクスを表 1 に示す論理行数や関数の数，クラス当たりメソッド数などが上位に挙がっており，上級者は処理を関数やクラスなどに分割する傾向があることが分かる．

また，堤の研究 [2] における表 10 から一部抜粋する形で，上級者と比較して初級者の計測値が大きいメトリクスを表 2 に示す．各関数の深さの平均値や，全体の論理行数に占める分岐分の割合などが上位に挙がっており，初級者は分岐文を多用して，ネストが深くなる傾向にあることが分かる．

堤はこのように，プログラミングコンテストでは，上級者のソースコードは初級者のものと比較して処理を関数やクラスといったブロックに分割されており，複雑度が小さい傾向にあることを示した．

2.2 解答提示システム TAMBA

プログラミングコンテストサイトでは，Web 上でプログラミングコンテストに参加することができ，ユーザは公開されている問題を選択してその解答となるソースコードを提出できる．これらのサービスはオンラインジャッジシステムと呼ばれる採点システムを利用して自動で採点が行われ，ユーザに結果が通知される．

多くのプログラミングコンテストサイトでは，過去に提出されたソースコードがリポジトリに保存されており，自由に閲覧することができる．プログラミングコンテストサイトのユーザはほかのユーザのソースコードを自身のプログラミング学習の参考にすることができる．ユーザの記述したソースコードに誤りがあり期待される動作をしない場合には，正しいソースコードを参考にして自身のプログラムを修正することで理解を深めることができる．また，ユーザの提出したものが採点上誤りのないソースコードであっても，過去に提出されたソースコードを参考にしてより効率的な実装が可能になると考えられる．

ところがプログラミングコンテストサイトは，使用言語や対象の問題などのソースコード以外のメタ情報による検索機能しか提供していない．そのため検索結果のソースコードがユーザのものと全く異なる実装方法であったり，ユーザのものより非効率なものであったりするため，全ての検索結果のソースコードがユーザにとって参考になるとは限らない．

この問題を解消するために，藤原らはプログラミング学習者向け漸進的ソースコード提示システム TAMBA [1] を開発した．TAMBA は対象とする問題の提出済みのソースコードについて，N-gram IDF を用いて各ソースコードを語彙的特徴で固定数のクラスタに分類する．N-gram IDF とは，白川によって開発された単語重要度を測る手法の一つであり [6]，単語単位の N-gram を入力として与え，全文書における特徴的な単語 N-gram の抽出を行うことができる．次に，コードのサイズ，メモリ使用量，CPU 時間の品質メトリクスに基づいてそれぞれのクラスタをさらに複数のクラスタに分類する．最後に，提出履歴情報を用いて各クラスタを頂点とする有向グラフを作成する．ユーザが TAMBA に，対象の問題に対する自身のソースコードを入力すると，TAMBA は入力したソースコードと最も特徴が近いソースコードが含まれているクラスタと，有向グ

