

プログラミング教育のための 類題出題システムの提案

吉村 涼矢¹ 坂本 一憲¹ 鷲崎 弘宜¹ 深澤 良彰¹

概要: プログラミング初学者を対象としたプログラミング教育において、オンラインジャッジを使用するケースがある。オンラインジャッジはコーディング能力の向上に役立つ反面、課題の難易度が適切でない場合、十分な教育効果が得られない可能性がある。そこで、本研究では課題に対して類似している問題の内容およびその問題に正答するソースコードの例を学生に公開するアプローチを提案する。評価のため、早稲田大学の講義において介入実験を実施した。その結果、介入しなかった年度の正答数と比較して、介入した年度の正答数が統計的に有意に向上した。したがって、本提案手法は課題の理解を向上させることができ、課題の教育効果を改善できることが示唆された。

1. 導入

一般に、プログラミング教育において、学生がアルゴリズムを理解して実装できるように指導することが求められている。そしてアルゴリズムの学習を支援するシステムとして、オンラインジャッジ [1] が利用されることがある。オンラインジャッジは学生に実装すべきコードの仕様を提示して、学生からコードの提出を受け付け、受け取ったコードが仕様を満たすかどうかを判定する。オンラインジャッジを利用することで、学生はコードを記述する能力に加えて、問題文を読み解く能力や、時間計算量と空間計算量を考慮してコードを記述する能力、デバッグ能力などを鍛錬できる。

しかし、オンラインジャッジが出題する課題の難易度が学生にとって適切でない場合、学生が早期に解答することを諦めてしまい、適切な学習効果が得られないケースがある。講義中の演習課題であれば、解答に苦労している学生に対して、教員やティーチングアシスタントなどが個別指導することで、各学生にとって適切な難易度に課題を調整する工夫が行われている。しかし、従来のオンラインジャッジは投稿されたコードが仕様を満たすか否かを評価する側面が強く、解けない学生をフォローする仕組みが提供されていない。模範解答を公開することは可能ではあるものの、学生はそのまま模範解答をオンラインジャッジに投稿して正答判定を受けるだけで満足してしまい、学生はそれ以上の学習を行わない可能性がある。

そこで、本研究では難易度の高い課題の模範解答を公開する代わりに、課題に類似する問題（以降、類題）の内容および類題に正答するソースコードの例（以降、参考実装）を公開するアプローチを提案する。学生は類題の内容および参考実装を読んで理解することで、難易度の高い課題の解き方を理解できる。また、学生は類題の参考実装をオンラインジャッジへ提出しても、課題の解答としては正答判定を得ることができない。そのため、学生は類題の内容および参考実装を閲覧しても、依然として難易度の高い課題を解くために考える必要がある。

以上のアプローチの有効性を検証するため、類題出題システム（WOJ Recommender と名付ける）を提案する。本システムは課題の模範解答を使用して、既存のオンラインジャッジ上で公開されている問題（以降、問題セット）のうち課題に類似する問題および参考実装を、オンラインジャッジを用いる講義を担当する教員に提示する。教員は提示された複数の類題候補の中から使用する類題を選ぶ。同様に、選んだ類題の参考実装の候補の中から使用する参考実装を選び、必要に応じて修正を行う。学生は類題および参考実装を読むことで、アルゴリズムの概念の実装に対する理解を向上させ、それによって、学生が諦めずに学習を続けるように支援する。本研究における研究課題は以下の2つである。

RQ1 講義を受講する学生のうち何名の学生が、WOJ Recommender を介して提供する課題の類題および参考実装を閲覧するか。

RQ2 WOJ Recommender を提供することで、学生の課題における正答数は向上するか。

¹ 早稲田大学
Shinjuku, Tokyo 169-8050, Japan

以上の研究課題に回答するため、Waseda Online Judge (WOJ) [2] というオンラインジャッジを用いる、早稲田大学のアルゴリズムの演習を主体とした講義において、本システムを提供する介入実験を実施した。また、類題および参考実装を取得するためのオンラインジャッジとして、Aizu Online Judge (AOJ) [3] を採用した。その結果、介入しなかった年度の正答数と比較して、介入した年度の正答数が統計的優位に向上した。したがって本手法は課題の理解を向上させ、課題の教育効果を改善できることが示唆された。本研究の貢献は以下のとおりである。

