特徴トークンに注目した Smith-Waterman アルゴリズムに基づく剽窃ソースコードの自動検出手法

情報数理応用研究　5211C034-1　日比健太
指導教員　後藤正幸

Automatic Detection of Plagiarized Source Codes Based on Smith-Waterman Algorithm Focusing on Feature Terms

HIBI Kenta

1　研究背景と目的

教育機関におけるプログラミング技術に関する演習では，コピー＆ペーストなどを用いた剽窃が問題となっている．剽窃が行われることで“ 教員が受講者の適切な評価を行うことが出来ない ”,“ 受講者の真の理解度を把握できない ”,“ 受講者のプログラミング技術が上達しない ” といった問題につながるため，学生のソースコードの剽窃を検出する技術を確立することの意義は大きい．一方で，教育機関におけるプログラミング技術に関する演習科目では，剽窃レポート等には厳正な処罰が課されると明文化されていることが多く，剽窃を誤検出した場合，学生に不要な処罰が課されてしまう．そのため，自動的に剽窃を検出する手法においては，剽窃の誤検出は避けなければならない課題のひとつである．しかし，剽窃の誤検出を防ぐことと剽窃を検出することはトレードオフの関係がある．この様な状況に適した剽窃検出を行うため，本研究では剽窃の誤検出を可能な限りゼロに防ぎつつ，より多くの剽窃の検出を前提とした手法について検討を行う．

剽窃ソースコードの自動検出手法としてソフトウェアバースマークを用いてアルゴリズムの類似度を比較する手法 [1]，Smith-Waterman アルゴリズム [2] を用いて単語列の一致性を検出し，類似度を計算する手法 [3] が提案されている．これらの手法は，一般的なソースコードの剽窃検出において優れた性能を示すことが知られている．しかし，学生の記述するソースコードは「教科書や授業内容等の知識の参照元が同じである」という特徴から，アルゴリズムやデータ構造が類似しやすいため，単語列の一致性のみによって類似度を計算することは，誤判定に結びつく可能性がある．

一方，上記の問題点の影響を受けずに剽窃を検出する手法として，コーディングスタイルである作成者の書く癖を比較する大野らの手法 [4]，及び大野らの手法を改良した Hibi らの手法 [5] が提案されている．しかし，これらの手法は，コーディングスタイルのみに着目して類似度を計算しているため，全単語列の一致性から類似度を計算する手法 [3] に比べて，扱える情報量が少なく，改善の余地があると考えられる．以上のことから，単語列の一致性とコーディングスタイルの類似性を同時に評価して類似度を計算することができれば，学生の剽窃ソースコード検出において，優れた性能を示すと考えられる．

そこで本研究では，単語列の一致性とコーディングスタイルの類似性の双方を加味した新しい手法を構築することで，剽窃の誤検出を可能な限りゼロに防ぎつつ，既存手法よりも剽窃の検出率を高める手法を提案する．提案手法を学生により作成されたソースコードに適用することで，既存手法よりも優れた性能が得られることを示す．

2　既存研究

2.1　Smith-Waterman アルゴリズムを用いた手法 [3]

Smith-Waterman アルゴリズムは局所的に酷似した部分を特定するために２つの系列の比較を行う手法である [2]．これは，動的計画法の一種であり，２つの系列が与えられた時，系列を構成する要素ごとに順に比較を行い，系列全体の比較を行う．

系列を単語系列として Smith-Waterman アルゴリズムを文書に適用した Irving の手法 [3] では，I 個の単語からなる文書$X=(x_1,\cdots,x_i,\cdots,x_I)$ と J 個の単語からなる文書 $Y=(y_1,\cdots,y_j,\cdots,y_J)$ 間での類似した連続単語系列を検出するために，これら単語系列の類似度を算出している．ここで，単語の組 (x_1,y_1) から単語の組 (x_i,y_j) までの累積スコアを $S_{i,j}$ とし，アルゴリズムを以下に示す．下記アルゴリズムにおいて，α は x_i と y_j が一致した際にスコアを増加させるためのパラメータ，β，γ，δ は，それぞれ x_i に挿入，削除，入れ換えが行われた場合にスコアをどれだけ下げるかを意味するパラメータである．

