論文

コピペ防止エディタにおける丸写し検出機能の開発
Development of Function for Detection of Entirely Copy by Key Input in Copy and Paste Prevention Editor

大木　優[1)]
Masaru Ohki

馬場　博己[2)]
Hiromi Baba

高橋　圭一[3)]
Keiichi Takahashi

■Abstract

In this paper, we describe the development for the function of detection of entirely copy by key input in the Epron editor that prevents copy and paste. The Epron Editor is an editor that cannot copy and paste text from outside. Since the student's ID number is used as the encryption key, the text using the Epron editor can be estimated that the student has written. By submitting a report using the Epron editor, it is not able to submit a report copied another person's text. However, even using the Epron editor, there is the possibility to create a report by coping entirely the other's text by key input. The function to detect entirely copy by key input presents the text that may be entirely copy by using the key input time interval. By adding this function to the editor, it is able to prevent copy and paste, and entirely copy by the key input by using the Epron editor. We think that the improvement of learning in the subject which submits a report electronically is expected,

Key Words; Copy protection, Editor, Detection of Entirely Copy

1．はじめに

著者らは、これまで、レポート演習やプログラミング演習などでコピペ（コピーアンドペースト）を防ぐエディタを開発してきた。パソコンを使ったレポート課題などでは、コピペを容易に行えるため、自分で書いた、あるいは解いたものではなく、他の人やネットからのコピーをしたものを提出するという問題が指摘されている[1]。この問題を解決するために、筆者らは外部からのテキストをエディタ内部に取り込めないようにすることにより、電子的なコピペを防止するエディタEpronを開発し、演習等に使用してきた[2]。

Epronエディタを使用することにより、コピペの可能性はなくなり、テキストは受講生がキー入力をしたものと保証される。教師の立場では、コピペではなく自分でキー入力して課題を完成させたということは、教育効果が大きいと考えている。しかし、他の受講生のレポートを見ながらキー入力してレポートを作成している可能性もあるため、丸写しを検出する機能を開発することにした。コピペ防止機能と丸写し検出機能を組み合わせることにより、受講生はより自ら課題に取り組まざるを得なくなり、パソコンを使った演習やレポート作成の課題での学習効果が向上すると考えている。

本論文では、Epronエディタへの丸写しを検出する機能の開発について述べる。丸写しの検出は、キー入力時間間隔から、考えながらのキー入力と、丸写しのキー入力を識別することにより実現する。本開発では、被験者が考えながらと丸写しの条件で入力したキー入力履歴から、それらの条件のキー入力の識別を行う条件を求め、それをもとに入力されたテキストが丸写しであるか否かの判定を行う。エディタでは、編集操作も行われるため、判定では、キー入力時間とエディタでの文字の入力時間を一致させる必要がある。本開発では、エディタで編集時のキー入力時間を保持することにより、エディタ上での文字の入力時間を求める。

丸写ししたと推定された文字列はエディタ上に判明できる形で表示する。これにより、丸写しをした可能性がある範囲の把握が可能となり、丸写ししたと推定される部分への口頭試問などが可能となる。これにより、丸写しの抑止効果になると考えている。

2．関連研究

コピペの問題は、大学の教育でも大きな問題である[1]。コピペを防ぐには、①提出された文書が、インターネットなどの資料を流用しているか類似性を調べる、②物理的にコピペを許さない、③作成過程からコピペを検出する、アプローチがある。報告者らは、②のコピペが物理的にでき

1）近畿大学産業理工学部情報学科　教授　ohki@fuk.kindai.ac.jp
2）近畿大学産業理工学部情報学科　講師　baba@fuk.kindai.ac.jp
3）近畿大学産業理工学部情報学科　准教授　ktakahas@fuk.kindai.ac.jp

