МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра информатики и программирования

ПРИМЕНЕНИЕ МЕТОДОВ ПОДСЧЕТА СХОЖЕСТИ СТРОК И МАШИННОГО ОБУЧЕНИЯ В ЗАДАЧЕ ПОИСКА ПЛАГИАТА В ПРОГРАММНОМ КОДЕ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 441 группы
направления 02.03.03 — Математическое обеспечение и администрирование информационных систем
факультета КНиИТ
Огневой Татьяны Алексеевны

Научный руководитель
ст. преп. _____ Е. Е. Лапшева

Заведующий кафедрой
к. ф.-м. н., доцент _____ М. В. Огнева

Саратов 2023

ВВЕДЕНИЕ

Актуальность темы. Задача поиска плагиата в программном коде особенно актуальна в настоящее время для образовательного процесса. Количество курсов, связанных с написанием программ, постоянно увеличивается, также как и число их участников. Не секрет, что всегда есть недобросовестные студенты, которые пытаются предоставлять чужие решения (возможно, немного видоизмененные) или использовать фрагменты чужого кода. В программе могут быть изменены имена переменных и функций, переформатирован исходный код, вставлены лишние переменные или функции и части кода, которые не используются, заменены синтаксические конструкции на конструкции аналогичного действия (например, цикл for на цикл while), изменены типы переменных (например, целый знаковый на целый беззнаковый) и так далее.

Простым сравнением файлов не всегда можно распознать такие случаи, требуется специальное программное обеспечение (ПО).

Существуют различные методы поиска плагиата в программном коде. Каждый из них имеет свои достоинства и недостатки и может применяться для различных видов заимствований.

Цель бакалаврской работы — разработать систему поиска плагиата в программном коде.

Поставленная цель определила **следующие задачи:**

1. Провести обзор существующих систем поиска плагиата в программном коде.
2. Разобрать методы обнаружения плагиата в программном коде.
3. Разобрать методы кластеризации.
4. Реализовать расстояние Левенштейна, метод выравнивания, расстояние Дамерау-Левенштейна, алгоритм Хескела.
5. Реализовать предобработку для Python и C++.
6. Реализовать систему, позволяющую работать с большим количеством файлов.
7. Протестировать и отладить работу системы на программных решениях разного размера, выгруженных с сайта school.sgu.ru.
8. Проанализировать полученные результаты.
9. Реализовать выделение кластеров студентов со схожими решениями с помощью кластеризации.

Практическая значимость бакалаврской работы заключается в том, что разработанная система может использоваться для выявления кластеров студентов с заимствованными решениями. Она распознает наиболее распространенные выиды заимствования, такие как случаи переформатирования кода (изменение отступов, добавление / удаление комментариев), переименования идентификаторов (переменных, функций, библиотек), изменения типов переменных и значений, которые возвращают функции.

Структура и объём работы. Бакалаврская работа состоит из введения, 5 разделов, заключения, списка использованных источников и 19 приложений. Общий объем работы — 79 страниц, из них 53 страниц основное содержание, включая 10 рисунков и 17 таблиц, список использованных источников информации — 29 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Обзор существующих детекторов плагиата в программном коде» посвящен обзору существующих детекторов плагиата в программном коде.

SIM — это система проверки сходства программного обеспечения для программ. Она обнаруживает сходство между программами путем оценки их правильности, стиля оформления и уникальности.

JPlag был разработан Гвидо Мальполем в Университете Карлсруэ. JPlag преобразует программы в строки маркеров, которые представляют собой структуру программы. Для сравнения двух токенов JPlag использует алгоритм жадного строкового замощения.

MOSS — это автоматическая система для определения сходства программ. Он предоставляется в виде веб-сервиса, к которому можно получить доступ с помощью скрипта. Для измерения сходства между документами, MOSS использует алгоритм «отпечатков пальцев».

Второй раздел «Существующие методы распознавания плагиата в программном коде» посвящен методам и алгоритмам детектирования плагиата в программном коде.

Наиболее продуктивными и известными алгоритмами поиска плагиата в программном коде являются алгоритмы, использующие сравнение фактических строк исходного кода программы, лексический анализ, деревья синтаксического анализа, и графы зависимостей.

