АНАЛИЗ СИСТЕМ ПОИСКА ПЛАГИАТА В ТЕКСТАХ КОМПЬЮТЕРНЫХ ПРОГРАММ

Лось Н.С.
Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь
Черкас Н.Л. – кандидат физ.-мат. наук

Рассмотрена классификация методов поиска плагиата в исходном коде программ.

Программа для компьютера отличается от обычного текста и ей присущи специфичные характеристики: структурированность, зависимость от входных данных. Кроме того, программа может быть показана в виде исходного текста (исходного кода), промежуточного (объектного) кода и исполняемого кода. Каждое из указанных представлений может использоваться как материал для поиска плагиата.

В исходном тексте находится много характеристик, свойственных конкретному автору (наименование идентификаторов, стиль написания кода, лингвистические особенности, количество разных типов данных и др.), большинство из которых при компиляции, утрачиваются. Но и в исполняемом коде хранится много индивидуальной информации: организация данных, информация об использованном компиляторе, системные функции и вызовы, специфические ошибки и т.д. В промежуточном и исполняемом кодах основное внимание уделяется логике выполнения программы и её управления.

Принято выделять следующие способы оценки близости программ: атрибутно-подсчетный, структурный и комбинированный, сочетающий в себе первых два.

Атрибутные методы появились исторически первыми. Их смысл заключается в численном выражении некоторых характеристик программы, например, размер файла программы, количество исполняемых конструкций или число переменных, и сравнении полученных чисел для разных программ, чаще всего такую информацию называют метаданными, их легко вычислить и в связи с небольшим объемом эти данные можно хранить вместе с самой программой. Программы с близкими численными характеристиками атрибутов потенциально похожи. Можно комбинировать несколько параметров так, чтобы программа была представлена набором числовых характеристик. Две программы могут считаться похожими, если соответствующие числа из их наборов близки или совпадают. Таким образом, оценка близости программ сводится к сравнению чисел или векторов, которые получаются путем несложного анализа непосредственно исходного кода. Основным недостатком атрибутных техник является то, несвязанные между собой параметры программы плохо описывают ее в целом. Следовательно, при таком подходе разные программы получают близкие характеристики. Вдобавок, достаточно просто обмануть систему, построенную по данному принципу. Размер файла можно расширить большим количеством фрагментов кода, которые никак не влияют на непосредственные вычисления в программе, но увеличивают как ее размер, так и количество исполняемых конструкций.

Структурные методы исследуют свойства программы не изолированно, а как бы в контексте, устанавливают взаимосвязь различных параметров программы, их совместное поведение. Чтобы отбросить персонализированную информацию и выделить нужные зависимости, программа предварительно переводится в более компактное и абстрактное представление (выполняется токенизация исходного кода). Классическим примером структурного подхода является синтаксическое построение дерева программы с последующим сравнением деревьев для различных программ. Недостатком структурных методов является их сложность в построении, сравнении и вычислительная трудоемкость. Кроме того, структурные методы обычно разрабатываются под синтаксис каждого конкретного языка программирования отдельно. Адаптация метода для различных языков требует значительных усилий. Сложность реализации алгоритмов, использующих структурные методы, является платой за точность этих алгоритмов. Структурные методы наиболее приближены к поиску, который осуществляет человек при проверке текста, зачастую данные методы оказываются точнее оценки человека.

Комбинированный подход целесообразно использовать для поиска плагиата среди большого количества программ. Для этого на первом этапе с помощью атрибутных методов можно отсеять непохожие программы. На втором этапе выполняется более детальное сравнение оставшихся программ каким-либо структурным методом. Таким образом, из-за предварительного несложного анализа сокращается количество попарных сравнений при поиске плагиата, а, следовательно, растет эффективность.

Чаще всего программы подвергаются плагиату на уровне исходного текста, реже – на уровне исполняемого кода. В связи с этим, большинство алгоритмов и разработанных на их основе детекторов осуществляет поиск плагиата в исходном и промежуточном представлениях программ. [1]

Среди структурных методов обнаружения наиболее распространенными являются:

String-based ищет точные совпадения строк, например, 5 подряд одинаковых слов. Данный подход является очень быстрым, простым в реализации, но чувствительным к изменению имен идентификаторов;

Token-based – подход очень напоминающий первый по своей сути, но с небольшим отличием - предварительно исходный код программы шифруется в последовательность токенов. Таким образом, удаляются лишние пробелы, комментарии, имена переменных, делая систему более устойчивой к простым текстовым изменениям. Большинство академических программ-детекторов работают таким образом, применяя различные алгоритмы строкового сравнения для поиска дублирующих фрагментов последовательности токенов;

Tree-based – построение и сравнение деревьев разбора. Это позволяет обнаруживать схожести более высокого уровня. Например, сравнение деревьев может нормализовать условные конструкции и определить их схожесть даже с глобальными синтаксическими изменениями. Для каждого файла с исходным кодом строится абстрактное синтаксическое дерево, затем полученные деревья сравниваются между собой;

PDG-based (PDG - Program Dependency Graph) - строится граф зависимостей, который отражает зависимость между управляющими конструкциями программы (control dependency). т.е. как переходит управление из одной точки программы в другую, и между потоками данных (data dependency). Затем полученные графы для каждого файла с исходным кодом сравниваются между собой. Граф представляет собой полный поток контроля в программе, что позволяет найти гораздо более сложные уровни плагиата, хоть и ценой большой сложности в реализации и времени работы;

Metrics-based данный подход заключается в оценке метрик программы, например, количестве используемых переменных, циклов, условных операторов и т.д. Сравнение происходит после предварительного преобразования программы, поэтому данный метод относится к комбинированному, а не к атрибутному или структурному. Далее две программы сравниваются по соответствующим метрикам. Но следует осторожно относиться к данному методу, так как возможен вариант абсолютного совпадения метрик у совершенно разных по смыслу программ;

Hybrid-based - гибридные подходы основаны на сочетание нескольких из выше перечисленных подходов. [2]

Список использованных источников:

1. Макаров, В.В. Идентификация дублирования и плагиата в исходном тексте прикладных программ / В.В. Макаров// Лаборатория компьютерной графики [Электронный ресурс]. - 2016г. - Режим доступа: http://lab18.ipu.rssi.ru/projects/conf2006/1/%D0%92.%D0%92.%D0%9C%D0%B0%D0%BA%D0%B0%D1%80%D0%BE%D0%B2.htm. - Дата доступа: 20.09.2019.

2. Manber, U. Finding similar files in a large file system. / U. Manber // USENIX Winter 1994 Technical Conference reference book. - San Francisco, 1994 – p. 1–10.