ないエディタを開発してきた[2]。ただし、この方法は、丸写しでの入力を検知できない。

①の研究では、上田らの研究[3]がある。レポート同士の類似性を評価するものである。プログラミング演習では、そもそも解答が似ているため、類似度の判定は、論述型の課題レポートより難しい可能性もある。また、この方法は、レポートがすべて提出されないと判定できない。

②のアプローチの先行研究には、Eclipse上のエディタを独自で開発したVamplewらの研究[4]がある。ローカルなコピーアンドペースト用のクリップボードを実装し、外部からのコピペを許さない方法である。外部へのコピーができるように、コピーする際、通常のクリップボードにもコピーするテキストを格納する。ペーストはローカルなクリップボードからしかできない。

③のアプローチとして松木らの研究[5]は、専用エディタでのキー入力やマウスクリックを記録し、レポートの評価を行うものである。評価は、記録した過程を再生し、レポートだけでなく作成過程も評価する。これにより、不正なコピペを行ったかどうか評価できる。レポートの再生課程を見ないと、コピペをしたかどうか判断できない。

森田の研究[6]は、キー入力やマウスクリックを記録し、プログラミングの作成過程を記録し、再生することにより、学習指導や評価を行うものである。画面のデータも取得している。この研究も、作成過程を再生して評価を行うため、大人数のレポートのコピペを判断するためには、時間がかかるものと思われる。

3．コピペ防止エディタ　Epron

コピペ防止エディタEpronは、Microsoft社Windowsに標準搭載のワードパッドに近い外見と機能を持つエディタである。しかし、コピペ機能が制限されている。自分のファイル、あるいはエディタ内でのコピペ、外部へのテキストのコピーはできるが、エディタ外部からエディタ内へのテキストのペーストはできない。ただし、外部からの画像の取込みは可能である。これは、レポートを作成する際、外部の画像などを使用できるようにするためである。

提出者本人が書いたことの証明は、図1に示すように、レポートに斜めに印刷された提出者の学籍番号（図1では、1234567890と印刷されている）で確認できる。学籍番号は、Epronエディタの初回利用時に登録する。この学籍番号のレポートは、その学籍番号の受講生のレポートとなる。エディタでのコピーとペーストの処理は、学籍番号に基づいた暗号化するため、学籍番号が違うエディタ間でのコピペはできない。そのため、他者の学籍番号のレポートを自分の学籍番号のエディタ内に取り込めない。

Vamplewらの研究らとの大きな違いは、汎用的なレポート課題にも使えるように、画像を取り込めることである。Vamplewらの研究は、プログラミング演習専用であるため画像の取り込みはできない。また、Epronエディタでは、自分で作成した過去のレポートからのコピペは認めている。レポート課題を提出する際、以前に提出したレポートの一部をコピーすることがあるためである。画像の取り込み、あるいは自分が作成したほかのファイルからの取り込みは、コピペにローカルのクリップボードを使うVamplewらの研究の実現方法では実現できないと考えられる。

しかし、Vamplewらの研究は、プログラミング演習を目的としてEclipseに組込んでいるため、プログラミング演習で使用するには、不便さが少ない。一方、Epronエディタを使用する際は、一旦、プログラム開発環境にプログラムコードをコピーして、実行させる必要がある。また、修正は、再度、Epronエディタで変更を行い、プログラミング開発環境で再度、実行させる手間が生じる。しかし、この方法は、使用するプログラミング開発環境に依存せず、広く使用できる利点がある。課題のレポート作成にはEpronエディタを使用することにより、レポートは、提出者が入力したものと保証される。

著者らは、データベースの演習にEpronエディタを利用している。使用しているデータベース開発環境に、Vamplewらの研究のように、追加の機能を組み込むことはできない。組込みができない開発環境を使用するためには、Epronエディタのようなアプローチが有力な一つの方法だと考えられる。データベース演習等では、受講生がSQL文をEpronエディタで作成し、データベース環境で実行した結果を画像として、Epronエディタに取込ませて、レポートを完成させるようにしている。実行結果などを取り込むためには、画像の取込みは不可欠である。また、

