コーディングスタイルモデルに基づく剽窃ソースコードの自動検出手法

1X07C100-5　日比健太
指導教員　後藤正幸

1　研究背景

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，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，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．

ソースコードの自動剽窃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 アルゴリズムを用いて局所的な類似部分を検出する手法 [2] が提案されている．しかし学生のソースコードは「教科書や授業内容等，知識の参照元が同じ」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，上記の手法では適切に剽窃を検出をすることが難しい．そこで本研究では，作成者の書く癖を比較する大野らの手法 [3] の視点を基に，剽窃の誤検出を可能な限りゼロに防ぎつつ，従来手法よりも剽窃検出の網羅性を高める手法を提案する．提案した手法を実際のソースコードに適用することで，従来手法よりも優れた性能となることを示す．

2　従来手法

2.1　モデルの構築

大野ら [3] 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表現した後，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，頻繁に用いられる文字である“ { ”，“ (”,“ + ”などの 14 種の基準トークン z_k $(1 \leq k \leq 14)$ とその前後に出現する「空白」「改行」「タブ」の 3 種の着目トークン $y_l (1 \leq l \leq 3)$ に対して，空白の繰返し回数のみを考慮した着目トークンの組み合わせ $x_m (1 \leq m \leq 6)$ の出現頻度を特徴量として抽出し，ソースコード作成者の記述スタイルモデル$\boldsymbol{M}$をマルコフモデルにより構築する．

ここで，記述スタイルモデル$\boldsymbol{M}$は各基準トークンの前方の記述スタイルを表す $M^{f,k}$(f：前方) と各基準トークンの後方の記述スタイルを表す $M^{b,k}$(b：後方) から成り立つ要素モデルの集合と定義する．また $M^{f,k}$，$M^{b,k}$ は状態，初期確率，遷移確率，観測確率によって構成される．図 1, 2 に $M^{f,k}$ と $M^{b,k}$ の状態遷移図を示す．

- 状態 S_1^f,S_2^b：着目トークンを観測できる状態
- 状態 S_2^f,S_1^b：基準トークンを観測できる状態
- 状態 S_3^f,S_3^b：終了状態
- 初期確率 π_1^f,π_2^f：S_1^f,S_2^f から始まる確率
- 初期確率 π_1^b：S_1^b から始まる確率
- 遷移確率 a_{ij}^f：S_i^f から S_j^f へ遷移する確率 $(1 \leq i, j \leq 3)$
- 遷移確率 a_{ij}^b：S_i^b から S_j^b へ遷移する確率 $(1 \leq i, j \leq 3)$
- 観測確率 c_m^f,c_m^b：x_m が観測される確率

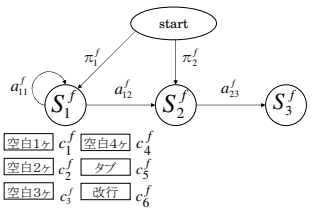

図 1. モデル $M^{f,k}$

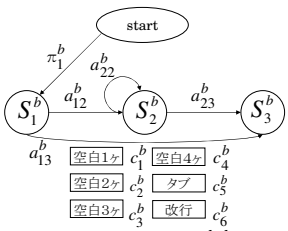

図 2. モデル $M^{b,k}$

2.2　剽窃検出方法

ソースコード作成者の集合を$\boldsymbol{Q}$とし，作成者 $q \in \boldsymbol{Q}$ により作成されたソースコードを d_r^q $(1 \leq r \leq R_q)$ とする．ここで R_q は各作成者 q により作成されたソースコードの総数である．q により作成された R_q 個のソースコードを学習させ q の記述スタイルモデルを表す$\boldsymbol{M_q}$=$\{M_q^{f,k}, M_q^{b,k} : 1 \leq k \leq 14\}$ を構築する．ここに新たなソースコード $\tilde{d}$ が与えられた時，$\tilde{d}$ の作成者が q である確率 $P(q|\tilde{d})$ を計算する．