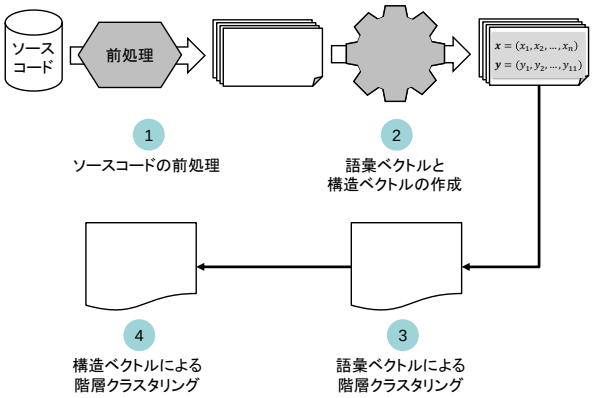


図 1: 提案手法の全体像

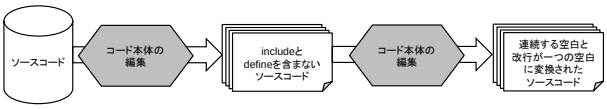


図 2: ステップ 1 の処理の概要

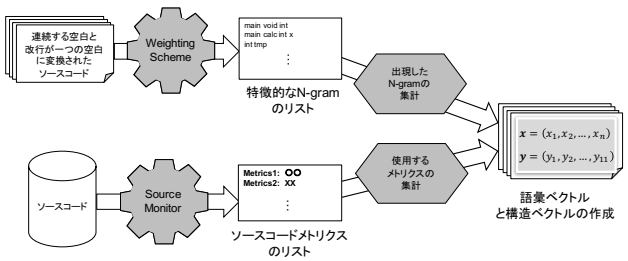


図 3: ステップ 2 の処理の概要

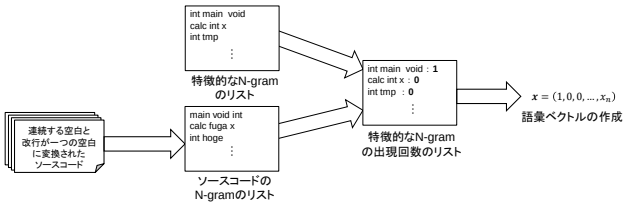


図 4: 語彙ベクトル作成の概要

ラフ上でそのクラスタと隣接しているクラスタを特定する．それらのクラスタに含まれている全てのソースコードを，ユーザの入力したソースコードと類似したソースコードとして提示する．最も性能が良いと考えられるソースコードを初めから提示するのではなく，他のユーザの提出履歴に沿ってソースコードを提示することで，ユーザは漸進的に自身のソースコードを改善できる．

3. 提 案 手 法

前節で説明した TAMBA にはソースコードの分類において，ソースコードサイズ以外のソースコードメトリクスの値が反映されていない．堤の研究を踏まえると，ソースコードメトリクスに基づいた分類を行えば，ユーザの学習レベルに近い解答を検索できる可能性がある．そこで，本研究ではソースコードメトリクスを用いて，語彙的な特徴に限らず構造的にも類似したソースコードの分類手法を提案する．

図 1 に提案手法の概要図を示す．分類対象とする問題の提出済みのソースコードについて，以下の 4 ステップを適用する．

（1） ソースコードの前処理

（2） 語彙的な特徴を表すベクトル (以下，語彙ベクトルと呼ぶ) と，ソースコードメトリクスを集約したベクトル (以下，構造ベクトルと呼ぶ) の作成

（3） 語彙ベクトルによるクラスタリング

（4） 構造ベクトルによるクラスタリング

それぞれのステップで行う処理について，各小節で説明する．

3.1 ステップ 1

ステップ 1 では，ソースコードの特徴量計測のための前処理を行う．処理の概要を図 2 に示す．

a） プリプロセッサの削除

プログラミングコンテストに提出されるソースコードの特徴として，不要なプリプロセッサの存在が挙げられる．ユーザはしばしば，プログラミングコンテストによく使われる項目をマクロとして記述したテンプレートを用いて，解答を作成する．一方で，今回解析対象とするプログラミングコンテストではソースコードの長さに制限が設けられていないため，不要な include 文や define 文によるエイリアスを含むソースコードが存在することになる．そこで本研究では語彙ベクトル作成に際して，プログラムの動作に影響を与えるかどうかに関わらず，プリプロセッサを全て削除する．

b） 改行と連続した空白の変換

ソースコードに含まれる改行と連続した空白を 1 つの空白に変換する．これは，N-gram IDF 値の計算に用いるツール入力フォームに合わせるためであり，TAMBA でも同様の処理を行っている．

3.2 ステップ 2

ステップ 2 では，ソースコードの特徴を表す語彙ベクトルと構造ベクトルの作成を行う．処理の概要を図 3 に示す．

a） 語彙ベクトルの作成

ステップ 1 で前処理を施したソースコードに対して，N-gram IDF 値を用いた全文書間における特徴的な単語単位の N-gram [6] の抽出と，その出現回数を用いた特徴ベクトルの作成を行う．語彙ベクトル作成の概要を図 4 に示す．

単語 N-gram の抽出には Weighting Scheme [7] を用いる．これは白川らが提案した，全文書における特徴的な単語 N-gram

表 3: 調査対象のメトリクス一覧

メトリクス	説明
avg_complexity	各関数の循環的複雑度の平均値
max_complexity	各関数の循環的複雑度の最大値
avg_depth	各関数のネストの深さの平均値
max_depth	各関数のネストの深さの最大値
methods_per_class	クラス当たりのメソッド数
n_class	クラス数
n_func	関数の数
n_lines	物理行数
n_statements	セミコロンで区切られた論理行数
percent_branch_statements	全体の論理行数に占める分岐文 (if, else, for, while, goto, break, continue, switch, case, default, return を用いた文) の割合
percent_comments	全体の物理行数に占めるコメントの割合