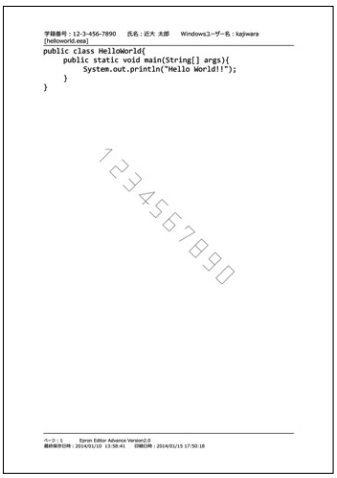

図1　Epronエディタの印刷例

普通の課題レポートでも、画像の取込みは必要である。画像を取り込める機能は、Epronエディタの利点である。

4．丸写し検出機能

Epronエディタを使用することにより、電子的コピーによるレポートの提出はなくなった。しかし、他の人のレポートを見て写して提出する可能性がある。これを検知するためには、2章で述べたレポート間の類似性を調べる①のアプローチが一つの方法である。しかし、プログラミング演習などでは、答えがあるため、各レポートは似てくる可能性があり、①の類似性を検知する方法では、簡単に検出できない可能性がある。また、すべてのレポートが提出された後でないと、類似性の検知ができない。

本開発では、2章の③のアプローチを使う。ただし、作成過程からコピペを判断するのではなく、作成した結果をもとに丸写しを検出する。本開発では、Epronを使ったキー編集履歴をもとに丸写しか否かの判定を行う。

本開発では、キーが丸写しされたという判定は、一定のキー入力数で長いキー入力時間間隔が少ない場合は、そのキー入力は丸写しをしたと判定する。図2は、自力での入力の条件でのキー入力数と入力の累積時間を示したものである。丸写しではなく、考えながら入力しているため、ところどころで入力はないのに時間が過ぎている個所がある[7]。グラフでは垂直の直線の部分がそれに対応している。図3は丸写しの条件でのキー入力数と入力の累積時間を示したものである。考えながらキー入力を行っていないため、なだらかな右肩上がりの線になっている。本開発では、考えながらキー入力を行うと、キー入力の停滞があることを利用する。

キー入力の時間間隔を記録した後、編集履歴から編集の最終結果である編集結果を作成し、そのキー入力による文字列に丸写し判定の結果を付与する。キー入力時間間隔を基にしているため、丸写しの検出精度は高くないと考える。

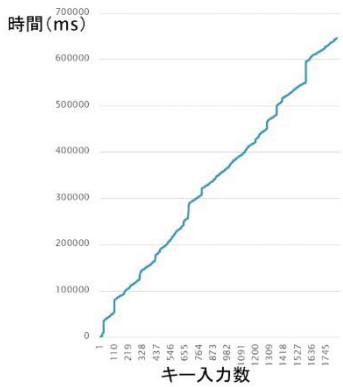

図２　自力での入力の条件でのキー入力数と入力の累積時間

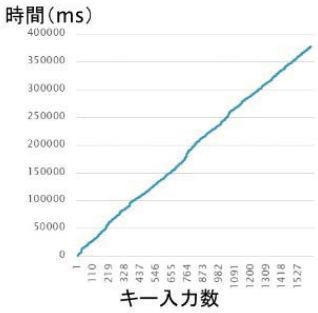

図３　丸写しの入力の条件でのキー入力数と入力の累積時間

る。精度を上げるためには、キー入力の時間間隔以外の量を使用する必要がある。例えば、Deleteキーの入力回数が考えられる。これも、これだけで高い精度が得られることはない[7]。

