2. ТЕХНОЛОГИИ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ

М. В. Аврискин, Е. А. Павлова

Тюменский государственный университет, г. Тюмень

УДК 004.912

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ОПРЕДЕЛЕНИЯ АВТОРСТВА ПРОГРАММНОГО КОДА

Аннотация. В статье рассматриваются различные количественные признаки программного кода, методы и инструменты для их извлечения, их дальнейшая обработка и использование. Приводится пример определения авторства с использованием извлекаемых признаков.

Ключевые слова: интеллектуальный анализ программного кода, машинное обучение, анализ текстовых данных, конструирование признаков.

Введение

Интеллектуальный анализ программного кода играет важную роль в помощи написания и улучшения программного кода, борьбе с плагиатом, аудите программного кода. Автоматизированный анализ может использоваться преподавателем при проверке студенческих работ, что может послужить улучшению качества процесса обучения [1].

При работе с большими объемами данных работа с программным кодом в исходном виде требует больших затрат ресурсов. Для уменьшения ресурсоемкости необходимо конструирование признаков программного кода. По полученным признакам оказывается возможной быстрая классификация и кластеризация исходных текстов используя типичные инструменты машинного обучения.

Цель исследовательской работы — разработать инструмент для конструирования признаков программного кода и дальнейшей обработки.

Конструируемые признаки

Признаки кода делятся на пять категорий: лексические признаки, признаки форматирования, синтаксические признаки, стилевые признаки и качественные признаки [2].

К лексическим признакам относится использование тех или иных лексем, их длины, и частоты:

- количество ключевых слов (if, while, def и др.);
- наиболее часто встречающиеся лексемы и их частоты;
- средняя длина лексем;
- количество литералов;
- количество комментариев;
- количество функций;
- среднее количество аргументов функций;
- среднее отклонение количества аргументов функций;
- средняя длина строки;
- среднее отклонение длин строки.

К признакам форматирования относятся признаки расстановки непечатных знаков:

- количество символов табуляции;
- количество символов пробела;
- количество непечатных символов;
- количество пустых строк;
- наличие переноса строки перед фигурными скобками;
- использование знаков табуляции вместо пробелов в начале строки.

Для получения синтаксических признаков необходимо, чтобы код был синтаксически корректным, т.е. код должен следовать синтаксическим правилам языка программирования. В таком случае строится абстрактное синтаксическое дерево (АСД) — дерево, в котором внутренние вершины сопоставлены с операторами языка программирования, а листья — с соот-

ветствующими операндами, и параметры данного дерева используются как количественные признаки.

Для примера конструирования синтаксических признаков рассмотрим дерево, полученное из кода на языке C++ с помощью библиотеки Joern (рис. 1).

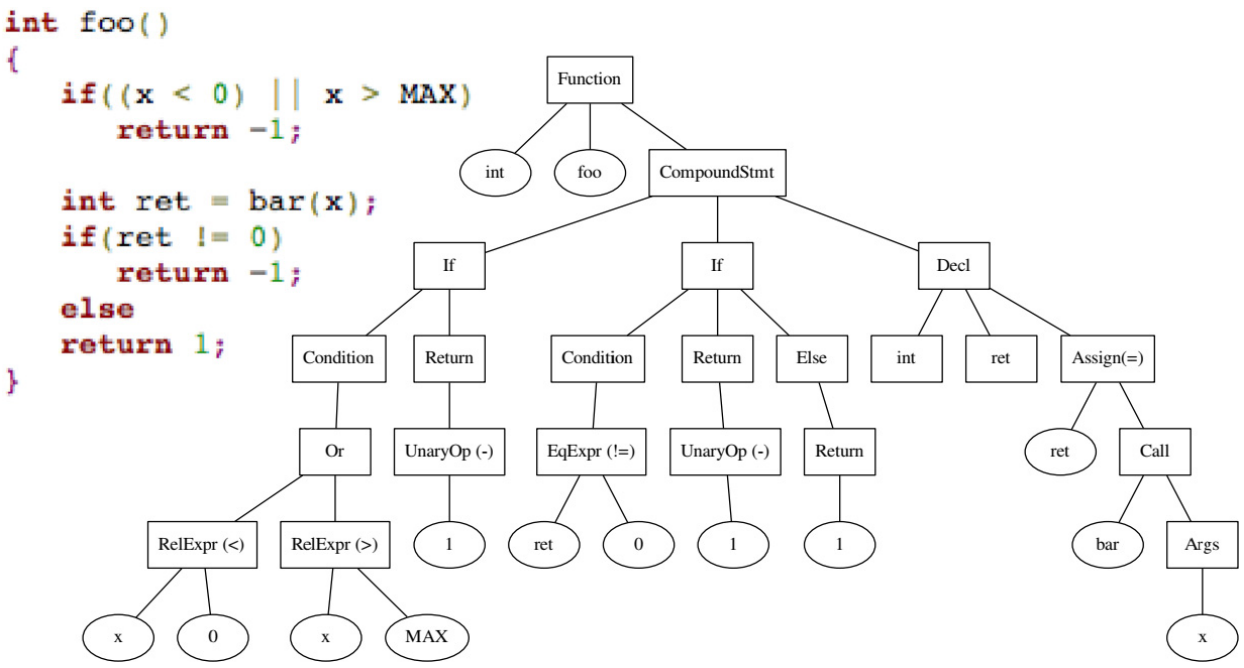

Рис. 1. Пример абстрактного синтаксического дерева.

