7H-02

C言語プログラムを対象とした盗用検知手法の提案および実装

東 拓磨　　篠埜 功

芝浦工業大学大学院 理工学研究科 電気電子情報工学専攻

1. 研究背景と目的

プログラミング教育において、学生が課題に取り組む際に、プログラムの盗用を行う可能性がある。盗用の発見は採点者にとって時間がかかる作業であり、分担してプログラムを採点することもある。このため盗用の発見を支援する環境が望まれる。プログラムの盗用検知には様々な手法があるが、コードクローンおよび類似度に基づいた手法に分けられる。これまでプログラムの類似度は、字句および構文の情報を用いた方法で算出されていた。

本研究では、従来文章の類似度判定に使用されていた N-gram 法とレーベンシュタイン法により、盗用検知を行うシステムを実装する。実装したシステムを用いて実験を行い、盗用の疑いのあるプログラムを概ね発見できることを示す。

2. 既存研究

盗用検知を行うために、コードクローン検出手法とレーベンシュタイン法を組み合わせて用いる手法[1]や、N-gram 法を用いる手法[2]などが提案されている。N-gram 法を用いる利点は、利用する文書の形式を問わない点である。この特徴により、様々なプログラミング言語のソースコードに対して N-gram 法がそのまま適用できる。また、N-gram 法は計算量が比較的小さいことも利点である。N-gram 法は 2 つの文字列に対して、長さ N の部分文字列の出現頻度の分布の近さを数値化する方法であり、プログラムの構造が異なっていても、分布が近ければ誤って盗用と判定される場合がある。またレーベンシュタイン法もプログラムの順序の入れ替えが発生した場合距離が増えてしまい、盗用の判定を誤る場合がある。

3. 提案手法

本研究では N-gram 法とレーベンシュタイン法を組み合わせることで盗用検知を行う。

3.1 システム構成

提出された C 言語プログラムに対して前処理を行い、N-gram 法とレーベンシュタイン法を実行する。前処理として、空行と注釈を削除する。システムの概要を図 1 に示す。

3.2 盗用発見手法

3.2.1 盗用について

プログラムの盗用は、入手した他者のソースコードの一部を変更する偽装が行われることが多い。例えば、空白文の挿入やコメント文の変更・削除、変数名や関数名、定数の変更、条件文の書き換え等が挙げられる。

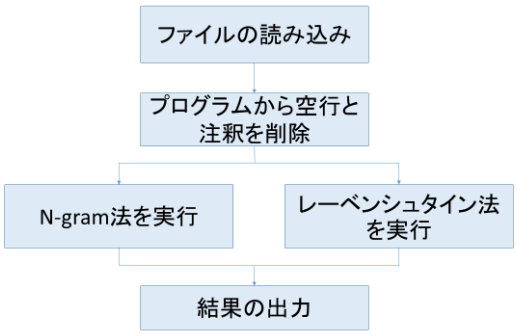

図 1. システムの概要

3.2.2 類似度算出手法

ソースコード間の類似度は N-gram 法とレーベンシュタイン法を用いて得られた類似度がそれぞれで設定した閾値を越えた場合、盗用と判断する。

本システムでは 2-gram 法を用いる。つまり 2 つのソースコードから 2 文字単位(2-gram)で切り出した要素の出現回数をベクトル化し、それらのベクトル間の角度の余弦を計算する。

編集距離を求める一つの方法であるレーベンシュタイン法は、ある文字列に対して 1 文字の削除あるいは 1 文字の追加によって得られる文字列との距離を 1 としたとき、文字列全体を目的の文字列に置き換えるために必要となる編集操作による距離の最小値を動的計画法により求める手法である。レーベンシュタイン法の時間計算量は、2 つの文字列の長さを n と m とすると O（nm）である。

4. 評価実験

4.1 実験概要

芝浦工業大学工学部情報工学科 1 年生を対象に実験を行った。学生から提出された課題すべてに対して総当りで本提案手法を実行した。出力された数値を表にし、閾値ごとに色付けを行った。色分けされた結果に対応するプログラムを目視し、疑わしいプログラムを発見できたかを分析した。

今回の実験では、2-gram 法による類似度が 0.9 以上の場合に色付けを行い、さらにレーベンシュタイン法による類似度が 1 以上 10 未満の場合は赤、10 以上 50 未満は橙、50 以上の場合は黄色で色分けを行った。レーベンシュタイン法による色分けの基準を表 1 に示す。

表 1. レーベンシュタイン法により求めた編集距離による色分け

編集距離	色（盗用の疑いの度合い）
1~9	赤（大）
10~49	橙（中）
50~	黄（小）

4.2 実験結果

実験結果を表 2 に示す。2-gram 法による類似度が 0.9 以上の場合に色付けを行い、色付けしたマスを表 1 の基準でさらに色分けを行っている。

表 2. 2-gram 法による類似度が 0.9 以上の場合に色付けし、色付けしたマスを表 1 の基準で色分けした編集距離

	1	2	3	4	5
1		53	50	52	43
2			82	83	39
3				81	64
4					77
5					

	6	7	8	9	10
1	22	126	76	37	37
2	52	101	112	30	30
3	53	144	85	70	68
4	49	157	52	68	73
5	42	108	95	22	19
6		128	80	32	40
7			176	107	105
8				92	93
9					8
10					

例えば表 2 の赤いマスにおいては、2-gram による類似度が 0.9 以上であり、かつレーベンシュタイン法による編集距離が 8 である。これは非常に盗用の疑いが大きい。該当のプログラムを目視で比較したところ、3.2.1 節でも述べたように、空白の挿入や変数名の違い、if 文等の条件式の違いなどが見られた。

他のプログラムの組み合わせについても 3.2.1 節で述べたような傾向が見られ、盗用の疑いのあるプログラムを検出できたと考えられる。

しかし、偶然類似度が高くなる場合や、授業内で提示されたサンプルプログラムを参考にすることによって類似度が高くなってしまう場合がある。表 3 では授業の履修者以外の学生が他のプログラムを見ずにプログラムを作成し、表 2 の 10 名の学生のプログラムとの類似度の計算を行った。

表 3. 履修者以外の学生のプログラムとの比較

	1	2	3	4	5
作成	28	57	49	65	51

	6	7	8	9	10
作成	33	123	83	44	44

今回の実験に用いた課題は簡単であり、またサンプルプログラムが提示されていたこともあり、意図せず類似してしまう場合があった。

5. まとめと今後の課題

2-gram 法とレーベンシュタイン法を組み合わせることにより盗用検知の判定がより正確になった。またレーベンシュタイン法において、変数名や if 分の条件式の違い程度では大きな差が無いことがわかった。

しかし、今回の実験の課題は簡単であり、プログラムが 50 行に満たないものが多い。また、サンプルプログラム無しの課題においても偶然類似度が高くなる場合もあった。

今後の課題としてより難しい課題の場合で実験を行うことが挙げられる。また、類似度が高い場合に故意か偶然かを判定する手法を検討することが挙げられる。

参考文献

[1] 岩本舞, 小島俊輔, 中嶋卓雄, 学生のプログラミング演習におけるトークンベースのコードクローン検出手法, 情報処理学会研究報告, 2012.

[2] 和田修平, 井上潮, 盗用発見と自動採点によるプログラミング演習課題の評価支援システム, 電子情報通信学会 DEIM フォーラム, F8-5, 2010.