[Smith-Waterman アルゴリズム]

Step1) 単語の組 (x_1,y_1) から探索し，最初に一致する単語の組 $(x_{i'},y_{j'})$ を始点とする．ここで，$S_{i'-1,j'}=S_{i',j'-1}=S_{i'-1,j'-1}=0$ とおく．

Step2) i，j を増やしながら，始点以降の累積スコアを次式を用いて計算する．$S_{i,j}=0$ となる時，Step3 へ．

$$S_{i,j}=\begin{cases} S_{i-1,j-1}+\alpha, & \text{if} \quad x_i=y_j \\ \max\begin{pmatrix} 0, \\ S_{i-1,j}-\beta, \\ S_{i,j-1}-\gamma, \\ S_{i-1,j-1}-\delta \end{pmatrix} & \text{otherwise.} \end{cases} \tag{1}$$

Step3) $S_{i,j}>0$ を満たす範囲の中で，$x_i=y_j$ となる単語の組のうち，始点から最も離れた単語の組を終点とする．始点から終点までを類似した連続単語系列として抽出する．

2.2 前後方記述スタイルモデルを用いた手法 [4]

大野ら [4] はソースコード記述における作成者固有の癖を“ 記述スタイル ”として確率モデルで推定した後，他のソースコードとの類似度を比較し，剽窃を検出する手法を提案している．具体的には，“ { ”，“ (”，“ + ”などの頻繁に用いられる 14 種の文字を基準トークン z_k $(1 \leq k \leq 14)$，その前後に出現する「空白」「改行」「タブ」の 3 種の文字を着目トークン $v_l (1 \leq l \leq 3)$ と定義し，これらのトークンの出現順のみを考慮したモデルを構築する．

2.3 コーディングスタイルモデルを用いた手法 [5]

Hibi らの手法 [5] で提案されたモデルでは，作成者の記述スタイルをより詳細に表現するために，考えられる全ての関係をモデル化することを目的とする．注目する記号は，大野らの手法 [4] と同じく基準トークン z_k および着目トークン v_l の全 17 種としている．ここで，大野らの手法 [4] では，基準トークン z_k に用いられる 14 種類の記号が明記されていない．そこで，Hibi らの手法 [5] では，基準トークン z_k に用いられる 14 種類の記号を定義している．基準トークン及び着目トークンを表 1,2 に示す．

表 1. 基準トークン

k	1	2	3	4	5	6	7
z_k	+	-	/	*	%	{	}
k	8	9	10	11	12	13	14
z_k	(	)	=	’	”	〈	〉

表 2. 着目トークン

l	1	2	3
v_l	空白	TAB	改行

大野らの手法 [4] では，各着目基準トークン間の関係は考慮されていない．しかし，ソースコード記述における作成者固有の癖は“); ”や“){ ”などの記号列に現れる．そこで，Hibi らの手法 [5] では，基準および着目トークン間の全ての関係を考慮したコーディングスタイルモデルNを提案している．

基準および着目トークンの全 17 種の 1 次マルコフで表現することからコーディングスタイルモデルNは，各関係の出現確率を示すパラメータ $\theta = (\theta_1, \cdots, \theta_h, \cdots, \theta_{289})$ によって規定され，最尤推定法によって推定される．

3 既存研究の問題点と本研究への展開

Smith-Waterman アルゴリズムを用いた手法 [3] では，剽窃検出において優れた性能を示すことが知られている．しかし，学生のソースコードは，アルゴリズムやデータ構造が類似しやすいため，剽窃でない作成者を剽窃であると判定してしまう可能性がある．一方，手法 [4],[5] では，この問題点の影響を受けずに剽窃を検出することが可能であるが，手法 [5] に比べて扱える情報量が少ないため，低い性能となることが考えられる．

