УДК 004, 378.147.88

ВЫЯВЛЕНИЕ ПЛАГИАТА В ИСХОДНОМ КОДЕ ПРОГРАММ СТУДЕНТОВ

Е.Ю. Суханова, студент гр. ПИ-101, V курс
Научный руководитель: В.С. Дороганов, ст. преподаватель
Кузбасский государственный технический университет имени Т.Ф. Горбачева
г. Кемерово

На сегодняшний день плагиат получил широкое распространение не только в литературе и искусстве, но и в образовании. В Интернете существует масса ресурсов предоставления уже готовых рефератов, курсовых и даже дипломных работ по различным темам. Проблема плагиата в образовательном процессе заключается не только в нарушении авторских прав, но и в ухудшении качества образования. Скачивая реферат из Интернета или «одалживая» у товарища, студент не усваивает ничего нового, не учится думать самостоятельно. И даже, наоборот, привыкает пользоваться чужими мыслями, считая это нормой. Преподаватель в свою очередь лишается возможности объективно оценить уровень знаний своего подопечного – зачастую он даже не догадывается о том, что проверяет плагиат. [1]

Таким образом, преподавателям просто необходим сервис для проверки работ студентовна оригинальность. Подобных решений в Интернете не мало, например AntiPlagiat.ru – детектор, который доступен в режиме on-line. Данный сервис сравнивает проверяемую работу с источниками из Интернета и базами научных статей и рефератов.[1] Но такие сервисы могут проверять только текстовые работы.

В ВУЗах, подготавливающих будущих IT-специалистов, самостоятельные работы студентов заключаются в написании программного кода. Найти готовый программный код в Интернете гораздо сложнее, чем скачать готовый реферат, но это не мешает особо хитрым студентам брать лабораторные работы своих коллег или товарищей со старших курсов и выдавать их за свои. Так же студент может присвоить себе не всю работу целиком, а только ее часть, что так же является плагиатом.

Преподавателям при проверке работы, состоящей из программного кода, гораздо сложнее уличить студента в плагиате, чем при проверке реферата или курсовой. Студент может преобразить чужой код до неузнаваемости, изменив имена переменных, процедур и функций, изменив месторасположение некоторых блоков программ. Все эти модификации никак не сказываются на алгоритме, но делают практически невозможным визуальное сравнение работ между собой. Для выявления таких заимствований, необходимо сначала модифицировать программный код при помощи одного из нескольких методов:

1. Представление в виде элемента n-мерного пространства.

Суть данного метода заключается в представлении программы, как точки, расположенной в n-мерном пространстве натуральных чисел с нулем. Каждая i-ая координата точки – это количественная характеристика какого-либо свойства всей программы: количество объявленных переменных, средняя длина строк кода, имен переменных и т.д. Если точки двух программ лежат рядом, то одна из них предполагается плагиатом другой.

Для оценки расстояния между точками двух программ определяется их корреляция. Вычислив очередную характеристику на протяжении всей программы, получится усредненное значение. Таким образом, теряется много информации о структуре программы.

Данный метод имеет два недостатка: он выдает очень много ложных совпадений, и его легко обмануть даже при поверхностном изменении кода. Соответственно, если студент позаимствует только часть кода, крайне мала вероятность того, что метод обнаружит плагиат.

2. Метод, основанный на построчном сравнении исходного кода программ.

Данный метод работает как обычный сервис поиска заимствований в текстовых работах. В данном случае исходный код рассматривается "как есть". Очевидно, что такое сравнение программ крайне неэффективно. Любой студент, обладающий даже начальными навыками программирования, сможет обмануть детектор, основанный на этом методе.

3. Представление абстрактного синтаксического дерева

Для того чтобы при сравнении программ не учитывать бесполезные характеристики (например, порядок следования независимых операторов), можно воспользоваться методом представления исходного кода в виде дерева. Данный метод при модификации кода учитывает только логику управления программы.

Суть метода заключается в представлении кода в виде дерева, описание которого хранится в формате XML.И несомненным плюсом использования XML является значительное количество стандартных инструментов для работы с ним, что упростит архитектуру разрабатываемого детектора. Процедура сравнения деревьев сведена к анализу числовых матриц, что является одним из плюсов данного метода, т.к. такой анализ не требует вычислительных сложностей.

Минусы метода заключаются в ограниченности его применения: он подходит только для процедурных языков. А также при сравнении нескольких программ, оценка близости осуществляется попарно.

4. Токенизация.

Алгоритм токенизации достаточно простой и состоит всего из двух этапов:

*1. Преобразование операторов языка в токены.*Каждому из операторов языка программирования, кроме операндов, присваивается уникальный код

(токен). Значение каждого токена назначается заранее и соответствует определенному классу операторов.

*2. Преобразование исходного кода оцениваемой программы в последовательность токенов.*Изполученных на первом этапе токенов строится строка, с сохранением порядка их следования в исходном коде программы. На этом этапе отбрасываются бесполезные при сравнении программ характеристики (пробелы, отступы, имена процедур, переменных и т.д.).

Отбрасывание бесполезных характеристик сравнения является одним из преимуществ данного метода. Также при определенном методе сравнения токенов можно выявить одинаковые места кода даже тогда, когда они находятся в разных местах программы. Другими словами, студент не сможет обмануть преподавателя, изменив месторасположение заимствованной процедуры или функции. Метод представления исходного кода в виде дерева не справится с такой задачей.[2]

Таким образом, для разработки собственного детектора проверки программного кода на плагиат, за основу был взят метод токенизации. Сравнение исходного кода программы происходит с базой лабораторных работ, сделанных другими студентами.

