Компьютерные инструменты в образовании, 2020
№ 4: 79–92
УДК: 004.912:004.42
http://cte.eltech.ru
doi:10.32603/2071-2340-2020-4-79-92

АНАЛИЗ И ВИЗУАЛИЗАЦИЯ ПЛАГИАТА ИСХОДНОГО КОДА В ПРАКТИЧЕСКИХ КУРСАХ ПО ПРОГРАММИРОВАНИЮ

Ефимчик Е. А.[1], кандидат технических наук, ✉ eugene.efimchick@gmail.com
Цибин А. И.[2], инженер-программист, ✉ tsibin.andr@gmail.com

[1]Национальный исследовательский университет ИТМО,
Кронверкский проспект, д. 49, лит. А, 197101, Санкт-Петербург, Россия
[2]ООО «ЭПАМ Систэмз», наб. Черной Речки, д. 41, 197342, Санкт-Петербург, Россия

Аннотация

Проблема недобросовестного заимствования в академической среде по-прежнему является актуальной. Недобросовестные заимствования, или плагиат, встречаются сегодня в различных формах академической активности, начиная от семестровых работ студентов и заканчивая диссертациями ученых. Развитие коммуникаций, глобальный характер взаимодействия привели к широкой доступности материалов, которые легко скопировать. Это приводит к тому, что студентам становится проще найти решение, чем его составить. Отдельной проблемой являются недобросовестные заимствования в работах обучающихся учебных заведений, которые они выполняют в рамках практических курсов по программированию. Как и в случае с текстом, выявлять плагиат вручную является возможным только в самых небольших подвыборках данных. К счастью, на сегодняшний день существует довольно большое количество систем, позволяющих автоматизировано выявлять сходство исходного кода. Более того, существуют средства, позволяющие агрегировать результаты поиска плагиата несколькими различными системами, что также увеличивает вероятность обнаружения случаев недобросовестного заимствования. При этом применение данных средств по-прежнему не так широко распространено в образовательных учреждениях. В настоящей статье приводится описание процесса анализа плагиата, построенного для использования в рамках практических курсов по программированию, а также рассмотрен инструмент интерактивной графовой визуализации результатов анализа плагиата.

Ключевые слова: *дистанционное обучение, электронное обучение, смешанное обучение, плагиат, академическая недобросовестность.*

Цитирование: Ефимчик Е. А., Цибин А. И. Анализ и визуализация плагиата исходного кода в практических курсах по программированию // Компьютерные инструменты в образовании. 2020. № 4. С. 79–92. doi: 10.32603/2071-2340-2020-4-79-92

1. ВВЕДЕНИЕ

Задача выявления плагиата применительно к практическим курсам по программированию становится с каждым годом все более актуальной [15]. Открытые онлайн-курсы

по программированию, зачастую имеют целые базы открытых решений. Чаще всего они публикуются в рамках коммуникации обучающихся, выполнивших задания, однако недобросовестно использовать их может кто угодно. Бороться с публикацией решений не представляется возможным, поскольку решения заданий могут публиковаться на различных платформах. По этой причине бороться необходимо с заимствованиями решений, то есть искать плагиат в работах обучающихся. В настоящий момент контроль добросовестности становится в большей степени необходимостью, чем второстепенной возможностью [1]. Кроме того, как показывают некоторые исследования, существование механизма анализа плагиата в образовательном процессе повышает общее качество обучения навыкам программирования [6, 13].