Первые два типа алгоритмов используют методы сравнения схожести строк, рассмотрим некоторые из них подробнее.

Начнем с алгоритма, основанного на выравнивании строк. Пусть у нас есть два исходных текста программ, представим их в виде строк токенов s и t соответственно (возможно, различной длины). Выравнивания двух строк s' и t' соответственно получаются с помощью вставки в них пробелов таким образом, чтобы их длины стали одинаковыми.

Цена выравнивания — это сумма индивидуальных стоимостей всех пар s' и t', наибольшее значение этой целевой функции для всех i и j ($i \leq j \leq |s'| = |t'|$) на строках $s'[i..j]$ и $t'[i..j]$ — величина выравнивания.

Оптимальное выравнивание между двумя строками — максимальное значение целевой функции среди всех выравниваний. Это значение может быть

вычислено с помощью динамического программирования.

Алгоритм Хескела также работает с парами строк токенов. Результатом работы алгоритма является числовое значение — некоторый коэффициент сходства, который в литературе обозначается как коэффициент Жаккара. В общем случае алгоритм можно разбить на 3 шага: формирование *N*-грамм, подсчет уникальных *N*-грамм в каждой из программ, и вычисление коэффициента Жаккара.

Полученный коэффициент можно считать приблизительной оценкой сходства двух программ.

Расстояние Левенштейна (редакционное расстояние) — минимальное количество операций вставки, удаления или изменения символа, необходимое для преобразования одной строки в другую.

Цены операций могут зависеть от вида операции (вставка, удаление, замена) и/или от участвующих в ней символов, отражая разную вероятность разных ошибок при вводе текста, и тому подобное. Для решения задачи о редакционном расстоянии необходимо найти последовательность замен, минимизирующую суммарную цену. Расстояние Левенштейна является частным случаем этой задачи, когда цены всех операций равны 1.

Расстояние Дамерау-Левенштейна — минимальное количество операций вставки, удаления, изменения и замены символа, необходимое для преобразования одной строки в другую.

В третьем разделе «Характеристики стиля оформления программного кода» рассмотрены правила оформления программного кода.

Значимой отличительной особенностью программного кода является именование составных идентификаторов, состоящих из двух и более слов. Слова в идентификаторе могут быть разделены знаком подчеркивания.

В C/C++ любая обособленная часть кода должна выделяться фигурными скобками ({ и }). Существует несколько способов выделения логически-обособленных фрагментов.

В соответствии со стилем “K&R“ скобка переносится на новую строку при определении пространств имен, классов, функций, в остальных случаях остается на той же строке, где располагается часть кода, к которой она (скобка) принадлежит.

В соответствии со стилем Олмана открывающая скобка переносится на

следующую строку на том же уровне, что и оператор (заголовок функции).

Стиль Уайтсмита — использование отступов для скобок — популярен из-за примеров, шедших с Whitesmiths C — ранним компилятором с языка C.

Во многих языках программирования отступы используются для форматирования исходного кода программы для улучшения читабельности. Однако некоторые языки программирования (Haskell, Occam, Python) используют отступы для синтаксического выделения блоков кода вместо операторных скобок.

Знаки пробела, используемые при оформлении логических и арифметических выражений, не влияют на логическую структуру программного кода. Они позволяют представить выражение в более удобной для чтения форме.

Еще одной очень важной составляющей оформления программного кода является его грамотное документирование. Комментирование программного кода не является обязательным требованием, поэтому, зачастую, факт наличия комментариев и их содержание, является характерной чертой индивидуального стиля программирования. В случае анализа студенческих работ, наличие комментариев может судить о самостоятельном написании программы так как, в отсутствии обширного опыта программирования, большинство студентов не комментируют свой программный код.

«Родимые пятна» — это особенности, которые присущи стадии разработки программы. Например, части кода из более ранних версий программы, фрагменты, написанные при разработке программы, но не удаленные после того, как они стали ненужными, ошибки.

Воспроизведение ошибки может служить доказательством заимствования, поскольку маловероятно, чтобы один человек самостоятельно и независимо повторил ту же ошибку, что и другой.

