2016 年 2 月 4 日

平成 27 年度　情報・通信工学科　コンピュータサイエンスコース　卒研発表会

プログラミング入門教育におけるソースコードの自動分類

情報・通信工学科　学籍番号：1211215　寺田研究室　渡邊裕貴

1　背景

情報工学教育において, あるプログラムを作成するという課題があったとき, 学習者はその課題を解決するにあたり, 様々なアプローチで実装を行うことになる.

これに対して, 教員は学習者から提出されたソースコードを集め, それをもとに評価を行う. しかし, ソースコードが膨大な数となると, 各学習者がどのようなアルゴリズムを用いたか, また文法を用いたかといった, 出題者が課題への理解度を図る上で重要な要素を把握しきることが困難になると予想される.

本研究では, 同じプログラミングの課題に対して学習者から提出されたソースコードが多数あったとき, その中から似ているもの, そうでないものの自動分類を行う. また, 似ているコードのグループを特定することで, 各グループがどのような手法で課題に取り組んだかを明らかにし, それが出題者のねらいにそったものであるかの確認を行う. このようにしてクラスタリングの結果を各学習者の課題への理解度を図る素材とし, 情報工学教育の支援を行うシステムを作ることが本研究の目的である.

2　関連研究

本研究の関連研究として, 電気通信大学の寺田によるプログラムの類似性の判定とクラスタリングがある [1]. この研究は, プログラミングの課題に対して, 学習者から提出されたソースコードを解析し, 類似プログラムを発見することを目的とする点で, 本研究と共通している. 本研究では, この研究が用いている字句解析による特徴ベクトルが含む字句の種別を増やし, また k 平均法と階層化クラスタリングを新たに用い, 特徴ベクトル内での代表要素の発見も新規に行う.

また, ソースコードの内容の一致, もしくは類似を発見する研究としては, 大阪大学の井上らが行っているコードクローン検出法がある [2]. この研究は大規模ソフトウェアの開発などで, コピー・ペーストの繰り返しや, 意図的に同一処理を繰り返し書くことによって生じる, ソースコード中の類似または一致した部分 (コードクローン) 検出するというものである.

前述したコードクローンの研究に関連して, 産業技術総合研究所の神谷によって開発実装されている CCFinder がある [3]. CCFinder では, ソースコードに字句解析を行い, トークン列へと変換することで, トークン列同士の比較を行い一致を調べる. この手法では, いくつかのソースコードを比較したとき, 比較的簡単に一致部分を発見できるが, 本研究で提案する大量のソースコードの分類に適用することは難しい.

3　提案システム

3.1　特徴ベクトルへの変換

はじめに, ソースコードを以下のような手順で特徴ベクトルへと変換する.

1. ソースコードに対し字句解析を行う.
2. 字句解析の結果からソースコード中に出現した字句の数を種別ごとにカウント.
3. 字句の種別ごとの出現数をそのままベクトルの各次元の要素とする.

実際には, 同じ課題について集められたソースコードの特徴ベクトルの集合に対して, よりベクトルの間の特徴の差を見つけやすくするため, 各次元に関してして集合の中の特徴ベクトルでもっとも数が多いものを 1.0 として, 基準の値とした.

なお, 本研究ではクラスタリングするコードの対象として C 言語のコードを用いた.

3.2　クラスタリングの手法

本研究では, 前述した特徴ベクトルの自動分類を行うにあたって, k 平均法および階層化クラスタリングを用いた. なお, いずれの手法においても二つの n 次元ベクトル a, b 間の距離は次のように定義する.

$$d = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \tag{1}$$

階層化クラスタリングについては, 重心法と呼ばれる, クラスタの中心をクラスタに属するベクトルの重心で表す手法を用いた [4].

3.3　特徴ベクトルの代表要素の発見

さらに, 特徴ベクトルの要素から, さらにクラスタリングを行う上で重要となる, すなわち他のベクトルとの差が現れやすい要素を見つけ出し, これらの要素のみで構成された特徴ベクトルによるクラスタリングも行う. 具体的な代表要素の発見手法は, ソースコードの集合から各要素について分散の値を計算し, その値が大きい要素をいくつか選び代表要素とするものである.

4　実装

4.1　字句解析

C 言語の字句解析には, C コンパイラである clang のライブラリの LibTooling を用い, C++ のプログラムで行った. 字句解析の結果, 92 次元の特徴ベクトル (すなわち, 92 種類の字句を含む) を出力する仕様となっている.

4.2　クラスタリング

クラスタリングを行うプログラムは, いずれも Java を用いて実装を行った. また, 特徴ベクトルの代表要素を選ぶプログラムも, Java によって実装を行った. このプログラムは, 代表要素だけによって構成される特徴ベクトルを字句解析プログラムと同様の様式で出力するものである. これにより, 通常の特徴ベクトルのためのものと同じクラスタリングプログラムに入力することができる.

5　実験

提案手法の有効性を示すにあたり, 以下のような条件のもと実験を行った.

