ПРИМЕНЕНИЕ АЛГОРИТМОВ ДЛЯ ОПРЕДЕЛЕНИЯ ПЛАГИАТА В ПРОГРАММНОМ КОДЕ

Удовин И. А., Воронова В. В.
Кафедра информатики, Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь
E-mail: wilcot@ya.ru, veronika.voronova31@gmail.com

В данной статье рассматриваются алгоритмы применяемые для определения плагиата в программном коде. Проанализированы основные проблемы связанные со сложностью простого сравнения исходного кода программ, а также алгоритмы, позволяющие избежать данные проблемы. Была рассмотрена поисковая система, которая позволяет масштабировать обнаружение плагиата на больших объемах программного кода.

Введение

Плагиат является довольно распространенной проблемой в оцениваемых заданиях. На курсах по программированию студенты часто прибегают к плагиату исходных кодов программ, доступных в открытом доступе. Плагиат программного кода встречается и на соревнованиях по спортивному программированию, где количество проверяемых программ может достигать нескольких тысяч или десятков тысяч. Ручное определение плагиата в таких случаях нецелесообразно, если вообще осуществимо, за разумные сроки. Поэтому для определения плагиата необходимо использовать автоматизированные алгоритмы, позволяющие обрабатывать большие объемы исходных кодов программ.

Плагиат – это выдача чужого произведения за своё или незаконное опубликование чужого произведения под своим именем. Касательно программного кода, это относится к случаям копирования чужого кода. Причем в одних случаях копирование может осуществляться путем некоторого изменения названий объектов, изменением порядка кода, а в более сложных случаях – изменением структуры и замены одних конструкций другими. Также бывают случаи плагиата, когда программный код переписывается с одного языка программирования на другой. Если в случае простого копирования чужого кода можно обойтись простыми алгоритмами, например поиска наибольшей общей подстроки, то в остальных случаях необходимы более сложные алгоритмы, в том числе использующие методы машинного обучения.

Есть два фактора, которые усложняют обнаружение плагиата — это обилие доступных ресурсов и разнообразие методов, используемых для маскировки скопированных материалов. Есть ряд подходов для обнаружения плагиата в исходном коде программы, некоторые из которых будут рассмотрены в этой статье.

I. Сравнение факторов

При сравнении факторов, сходство двух программ оценивается по сходству различных показателей программного обеспечения, таких как среднее количество символов в строке, количество строк комментариев, количество строк с отступами, количество пустых строк и количество маркеров (например, ключевые слова, символы оператора и имена модулей стандартной библиотеки). Один из подходов сравнивает две программы на основе трех профилей:

- Физический профиль характеризует программу на основе ее физических атрибутов, таких как количество строк, слов и символов;
- Профиль Холстеда характеризует программу на основе использования типов токенов и их частот. Они включают количество вхождений токенов (N), количество уникальных токенов (n) и их объем ($Nlog_2n$);
- Составной профиль комбинация физического профиля и профиля Холстеда.

Для выявления плагиата профили каждой программы вычисляются, а затем нормализуются. Сходство двух программ оценивается путем вычисления евклидового расстояния между их профилями. Системы, которые используют сравнение факторов, могут быть очень нечувствительными (могут быть легко введены в заблуждение) или очень чувствительными (приводящими к множеству ложных срабатываний), поскольку они игнорируют структуру программы. Нарушители могут легко добавлять или удалять комментарии, переменные или избыточные строки кода, чтобы избежать обнаружения плагиата [1].

II. Сравнение структуры

Этот подход основан на убеждении, что сходство двух программ можно оценить по сходству их структур.

При таком подходе программы сравниваются в два этапа. Первый этап анализирует код и генерирует последовательности токенов, в то время как второй этап сравнивает токены.