の抽出の手法 [6] を実行するツールである．

前処理を行ったソースコードを Weighting Scheme に入力すると，そのソースコードにおける特徴的な N-gram の一覧が出力される．出力された N-gram の出現回数をソースコードごとに求め，次元数を出力された N-gram の数，各成分を対応する N-gram の出現回数とするベクトルを作成し，これを語彙ベクトルとする．

b） 構造ベクトルの作成

前処理を行う前のソースコードに対して，Source Monitor が対象とする 11 種類のメトリクスについて計測する．それらを表 3 に示す．各成分をそのメトリクスの値とするベクトルを作成し，これを構造ベクトルとする．

3.3 ステップ 3

ステップ 3 では，ステップ 2 で作成した語彙ベクトル間の類似度を求めて凝縮型階層クラスタリングを実行し，全ソースコードを 3 つのクラスタに分類する．凝集型階層クラスタリングは同じ次元を持つ複数のベクトルを，定めたクラスタ間の距離計算アルゴリズムと計算距離によって併合していき，複数のクラスタに分類するアルゴリズムである [8]．本研究ではクラスタ間の距離計算アルゴリズムには最長距離法を，計算距離にはコサイン距離を用いる．凝集型階層クラスタリングの概要を以下に示す．

（1） 分類するベクトルと同数のクラスタがそれぞれ 1 つのベクトルを持った状態からスタートする

（2） 各クラスタ間でベクトルの最大距離を全て求め，全てのクラスタの組み合わせのうち，最小なクラスタの組を 1 つ抽出する

（3） 抽出したクラスタの組について，1 つに併合する

（4） クラスタが目標の数になるまで上記を繰り返す

3.4 ステップ 4

ステップ 4 ではステップ 3 で分類した 3 つのクラスタごとに，凝集型階層クラスタリングを実行し，さらに 3 つのクラスタに分類する．ステップ 3 と同様にクラスタ間の距離計算アルゴリズムには最長距離法を，計算距離にはコサイン距離を用いる．ステップ 3，4 の手順によって全ソースコードは 9 つのクラスタに分類される．

4. ケーススタディ

本章では TAMBA による分類手法と提案手法とで，分類後のソースコードのソースコードメトリクスの分布にどのような違いが見られるかを調査する．

4.1 分類結果の可視化

本研究の目的であるソースコードメトリクスを反映した分類が達成できているかを視覚的に確認するため，ソースコードの構造ベクトルを二次元グラフ上にマッピングし，その上に分類結果を可視化する

4.1.1 主成分分析

主成分分析とは，多次元データのもつ情報を可能な限り損なわずに低次元の空間に情報を縮約する手法の一つである．多次元ベクトルに対して主成分分析を実行することで，ベクトル群の圧縮前の次元と比較して情報 (ベクトル群の分散の合計) の最大量を保持しながら，元の分布を少数のベクトルで表すことができる．

本研究では，主成分分析を可視化に用いるにあたり採用する主成分の数を 2 とし，構造ベクトル群を 2 次元グラフ上にマッピングした．さらに，主成分分析による次元削減後の第一主成分・第二主成分の寄与率も計算し，次元削減により減少した情報量も確認する．

4.1.2 主成分分析の実行方法

提案手法のステップ 3 で示した構造ベクトルを，主成分分析における次元削減対象とする．第一主成分，第二主成分をそれぞれ横軸，縦軸とし，各構造ベクトルをプロットしたグラフを作成する．

TAMBA による分類手法と本研究の提案手法では，分類後に 9 つのクラスタに分類される．それぞれのクラスタに色を付け，第一主成分，第二主成分をそれぞれ横軸，縦軸とするグラフ上でそれぞれのクラスタがどの位置にプロットされたのかを明確にする．これにより，主成分分析によって次元削減したグラフ上での構造的な特徴の分布を可視化し，比較することができる．