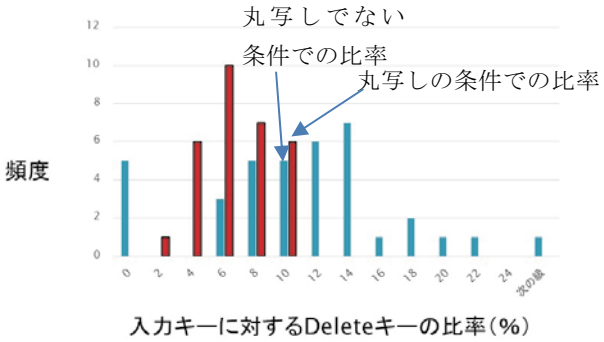

図４　３分ごとのキー入力数に対するDeleteキーの比率

図4は、3分おきのDeleteキーの比率を示したグラフである。濃い棒グラフ（隣接している右側のグラフ）が、丸写しの条件でのDeleteキーの比率であり、濃くない棒グラフは丸写しではなく考えながら入力したDeleteキー比率である。ただし、この実験からは、Deleteキーの比率で、丸写しか否かを判定することは十分ではないと判断した。どのような操作を組み合わせて、丸写しの高い精度を出すかは、今後の課題と考えている。

本機能の丸写し検出では、丸写しをしたと推定する文字列を提示する。丸写しの検出はキー入力時間間隔を使用する。しかし、レポート作成者は、編集操作を行うため、キー入力時間と編集結果の文字列は異なるのが一般的である。例えば、ある部分の文字列が別の位置に移動やコピペが行われる。編集結果の文字列に丸写しの推定結果を表示するためには、キー入力時間を編集結果の文字列にマッピングする必要が出てくる。本開発では、キー入力とその時のキー入力を編集操作にしたがって、編集操作を実行し、編集結果の文字列でのキー入力時間を求め、その時間に従って、丸写しの範囲を表示する。

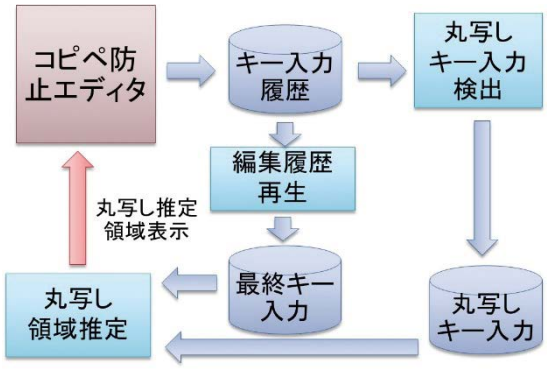

図５　丸写し機能のプログラム構成

本論文での丸写し機能のプログラム構成は図5のとおりである。キー入力履歴は、キー入力時間、キー入力終了時間、入力タイプ、入力前のカーソルの位置、入力前の文字数、入力前の文字、入力後のカーソルの位置、入力後の文字数、入力後の文字の情報を保持する。たとえば、文字列”34”が選択されているところに、“A”を入力すると、以下のような入力履歴が生成される。

9541, 0, inputChar, 2, 2,“34”, 2, 1,“A”

この履歴の形式は以下のとおりである。

① キー入力時間
② キー入力終了時間（検出できた場合のみ）
③ 入力タイプ
④ 入力前のカーソルの位置
⑤ 入力前の文字数
⑥ 入力前の文字
⑦ 入力後のカーソルの位置
⑧ 入力後の文字数
⑨ 入力後の文字

先の例では、開始9541ミリ秒後に、2カラム目の“34”という2文字列を、2カラム目に“A”という1文字で置き換えたことを示している。入力タイプの“inputChar”は文字の入力であることを示している。その次の“2, 2”は2カラム目からの2文字が選ばれている状態であることを示している。“34”は選ばれている文字列である。そのあとの、“2, 1”は、2カラム目に1文字が入力されたことを示している、ここでは“A”が入力された。この様な形式で操作履歴がすべて記録される。

