予約語の影響を考慮した Smith-Waterman アルゴリズムの剽窃発見

1X11C091-1　野村幸弘
指導教員　後藤正幸

1　研究背景

大学などで実施されるプログラミング演習科目において，学生間の剽窃は様々な問題を引き起こす．学生間で剽窃が行われた場合,「学生のプログラミング技術を教員が正しく評価できない」,「学生のプログラミング技術が上達しない」などの問題につながる．このため，教員は剽窃を発見し適切な指導を行うことで，学生のプログラミング技術取得を促す必要がある．しかし，学生の全プログラムを人手で確認することによる剽窃発見は非常に大きな労力を伴う．このような作業を軽減するための方法として剽窃自動検出アルゴリズムが存在し，本研究ではこの技術に着目する．

剽窃自動検出アルゴリズムには様々なものが存在するが，本研究ではこのうち特に精度の高い Smith-Waterman アルゴリズム [1] をベースとした日比らの剽窃発見手法 [2] に着目する．この手法は作成者のコーディングスタイルを考慮しつつ，ソースコードを文字単位で比較し，その一致度合いをスコアとして数値化することで剽窃の発見を行う．しかし，この手法は予約語など必ずまとめて用いられる文字列も文字単位で比較する．予約語同士が一致する場合はその文字列全体が一致するため，予約語以外の文字同士が一致する場合よりも，スコアが高く算出されてしまう．さらに，予約語は剽窃の有無に関わらず必ず使用されるため，不要にスコアが上がってしまい，剽窃でないソースコードまで誤検出する恐れがある．予約語の中には C 言語の for や while のように制御構造を規定するものがあり，これらに続く文字列はアルゴリズムの構造を規定しているため，ここが一致しているプログラム同士はアルゴリズムが類似していると考えられる．

そこで本研究では，上記の問題点を解決するため日比らの手法に加え，予約語などの使用頻度が高い単語については文字単位ではなく単語単位で処理を行い，後続する文字列がソースコードの構成に関わる可能性が高い予約語については，それらの後続文字列のスコアを一定値上昇させることで剽窃発見を行う方法を提案する．これにより予約語以外の剽窃箇所が相対的に際立ち，剽窃の検出精度が上がると考えられる．提案手法の有効性を学生が作成した C 言語のソースコードを用いた実験を行い，有効性を示す．

2　日比らの剽窃発見手法 [2]

Smith-Waterman アルゴリズム [1] を基にしたソースコード間の剽窃検出手法として，日比らの手法が存在する [2]. この手法は，作成者の癖を表すと考えられる記号を「基準トークン」,「着目トークン」として定義し，これらの文字とそれ以外の文字とでスコアの値を変えることで従来の Smith-Waterman アルゴリズムよりも高い精度で剽窃の検出を可能としている．ただし，ここでの基準トークンとはプログラミング上で必ず出現する 14 種類の記号を表したものであり，これを $Z_k(1 \leq k \leq 14)$ とする．着目トークンは「空白」,「改行」,「タブ」の 3 種類でこれを $v_l(1 \leq l \leq 3)$ と定義する．これらを要素とする特徴語集合 $\mathcal{W}$ を $\mathcal{W} = \{v_l, z_k : 1 \leq l \leq 3, 1 \leq k \leq 14\}$ とし，比較する文字が $\mathcal{W}$ の要素か否かで Smith-Waterman アルゴリズムでのスコアを調整する．

いま I 文字からなるソースコードを $\boldsymbol{X} = (x_1, .., x_i, .., x_I)$ とする．ただし，x_i はソースコード中の i 番目の文字とする．日比らの手法では特徴語集合 $\mathcal{W}$ に一致する文字が属するかどうかで加点スコアを α_a(文字が特徴語集合 $\mathcal{W}$ に属する場合は $a = 1$, そうでない場合は $a = 2$) として Smith-Waterman アルゴリズムに適用する．加えて，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についても x_i が特徴語集合 $\mathcal{W}$ に含まれるか否かでスコアの減少量を変化させる．挿入，削除，入れ替えのそれぞれに対応するパラメータを β_a，γ_a，δ_a(文字が特徴語集合 $\mathcal{W}$ に属する場合は $a = 1$, そうでない場合は $a = 2$) とし Smith-Waterman アルゴリズムに導入する．従来手法における剽窃発見アルゴリズムは次の通りとなる．