4.2 データセットの準備

本研究では，分類するソースコードのデータセットとして，堤の研究で作成されたデータセット [2] を用いた．これは，プログラミングコンテストの 1 つである Codeforces に提出されている，ソースコード及び提出時間などのメタ情報を格納したデータベースによって構成されるデータセットである．プログラミング言語は C/C++ を対象に，データ収集は 2016/5/19 から 2016/11/15 の間で行われ，対象となったソースコードのファイル数は 1,644,636 個，ソースコードの提出者の数 (データセットに含まれるプログラミングコンテストのユーザの数) は 739 人，問題数は 3,218 個である．このデータセットの中から特定の問題のソースコードを抽出してそれぞれの手法で分類と可視化を行う．

今回対象とした問題の情報を表 4 に示す．問題 ID に付与さ

表 4: 調査対象の Codeforces の問題

問題 ID	ファイル数
489A	131
489D	221

表 5: 凡例の番号の説明

凡例の番号	説明
1, 2, 3	語彙ベクトルによる分類で 1 つ目のクラスタに分類された
4, 5, 6	語彙ベクトルによる分類で 2 つ目のクラスタに分類された
7, 8, 9	語彙ベクトルによる分類で 3 つ目のクラスタに分類された

れている数字はコンテスト開催毎に付与される番号であり，アルファベットは問題作成者が付与した難易度を表している．問題によってその程度は異なるが A から F までであり，A が比較的容易であり F は比較的難しい．また，それぞれの問題でのデータセット中の解答についてそれぞれのソースコードメトリクスの分散を計算したところ，489A では小さく，489D では大きいことが確認できている．

4.3 分類結果

調査対象の問題について，次元削減を行った結果と，それぞれの分類結果をその上にマッピングした結果を示す．主成分分析の結果を示した図において，凡例のラベルは分類に際し作成されたクラスタにつけた番号を表す．番号は表 5 に示すように，ステップ 3 における語彙ベクトルによる分類によって生成された 3 つのクラスタに対応している．各クラスタのプロットが他のクラスタのプロットと比較して重なりが少なく，一箇所に集まっている状態ほど，最終的に作成されるクラスタにソースコードメトリクスの値を反映できていると言える．

4.3.1 489A の分類結果

問題 ID 489A に対する解答ソースコードを TAMBA の分類手法と本研究の提案手法とで分類し主成分分析を用いて次元削減したときの，第 1 主成分の寄与率は 88.9%，第 2 主成分の寄与率は 7.10% であった．また，可視化の結果をそれぞれ図 5，6 に示す．主成分分析の累積寄与率は 95% を超えており，可視化に伴うデータ損失量は小さいといえる．

2 次元マッピングの結果について見てみると，点全体が原点の近くに偏っている．A 問題はコンテストの問題において比較的簡単な問題であり，実装方法にそれほど差が生じないことに起因していていると考えられる．

また，TAMBA の可視化したグラフと比較して，提案手法による分類では 1 つのクラスタに多くの解答が集中している．これは平易な問題ほどソースコード間におけるソースコードメトリクスの差が小さく，分類に偏りが生じるためであり，提案手法によって作成されるクラスタが，TAMBA と比べてソースコードメトリクスの値を強く反映していることを示している．

4.3.2 489D の分類結果

問題 ID 489D に対する解答ソースコードを TAMBA の分類手法と本研究の提案手法とで分類し主成分分析を用いて次元削減したときの，第 1 主成分の寄与率は 88.7%，第 2 主成分の寄与率は 6.31% であった．また，可視化の結果をそれぞれ図 7，8 に，ステップ 3 における語彙ベクトルによる分類によって作成した 3 つのクラスタごとにプロットしたものをそれぞれ図 9，図 10，図 11 に示す．主成分分析の累積寄与率は 95% を超えており，可視化に伴うデータ損失量は小さいといえる．