そこで本研究では，アルゴリズムやデータ構造が類似した場合でも，適切に剽窃を検出できるように，コーディングスタイルを考慮した Smith-Waterman アルゴリズムの拡張を行う．

4 提案手法

本研究では，コーディングスタイルを用いた手法 [3] の拡張を行うため，x_i が特定の文字であるか否かによって，重みを分けて累積スコア $S_{i,j}$ を計算する．ここで，手法 [4],[5] において，着目トークン v_l，及び基準トークン z_k がコーディングスタイルをよりよく表現できていることが示されている．そこで，提案手法では，着目トークン v_l，基準トークン z_k から成る特徴トークン集合 $\mathscr{W} = \{v_l, z_k : 1 \leq l \leq 3, 1 \leq k \leq 14\}$ を定義し，x_i が特徴トークン集合 $\mathscr{W}$ に含まれるか否かによって場合分けを行うことで，コーディングスタイルを用いた手法 [3] の拡張を行う．

具体的には，x_i と y_j が一致している場合，x_i が特徴トークン集合 $\mathscr{W}$ に含まれるか否かによって，スコアを変えられるものとし，その値を決めるパラメータ α_1，α_2 を Smith-Waterman アルゴリズムに導入する．また，x_i に対して挿入，削除，入れ換えのいずれかの操作が行われた場合，x_i が特徴トークン集合 $\mathscr{W}$ に含まれるか否かによって，スコアを減少させる量を決めるパラメータ β_1，β_2，γ_1，γ_2，δ_1，δ_2 を Smith-Waterman アルゴリズムに導入する．従って，上記 8 つのパターンに場合分けすることができ，提案アルゴリズムのパラメータ $\lambda = (\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2, \delta_1, \delta_2)$ を構成する．この拡張により，スコア $S_{i,j}$ を計算する際，特徴トークンが一致または変更された文字の重みを加えることができ，文字列の一致性とコーディングスタイルを同時に評価して類似度を計算できるようになる．

また，Smith-Waterman アルゴリズムでは，単語系列を比較してスコア $S_{i,j}$ を計算している．しかし，単語“ n ”と単語“ n1 ”が与えられた時，文字“ n ”に文字“ 1 ”が加わったと解釈する方が，2 文書間の類似度をより詳細に測ることができる．また，提案手法では，特徴トークンが文字単位で定義されている．そのため，提案手法では，文字単位でスコア $S_{i,j}$ を計算することとした．以下に，提案アルゴリズムを示す．

[提案アルゴリズム]

Step1) 文字の組 (x_1, y_1) を始点とし，$S_{0,1} = S_{1,0} = S_{0,0} = 0$ とする．

Step2) i，j を増やしながら，始点以降の累積スコアを次式を用いて計算する．

$$
S_{i,j} = \begin{cases}
S_{i-1,j-1} + \alpha_1, & \text{if} \quad x_i = y_j \cap x_i \in \mathscr{W} \\
S_{i-1,j-1} + \alpha_2, & \text{else if} \quad x_i = y_j \cap x_i \notin \mathscr{W} \\
\max \left(\begin{array}{l} 0, \\ S_{i-1,j} - \beta_1, \\ S_{i,j-1} - \gamma_1, \\ S_{i-1,j-1} - \delta_1 \end{array} \right) & \text{else if} \quad x_i \in \mathscr{W} \\
\max \left(\begin{array}{l} 0, \\ S_{i-1,j} - \beta_2, \\ S_{i,j-1} - \gamma_2, \\ S_{i-1,j-1} - \delta_2 \end{array} \right) & \text{otherwise.}
\end{cases} \tag{2}
$$

Step3) 終点 (x_I, y_J) までスコアを計算し，スコア $S_{I,J}$ をソースコードXとソースコードYにおける最終的なスコア $S_{X,Y}$ とする．