Для сравнения программного кода, написанного на разных языках программирования, были созданы таблицы сопоставления операторов. То есть однотипные операторы разных языков определяются как одинаковые. Результатом сравнения двух является график (рис. 1). На оси ординат расположены порядковые номера операторов. На оси абсцисс – место оператора в исходном коде. Лабораторные работы отражаются разными по цвету линиями, и их наложение друг на друга может означать заимствование кода.

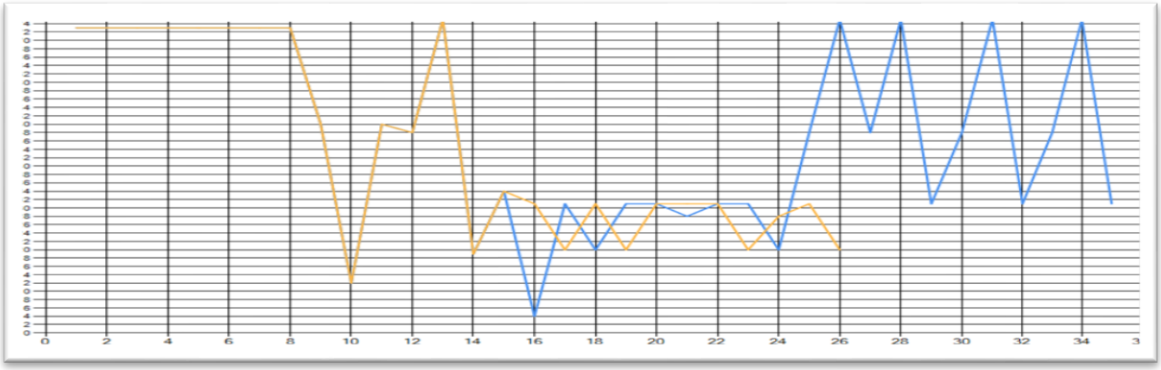

Рис. 1. Сравнительный график двух лабораторных работ

На представленном графике (рис. 1) изображено сравнение двух лабораторных работ. Программный код, отображенный синей линией, состоит из 35 операторов, оранжевой – из 26 операторов. С первого по четырнадцатый оператор происходит полное наложение линий. Это означает, что в этом месте в обеих программах использовались одинаковые операторы. Можно сделать вывод о заимствовании части программы.

Для увеличения скорости работы программы, лабораторные работы разбиты на темы. Сравнение происходит только внутри темы, что позволяет не сравнивать между собой изначально разные лабораторные работы. [3]

Анализ данного метода показал, что она не подходит для сравнения программ, написанных на современных языках. Причиной этому является объектно-ориентированный и процедурный подход: перемена места расположения процедуры не изменит работы программы, но данный алгоритм не заметит заимствования. Для улучшения работы была разработана методика, суть которой заключается в том, что сравнивается не полностью совпадение всех операторов программы, а отыскиваются идентичные последовательности команд. Заимствованием считается одинаковая последовательность операторов в количестве более трех.

Итогом такого анализа является таблица (таблица 1), в которой можно увидеть авторов (столбцы 1-2) и размер работ (столбец 3), с которыми сравнивали оцениваемую программу, максимальную длину заимствованной последовательности (столбец 4), количество найденных заимствований (столбец 5), среднюю длину заимствований (столбец 6) и процент исходного кода, который был обнаружен в других работах (столбец 7). Можно увидеть, что один студент упоминается дважды: его работы были выполнены на разных языках и участвовали в сравнении.

Таблица 1. Сравнительная таблица лабораторных работ данной темы

Фамилия	***Группа***	***Количество операторов***	***Максимальная цепочка***	***Количество цепочек***	***Средняя длина цепочки***	***Процент заимствования***
1	***2***	***3***	***4***	***5***	***6***	***7***
Иванов	ПИб-00	81	4	1	4	22
Петров	ПИб-00	17	4	1	4	22
Петров	ПИб-00	28	0	0	0	0
Сидоров	ПИб-00	23	5	2	5	53

Данный программный продукт поможет преподавателям выявлять участки заимствования в программном коде. Сравнение лабораторных работ происходит не только между студентами, учащимися в одной группе, но и с другими студентами, когда-либо выполнявших данное задание. Таким образом, будет исключено заимствование лабораторных не только у друзей из своей группы, но и у студентов старших курсов.[4]

Список литературы:

1. Интернет-сервис Антиплагиат.Ру [электронный ресурс]. – Режим доступа:http://www.antiplagiat.ru/index.aspx, заглавие с экрана.
2. Обзор автоматических детекторов плагиата в программах [Электронный ресурс].– Режим доступа:http://logic.pdmi.ras.ru/~yura/detector/survey.pdf, заглавие с экрана.

3. Дороганов В.С, Суханова Е.Ю. Способы выявления плагиата в исходном коде программ // Информационно-телекоммуникационные системы и технологии (ИТСиТ-2014). - Кемерово: Кузбас. гос. техн. ун-т им. Т.Ф. Горбачева, 2014. - С. 176.
4. Дороганов В.С, Суханова Е.Ю. Способы выявления плагиата в исходном коде программ студентов // Информационные системы и технологии в образовании, науке и бизнесе (ИСиТ-2014). - Кемерово: Кузбас. гос. техн. ун-т им. Т.Ф. Горбачева, 2014. - С. 85-86.