ПОИСК ПЛАГИАТА В ИСХОДНЫХ ТЕКСТАХ ПРОГРАММ

А. И. Логинов

ВВЕДЕНИЕ

В последнее время нередки скандалы в сфере высоких технологий о присвоении одной компании интеллектуальной собственности другой: в апреле 2011 года компания Oracle подала в суд на компанию Google о незаконном копировании последней более ста тысяч строк кода. Судебные тяжбы длились больше года, от вынесенного вердикта мог быть создан прецедент в мировой судебной практике.

Присяжные не смогли дать точного решения по этому вопросу: было решено, что Google не нарушал права Oracle на комментарии и документацию в коде, но нарушил копирайт, скопировав 9 строк кода.

Произошедшая ситуация является достаточно серьезной не только для enterprise-мира, но и для многих других областей, связанных с информационными технологиями. Не осталась в стороне и сфера образования.

Во всем мире студенты постоянно сталкиваются с заданиями по программированию алгоритмов на различных языках. Зачастую их преподаватели невольно испытывают ощущения, что некоторые из этих программ являются, по сути, копиями – явными или с некоторыми изменениями – программ, которые были написаны другими студентами.

Именно поэтому мощная автоматизированная система была бы очень актуальна для поиска нечестных заимствований.

РАЗРАБОТКА СИСТЕМЫ ВЫЯВЛЕНИЯ ПЛАГИАТА

Методы и алгоритмы разработанной программы

Проведя изучение предметной области, было выделено следующее:

- каждому исходному коду исследуемой программы необходимо поставить в соответствие ее токенизированное представление;
- используемые методы и алгоритмы не должны быть ресурсоемкими.

На текущий момент реализованная программа способна отыскать плагиат в исходных текстах на языке Java несколькими способами: с помощью анализа характеристик программ и построения по ним наглядной

диаграммы, а также с использованием метода отпечатков (как с токенизацией, так и без нее).

Токенизация исходного кода программ

Для токенизации исходного кода необходимо прочитать последовательность символов, а также создать структуру данных, которая будет отражать эту последовательность. В настоящее время есть бесплатные удобные средства, которые позволяют генерировать синтаксические анализаторы, стоит написать лишь грамматику. Для создания API, выполняющего токенизацию, использовался фрйэмворк SableCC.

Поскольку каждая программа, так или иначе, использует стандартный набор API функций, предложенный в свое время компанией Sun, то было принято решение сохранять в процессе токенизации все типы параметров, передаваемых в методы, названия классов и интерфейсов, участвующих в наследовании.

Все встречающиеся числовые и символьные литералы, циклы и ветвления заменяются заранее заданными токенами.

Метод отпечатков

Метод отпечатков (другое название «метод идентификационных меток») позволяет находить как копии, так и частичные копии файлов в гипотетической базе большого размера[1]. Суть метода заключается в том, что на вход поступает строка, затем она разбивается на подстроки длины k, которые последовательно хэшируются.

Для большей эффективности необходимо рассматривать не все множество полученных хэш-значений, но выбирать какое-то их подмножество. Для этого существует несколько подходов. Наилучшим из них считается метод просеивания[2], описание которого приведено далее.

Необходимо ввести несколько обозначений: пусть есть окно размера w, которое представляет собой w последовательных хэш-значений полученных k-грамм в документе. Выбор коэффициента k (шумовой порог) позволяет контролировать ложные срабатывания. Ясно, что чем размер шумового порога выше, тем меньше будет случайных совпадений. В то же время, с ростом k падает устойчивость метода к перестановкам.

Идея алгоритма состоит в следующем: продвигается окно размера $w = (t - k + 1)$ вдоль последовательности $h_1 \ldots h_n$. На каждом шаге окно перемещается на одну позицию вправо. Меткой назначается минимальное h_j в окне. Если в одном окне два элемента принимают минимальное значение, то меткой считается крайнее правое.

В процессе реализации нетривиальным оказался подбор коэффициентов: k – длина игнорируемой строки-совпадения и t – гарантированная длины подстроки, в которой совпадения всегда может быть определено.

В различных источниках говорится, что за длину шумового порога (т.е. игнорируемой строки) следует взять среднюю длину всех слов текста. В данном же случае применительно к языкам программирования рекомендовалось взять среднюю длину всех ключевых слов. Длине же гарантированный подстроки t полагалось быть как минимум не меньше длины шумового порога.

Однако, несмотря на такие рекомендации, проведя множество тестов, получились неоднозначные результаты: процент совпадений был чрезвычайно велик (более 80% для не сильно связанных между собой программ и более 90% для программ, факт плагиата в которых просто неоспорим). Поэтому, проанализировав исходные коды нескольких контрольных работ младших курсов, получилось достичь приемлемого достоверного результата. Определив длину шумового порога не меньше, чем суммарная длина двух ключевых слов, удалось уменьшить завышенные оценки. Теперь очень далекие друг от друга программы показывают соответствующий низкий показатель совпадений (около 15%). В то же время показатель довольно схожих программ остался практически на уровне (около 70%).