編集履歴再生部では、入力履歴をもとに最終編集結果を作成する。入力履歴には、最終的な編集結果を作成するために必要な操作を記録する。キー入力だけではなく、削除、置換、コピー、ペースト、redo、undoなども記録される。

最終編集結果のそれぞれの文字には、キー入力の時間が付与されており、丸写しキー入力検出で検出された丸写しキー入力判定を使って、丸写しの判定が付与される。

エディタでは、編集結果に対して、丸写し判定結果を反映する。現時点では、エディタの文字に、丸写しの判定に従って、文字を色付けする。これにより、教員は学習者に対して、丸写したと判定された部分への口頭試問を行うことができるようになる。この結果、受講者のレポート取組みが改善されると期待できる。

例えば、“12”、“学校”と入力し、“学”を削除し、さらに、その削除をundoし、“12”を“学校”の後にコピーした場合、表1のようなキー入力の履歴が作成される。この履歴で作成されたキー入力時間は表2のようになる。削除されて、undoされた“学”の入力時間は、元のキーの入力時間である。また、8、9行目の“1”、“2”は、1、2行目の“1”、“2”をコピーしたものである。それらの入力時間は、実際にキー入力された時間となる。

表１　キー入力履歴の例

操作	入力履歴
“1”入力	2957, 0, inputChar, 0, 0, “”, 0, 1, “1”
“2”入力	3281, 0, inputChar, 1, 0, “”, 1, 1, “2”
改行入力	7233, 0, inputChar, 2, 0, “”, 2, 1, “¶”
“学校”入力	8858, 10491, inputNihongo, 3, 0,“”, 5, 4,“学校”
改行入力	12146, 0, inputChar, 5, 0, “”, 5, 1, “¶”
4文字目カーソル	14425, 0, setCursor,-1, 0, “”, 4, 0, “”
“学”をカット	15805, 0, cut, 3, 1, “学”, 3, 0, “”
undo	16783, 0, undo, 3, 0, “”, 3, 1, “学”
2文字目カーソル	18740, 0, setCursor, -1, 0, “”, 2, 0, “”
“12”をコピー	20068, 0, copy, 0, 2, “12”, 0, 2, “12”
6文字目カーソル	20394, 0, setCursor, -1,0, “”, 6, 0, “”
“12を”ペースト	21280, 0, paste, 6, 0, “”, 8, 2, “12”

注）¶は改行記号を示している。

表２　生成された最終キー入力

キー	入力時間
1	2957
2	3281
¶	7233
学	8858
校	9674
¶	12146
1	2957
2	3281

丸写しの領域の推定は、図6のように行う。表2のように作成された最終編集結果のキーの時間とキー入力時間をあ

わせて、丸写しはキー入力が単調であるテキスト領域を丸写し領域として推定する。一定のキー入力回数以上で単調な入力であると、エディタ上のそれらのキー入力に該当する文字を丸写しであると判断する。

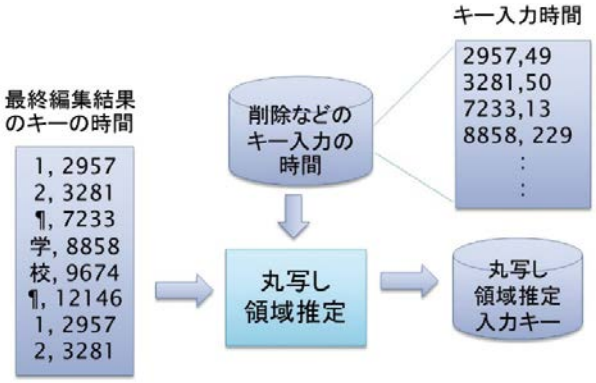

図6　丸写し推定

丸写しの基準作成には、被験者10名が丸写しで入力したデータと、自分で考えて入力したデータを利用した。基準には、キー入力の時間間隔を使用した。丸写しか否かを判定するキー入力数、そのキー入力の中で丸写しではないと判断するキー入力時間の時間間隔、その時間間隔を超えているキー入力数を変え、被験者が丸写しで入力したデータと、自分で考えて入力したデータを分離できる閾値を求めた。その結果は以下の通りである。