Step1) ソースコード $\boldsymbol{X} = (x_1, .., x_i, .., x_I)$，$J$ 個の文字からなるソースコード $\boldsymbol{Y} = (y_1, .., y_j, .., y_J)$，文字の組 (x_i, y_j) を始点とし，$S_{0,0} = 0$ 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．

$$S_{i,j} = \begin{cases} S_{i-1,j-1} + \alpha_1, & \text{if} \quad x_i = y_j \cap x_i \in \mathcal{W} \\ S_{i-1,j-1} + \alpha_2, & \text{else if} \quad x_i = y_j \cap x_i \notin \mathcal{W} \\ \max \begin{pmatrix} 0, \\ S_{i-1,j} - \beta_1, \\ S_{i,j-1} - \gamma_1, \\ S_{i-1,j-1} - \delta_1 \end{pmatrix} & \text{else if} \quad x_i \in \mathcal{W} \\ \max \begin{pmatrix} 0, \\ S_{i-1,j} - \beta_2, \\ S_{i,j-1} - \gamma_2, \\ S_{i-1,j-1} - \delta_2 \end{pmatrix} & \text{otherwise.} \end{cases} \tag{1}$$

Step3) 終点 (x_I, y_J) までスコアを計算し，スコア $S_{I,J}$ を $\boldsymbol{X}$ と $\boldsymbol{Y}$ における最終的なスコア $S_{X,Y}$ とする．

Step4) 類似度を「得られたスコア/得られる最大スコア」で定義し，$\boldsymbol{X}$，$\boldsymbol{Y}$ の類似度 $\mathrm{Sim}_{X,Y}$ を以下の式 (2) で算出する．

$$\mathrm{Sim}_{X,Y} = \frac{S_{X,Y}}{N\alpha_1 + (I - N)\alpha_2} \tag{2}$$

ただし，N は $\boldsymbol{X}$ に含まれる文字のうち，$\mathcal{W}$ に含まれる文字数とする．

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

3　提案手法

従来手法では，予約語のような必ず一意に用いられるような文字列に関しても文字単位でスコアを計算している．予約語はソースコード内に必ず出現するが，Smith-Waterman アルゴリズムを用いてこれを文字単位で比較すると総スコアが過大に加点され，剽窃の誤検出が発生する可能性がある．そこで本手法では剽窃検出の性能向上のため，ソースコードを比較する際に予約語については文字列を 1 つの単位として比較することを考える．これにより，予約語が一致した場合の過度のスコア加算と誤検出の発生を防ぐことができると考えられる．また，一部の制御構造を規定する予約語の直後はアルゴリズムの構造を決定する部分であるため，剽窃，被剽窃ソースコード間で酷似すると考えられる．予約語に続く文字の加点を大きくすることで，剽窃箇所を際立たせ剽窃検出の精度を上げる．

3.1 予約語の考慮

前述の通り，過度にソースコードを「類似している」と判断させないため，予約語を文字列の単位で用いる．提案手法ではこれにより，前述の Smith-Waterman アルゴリズムの問題の解決を図る．予約語は言語ごとに数多く存在するが，本研究ではプログラム言語として C 言語を対象とする．これら予約語の中でも本研究では，ソースコード中で特に使用頻度が高いと想定される表 1 の予約語を用いる．加えて表 2 で示す制御構造を規定する予約語を読み込んだ場合，その直後に連続する A 文字の加点スコアを R 倍する．

表 1. 予約語トークン

n	1	2	3	4
p_n	#include	int	void	main
n	5	6	7	8
p_n	cout	endl	double	return
n	9	10	11	
p_n	break	delete	system(”pause”)	

表 2. 制御構造に用いられる予約語トークン

m	1	2	3	4	5	6
q_m	printf	scanf	for	if	else	while

3.2 提案手法のアルゴリズム

いま，トークンに属さない文字の集合を $\mathcal{W}_1$, 基準・着目トークン集合 $\mathcal{W}_2 = \{v_l, z_k : 1 \leq l \leq 3, 1 \leq k \leq 14\}$, 予約語トークン集合を $\mathcal{W}_3 = \{p_n : 1 \leq n \leq 11\}$, 直後にプログラムの制御構造を規定する予約語トークン集合を $\mathcal{W}_4 = \{q_m : 1 \leq m \leq 6\}$ として定義する．

提案手法では従来手法同様に **Step1**〜**Step5** の手順に従い 2 つのソースコード間のスコアを算出する．その中で **Step2** における累積スコア算出は表 1 で示した予約語を一つの文字列として扱い，$\mathcal{W}_4$ に属する文字列を読み込んだ場合はその直後から A 文字の加点スコアを ($R > 1.0$) 倍することで求められ，計算式は以下の式 (3) で表わされる．

$$S_{i,j} = \begin{cases} S_{i-1,j-1} + \alpha_b, \\ \quad \text{if} \quad x_i = y_j \cap x_i \in \mathcal{W}_b (b = 1, 2, 3) \\ S_{i-1,j-1} + \alpha_4 \times R, \\ \quad \text{if} \quad x_i = y_j \cap \forall x_c \in \mathcal{W}_4 \\ (c = i-1, i-2, \cdots, i-A) \\ \max \begin{pmatrix} 0, \\ S_{i-1,j} - \beta_b, \\ S_{i,j-1} - \gamma_b, \\ S_{i-1,j-1} - \delta_b \end{pmatrix} \\ \quad \text{if} \quad x_i \neq y_j \cap x_i \in \mathcal{W}_b (b = 1, 2, 3, 4) \end{cases} \tag{3}$$