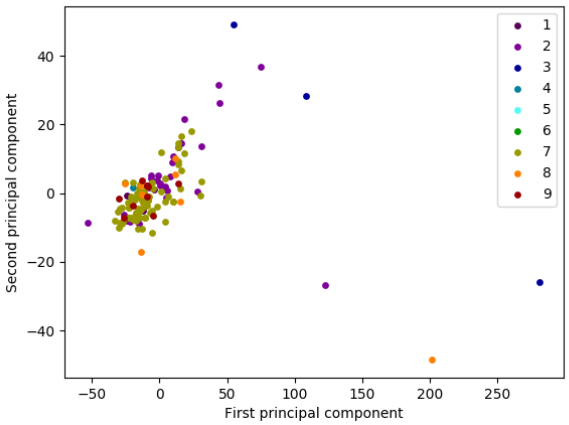


図 5: TAMBA の分類手法による問題 489A の分類結果

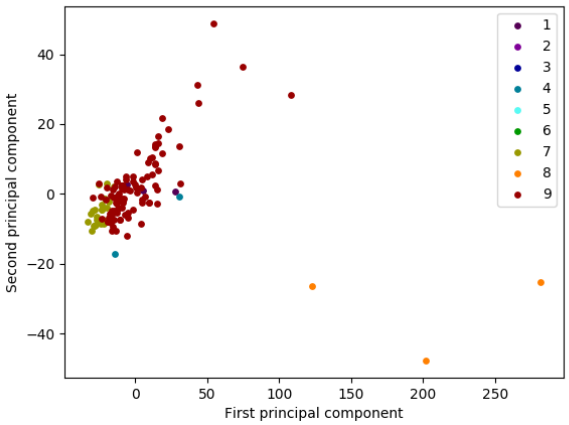


図 6: 提案手法による問題 489A の分類結果

2 次元マッピングの結果について見ると，提案手法の分類結果の方が，TAMBA の分類よりも各クラスタのプロットが一か所に集まる．語彙ベクトルによる分類で作成したクラスタごとにプロットしたものを見るとその傾向がより顕著に確認できる．今回分類の対象にした D 問題は比較的難易度が高く，問題を解くために用いるアルゴリズムやユーザの癖によってソースコードメトリクスの値も異なってくると考えられる．

この結果から，提案手法は TAMBA とは異なるソースコードメトリクスの観点からソースコードを分類することができたといえる．また，TAMBA で分類に用いられていた品質メトリクスは，提案手法で用いたソースコードメトリクスとは異なる傾向を示すこともわかる．

5. まとめと今後の課題

本研究では，プログラミング学習者向けの解答提示システム

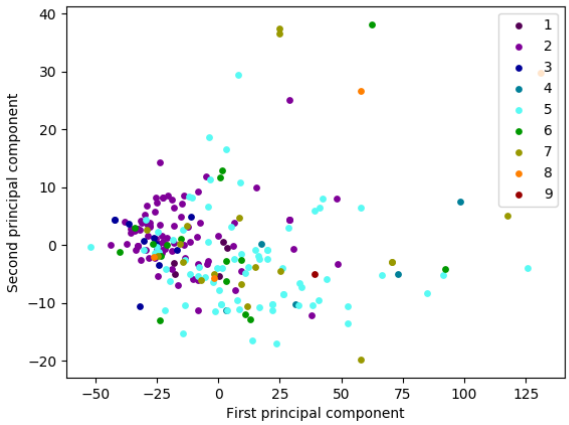


図 7: TAMBA の分類手法による問題 489D の分類結果

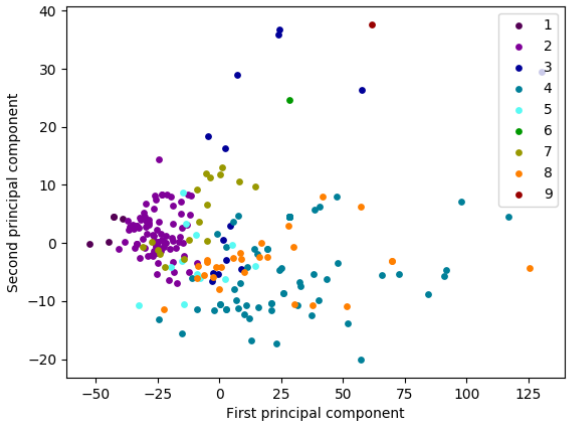
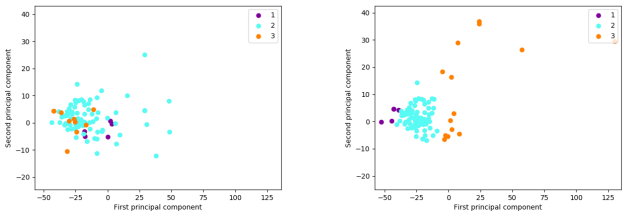
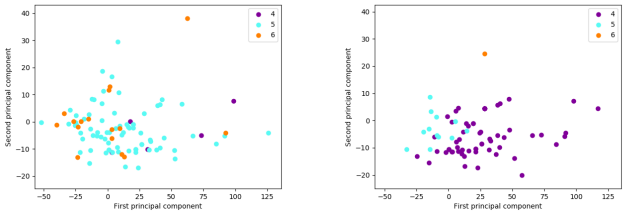