$$P(q|\tilde{d}) = \frac{P(\tilde{d}|q)P(q)}{P(\tilde{d})} \tag{1}$$

ここで $P(q)$ は一様と仮定すれば，$P(\tilde{d})$ は q によらない定数とみなせるので $P(q|\tilde{d}) \propto P(\tilde{d}|q)$ となる．ただし，$P(\tilde{d}|q)$ を以下のとおり計算する．

$$P(\tilde{d}|q) = \frac{1}{2} \sum_k \left\{ P(M_q^{f,k}|\tilde{d}) + P(M_q^{b,k}|\tilde{d}) \right\} \tag{2}$$

この式 (2) が最大となる q を $\tilde{d}$ の作成者と推定する．

3　従来手法の問題点と本研究への展開

従来手法では，基準トークンの前方と後方に分けて特徴量が抽出されている．基準トークンの前方と後方を同じ重みで扱うが，実際には，前方と後方で記号列の長さに違いがあり等しく扱うべきではないと考えられる．また基準トークン間の繰り返しなど従来手法のモデルでは表現できていない特徴量がある．これらの問題点を鑑み，本研究では記述スタイルをより詳細に表現できるモデルを提案する．

また，従来手法では作成者毎にモデルが構築されていることが前提であるため，作成者毎に一定量の学習用ソースコードが必要である．しかし，実際の演習では同じ作成者のソースコードを複数入手するためには複数回の事前課題提出が必要であり，その中にも剽窃が含まれていることが考えられるため実用的でないといえる．また従来手法では，ソースコードと記述スタイルモデルの類似度を計る尺度として $P(\tilde{d}|q)$ を用いているが，$P(\tilde{d}|q) > 1$ となる可能性があり，これを確率として扱うことも問題である．すなわち，式 (1) との整合性に乏しい．そこで，本研究ではソースコードを確率モデルで表し，確率モデルと整合性のある尺度を用いて，ソースコード同士を直接比較する剽窃検出方法を提案する．

4　提案手法

提案として，新しい記述スタイルモデル及び実用的な剽窃の検出方法を提案する．

4.1　目的

本研究では，学生のソースコードを対象とし，自動的に剽窃および被剽窃となるソースコードのペアを検出することを目的としている．ここで，演習で提出するレポート等での剽窃は近年，厳正な処罰が課されることが明文化されていることが多く，剽窃を誤検出することで，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．そこで本研究では，剽窃の誤検出を可能な限りゼロに防ぎつつ，従来手法よりも剽窃の検出の網羅性を高めることを目的とする．

4.2 モデルの構築

提案モデルでは，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，従来手法と同じく基準トークン z_k および着目トークン y_l の全 17 種とする．

従来手法では，着目および基準トークン内の個別の記号同士，また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，本研究では，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$\boldsymbol{N}$を提案する．

注目する記号が基準および着目トークンの全 17 種であることからコーディングスタイルモデル$\boldsymbol{N}$は，パラメータ $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_h, \cdots, \theta_{289})$ によって構成される．図 3 に記述スタイルモデル$\boldsymbol{N}$を表現したものを示す．

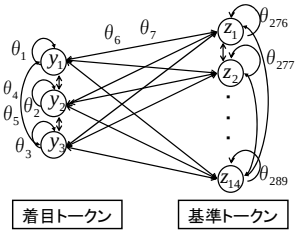

図 3. 提案モデル$\boldsymbol{N}$

4.3 剽窃検出方法

各ソースコードからコーディングスタイルモデル$\boldsymbol{N}$を構築してモデル同士の分布の類似度を計る．本研究では分布間の類似度を計る尺度として，確率モデルと整合性のある Jensen-Shannon 情報量 [4]（JS 情報量）を用いる．