Из построенного дерева выделяем следующие синтаксические признаки:

- высота АСД;
- частоты узлов, не являющихся листами (TF и TF-IDF меры);
- средняя глубина для каждого из 58 типов узлов, не являющихся листами;
- частоты 84 ключевых слов C++;
- частоты юниграмм и их средние глубины.

При конструировании признаков кода, написанного на других языках программирования, процесс выполняется аналогично, количество синтаксических признаков зависит от лексики языка.

Кроме синтаксических признаков, выделяют стилевые и качественные признаки программного кода [3].

К стилевым признакам отнесем признаки, рекомендуемые, но не обязательные при написании кода, такие как:

- количество лексем, состоящих из английских слов;
- количество функций, названых глагольными словами;
- количество классов, названых существительными словами;
- использование snake case и camel case в лексемах, состоящих из нескольких слов.

К качественным признакам можно отнести степень несоблюдения правил написания удобочитаемого кода:

- длина лексем;
- длина тела функций и описаний классов;
- использование неименованных констант;
- написание функций с большим количеством аргументов.

Таким образом, в пяти категориях выделяется 30 признаков, 6 из которых являются векторами количественных признаков. Если привести их к набору числовых значений, получится несколько сотен признаков, обработка которых будет слишком ресурсоемкой. Поэтому для дальнейшей обработки проводится отбор этих признаков по критерию энтропии, используя обучающий набор входных текстов.

Инструменты для извлечения признаков

Некоторые признаки (количество пробельных символов, длины строк, количество пустых строк и комментариев) не требуют сложного анализа и могут легко применяться к различным языкам программирования. Для конструирования таких признаков в процессе предобработки подсчитывается количество символов пробелов, символов табуляции в тексте и других простых параметров.

Для построения синтаксических деревьев и конструирования признаков из всех пяти категорий необходимо использование инструментов статического анализа кода для конкретного языка. Примерами этих инструментов являются платформа Joern для анализа кода на C/C++ и платформа Roslyn для анализа кода C#/Visual Basic.

```
var tree = CSharpSyntaxTree.ParseText(code);
var classInfos = tree.GetRoot()
    .DescendantNodes()
    .Where(t => t.Kind() == SyntaxKind.ClassDeclaration)
    .Cast<ClassDeclarationSyntax>()
    .Select(t =>
    new
    {
        ClassName = t.Identifier.ValueText,
        Methods =
    t.Members.OfType<MethodDeclarationSyntax>()
    })
    .Select(t =>
    new
    {
        ClassName = t.ClassName,
        MethodDetails = t.Methods
    .Select(m => new
    {
        Name = m.Identifier.ValueText,
        Lines = m.Body.Statements.Count,
        CommentsSL = m.Body.DescendantTrivia()
                        .Count(b => b.Kind() == SyntaxKind.SingleLineCommentTrivia)
                        ,
        CommentsML = m.Body.DescendantTrivia()
                        .Count(b => b.Kind() == SyntaxKind.MultiLineCommentTrivia)
    })
    });
```

Рис. 2. Подсчет количества однострочных и многострочных комментариев в коде.

На рисунке 2 показан пример подсчета количества комментариев с помощью библиотеки Roslyn.CodeAnalysis — по данному коду строится синтаксическое дерево, обходятся узлы, соответствующие методам и подсчитываются узлы, соответствующие комментариям.

Пример классификации программного кода на языке C++

Для примера классификации кода были взяты исходные коды программ, являющихся решениями задач заключительного этапа всероссийской

олимпиады по информатике 2018 года. Эта выборка состоит из 17215 образцов исходного кода на C++, принадлежащих 479 участникам олимпиады.

Для каждого образца программного кода были высчитаны значения пяти признаков: доля комментариев в коде, средняя длина идентификаторов, доля пустых строк, средняя длина строки и среднее отклонение длины строки.