図 8: 提案手法による問題 489D の分類結果



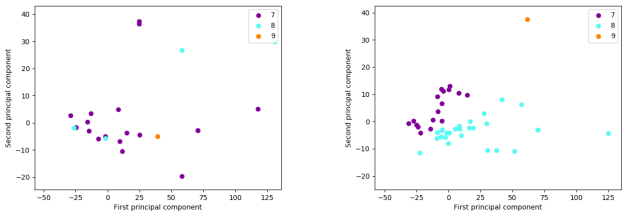
(a) TAMBA (b) 提案手法

図 9: クラスタ 1, 2, 3 の分類結果



(a) TAMBA (b) 提案手法

図 10: クラスタ 4, 5, 6 の分類結果



(a) TAMBA (b) 提案手法

図 11: クラスタ 7, 8, 9 の分類結果

TAMBA で用いられている特徴量に加えてソースコードメトリクスを用いて分類する手法を提案し，その結果を主成分分析により次元削減したソースコードメトリクスのベクトルを 2 次元グラフにプロットすることによって可視化し，TAMBA による分類結果と比較を行った．

分類を実行した 2 つの問題のうち，難易度が低い問題では，提出されたソースコードのソースコードメトリクスの分布が一点に偏っており，1 つのクラスタに含まれるソースコードの数が全く異なる結果となった．一方，比較的難易度の高い問題では 1 つ 1 つのクラスタのソースコードメトリクスの分布に違いが見られた．TAMBA の分類手法による分類結果は 1 つのクラスタのプロットのばらつきが大きく，異なるクラス間のプロットの重なりが多かった．それと比較して本研究の提案手法による分類結果は，各クラスタの分類が一箇所に集まっているのを確認できた．このことから，提案手法によりソースコードメトリクスの値が互いに近いソースコードが同一のクラスタに含まれることが確認できた．

今後の課題として，ソースコードメトリクスが実際に学習に有用であるかを評価することが挙げられる．本研究では提の研究に基づきソースコードメトリクスがユーザの学習レベルにあわせた解答検索を可能にすると考え分類を行ったが，TAMBA で行われていたユーザテストによる分類結果の有用性の評価までは行っていない．そのため，ユーザテストを行いソースコードメトリクスがどの程度学習に有用であるかを調査する必要がある．

文　献

[1] 藤原新, 中川尊雄, 畑秀明, 松本健一. プログラミング学習者向けソースコード提示システム tamba. ソフトウェアエンジニアリングシンポジウム 2016 論文集, 第 2016 巻, pp.34-41, 2016.

[2] 堤祥吾, プログラミングコンテスト初級者・上級者間におけるソースコード特徴量の比較, 大阪大学大学院情報科学研究科修士論文, 2018.

[3] Codeforces. http://codeforces.com/

[4] Sourcemonitor v3.5. http://www.campwoodsw.com/sourcemonitor.html

[5] Jacob Cohen. HStatistical Power Analysis for the Behavioral Sciences, 2nd edition. Lawrence Erlbaum Associates, 1988.

[6] 白川真澄, 原隆浩, 西尾章治郎. コルモゴロフ複雑性に基づく idf の単語 n-gram への適用. データ工学と情報マネジメントに関するフォーラム (DEIM 2015), 2015.

[7] iwnsew/ngweight: N-gram weighting scheme. https://github.com/iwnsew/ngweight

[8] Zhao Ying, Karypis George Fayyad Usama, "Hierarchical Clustering Algorithms for Document Datasets"