- 難しい課題の理解を補助する方法として、類題および参考実装を提供するアプローチを提案したこと。
- 上記アプローチを実現するために、WOJ Recommender を開発したこと。
- WOJ Recommender を用いて介入実験を実施し、介入を行わなかった年度と比較して課題の正答数の向上を確認したこと。

2. 関連研究

Q.Cutts らは、Thinkathon という紙を用いた任意参加の補講を実施した [4]。対象講義では元々 Peer Instruction を用いて、コンピュータ上で演習に取り組むスタイルを採用していた。しかし、対象講義の試験は紙を用いたものであったため、補講では試験問題の形式に近い問題を扱った。その結果、補講に参加した学生の成績が向上した。さらに、本補講の実施を通して、学生は問題を解く際に必要以上にコンピュータに頼ってしまっており、紙で解かせることで、より深い学びを与えられることが示唆された。本研究においては、学生が取り組む問題に類似している問題を学生に解かせるという観点で共通している。一方で、Q.Cutts らは類題を教員が生成したが、本研究ではオンラインジャッジより選出している。

H.Yuan らは、Hybrid Pair Programming という形式の講義を実施した [5]。対象講義ではペアプログラミングを採用していたが、ペア間の力量の差から片方の学生のみが課題を解き、もう片方の学生が何もしない現象が起きていた。彼らは該当現象を FreeRide と名付けている。その対処法として彼らは、ペアプログラミングの講義を2段階に分割する Hybrid Pair Programming を実施した。第1段階では、学生が一人で簡単なレベルの課題を解くようにさせた。第2段階では、第1段階の知識を用いた発展的課題をペアプログラミングの形式で解かせた。その結果、Hybrid Pair Programming を行った学生と最初から最後までペアプログラミングを行った学生を比較すると、前者の方が成績が良い傾向にあった。本研究においても、課題の模範解答を提示してしまうと、学生が内容を理解せずにそのまま模範解答を投稿してしまうという現象を考慮しており、FreeRide と同様の現象を扱っている。本研究では、類

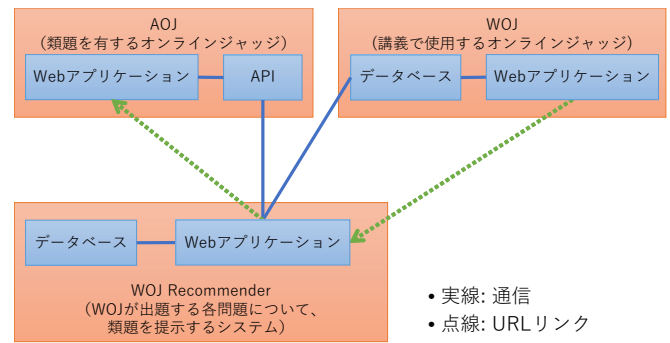


図1 システムの全体像

題及び参考実装を提示することで、本現象を回避している。

藤原は、TAMBA というソースコード提示システムを作成した [6]。TAMBA は、学生が解答コードを提出しても、オンラインジャッジが仕様を満たさないと判定した際に、学生のデバッグ作業を支援するために、過去に他者が提出した正答コードを提示する。しかし、同一問題において、解法および実装方法が多岐に渡ることがあるため、学生が採用している解法および実装方法に類似するコードを提示しなければ、デバッグの役に立たない。そこで、TAMBA は過去の提出コードの中から、ユーザーが投稿したコードに類似するコードを収集して、最も有用なコードを提示する仕組みを提供する。それを実現するためのシステムとして TAMBA を開発した。AOJ の問題セットを用いて、被験者による評価実験を行った結果、AOJ 上の提出一覧を閲覧するよりも効率よく誤答の原因を特定し、デバッグに役立つことが示唆されるものとなった。TAMBA は、同じ問題の参考となるコードを N-gramIDF [7] を用いてソースコードを特徴ベクトルに変換し、コードの類似度を計算する。本研究では、類題および参考実装を探すために、N-gramIDF に近い特性を持ちながら、計算量や実装が比較的軽量の TFIDF [8] を用いた。

3. 類題出題システムの提案

3.1 全体像

WOJ Recommender は、オンラインジャッジを用いる講義において、講義を担当する教員には出題する課題の類題やその参考実装の候補選出を行う機能を、講義を履修する学生には教員が選択した類題と参考実装を表示する機能を実現する。