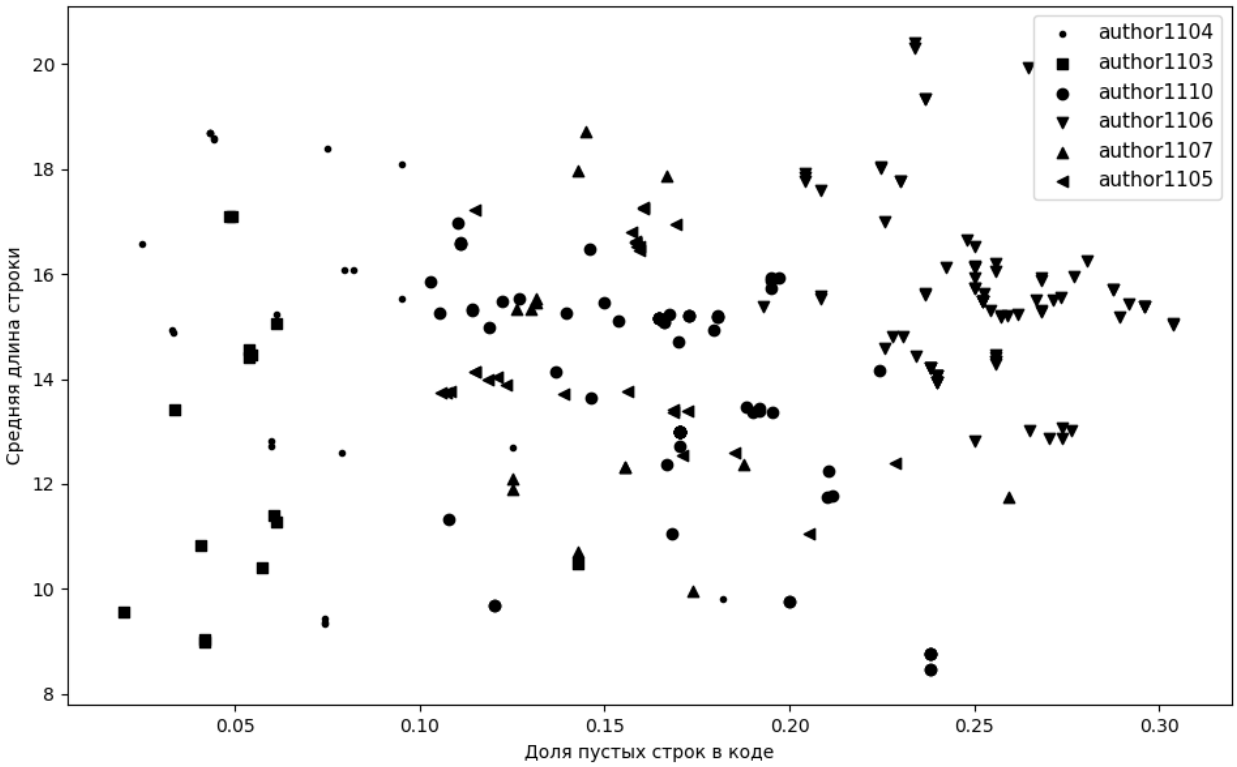

Рис. 3. Диаграмма рассеяния признаков длины строки и доли пустых строк.

На рисунке 3 приведена диаграмма рассеяния, показывающая значения признаков «средняя длина строки» и «доля пустых строк в коде». Для наглядности на диаграмме отображен 241 образец от 6 авторов. Точки, соответствующие образцам программного кода разных авторов, изображены разными маркерами.

Для данной выборки по 5 признакам обучена модель классификации с алгоритмом случайного леса в программном пакете WEKA (Waikato Environment for Knowledge Analysis). Перекрестная проверка показала, что данный классификатор обладает точностью классификации около 77%,

т.е. образцу кода ставится в соответствие верный автор в 7 из 9 случаев. При большем количестве признаков и использовании других алгоритмов классификации можно получить большую точность.

Заключение

В результате проведенного исследования были выделены 30 признаков, разделенных на 5 категорий, для анализа программного кода. Разработан инструмент для конструирования признаков и определения авторства программного кода.

СПИСОК ЛИТЕРАТУРЫ

1. Воробьева, М.С., Павлова Е.А. Разработка конструктора вариантов индивидуальных заданий на основе шаблонов / М. С. Воробьева, Е.А. Павлова // Современные исследования социальных проблем. – 2017. №8. – С. 214-217.
2. Aylin Caliskan-Islam, Richard Harang, Andrew Liu, Arvind Narayanan, Clare Voss, Fabian Yamaguchi, and Rachel Greenstadt. 2015. De-anonymizing programmers via code stylometry. In 24th USENIX Security Symposium (USENIX Security), Washington, DC.
3. Sudipta Mukherjee. 2017. Source Code Analytics with Roslyn and Javascript Data Visualization (1 издание). Apress, Berkely, CA, USA
4. Стрельчёнок, Г.В. Ступенчатый метод проверки исходного кода программы на плагиат, Санкт-Петербургский государственный университет, 2016 // URL: https://nauchkor.ru/pubs/stupenchatyy-metod-proverki-ishodnogo-koda-programmy-na-plagiat-587d36555f1be77c40d58cf6 (дата обращения: 15.05.2019).