2017 年 2 月 10 日

平成 28 年度　情報・通信工学専攻　コンピュータサイエンスコース　修士論文発表会

相関ルールを利用したソースコードの識別子推薦手法

寺田研究室 学籍番号:1531003 阿部 真之 指導教員 寺田 実, 岩崎 英哉

1　はじめに

大規模なプログラムの開発を行うプロジェクトでは, 既存のコードに手を加える形でコーディングを行う場面が多くなる. ソフトウェアの保守や作成では, プログラムを理解するために, 識別子名からその関数や変数の役割を推測する. この時, 識別子の名前付けが正しい役割を表現していなければ, プログラムの理解により多くの時間がかかってしまう. コードの可読性やプログラミング効率を向上させるために, 変数名や関数名といった識別子には適切な名前付けを行うことが重要となる.

一方, 識別子の中には互いに関連のある名前を持つものが存在する. そして, これらの識別子は同じ関数内で使われることが多い. そのため, 関連のある識別子の内, 一方を入力すると, もう一方の識別子を自動的に推薦するようにすると, 開発者が識別子に対して適切な名前付けを行うための支援ができると考えられる.

本研究では, 相関ルールマイニングを用いて識別子同士の一対一の共起関係を抽出し, コード編集中に識別子を推薦するシステムを作成し, 適切な候補を推薦できたか評価を行った.

2　関連研究

Li ら [2] は, ソフトウェアのソースコードから暗黙のプログラミングルールを見つけ, そのルールを用いてバグを検出する手法を提案している. この手法を実装した PR-Miner は, Linux カーネルのソースコードから 16 個のバグを検出した. また, Livshits ら [3] は, バージョン管理システムの改版履歴を解析し, 同時に変更されたメソッド呼び出しの組み合わせを抽出することでルールを見つけ, バグ検出に応用するツール DynaMine を開発している. これらの研究では, 既存のソースコード集合から共起関係のある識別子のルールを見つけているが, それをバグ検出に利用している点で本研究と異なる.

3　提案手法

3.1　相関ルール

相関ルールマイニング [1] は Agrawal らが提案したデータマイニングの手法である. 相関ルールとは, ある事象 A が発生したときに, 別の事象 B も発生するといった関連を示すものであり, $A \Rightarrow B$ と表される. ここで, ルールの A の部分を条件部, B の部分を結論部と呼ぶ.

有用な相関ルールを抽出するための評価指標として, 支持度, 確信度, リフト値がある. 基本的に確信度が大きいほど関連が強いといえる.

3.2　識別子同士の共起関係の解析

共起関係を解析するためのソースコードコーパスは, 開発者が機能追加や編集を行いたいプロジェクトの全ソースコードとする. これらのソースコードの構文解析を行い, 1 つのメソッドの中に出現する識別子を取得し, これを 1 つのトランザクションデータとする. 構文解析により得られたトランザクションデータはデータベースに保存し, 相関ルールマイニングに利用する.

3.3　識別子の推薦

識別子の共起関係の推薦には相関ルールを基にした指標を用いる. 提案手法では, 相関ルールの抽出の際, 条件部と結論部のアイテム数が 1 つとなる集合のみに限定する.

推薦元となる識別子は, 編集中のメソッド内に出現するフィールド変数, ローカル変数, メソッド呼び出しの識別子である. これらの識別子を相関ルールの条件部としてもつルールをデータベースから検索し, 確信度の高い順に並べることで推薦候補とする. 以降の節で, 確信度にたいしてスコアを付加した推薦指標を提案する.

また, 異なる条件部から同じ結論部が推薦されることがある. このとき, 推薦候補の評価値の値として 2 通りの方法が考えられる. 1 つ目は確信度の合計値を評価値にする方法で, 2 つ目は確信度の最大値を評価値にする方法である. 以降, これらの計算方法をそれぞれ合計確信度推薦, 最大確信度推薦と呼ぶ.

3.4　推薦精度の向上

提案手法では, 確信度に対して, 追加のスコアによる重み付けを行うことで, 上位に現れる推薦候補が増えるかを検証した.

型名による重み付け　同じ型名を持つ識別子は, 互いに関連している可能性が高いと考えた. そこで, 推薦候補の識別子が持つ型名と, 推薦元の識別子が持つ型名が一致した場合に, 確信度に対して重みづけを行うことで, 上位候補の数が増えるかを検証した.

重要語による重み付け　関数名の動詞部分, 変数名の名詞部分をそれぞれ重要語として取り出し, 重要語同士の確信度を計算した. この確信度を用いて重みづけを行うことで, 上位候補の数が増えるかどうかを検証した.

多対一のルールによる推薦　これまでの付加スコアでは, 一対一の対応関係のルールのみを扱っていた. そこで, 重み付けを行わずに, 多対一の対応関係も含めたルールを扱った場合についても評価を行う.

4　評価実験

4.1　実験対象

実験対象のプロジェクトとして, Apache Ant を利用した. 構文解析により 10,710 個のメソッドを抽出し, その中に含まれる識別子をデータベースに格納してコーパスとした.