5.1　実験で用いたソースコード

[1] で寺田が扱った, プログラム課題に対する解答の C 言語のソースコードの集合を使用した.

```
int findstr(char *string, char *pattern)
{
  int i;
  int n = 0;

  while(string[n] != '\0'){
    for (i = n; string[i] != pattern[0]; i++)
      if (string[i] == '\0') return -1;
    for (n = i; string[n] == pattern[n - i] && string[
      n] != '\0'; n++);
    if (pattern[n - i] == '\0') return i;
    else n = i + 1;
  }
  return -1;
}
```

コード 1　実験で使ったコードの一例

5.2　クラスタリング

5.2.1　k 平均法

初期ベクトルとして, ソースコードの平均ベクトルについて各要素を 0.5 倍から 0.9 倍に変化させたもの (初期条件 A) と, 0.6 倍から 1.5 倍まで変化させたもの (初期条件 B) を用意し, $k = 5, 10, 15, \cdots, 50$ で結果を比較した. また, 92 次元の特徴ベクトルのクラスタリング以外に, 後述する 10 次元の特徴ベクトル (後述) についてもクラスタリングを行った.

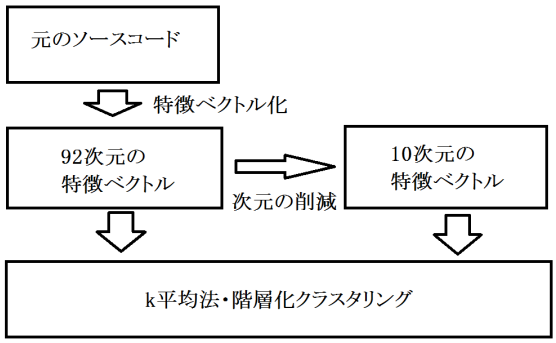
5.2.2　階層化クラスタリング

k 平均法と同様, 92 次元と 10 次元の特徴ベクトル 2 つについてクラスタリングを行った.

5.3　ベクトルの代表要素の発見

全体の集合の中で出現数が少なく一部のソースコードのみにしか現れないために分散が大きくなる字句を除くために, 出現数の平均が 3 を越える要素のみを対象にして代表要素を 10 個選び出しそれを新たな特徴ベクトルとした.

図 1　実験の流れ



6　結果と考察

6.0.1　k 平均法

二種類のベクトルどちらについても, 実験を行ったすべての k の値において, 誤差関数 E(クラスタの代表ベクトルとそのクラスタに属するベクトルの間の距離の和の合計) の値が初期条件 B の方が小さかったため, 初期条件 B の結果を採用した.

クラスタリングの結果から, グループごとの学習者の傾向 (あるグループでは goto 文を使っているために, 字句 “:” を唯一用いているなど) が見られた.

6.0.2　階層化クラスタリング

10 次元, 92 次元のベクトルに対するクラスタリングのいずれの場合でも, いくつかの字句の数値の大きさが階層構造に対応したものであることが観察された.

しかし, 階層の上位で学習者を分類する字句の多くは, 学習者のコードの特徴を表しているとは言い難いものであった (“(”, “)” など).

6.1　ベクトルの代表要素の発見

分散が大きくなるベクトルの要素が, なぜ大きくなるかを調べることにより, 極数の特殊な解答を発見することができた. たとえば, 他の学習者が使っていない字句を使っている学習者の一人は, まったく異なる課題の解答のコードを提出していた.

7　結論

k 平均法では, k の値をある程度小さくすることで, 分類されたソースコードの特徴からある手法を用いた学習者とそうでない学習者の分類を行うことができた. また, k の値を大きくすることで, クラスタのサイズが小さくなり, コピー・ペーストされたコードのペアの発見が容易となった.

階層化クラスタリングでは, 階層の上層部でどのような字句が使用されているかの差を見ることによって, ソースコードの集合を大まかに分離する要素を特定したが, クラスタの分離では, 大きなクラスタと小さなクラスタに分かれることが多々あり, 各クラスタの特徴を見ることは難しくなった.

また, 特徴ベクトルを 10 次元にする実験では, クラスタリングの結果から各クラスタの特徴がとらえやすくなった.

8　今後の課題

提案手法では, 特徴ベクトルの代表要素について, 各特徴ベクトルの代表となる要素を分散の大きさで機械的に選び出したために, “[” と “]” のような必ずペアとなって出現する字句を代表要素としてしまい, その分無駄な情報を含んでしまっていた.

また, 代表要素については, 元のベクトルからもっとも情報量が大きくなる組み合わせと考え, 分散の大きい要素を選んでいるが, 実際にクラスタリングをする上で有効となる組み合わせは今後研究していきたい.

さらに, 本研究の提案手法では字句解析の結果のみを使用しているために, プログラムの構造的な違いを判断することができない. そのため, プログラムの構造の類似に基づくクラスタリングは不可能である. そういったクラスタリングをより正確に行うには, クラスタリング対象となる情報に構造を示す要素を含める必要があり, その手法については今後の課題である.

参考文献

[1] 寺田実, プログラムの類似性とクラスタリング, プログラミング・シンポジウム, 1997.

[2] 井上 克郎, コードクローン検出法, コンピュータ ソフトウェア, Vol. 18 No. 5, pp47-54, 2001.

[3] 植田 泰士, 神谷 年洋, 楠本 真二, 井上 克郎, 開発保守支援を目指したコードクローン分析環境, 電子情報通信学会論文誌 D-I, Vol.86-D-I, No.12, pp.863-871, 2003.

[4] 宮本 定明, クラスター分析入門—ファジィクラスタリングの理論と応用, 森北出版, 1999.