従来同様に加点パラメータを α_b，減点パラメータを $\beta_b, \gamma_b, \delta_b$(比較する文字がトークンに属さない文字の集合 $\mathcal{W}$ に属する場合は $b = 1$, 基準，着目トークン集合 $\mathcal{W}_2$ に属する場合は $b = 2$，予約語トークン集合 $\mathcal{W}_3$ に属する場合は $b = 3$，制御構造に用いられる予約語トークン集合 $\mathcal{W}_4$ に属する場合は $b = 4$) として定義する．

また，提案手法における類似度は以下のように定義する．

$$Sim_{X,Y} = \frac{S_{X,Y}}{\text{ソースコードが完全一致した場合のスコア}} \tag{4}$$

4 評価実験

提案手法の有効性を検討するために，実データを用いて剽窃検出の評価実験を行った．実験データの作成にあたってはプログラム演習で実際に用いられる問題 5 題に関して，18 人の学生に計 80 件のソースコードの作成を依頼した，13 人の学生に対してこれらを剽窃したソースコード 40 件の作成を依頼し，これををデータセット 1 とした．また，別の教科書 5 題に関して，14 人の学生に計 70 件のソースコードの作成を依頼した．加えて，7 人の学生に対してこれらを剽窃したソースコード 28 件の作成を依頼した．これらをデータセット 2 とした．これらを用いて，剽窃したソースコードが正しく検出されるかを確認する．

4.1 実験条件

今回は予備実験により従来手法, 提案手法のパラメータを決定した．データセット 1 に対しては，従来手法のパラメータは $\alpha_a = \beta_a = \gamma_a = \delta_a = 1$，提案手法のパラメータは $\alpha_1 = 2$, その他の $\alpha_b = 1$，$\beta_b = \gamma_b = \delta_b = 1$，$A = 10, R = 4$，とした．同様にデータセット 2 に対しては，従来手法のパラメータは $\alpha_a = 2$, その他の $\alpha_a = 1$, $\beta_a = \gamma_a = \delta_a = 1$, 提案手法のパラメータは $\alpha_1 = 2$, その他の $\alpha_b = 1$, $\beta_b = \gamma_b = \delta_b = 1$, $A = 10, R = 4$ とした．

評価尺度としては精度と検出率，F 値 を用いた．それぞれの定義は以下の通りである．

$$\text{精度} = \frac{\text{出力された剽窃ペア数}}{\text{出力された総ペア数}} \tag{5}$$

$$\text{検出率} = \frac{\text{出力された剽窃ペア数}}{\text{総剽窃ペア数}} \tag{6}$$

$$F \text{ 値} = \frac{2 \times \text{精度} \times \text{検出率}}{\text{精度} + \text{検出率}} \tag{7}$$

4.2 実験結果, 考察

表 3，表 4 にデータセット 1，2 それぞれに対する剽窃検出実験の結果を示す．

表 3. データセット 1 に対する剽窃検出結果

	従来手法	提案手法
精度	0.911	0.944
検出率	0.989	0.943
F 値	0.949	0.944

表 4. データセット 2 に対する剽窃検出結果

	従来手法	提案手法
精度	0.576	0.788
検出率	0.819	0.752
F 値	0.676	0.770

表 3，表 4 より，提案手法は従来手法よりも精度は高く，検出率は低いことが分かる．一方，両者の調和平均である F 値では，データセット 1 では同程度，データセット 2 では優れており，F 値の面で提案手法が優れているといえる．

提案手法の性能が優れている理由として以下の 2 点が考えられる．一点目は予約語をまとめたことで過度なスコアの加算が抑えられ，誤検出を抑えられたということ，二点目は，ソースコードを剽窃した場合，制御構造を規定する予約語直後の構造が類似する可能性が高いが提案手法ではこのような箇所の加点スコアが大きくなることで剽窃ソースコードのペアをより検出多くできたということである．

5 まとめと今後の課題

本研究では予約語をまとめ，構成に関わる部分を重視する剽窃検出アルゴリズムを提案した．実際のプログラムを用いて実験した結果，F 値を向上させることが出来た．今後の課題としては，さらなる予約語の追加や，最適なパラメータの自動決定などが挙げられる．

参考文献

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

[2] 日比健太, 雲居玄道，三川健太，後藤正幸，“特徴語に着目した 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，”電子情報通信学会技術研究報告，AI, 人工知能と知識処理，Vol.112(319), pp. 1–6, 2012.