図1で WOJ Recommender および連携システムの全体像を示す。WOJ とは講義で使用するオンラインジャッジである。WOJ の構成要素の代表的なものとして、Web アプリケーションとデータベースがある。AOJ とは WOJ とは異なるオンラインジャッジであり、WOJ から見ると類題や参考実装を有している。AOJ の構成要素の代表的なものには、Web アプリケーションとそれに表示される情報を取得できる API がある。WOJ Recommender とは WOJ が

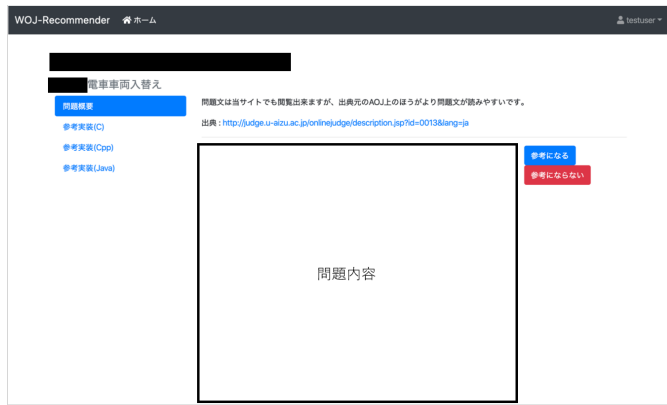


図 2 学生の閲覧する画面の例

出題する各課題について、類似する AOJ 上の問題やその参考実装を提示するシステムである。WOJ Recommender の構成要素の代表的なものには、Web アプリケーションとデータベースがある。各システムの関係性としては、WOJ Recommender が他のシステムや構成要素にアクセスしてデータを取得するという形になっている。WOJ Recommender は WOJ にアクセスして課題などの情報を取得したり、AOJ にアクセスして類題となりうる問題などの情報を取得したりする。

以下で学生が WOJ Recommender を使用する手順を示す。まず学生は、WOJ 上の課題を解く。WOJ 上の課題が解けない場合、学生は WOJ に配置されている URL リンクを踏み、WOJ Recommender のサイトにアクセスする。学生はそのサイトにおいて、図 2 のように表示される類題やその参考実装を閲覧し、WOJ の課題に解答する際の参考にする。学生が類題を解いて類題の解答を提出する際は、WOJ Recommender は類題及び参考実装の閲覧機能のみを提供しているため、AOJ 上で提出する必要がある。そのため、提出する際には WOJ Recommender に配置されている URL リンクを踏み、AOJ のサイトにアクセスする。

3.2 類題を選出する方法

3.2.1 TFIDF の説明

WOJ Recommender は、TFIDF 法とコサイン類似度を用いてソースコードの類似度を調べることで、複数の類題候補を選出する。TFIDF 法は主に文書のジャンル類似度を計算するための手法であるが、コードクローンの検出 [9] や、競技プログラミングにおける提出コードのクラスタリング [6]、またソースコード内で特徴的部分を抽出する [10] ことにも使われている。TF 値は対象の文書において、着目している単語の出現頻度である。IDF 値は母集団とする文書集合において、着目している単語の出現頻度の逆数である。そして、TFIDF 値は $TF \times IDF$ として計算される [10]。WOJ Recommender は、scikit-learn の