Step4) ソースコードXとソースコードYが完全に一致している時 $\mathrm{Sim}_{X,Y} = 1$，全く一致していない場合 $\mathrm{Sim}_{X,Y} = 0$ となるように，ソースコードXとソースコードY間の類似度 $\mathrm{Sim}_{X,Y}$ を次式で計算する．ただし，N はソースコードXに含まれる文字のうち，$\mathscr{W}$ に含まれる文字数とする．

$$\mathrm{Sim}_{X,Y} = \frac{S_{X,Y}}{N\alpha_1 + (I-N)\alpha_2} \tag{3}$$

Step5) ソースコードXとソースコードY間の類似度 $\mathrm{Sim}_{X,Y}$ が閾値 η_λ 以上の時，ソースコードXとソースコードYを剽窃ソースコードのペアであると判定する．

ソースコード全ての組み合わせに対して，Step1 から Step5 を行うことで，剽窃とみられるソースコードのペアを全て検出する．

5 評価実験

提案手法の有効性を検討するために実データを用いて評価実験を行った．

実験データの作成にあたっては，教科書で使用されている例題 5 題を対象とし，18 名の大学生に計 80 件のソースコードの作成を依頼した．一方，13 名の大学生にこれらのソースコードを剽窃した計 40 件の剽窃ソースコードの作成を依頼した．これらを実験データとし，剽窃したソースコードが正しく検出されるか否かを確認する．

5.1 実験条件

[テストデータ]

- 被剽窃ソースコード　80 件
- 剽窃ソースコード　40 件　計 120 件

[閾値]

大野らの手法 [4] 及び Hibi らの手法 [5] では，ソースコード間の JS 情報量 [6] の値が小さい程，2 つのソースコードが類似していると判定されるため，閾値は精度 1.00 を保つ最大の閾値とする．また Irving の手法 [2] 及び提案手法では，$\mathrm{Sim}_{X,Y}$ の値が大きい程，ソースコードXとソースコードYが類似していると判定されるため，閾値は精度 1.00 を保つ最小の閾値とする．

5.2 実験方法

Irving の手法 [3]，大野らの手法 [4]，Hibi らの手法 [5] では，それぞれ類似度の算出方法と剽窃の判定方法が異なる．本実験においては，提案した類似度の算出方法の有効性を示すため，大野らの手法 [4] 及び Hibi らの手法 [5] では，Hibi らの手法の剽窃判定方法を用い，Irving の手法 [3] 及び提案手法では，提案手法の剽窃判定方法を用いて実験を行った．また，Irving の手法 [3]，及び提案手法の両手法において，文字ごとにスコア $S_{i,j}$ を計算する．評価尺度としては，次式に示す精度と検出率を用いた．

$$\text{精度} = \frac{\text{出力された剽窃ペア数}}{\text{出力された総ペア数}} \tag{4}$$

$$\text{検出率} = \frac{\text{出力された剽窃ペア数}}{\text{総剽窃ペア数}} \tag{5}$$

6 実験結果と考察

6.1 実験結果

提案手法の λ=(1,1,3,1,1,3,3,3) における精度と検出率と閾値の関係を図 1 に示す．図 1 より，精度 1.00 を保つ最小の閾値は 0.41 程度であることが分かる．また，精度と検出率の両者をバランスよく保つ閾値は 0.23 程度であり，その時の精度と検出率は 0.95 程度の値を示している．