В четвертом разделе «Кластеризация» рассматриваются некоторые алгоритмы обучения без учителя и способы оценки их работы.

Кластеризация — это категория методов обучения без учителя, позволяющая обнаруживать скрытые структуры в данных, где мы заранее не знаем правильного ответа. Задача кластеризации состоит в том, чтобы отыскать в данных естественное разбиение по группам, такое что элементы в том же кластере более подобны друг другу, чем из других кластеров.

Существуют различные методы кластеризации, рассмотрим подробнее три из них: метод k средних, агломеративную кластеризацию и DBSCAN.

Алгоритм k средних принадлежит к категории кластеризации на основе прототипов. Кластеризация на основе прототипов означает, что каждый кластер представлен прототипом, который может быть либо центроидом (средним) подобных точек с непрерывными признаками, либо медоидом (наиболее представительной или наиболее часто встречающейся точкой) в случае категориальных признаков.

Альтернативный подход к кластеризации на основе прототипов — иерархическая кластеризация.

Существуют два основных подхода к иерархической кластеризации: агломеративный (объединительный) и дивизивный (разделяющий). В агломеративной кластеризации мы начинаем с каждого образца как отдельного кластера и объединяем ближайшие пары кластеров, пока не останется всего один кластер.

Еще один подход, применяемый в кластеризации — плотностная пространственная кластеризация приложений с присутствием шума (density-based spatial clustering of applications with noise, DBSCAN). Понятие плотности в DBSCAN определяется как число точек внутри указанного радиуса s.

Одно из основных преимуществ использования алгоритма DBSCAN состоит в том, что он не делает допущения о сферичной форме кластеров, как в алгоритме k средних. Кроме того, алгоритм DBSCAN отличается от кластеризации по методу k средних и иерархической кластеризации тем, что он с необходимостью не назначает каждую точку кластеру и одновременно способен удалять шумовые точки.

В отличие от классификации, трудно оценить качество результатов кластеризации. Здесь метрика не может зависеть от меток, а только от добротности разбиения.

Существуют внутренние и внешние метрики качества. Внешние метрики используют информацию об известном истинном разбиении, а внутренние метрики не используют никакой внешней информации и оценивают добротность кластеров только на основе исходных данных. Оптимальное количество кластеров обычно определяется относительно некоторых внутренних метрик.

Пятый раздел «Поиск и анализ плагиата в программном коде обучающихся на портале обучения информатики и программирования school.sgu.ru» посвящен реализации системы детектирования плагиата в программном коде для портала school.sgu.ru.

На базе Саратовского государственного университета с 2006 года существует специализированный портал school.sgu.ru, предназначенный для интернет-обучения основам алгоритмизации и программированиях. Ядром портала является автоматическая проверка заданий по программированию с помощью тестирующей системы — контестера. Он используется для обучения программированию школьников и студентов и содержит более 800 задач начального и среднего уровней сложности. При работе с контестером всегда находятся недобросовестные студенты, пытающиеся представить чужие решения или использовать фрагменты чужого кода.

Проверка «глазами» возможна только в случае решения нескольких задач небольшого размера, здесь же речь идет о группах и потоках студентов. Поэтому для решения проблемы поиска плагиата требуется специальное программное обеспечение.

Рассматриваемый алгоритм поиска плагиата в программном коде состоит из нескольких этапов.

Исходные данные — множество решений — представляют собой файл csv. Для удобства проверки эти данные преобразовывались в набор папок с файлами.

Из полученных папок производилось удаление решений следующим образом: если студент посылал одну задачу несколько раз, рассматривалось только последнее по времени решение.

В данной работе были реализованы следующие методы (с использованием языка программирования Python и его библиотек):

- нахождение расстояния Левенштейна,
- метод выравниваний,
- нахождение расстояния Дамерау-Левенштейна,
- метод Хескела.

Для всех методов на вход подается последовательность символов “K“, “O“, “I“, “N“ для Python и “K“, “V“, “O“, “I“, “N“ для C++; на выходе получаем меру схожести.