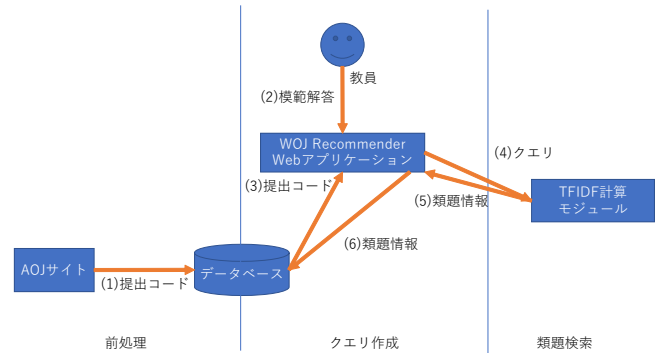


図 3 類題選出のための準備



図 4 教員の閲覧する画面の例

TfidfVectorizer^{*1} を用いて各ソースコードの TFIDF 値を計算する。

3.2.2 類題を選出する流れ

図 3 で類題を選出する流れを示す。オンラインジャッジ上には様々な問題および問題の解答コードが存在するので、事前にオンラインジャッジから解答コードをダウンロードしてデータベースに登録しておく (図 3 (1))。次に、教員は WOJ Recommender の Web ページにアクセスして、問題の管理画面を開く (図 4)。教員は WOJ Recommender に課題の模範解答を入力して、類題の候補を要求する (図 3 (2))。WOJ Recommender はデータベースから収集した解答コードを取り出して (図 3 (3))、模範解答と解答コード集に対して、各解答コードに対する類似度を計算するために、TFIDF 計算モジュールにクエリを送信する (図 3 (4))。TFIDF モジュールは類似度の高い解答コードを類題の候補として WOJ Recommender に情報を返す (図 3 (5))。最後に、WOJ Recommender は受け取った類題候補の情報をデータベースに格納する (図 3 (6))。以上の処理が完了すると、教員は WOJ Recommender の Web ページ上で、類題の候補が閲覧できるようになる。

3.2.3 問題の類似度を計算する方法

図 5 で問題の類似度を計算する流れを示す。まず、模範解答ソースコードと AOJ 上の全ての解答コードを母集団とした上で、各ソースコードにおける TFIDF 値を求める。AOJ 上の各解答コードについて、模範解答の TFIDF ベク

^{*1} https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

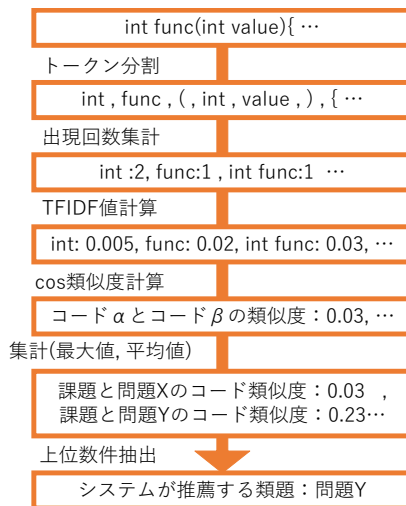


図 5 類題選出方法

トルと該当解答コードの TFIDF ベクトル間のコサイン類似度を計算することで、模範解答に対する AOJ 上の各解答コードの類似度をスカラー値として求める。その後、問題ごとに解答コードを集約して、各問題について存在する解答コード内の類似度の最大値もしくは平均値を類題候補の類似度とする。以上の方法で、出題する各問題について、類題となりうる問題の類似度が計算できる。WOJ Recommender では、類似度で上位 5 件を類題の候補として表示する。

3.3 参考実装を選出する方法

3.3.1 参考実装に適用するメトリクス

WOJ Recommender では参考実装に適用するメトリクスを、CC と token count を併用する $\max(1000 - 10 * CC - \text{tokencount}, 0)$ [11] とした。上記の式を用いた理由は 2 つある。1 つ目は、CC とソフトウェアの品質に相関があるためである [12]。2 つ目は、コードが長い場合、閲覧した瞬間にコードの理解を諦めてしまう可能性が上がるからである。そのため短いコードであれば、学生が少しでも閲覧して理解に繋がるようになると思った。ただしソースコードを 1byte でも短くすることを目指した極端なコードを除くため、ソースコードの行数が 1 行などの場合は除外した。

3.3.2 参考実装を選出する流れ

図 6 で参考実装を選出する流れを示す。まず事前にデータベースに AOJ 上のソースコードを登録する (図 6 (1))。その後、ソースコードの CC やトークン数を計算するため、lizard^{*2} というツールに先程述べたスコアの計算を行っていない提出コードを送信する (図 6 (2))。そうするとメトリクススコアを計算するために必要なメトリクスが返り値として与えられるため、データベースにメトリクススコア情報を設定する (図 6 (3))。設定後に教員がある課題に対する参考実装を要求した場合には (図 6 (4))、先程設定し