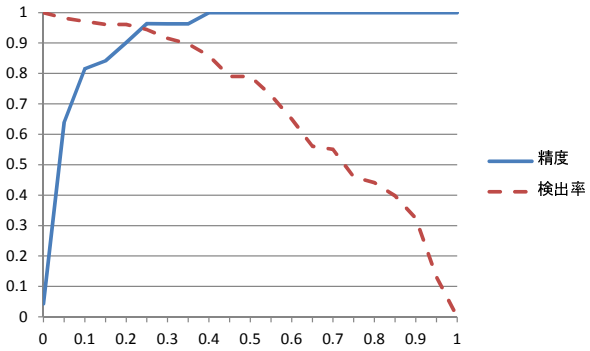

図 1. 精度と検出率と閾値の関係

次に各手法における検出率を図 2 に示す．ここで，Irving の手法 [3] 及び提案手法における設定すべきパラメータの組み合わせは無数に存在するため，パラメータの取りうる範囲を 1,2,3 に限定する．また，Irving の手法 [3] では，全てのパラメータの組み合わせ（3^4 通り）における検出率の最大値を用い，提案手法では，全てのパラメータの組み合わせ（3^8 通り）における検出率の TOP10 を用いるものとした．

図 2 より，全ての既存手法と比較し，提案手法の検出率が優れており，提案手法の有効性を示せた．また，Smith-Waterman アルゴリズムの検出率が手法 [4],[5] の検出率よりも高くなっている．

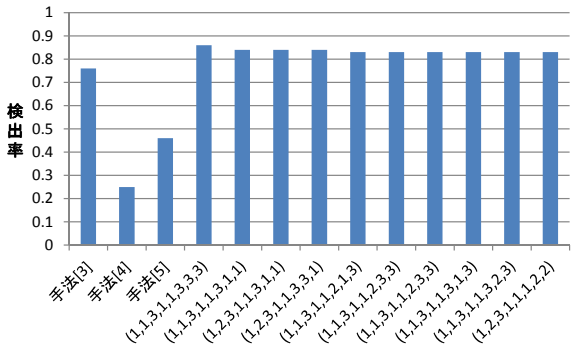

図 2. 各手法における検出率

提案手法では，β_1，β_2，γ_1，γ_2，δ_1，δ_2 のいずれかのパラメータが選択されてスコアが減少するため，その絶対値ではなく，他のパラメータとの相対的な値が意味を持っている．しかし，この相対的な関係を全てしらみ潰しに実験し考察することは困難であるため，パラメータ β_1，β_2，γ_1，γ_2，δ_1，δ_2 を 1 に固定し，α_1 のみを変化させた場合，及び α_2 のみを変化させた場合の検出率の推移をそれぞれ図 3，図 4 に示す．

図 3 より，α_1 の値を増加させた場合，検出率は低下していることが分かる．また図 4 より，α_2 の値を増加させた場合，α_2=3 までは検出率は増加し，その後 α_2=100 まで増加させると検出率は徐々に低下している．従って，α_2 の値を適切に設定した場合に検出率が向上していることから，提案手法において α_2 が α_1 より検出率の増加に寄与することが分かる．また，α_2=1 の場合の検出率と，α_2=100 の場合の検出率は同等である．

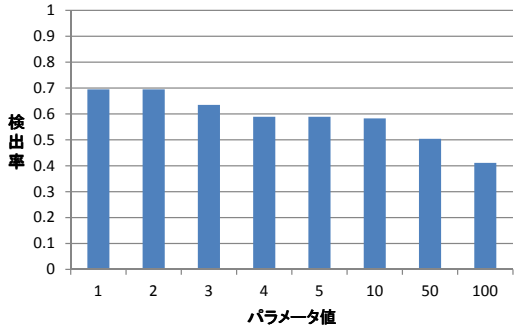

図 3. α_1 を変化させた時の検出率の推移

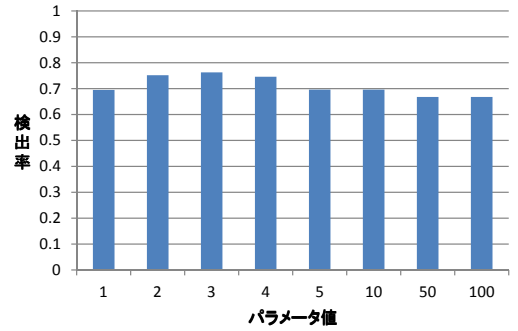

図 4. α_2 を変化させた時の検出率の推移

6.2 考察

提案手法のスコア計算において，α_1 の値を増やすことは，「x_i と y_j が共に特徴トークンであり，その文字が一致した時」のスコア $S_{i,j}$ を大きくすることを意味している．一方，特徴トークンはコーディングスタイルを表現しており，α_2 が α_1 より検出率の増加に寄与していることから，ソースコードの剽窃検出においては，コーディングスタイルよりも文字列の一致性を重視することで剽窃検出の性能を高められることが明らかとなった．