Для этих методов была реализована предобработка, которая включает в себя два этапа:

1. Токенизация. Каждому оператору выбранного языка программирования (в работе Python и C++) ставится в соответствие некоторая метка, назна-

ченная заранее для каждого класса операторов, метки также могут приписываться блочным операторам и операторам подключения библиотек. По полученному набору меток строится строка, причем порядок этих меток сохраняется в соответствии с их порядком следования в исходном коде программы. Замена производится при помощи словаря токенов на основе регулярных выражений. На вход подается исходный код программы; на выходе получаем токенизированное представление кода.

2. Нормализация — удаление пробельных символов, скобок, некоторых знаков пунктуации, комментариев и текстов внутри операторов вывода. На вход подается токенизированное представление кода; на выходе получаем нормализованную последовательность символов.

Для C++ этапы идут в обратном порядке.

Система была протестирована на программных кодах из системы автоматической проверки задач на сайте school.sgu.ru (40288 решений на Python и 42941 решений на C++).

После проверки каждого контестера создавался датасет, в который записывались результаты.

В процессе тестирования системы на решениях с контестера был выявлен и исправлен ряд недочетов:

1. обработка пустых файлов с решениями;
2. обработка некорректных программ;
3. обработка ключевых слов внутри идентификаторов;
4. обработка букв из русского алфавита в именах переменных;
5. обработка символа “@“ в коде программ;
6. некорректное распознавание пар символов “<<“ и “>>“.

Для того, чтобы убедиться в правильности работы фрагментов программы, реализующих предобработку, была написана специальная программа.

Для удобства анализа результатов все датасеты результатов проверки программ на одном и том же языке программирования были объединены.

Было проанализировано распределение длин программ в символах, в строках и в символах после обработки: посчитаны числовые характеристики.

С помощью фильтрации и построения корреляционных матриц были проанализированы коэффициенты схожести. Результаты были визуализированы с помощью диаграмм разброса.

Представление результатов в виде пар решений с коэффициентами не очень хорошо воспринимается глазами, так как студенты часто заимствуют программы группами больше, чем по двое.

Процесс выделения таких групп вручную очень сложен, нужно использовать кластеризацию.

Из методов кластеризации был выбран DBSCAN:

— Для него не нужно задавать число кластеров (в отличие от метода k средних и агломеративной кластеризации). Для задачи поиска плагиата заранее неизвестно количество групп похожих решений.
— Он выделяет шумовые точки. Для задачи поиска плагиата это студенты, не заимствовавшие решения у товарищей.
— Нужно отрегулировать максимальное расстояние между объектами, то есть границу коэффициента схожести, определяющую, заимствованы решения или нет.
— Нужно отрегулировать минимальное число объектов в кластере. Для задачи поиска плагиата это 2, потому что для заимствования кода нужно как минимум 2 студента.

Для отладки системы было выбрано 8 контестеров, в каждом 9-15 задач и 20-40 студентов, присылавших решения. По задачам были выделены группы с коэффициентами схожести более 0.9, 0.95 и 0.99. Кластеры, выделенные DBSCAN-ом с соответствующими метрикой и максимальным расстоянием между объектами (1 - коэффициент схожести), совпали с выделенными группами с точностью до номеров кластеров. Схожесть разбиений оценивалась с помощью метрики V-мера.

Затем система была протестирована на программных кодах из системы автоматической проверки задач на сайте school.sgu.ru (40288 решений на Python и 42941 решений на C++).

ЗАКЛЮЧЕНИЕ

В данной работе были рассмотрены некоторые системы поиска плагиата: SIM, JPlag, MOSS; разобраны следующие методы поиска плагиата в программном коде: алгоритм, основанный на выравнивании строк; алгоритм Хескела; расстояние Левенштейна; расстояние Дамерау-Левенштейна и методы кластеризации: алгоритм k средних, агломеративная кластеризация и DBSCAN.

Метод выравниваний, расстояние Левенштейна, алгоритм Хескела и расстояние Дамерау-Левенштейна были реализованы на языке Python. Кроме того, для них была реализована предобработка — нормализация (удаление пробельных символов (пробел, вертикальная табуляция, горизонтальная табуляция, перевод строки и возврат каретки), комментариев (однострочных и многострочных) и текстов внутри операторов вывода) и токенизация (замена фрагментов кода на поставленные им в соответствие токены), также на языке Python. Нормализация и токенизация осуществлялись для языков C++ и Python.