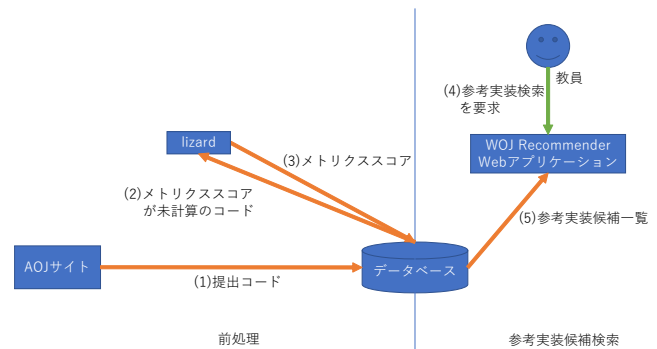


図 6 参考実装の選出を行う具体的な手順

表 1 各条件における該当学生人数 (介入年度)

条件	人数
講義履修	80
アンケート回答	52
システム使用	19

表 2 実験介入有無に対する全問正答者数および平均正答数の変動

	未介入年度	介入年度
学生人数	89	80
全問正答者数	71	73
平均正答数 (10 問)	9.24	9.73

たメトリクススコアの上位数件を保持するソースコードをデータベースから取得して (図 6 (5))、参考実装の候補一覧を教員側に提示する。

4. 実験

4.1 実験環境

WOJ Recommender の有効性を検証するために、早稲田大学における、アルゴリズム演習を中心とした講義で実験を実施した。講義を受講する学生数は介入年度においては 80 名で、学生はアルゴリズムの基礎的講義を履修済みである。介入は講義の初めの 5 回で行い、5 回目の講義時にアンケートへの回答を求めた。アンケートは、[13][14]を参考に、著者らによって作成された。

4.2 実験結果

本実験では、2019 年度の全ての受講生に介入するため、未介入である 2018 年度 (未介入年度と呼ぶ) と 2019 年度 (介入年度と呼ぶ) の正答数を比較することとした。

表 1 に介入年度における各条件下の学生の人数を示す。講義履修者全員を対象としたアンケートに対して回答を行った学生数は 52 名であった。そのアンケートにおいて実際に WOJ Recommender を使用したと答えた人数は 19 名であった。

表 2 に全問正答者数、平均正答数に関する結果を示す。

*2 <https://github.com/terryyin/lizard>

正答数の比較対象とした課題は、介入年度と非介入年度で同様の内容の課題で、かつ、出題した課題と類題が過度に類似しないような課題とした。その結果、10問が比較対象の課題となった。未介入時は平均正答数は10問中9.24問であったが、介入時には9.73問にまで向上した。また全問正答者は、未介入時は全体の凡そ79.8%であったが、介入時は91.3%にまで向上した。

統計的に有意な差があるかどうかを検証するために、マンホイットニーのU検定 [15] で両側検定を用いた。本検定は、母集団の分布を仮定せず、対応のない2群に対して有意差を調べるノンパラメトリックな検定である。本研究で比較する2群は介入年度の履修者と未介入年度の履修者であるため対応が無く、正答数を正規分布と仮定する根拠がなかったため用いた。なお有意水準は、教育分野で一般的に用いられる0.05を使用する。検定の結果p値は0.032であったため、未介入年度と比較して学生の正答数は有意に向上したと言える。

アンケートでは、WOJ Recommenderを使用した人にはシステムの良かった点や改善してほしい点などを質問した。使用しなかった人にはどういった機能が必要かなどを質問した。その結果、使用者の複数名が以下のように答えた。

- 問題文が分かりにくい類題がある。
- 考え方の似ている類題やアルゴリズムを参照できて参考になった。

また使用者未使用者に共通して表れたものとして以下のようなものがあった。

- 課題との類似度が上がって欲しい。
- アルゴリズムを図表などを用いて解説してほしい。

4.3 実験結果の考察

表1から、WOJ Recommenderを使用した学生は80名のうち19名が使っており、約25%が使用していた(RQ1)。25%という数字は少なく見えるが、WOJ Recommenderは課題が解けない学生向けの仕組みであることを考えると、そもそも全ての受講生が使用することは期待できない。表2から、未介入年度では学生の79.8%が全問正解しており、介入年度も同様の割合の学生が介入しなくとも全問正解できたことが想定できる。全問正解できない学生が約20%存在する状況で、約25%の学生がWOJ Recommenderを使用したのであれば、十分多くの学生が使用したと評価できるだろう。