また，今回の実験用プログラムを作成した大学生は，ある程度のプログラミング技術を有していたため，教科書等を参考にせずに自力で課題に取り組んだと考えられる．従って，本実験に用いた各作成者のソースコードは，アルゴリズムやデータ構造がそれほど類似しておらず，Irving の手法 [3] でも，適切に剽窃を検出できたことが考えられる．そのため，類似度計算により多くの情報を扱える Irving の手法 [3] の検出率が Hibi らの手法 [5] よりも高くなった可能性がある．しかし，このようにアルゴリズムやデータ構造が類似していない場合でも，提案手法の検出率が Irving の手法 [3] の検出率よりも優れている．この提案手法の検出率と Irving の手法 [3] の検出率の差は，文字列の一致性にコーディングスタイルを加味したか否かによって発生している．従って，アルゴリズムやデータ構造が類似しているか否かを問わず，文字列の一致性のみではなく，コーディングスタイルも考慮することで検出率の増加が期待できることが明らかとなった．

また，提案アルゴリズムでは，スコア計算におけるパラメータを増やしたことにより，Smith-Waterman アルゴリズムに比べてより詳細に文書間の類似度を表現することが可能となった．しかし，アルゴリズムが複雑になったため，時間計算量がかかるという問題がある．提案手法においては，1 つ 1 つのソースコードの記述量（長さ）が増えた場合，時間計算量はかなり増加してしまう．しかしながら，ソースコードの数を増やした場合の時間計算量の増加はそれほど大きくはない．そのため，提案手法は，多数の学生が参加し比較的簡単な演習を行う，教育機関におけるプログラミング技術に関する演習に適した手法であるといえる．

7 まとめと今後の課題

本研究では，コーディングスタイルの類似性を考慮するために，特徴トークンに注目した Smith-Waterman アルゴリズムに基づく剽窃検出手法を提案した．実際に大学生の書いたソースコードの剽窃検出に適用することにより，既存研究よりも検出率を高められるという点で提案手法の有効性を示すことができた．

今後の課題として，提案手法で用いるパラメータ λ の最適な設定方法を考察することで，様々なソースコードに適用できるより実用的な手法となることが考えられる．

参考文献

[1] H. Tamada, M. Nakamura, A. Monden, and K. Matsumoto, “Java Birthmarks-Detecting the Software Theft,” *IEICE Transactions on Information and Systems*, Vol. E88-D, No. 9, pp. 2148–2158, 2005.

[2] T.F. Smith, and M.S. Waterman, “Identification of common molecular subsequences,” *Journal of Molecular Biology*, vol.147, pp. 195–197, 1981.

[3] R. W. Irving, “Plagiarism and collusion detection using the Smith-Waterman algorithm,” *Dept of Computing Science Technical Report*, pp.1–24 , 2004.

[4] 大野麻子, 村尾元, “前後方記述スタイルモデルによる授業課題ソースコード作成者特徴の抽出,” 第 37 回知能システムシンポジウム資料, pp. 99–104, 2010.

[5] K. Hibi, G. Kumoi, K. Mikawa, M. Goto, “Automated Source Code Plagiarism Detection based on a Coding Style Model,” *Asia Pacific Industrial Engineering and Management Society*, ID Number 147, 2011.

[6] B. Fuglede, and F. Topsoe, “Jensen-Shannon Divergence and Hilbert Space Embedding,” *IEEE International Symposium on Information Theory*, 2004.