4.2　実験方法

実験対象のソースコードコーパス内に含まれる全てのメソッドに対して, コーパスに登録された後半 5 割の部分に初登場する識別子を削除する. その後, 残った前半 5 割の部分のみから推薦を行い, 推薦候補の上位 10 個までを取得し, 推薦された候補が削除した後半部分に含まれているかを調べる. 候補が含まれていた場合, その候補の順位を記録する.

実験の評価尺度については, 推薦成功率と MRR を用いる. 推薦成功率とは, 実験対象となったメソッドに対して, 推薦が成功した確率である. 提示された推薦候補が削除した後半部分に 1 つでも含まれていれば成功したとする.

4.3　平均逆順位 (MRR)

実験の評価尺度の 1 つとして, 平均逆順位 (MRR) を用いる. これは, 順位付けされた推薦候補の評価に用いられるもので, 候補の中で最初に正解が現れた順位の逆数の平均によって求められる.

5　結果と考察

正解の推薦候補が提示された順位を調べ, その順位に提示された識別子数をグラフにすると図 1 のようになった. この結果を見ると, 上位に提示された推薦候補ほどよく使われていることが分かる. これにより, 提案手法による識別子の提示では推薦候補を適切に上位に並べられていることが分かる.

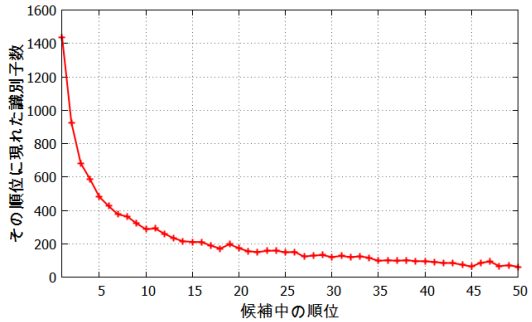

図 1　後半部分に含まれていた識別子の候補中の順位

各付加スコアでの推薦成功率と MRR を表 1 に示す. 型名と重要語は, 重み係数が 0.5 の場合の数値である.

表 1 の結果を見ると, 提案手法で述べた 3 つの付加スコアは一対一の重み無し確信度による評価との比較では, 推薦成功率と MRR の増加は見られなかった.

表 1　推薦成功率と MRR

	推薦	一対一	多対一	型名	重要語
推薦成功率	合計	0.581	0.569	0.579	0.531
	最大	0.563	0.554	0.561	0.501
MRR	合計	0.378	0.371	0.369	0.343
	最大	0.356	0.354	0.341	0.324

合計確信度推薦と最大確信度推薦を比較すると, どの付加スコアを用いた場合においても合計確信度推薦の方が, 推薦成功率, MRR ともに高くなっていることが分かる.

しかし, 2 つの推薦によって提示された候補全体のリフト値の平均を求めてみると, 合計確信度推薦の場合ではリフト値の平均が 74.8, 最大確信度推薦の場合ではリフト値の平均が 91.2 となった. リフト値は, 値が低いほど提示された推薦候補の出現率が高いことを意味しているため, 合計確信度推薦で提示される候補は, ありふれた名前の識別子であることが多いことが分かる.

更に, 型名と重要語では重み係数を 0 から 1 までの間で 0.1 ずつ変化させて実験を行った結果, 重み付けによる評価では推薦成功率や MRR が下がってしまうということが確認できた.

型名による重み付け推薦では, 評価実験において推薦候補として提示された識別子のうち, 後半 5 割部分でも使用されていた候補は, その候補の推薦元となった識別子と同じ型を持つものが増加していた. これにより, 開発者が, 提示して欲しい識別子の型がメソッド内に記述した識別子と同じ型を持っていることを予測できた場合に, この推薦を利用することで目的の識別子を見つけることができる可能性があると考えられる.

6　結論

本手法を用いることにより, 途中まで記述されたメソッドから約 58% の確率でそれ以降メソッド内で使用する識別子名を推薦することができることを評価実験により示した. また, 推薦候補の評価値としていくつかの付加スコアを提案し, これらの指標が有効であるかの検証を行ったが, 評価値の向上は見られなかった. 一方で, 本手法によって上位に提示された識別子ほどよく使われていることを示し, 適切な候補を上位に提示できていることを確認した.

今後は, 評価値の向上する付加スコアの考案や, ソースコードの文脈を考慮した推薦を行い, 文脈に応じた識別子名を提示するといった改善を行っていきたい.

参考文献

[1] Rakesh Agrawal, Ramakrishnan Srikant. “Fast Algorithms for Mining Association Rules” Proceedings of the 20th VLDB Conference (1994).

[2] Zhenmin Li, Yuanyuan Zhou. “PR-Miner: automatically extracting implicit programming rules and detecting violations in large software code” ESEC-FSE (2005).

[3] Benjamin Livshits, Thomas Zimmermann. “DynaMine: Finding Common Error Patterns by Mining Software Revision Histories” ESEC-FSE (2005).

対外発表

阿部真之, 寺田実. “相関ルールを利用したソースコードの識別子推薦手法” 第 58 回プログラミング・シンポジウム予稿集 pp.51-58 (2016).