На основе этих разработок была создана система проверки студенческих работ на плагиат. Работа этой системы была протестирована на файлах, выгруженных с сайта school.sgu.ru, написанных на языках C++ и Python. Были устранены недостатки, выявленные при тестировании данной системы и проведен анализ работы этих методов.

Для выделения кластеров похожих решений был использован DBSCAN с метриками подсчета схожести строк, минимальным числом объектов 2 и различными значениями минимального расстояния между объектами внутри одного кластера.

Отдельные части бакалаврской работы были опубликованы / представлены на конференциях:

- Огнева Т.А., Лапшева Е.Е. PLAGIARISM DETECTION IN SOURCE CODE ON THE UNIVERSITY SCHOOL.SGU.RU PORTAL// Proceedings of the XX International Multidisciplinary Conference “Innovations and Tendencies of State-of-Art Science“. Mijnbestseller Nederland, Rotterdam, Nederland. 2022. URL: https://www.elibrary.ru/item.asp?id=49194865
- Лапшева Е.Е., Огнева Т.А., ПОИСК ПЛАГИАТА В ПРОГРАММНЫХ РЕШЕНИЯХ В КОНТЕСТЕРЕ ПОРТАЛА SCHOOL.SGU.RU // Перспективы и возможности использования цифровых технологий в науке, образовании и управлении: сборник материалов Всероссийской научно-практической

конференции (г. Астрахань, 21-23 апреля 2022 г.) / под общ. ред. М.В. Коломиной, Е.А. Ивашиненко. – Астрахань: Астраханский государственный университет, 2022. – С. 94-97.– ISBN 978-5-9926-1375-9. URL: https://www.elibrary.ru/item.asp?id=49495454

— Огнева Т.А. ИСПОЛЬЗОВАНИЕ МЕТОДОВ «РАССТОЯНИЕ ЛЕВЕНШТЕЙНА» И ВЫРАВНИВАНИЯ СТРОК ДЛЯ ПОИСКА ПЛАГИАТА В ПРОГРАММНОМ КОДЕ // Научное сообщество студентов: МЕЖДИСЦИПЛИНАРНЫЕ ИССЛЕДОВАНИЯ: сб. ст. по мат. CXXXI междунар. студ. науч.-практ. конф. №24(131) URL: https://elibrary.ru/item.asp?id=47554221&pff=1

Основные источники информации:

1. Cheers, Hayden & Lin, Yuan & Smith, Shamus. (2021). Academic Source Code Plagiarism Detection by Measuring Program Behavioral Similarity. IEEE Access. PP. 1-1. 10.1109/ACCESS.2021.3069367.
2. Andrianov, Igor & Rzheutskaya, Svetlana & Sukonschikov, Alexey & Kochkin, Dmitry & Shvetsov, Anatoly & Sorokin, Arseny. (2020). Duplicate and Plagiarism Search in Program Code Using Suffix Trees Over Compiled Code. Proceedings of the XXth Conference of Open Innovations Association FRUCT. 26. 1-7. 10.23919/FRUCT48808.2020.9087465.
3. Yetthapu, Sudheer, “Source Code Plagiarism Detection Using JPlag & Stack Overflow Data"(2023). Masters Theses & Specialist Projects. Paper 3620.
4. Посов И. А., Допира В. Е., Методы поиска плагиата в кодах программ // Известия СПбГЭТУ «ЛЭТИ» №6/2019 С. 61-66
5. Густокашин М., Тренировки по алгоритмам 3.0 [Электронный ресурс]. — URL: https://yandex.ru/yaintern/algorithm-training (дата обращения 27.02.2023)
6. Рашка С. Python и машинное обучение / пер. с англ. А. В. Логунова. - М.: ДМК Пресс, 2017. - 418 с.
7. Онлайн-учебник по машинному обучению от ШАД. Раздел 7.1. Кластеризация [Электронный ресурс]. — URL: https://academy.yandex.ru/handbook/ml/article/klasterizaciya (дата обращения 09.11.2022)