全ての $q \in \boldsymbol{Q}$ に対し，作成されたソースコード d_r^q からモデル $\boldsymbol{N_q^r} = (\theta_{q,1}^r, \cdots, \theta_{q,h}^r, \cdots, \theta_{q,289}^r)$ を構築する．$q, q' \in \boldsymbol{Q}$ の作成したソースコードに対して，モデル $\boldsymbol{N_q^r}$ とモデル $\boldsymbol{N_{q'}^r}$ の分布の類似度を JS 情報量を用いて以下のとおり計算する．

$$
\begin{aligned}
JS(\boldsymbol{N_q^r}, \boldsymbol{N_{q'}^r}) &= \frac{1}{2}\sum_h \theta_{q,h}^r \log \frac{\theta_{q,h}^r}{(\theta_{q,h}^r + \theta_{q',h}^r)/2} \\
&+ \frac{1}{2}\sum_h \theta_{q',h}^r \log \frac{\theta_{q',h}^r}{(\theta_{q,h}^r + \theta_{q',h}^r)/2} \quad (3)
\end{aligned}
$$

$JS(\boldsymbol{N_q^r}, \boldsymbol{N_{q'}^r})$ が閾値 α 以下の時，ソースコード d_r^q とソースコード $d_r^{q'}$ のペアを剽窃であると判定する．ソースコード全ての組み合わせに対して，JS 情報量を計算することで剽窃ソースコードのペアを検出する．

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．

教科書で使用されている例題 4 題に対し，16 名の学生に独力でソースコードの作成を依頼した．一方，11 名の学生にこれらのソースコードを渡し，剽窃した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 (課題毎に作成)]

- 被剽窃ソースコード　16 件
- 剽窃ソースコード　3 件　計 19 件

※剽窃ソースコード 3 件の選び方は ${}_{11}C_3$ パターン

[閾値]

- 従来モデル $\alpha_1 = 0.001$
- 提案モデル $\alpha_2 = 0.015$

※ α_1，α_2 は，簡易的に作成した数件のソースコードによる予備実験において，精度 1.00 を保つ最大の閾値

5.2 実験方法

19 件のテストデータに対して提案手法の剽窃検出方法を用いてそれぞれ実験を行った．

5.3 評価方法

評価尺度として次式に示す精度と検出率を用いた．

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

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

5.4 実験結果

従来手法，提案手法における精度と再現率を表 1, 2 に示す．

表 1. 従来手法の結果

	課題 1	課題 2	課題 3	課題 4	平　均
精　度	1.00	1.00	1.00	1.00	1.00
検出率	0.08	0.16	0.43	0.25	0.23

表 2. 提案手法の結果

	課題 1	課題 2	課題 3	課題 4	平　均
精　度	1.00	1.00	1.00	1.00	1.00
検出率	0.43	0.45	0.52	0.36	0.44

5.5 考察

1. 全ての課題において提案手法の再現率が優れており，提案手法の有効性を示せた．提案手法では，表現能力が向上したことで作成者の書く癖をよりよく表現でき，検出率が向上したと考えられる．
2. 提案手法では，パラメータ数が 224 個から 289 個に増えたことで，パラメータの推定量に 0 を多く含むモデルとなっている．しかし，従来手法より検出率は向上しており，この様なスパースネスの問題は検出率に大きな問題を与えていないことが分かる．これは，推定量が 0 のパラメータは比較する 2 つのソースコード間で共通であり，パラメータ同士の距離を計算する JS 情報量に影響していないためだと考えられる．
3. 剽窃を検出できなかったものの多くは，被剽窃者の記述スタイルが剽窃者の記述スタイルに書き換えられたことが原因だと考えられる．

6 まとめと今後の課題

本研究では，モデルを改善することで従来手法よりも再現率を高めることができ，提案手法の有効性を示せた．

今後の課題として，教科書をモデル化し，教科書とソースコードの類似度を考慮することでより精度を保持し再現率を向上できると考えられる．

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] J. Ji, S. Park, G. Woo, and H. Cho, “Understanding the Evolution Process of Program Source for Investigating Software Authorship and Plagiarism,” *Proceedings of IEEE International Conference on Digital Information Management*, pp. 92–97, 2007.

[3] 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，2010.

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