・キー入力数：151
・丸写しではないと判断するキー入力時間の時間間隔：2750ミリ秒
・その時間間隔を超えているキー入力変化量の数：2

このように、キー入力数が151ストロークの中で、キー入力間隔が2750ミリ秒以上のキーストロークが2個ある場合は、丸写しではないと判断する。そうでない場合は、丸写しと判断する。

5．評価結果

176文字のあらかじめ与えられた文を丸写しにして、そのあと、自由に文を続ける課題で評価を行った。その評価結果を表3に示す。被験者は4名である。

表3　評価結果

	被験者	1	2	3	4
丸写し入力	丸写し文字数	176	176	176	176
	丸写し判定文字数	104	164	123	88
	自力入力判定文字数	72	12	53	94
	丸写し入力判定文字比率	59.1%	93.2%	69.9%	46.6%
自力入力	自力文字数	162	176	192	116
	丸写し判定文字数	0	61	144	5
	自力入力判定文字数	162	115	48	111
	自力入力判定文字比率	100.0%	65.3%	25.0%	95.7%

前半の丸写しのキー入力の範囲の評価は、被験者4を除いて、50%以上丸写しと判定している。自力入力の範囲では、被験者3の自力入力比率25%と低い。これは、被験者3のキー入力の熟練度の高さの結果と思われる。

各被験者の判定結果を図7から図10に示す。丸写しと判定された文字列は、並下線で示されている。本来の判定は、丸写しの範囲は赤文字、自力の判定は緑文字で表示される。

図7の判定結果の被験者1は、キー入力の熟練度は高くない可能性があり、丸写しの範囲でも自力での入力と判定されている。一方、自力で入力する範囲では、すべて自力入力と判定されている。

図8の判定結果の被験者2は、丸写しの文はほぼ丸写しと判定されている。自力の入力部分は、35%くらい丸写しと判定されている。図9の判定結果の被験者3は、キー入力の熟練度が高いため、自力入力の範囲も80%以上が丸写しと判定されている。図10の判定結果の被験者4は、キー入力の熟練度が高くなく、丸写しの文で55%程度自力入力と判定されている。