未介入年度と比較して、介入年度では正答数が10問中約0.5問向上しており、正答率を約5%向上できた。未介入年度と介入年度の正答数は統計的に有意な差があり、WOJ Recommenderによって正答数を向上できることを確認した(RQ2)。元々約80%の学生が全問正解できていたことを考慮すると、より難易度の高い課題を対象とすることで、WOJ Recommenderによる正答数の向上率はより高まる

ことが期待される。

最後にアンケートの自由回答の結果について考察する。まず、考え方の似ている問題を参照できて参考になったという意見があった。これにより、WOJ Recommenderの狙い通りに参考になった学生がいたということが分かる。次に、問題文が分かりにくい類題があるという意見があった。今回類題の探索先にAOJを採用しており、問題文を複雑に書くことの多いICPCという大会などの過去問が多く含まれていたことが要因と考えられる。したがって、類題の探索先である問題集を拡充することで改善できる可能性がある。また、課題との類似度が上がって欲しいという意見があったことに対しては、2つの要因が考えられる。1つ目は、問題セットの問題である。前述した通り、AOJは過去のプログラミングコンテストの問題を多く含んでおり、難易度の高い問題が多数存在している。実験対象の講義で扱った問題は、上記問題より難易度が低いため、そもそも適切な類題が存在しなかった可能性がある。そこで、より難易度の低い問題を多数含む問題集を採用することで、改善できる可能性がある。2つ目は類似度計算方法の問題である。WOJ Recommenderは、ソースコードに対してTFIDFを用いた類似度計算を行い、問題の類似度を計算した。しかしTFIDFでは、主にトークンの類似性が類似度に反映される。そのため、既存のコードクローン技術 [16][17][18] を活用して、構造的観点を含んだ類似度を計算することで、改善できる可能性がある。ただし、一般的に構造的観点を考慮した類似度の計算量は大きいため、まず、計算量の小さい類似度を用いて類題の候補を絞り込んだ上で、計算量の大きい類似度で最終候補を絞り込むという2段階での絞り込みを採用する必要があると思われる。最後に、アルゴリズムの解説が欲しいという意見があった。既存のオンラインジャッジには、問題と回答コードしか存在していないため、WOJ Recommenderでは扱えない領域だと考えている。

4.4 提案手法の制限および妥当性の脅威

WOJ Recommenderは、既存のオンラインジャッジの類題および参考実装を分析して、類題とその参考実装を学生に閲覧させることにより教育効果を図っている。そのため、類題が存在しない課題に対してはWOJ Recommenderは使用できない。

また、WOJ Recommenderは、教員が類題および参考実装を候補の中から選択するため、教育効果は教員のスキルに依存する。

本実験を通して本システムを動作させたところ、木構造やグラフに関する問題については類題がほぼ見つからなかった。木構造やグラフの問題では、実装が構造的には類似することはあるが、ソースコード中のトークンは多種多様になりやすい。その結果、TFIDFでは適切に類似度を

計算できなかったものと考えられる。今後は、構造的観点を含んだ類似度を採用することで、類題を発見できる問題の種類を増やしたい。

本実験では、年度の異なる学生の間において正答数を比較している。そのため、2群間に実力差があった可能性がある。これは、内的妥当性への脅威である。今後は同様の実験を繰り返したり、同一の母集団を2群に分割して実験することで、対処をしていきたい。

また年度が異なることにより、問題文の内容が一部変化していた。したがって、2群間で解いた問題の難易度に差があった可能性がある。これは、内的妥当性への脅威である。今後、同一の問題を対象として同様の実験を行うことで、対処したい。

本実験は早稲田大学の1つの講義に対して実施したものであり、他の母集団や他の問題で同様の結果が得られるかは分からない。これは、外的妥当性への脅威である。今後、他の母集団や他の問題を対象とすることで、対処したい。

5. 結論と展望

オンラインジャッジで課題を与える際に、課題の難易度を適切にしなければ、十分な教育効果が得られない可能性がある。そこで、課題に対して類似している問題の内容および参考実装を公開するアプローチを提案した。提案手法の検証のため、WOJ Recommender というシステムを作成して、早稲田大学の講義で実験を実施した。

その結果、学生全体のうち4人に1人が本システムを使用した。また非介入年度の正答数と比較して、介入年度の正答数は統計的優位に向上したことが分かった。これにより、本提案手法は課題の難易度を下げることができ、課題の教育効果を改善できることが示唆された。一方で学生のアンケート評価としては、学生側に表示した問題の課題に対する類似度や、表示した問題の問題文の分かりやすさといったものの改善が求められているものがあった。