Один из таких подходов – Sim. Он обнаруживает сходство между программами, оценивая их правильность, стиль и уникальность. Каждая программа сначала анализируется с помо-

щью лексического анализатора, который генерирует последовательность токенов. Токены для ключевых слов, специальных символов и комментариев предопределены, в то время как токены для идентификаторов назначаются динамически и хранятся в общей таблице символов, пробелы при этом отбрасываются. Поток токенов второй программы сгруппирован в разделы, каждый из которых представляет модуль программы. Каждый раздел отдельно выровнен с потоком токенов первой программы. Выравнивание двух строк выполняется путем вставки пробелов между символами для выравнивания их длины. Для расчета сходства используется схема оценки выравнивания. [2]

III. Сравнение промежуточного кода

Существующие системы обнаружения плагиата, основанные на сравнении факторов или структур, разработаны для обнаружения плагиата на определенном языке, таком как C, Java, Pascal и т.д. Мы называем это внутриязыковым плагиатом. Однако их такие подходы сложно применить для выявления межъязыкового плагиата, когда код на одном языке является плагиатом и отображается на другом. Для определения межъязыкового плагиата методом сравнения промежуточного кода необходимо разработать компилятор, способный генерировать промежуточный код для нескольких языков программирования.

Генерация промежуточного кода обычно происходит в несколько этапов:

1. Лексический анализ;
2. Синтаксический анализ;
3. Семантический анализ.

IV. Язык передачи регистров

Язык передачи регистров (RTL) GCC содержит ряд инструкций, представленных во вложенных круглых скобках. Каждая инструкция содержит номер строки, указатель на предыдущую инструкцию и указатель на следующую инструкцию, за которой следуют выражения. GCC обеспечивает три уровня оптимизации компилятора.

В оптимизированном RTL инициализация переменных выполняется путем сохранения значения в виртуальном регистре (представленном маркером reg), а не в виртуальном стеке (представленном reg и значением смещения). Это делает RTL-файл и, следовательно, индекс поисковой системы более компактным. Выбор оптимизации также приводит к вставке функций (инлайнинг), когда компилятор вставляет функции короче порогового значения (по умолчанию 600 строк) в вызывающий функцию код. На практике получается, что самый высокий уровень оптимизации приносит наибольшую пользу такому подходу, поскольку инструкции RTL максимально упрощены. Кроме того, оптимизация позволяет легко обнаруживать случаи, когда вставка функций используется для маскировки скопированного кода. GCC также предоставляет возможность компиляции для добавления отладочной информации в промежуточный код, однако некоторые эксперименты показывают, что эта дополнительная информация бесполезна.

V. Поисковая система

Система обнаружения плагиата, основанная на попарных сравнениях, плохо масштабируется. Хорошим альтернативным подходом является индексирование и запрос токенов с помощью поисковой системы. [3]

В таком случае, поисковая система может использоваться для выполнения двух задач:

1. На этапе индексирования индексируется сгруппированный RTL файлов исходного кода в коллекции;
2. На этапе обнаружения сгруппированный RTL исходного кода запроса используется для поиска по индексу, и программы из коллекции перечисляются в порядке убывания сходства с исходным запросом.

VI. Заключение

В данной статье были рассмотрены некоторые алгоритмы обнаружения плагиата в программном коде. Были рассмотрены методы основанные на сравнении факторов, сравнении структуры программ, а также методы основанные на сравнении промежуточного кода и RTL. Была также рассмотрена проблема масштабирования алгоритмов обнаружения плагиата в программном коде путем использования поисковой системы.

Список литературы

1. Arwin, C., Tahaghoghi, S. M. M. Plagiarism detection across programming languages. Proc. 29th Australasian Computer Science Conf.. – 2006. – Vol. 48. – P. 277–286.
2. Gitchell, D., Tran, N. Sim: a utility for detecting similarity in computer programs, in 'Proceedings of the Thirtieth SIGCSE Technical Symposium on Computer Science Education', ACM Press. – 1999. – P. 266–270.
3. Burrows, S., Tahaghoghi, S. M. M., Zobel, J. Efficient and effective plagiarism detection for large code repositories, in 'G. Abraham and B.I.P. Rubinstein Editors, Proceedings of the Second Australian Undergraduate Students' Computing Conference (AUSCC04)'. – 2004. – P.8-15.