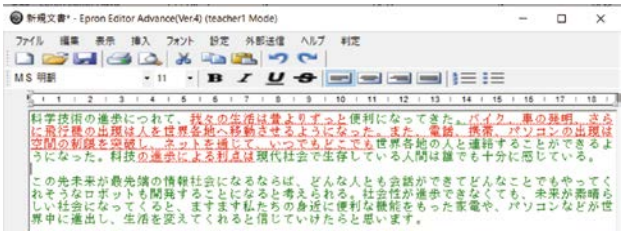

図7　被験者1の判定結果

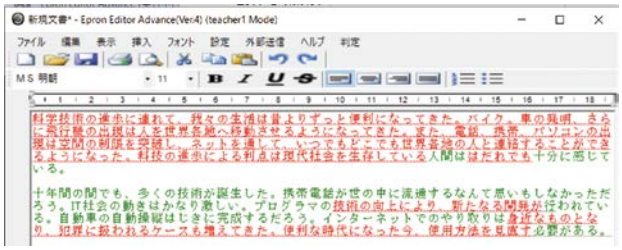

図８　被験者２の判定結果

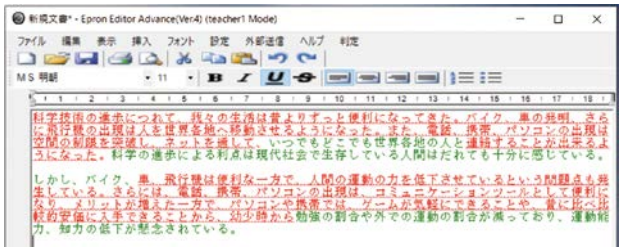

図９　被験者３の判定結果

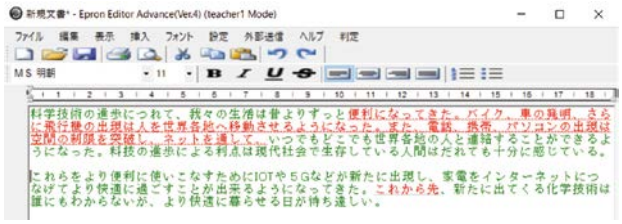

図10　被験者４の判定結果

表3に示すように、本開発での丸写し判定能力は高くない。これは、キー入力の熟練度に影響されると考えられる。本機能の利用方法としては、丸写しと判定された文章の範囲を口頭質問するのが一つの使い方と考えているため、精度が高い必要はないと考えている。

判定能力を上げるには、キー入力時間間隔以外の要素を取り入れることが考えられる。自力で入力している場合は、推敲に伴い、Deleteキーの入力が大きくなる傾向があるからだ。しかし、単純にDeleteキーの入力比率を考慮する実験の結果では大きな差は出なかった。これは、個人差が大きいと考えられる。改善案としては、受講生のキー入力速度で、キー入力間隔時間を正規化することは有力な方法かと考えている。しかし、この方法では、キー入力データの蓄積が必要である。

６．まとめ

本論文では、コピペ防止エディタEpronの丸写し検出機能について述べた。コピペ防止エディタEpronは、レポートのコピペ防止に使用できるエディタである。本機能をコピペ防止エディタに追加することにより、コピペ防止だけではなく、丸写しも抑制できると期待でき、学習者の学習取組みをより改善できる可能性がある。

今後、さらに、丸写し検出機能の評価を行いながら、丸写し推定能力の改善を図る計画である。4名の被験者の評価では、被験者のキー入力の熟練度によって、評価に大きな差があると考えられる。今後、熟練度を考慮した評価方式へ拡張したい。また、丸写し行った場合、キー入力が早くなる、キー入力速度の揺らぎが少なくなる、Deleteキー入力が減る、という傾向がある。それらの情報を活用していくことにより、丸写し検出の精度改善への対策を行う予定である。

謝辞　本システムの開発に当たって協力をいただいた、疋田公洋氏、梶原翔氏、田中太脩氏、本学部電算機センターに感謝いたします。

参考文献

[1] “大学レポートの「コピペ」、その深刻な問題とは”、ベネッセ 教育情報サイト、https://benesse.jp/kyouiku/201504/20150408-2.html（参照 2019-03-05）.

[2] 大木優、高橋圭一：“情報系演習向け演習支援システム”、情報処理学会論文誌教育とコンピュータ（TCE）, 1(4), 48-54,（2015-12-09）.

[3] 上田和志、富永浩之：類似性に基づくレポート剽窃の検出ツールの改良とソースコードへの適用、電子情報通信学会技術研究報告，ET，教育工学 110（453）, 119-124,（2011-02-25）.

[4] Peter Vamplew and Julian Dermoudy, An Anti-Plagiarism Editor for Software Development Courses, ACE ‘05 Proceedings of the 7th Australasian conference on Computing education - Volume 42, Pages 83-90.

[5] 松木保浩、稲垣嘉信、坂本久他：レポート作成過程評価システムの設計，情報処理学会研究報告コンピュータと教育（CE），2006，108（2006-CE-086），59-66,（2006-10-21）.

[6] 森田直樹：グローバルフックを用いたプログラミング過程可視化システム，情報科学技術フォーラム講演論文集，12(3)，61-64（2013-08-20）.

[7] 大木 優、田中太脩、馬場博巳、高橋圭一：情報系演習における受講態度の定量的評価の試み，電気学会全国大会（2016）.