今後の展望として、類題を検出する手法の改善が挙げられる。現在はトークン列としての類似性のみでソースコード間の類似度を測っているため、構造的な類似度を考慮していない。そのため、構造的類似度を考慮してソースコード間類似度を測る手法を採用したい。他には、問題セットの拡大が挙げられる。現在の制約下ではAOJのみが使用可能なオンラインジャッジとなっている。そのため検出される類題の限界が早い段階で来るとされる。より沢山の問題セットを用いることのできる制約となるようなシステムに置き換えることで、より教育効果の高い問題を提案できるようにするものと思われる。

参考文献

[1] Kurnia, A., Lim, A. and Cheang, B.: Online Judge, *Computers & Education*, Vol. 36, No. 4, pp. 299–315

(online), DOI: 10.1016/s0360-1315(01)00018-5 (2001).
[2] 中村慎司, 筑捷彦: プログラミング授業支援システム WOJ の開発, 情報処理学会第 77 回全国大会, pp. 939–940 (2015).
[3] 渡部有隆: オンラインジャッジの開発と運用 - Aizu Online Judge -, 情報処理, Vol. 56, No. 10, pp. 998–1005 (2015).
[4] Cutts, Q., Barr, M., Bikanga Ada, M., Donaldson, P., Draper, S., Parkinson, J., Singer, J. and Sundin, L.: Experience Report: Thinkathon - Countering an "I Got It Working" Mentality with Pencil-And-Paper Exercises, *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE '19*, pp. 203–209 (online), DOI: 10.1145/3304221.3319785 (2019).
[5] Yuan, H. and Cao, Y.: Hybrid pair programming - A promising alternative to standard pair programming (2019).
[6] 藤原新: プログラミング学習者向け全身のソースコード提示システム TAMBA (2016).
[7] Shirakawa, M., Hara, T. and Nishio, S.: N-gram IDF: A global term weighting scheme based on information distance, *WWW 2015 - Proceedings of the 24th International Conference on World Wide Web*, pp. 960–970 (online), DOI: 10.1145/2736277.2741628 (2015).
[8] Salton, G., Wong, A. and Yang, C.-S.: A Vector Space Model for Automatic Indexing, *Communications of the ACM*, Vol. 18, No. 11, pp. 613–620 (1975).
[9] 山中裕樹: TF-IDF 法と LSH アルゴリズムを用いた高速な関数単位のコードクローン検出法 (2014).
[10] 小林勇揮, 水野修: N-gram IDF を利用したソースコード内の特徴的部分抽出手法, ソフトウェア・シンポジウム 2017 in 宮崎, pp. 46–55 (2017).
[11] Kasahara, R., Sakamoto, K., Washizaki, H. and Fukazawa, Y.: Applying Gamification to Motivate Students to Write High-Quality Code in Programming Assignments, *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE '19*, pp. 92–98 (online), DOI: 10.1145/3304221.3319792 (2019).
[12] Chen, E. T.: Program Complexity and Programmer Productivity, *IEEE Transactions on Software Engineering*, Vol. SE-4, No. 3, pp. 187–194 (online), DOI: 10.1109/TSE.1978.231497 (1978).
[13] 菅民郎: 実例でよくわかるアンケート調査と統計解析, 株式会社ナツメ社 (2011).
[14] 岩佐英彦, 宿久洋: 授業評価・市場調査のための「アンケート」調査・分析ができる本, 株式会社秀和システム (2009).
[15] 池田郁男: 統計検定を理解せずに使っている人のために II, 化学と生物, Vol. 51, No. 6, pp. 408–417 (2013).
[16] CHANCHAL K.ROY: Detection and analysis of near-miss software clones (2009).
[17] Khatoon, S., Li, G. and Mahmood, A.: Comparison and evaluation of source code mining tools and techniques: A qualitative approach, *Intelligent Data Analysis*, Vol. 17, No. 3, pp. 459–484 (online), DOI: 10.3233/IDA-130589 (2013).
[18] Zhao, G. and Huang, J.: DeepSim: Deep Learning Code Functional Similarity, *ESEC/FSE*, pp. 141–151 (online), DOI: 10.1145/3236024.3236068 (2018).