Анализ характеристик программы

Любая программа имеет определенную иерархию структур, которые могут быть выявлены, измерены и использованы в качестве характеристик программы. Для этой цели лучше всего подходит последовательность операторов программы, поскольку модификация этой последовательности требует глубокого понимания логики функционирования программы и является трудоемким и нетривиальным процессом.

Для анализа размещения операторов в теле программы берутся две последовательности операторов и сравниваются с целью получения взаимной корреляции операторов этих последовательностей.

Сравнение проводится последовательным сдвигом меньшей последовательности относительно большей. На каждом смещении подсчитывается количество совпадений на одинаковых позициях, которое затем сохраняется.

По полученным на каждом сдвиге числам затем строится гистограмма, на абсциссе которой откладывается величина смещения одной последовательности операторов относительно другой, а на ординате – количество совпадений операторов при таком смещении.

Пик, наблюдаемый на какой-либо позиции, сигнализирует о том, что в этих программах встречаются одинаковые фрагменты.

Анализ полученных результатов

Еще при первоначальном изучении тематики возникла проблема, что же считать плагиатом. Несомненно, хочется думать, что плагиат – это какое-то переделывание чужих трудов, за которое может наступать определенная ответственность. Такое суждение порождает новые вопросы:

- насколько сильно должны отличаться программы, чтобы одну из них все еще можно было считать плагиатом;
- какие доказательства считать достаточными для плагиата.

С помощью смещения токенов одной программы относительно другой можно получить лишь небольшой шанс выявить плагиат, поскольку данный подход очень чувствителен к перестановкам строк, либо же вставкам кода из совершенно других программ. Тем не менее, можно выделить один неоспоримый плюс: его можно визуализировать так, чтобы человеку, ищущему совпадения в исходных кодах программ, было легко заметить использование плагиата.

Метод отпечатков с использованием токенизации устойчив как к перестановке строк, так и к различным переименованиям используемых единиц языка программирования. В то же время, ввиду того, что исходный код заменяется его токенизированным представлением, возможны случайные совпадения.

В моем проекте обе реализации метода отпечатков (как с токенизацией, так и без нее) после завершения поиска плагиата оценивают каждую пару входных файлов числом. Если одно из полученных чисел больше 50, то одна делается вывод о том, что одна из сравниваемых программ является плагиатом другой.

ЗАКЛЮЧЕНИЕ

К сожалению, в настоящее время практически нет какого-то одного общего универсального способа обнаружить плагиат. Такого способа, который был бы неподвластен времени. Злоумышленники постоянно будут пытаться находить новые способы борьбы с системами, специально разработанными против них.

Так или иначе, на данный момент все известные способы спрятать факт использования плагиата данной программой пресекаются. Если плагиат есть, то он будет обнаружен.

Литература

1. Интернет-адрес: http://www.cs.cmu.edu/afs/cs/user/nch/www/koala/main.html.
2. Интернет-адрес: http://theory.stanford.edu/~aiken/publications/papers/sigmod03.pdf.

МОНИТОРИНГ ИЗМЕНЕНИЙ ЗЕМНОЙ ПОВЕРХНОСТИ НА ОСНОВЕ СПУТНИКОВОЙ ИНТЕРФЕРОМЕТРИИ

А. А. Луговский

ВВЕДНИЕ

В настоящее время для эффективным средством для мониторинга крупномасштабных изменений земной поверхности является спутниковая РСА-интерферометрия. Метод РСА-интерферометрии применяется для мониторинга деформаций в случае землетрясений, вулканической деятельности, оползней, просадок грунтов и движения ледников[3; 4].

В частности, для нужд Республики, наиболее актуальной, является задача мониторинга деформации земной поверхности в районах интенсивной добычи полезных ископаемых.

В работе описан программный комплекс для решения задач в области мониторинга земной поверхности на основе технологии спутниковой интерферометрии, включающий средство интерферометрической обработки Doris и геоинформационную систему Saga. Приведены результаты применения приложения для выявлений смещений в районе Старобинского месторождения.

ПОСТАНОВКА ЗАДАЧИ

Целью данной работы является разработка программного комплекса для анализа и обработки интерферометрических данных. Для эффективной и качественной интерферометрической обработки приложение должно удовлетворять требованиям: включать основные этапы обработки данных; иметь модульную структуру; обладать удобным графическим интерфейсом; поддерживать форматы данных основных спутников; иметь набор механизмов обработки векторных и растровых данных.

Для решения поставленной цели были решены следующие задачи:

1. Проведен анализ существующих пакетов и выбрана система для интерферометрической обработки.

2. Проведен анализ существующих пакетов и выбрана система для визуализации данных.