Современные тенденции возрастания доли самостоятельной работы обучающихся, студенческой мобильности и развития дистанционных технологий обучения особенно остро ставят проблему организации практической работы в гибком, асинхронном режиме. Для решения этой проблемы в дисциплинах, связанных с практикой программирования, в Университете ИТМО была создана образовательная система Flaxo[1] для организации и проведения практических курсов по программированию. Система воплощает идею многофакторного автоматизированного анализа и оценивания работ обучающихся, включающую критерии оценки соответствия спецификации с помощью функционального тестирования, формальной оценки качества кода с помощью инструментов статического анализа и выявления заимствований между решениями. Причем более актуальной задачей является поиск заимствований среди обучающихся одного практического курса, поскольку, во-первых, система ориентирована на обслуживание курсов специализированных заданий, решения которых не так просто найти в открытых источниках, а во-вторых, работа организуется в относительно небольших группах обучающихся (до 100 человек), склонных к внутреннему распространению готовых решений, как например в группах и потоках студентов в вузе.

Система была разработана в рамках бакалаврской выпускной квалификационной работы [2, 3], и с начала учебного года 2018–2019 постепенно внедряется в дисциплинах образовательной программы «Нейротехнологии и программирование».

Далее в статье описаны технические решения, которые легли в основу модуля анализа плагиата системы Flaxo. В частности, описаны построенные процессы хранения и агрегации решений обучающихся, использованный инструмент поиска заимствований, а также разработанный инструмент визуализации найденных заимствований, который представляет из себя выделенный сервис графового представления результатов анализа плагиата.

2. СХЕМА ХРАНЕНИЯ ЗАДАНИЙ И РЕШЕНИЙ

Концепция образовательной платформы Flaxo включала хранение всех заданий и решений обучающихся в репозиториях системы контроля версий. Так, задания представляют собой ветки репозитория преподавателя, а решения — запросы на слияние, создаваемые обучающимися к этим веткам. Файлы, предоставленные преподавателем в его репозиториях, в дальнейшем будем называть *базовыми файлами*, а исходный код, который в них содержится, — *базовым кодом*. В решениях заданий обучающимся необходимо добавлять свой код, создавая новые файлы, но иногда необходимо изменять и базовые файлы, добавляя свой код к базовому или изменяя его. В текущей версии системы

[1] Образовательная платформа Flaxo. https://github.com/tcibinan/flaxo

реализована поддержка системы версионирования Git[2] и платформы GitHub[3], которая на сегодняшний день является одной из самых популярных систем для open-source проектов.

Для того чтобы провести анализ существующих решений, необходимо сначала собрать весь исходный код: базовый код, а также все изменения, сделанные обучающимися. Учитывая, что репозитории с заданиями и решениями помимо исходного кода могут содержать множество других файлов, например, скрипты системы сборки проекта или документацию, необходимо перед выгрузкой заданий и решений фильтровать содержания репозиториев по расширениям файлов. Также загружаемые файлы должны быть сгруппированы по обучающемуся, который их создал или изменил, чтобы дальнейший анализ мог различать автора того или иного фрагмента кода.

3. ИНСТРУМЕНТЫ АНАЛИЗА ЗАИМСТВОВАНИЙ

Одна из основных задач, возникающая при попытке внедрения системы анализа плагиата, — это, безусловно, подбор инструментов выявления заимствований программного кода. В [11] приведен обзор значительного количества существующих инструментов. Они различаются механизмом анализа плагиата, форматами входных и выходных данных, скоростью обработки программного кода, а также количеством поддерживаемых языков программирования. Помимо прочего, существуют унифицированные подходы к анализу плагиата [4, 14], которые агрегируют результаты работы нескольких анализаторов и позволяют минимизировать число ложных срабатываний каждого в отдельности. Кроме того, в [7, 9, 10] описаны и другие средства анализа плагиата.

На основе проанализированной литературы и описаний существующих инструментов при разработке программного модуля анализа плагиата для системы Flaxo был выбран инструмент MOSS [5].

MOSS является бесплатным веб-сервисом поиска плагиата, поддерживающим поиск плагиата на 26 языках программирования. Сервис предоставляет Perl-скрипт в качестве клиентского приложения, который позволяет загружать файлы с исходным кодом и инициализировать анализ плагиата. Наряду с обычными файлами, MOSS использует упомянутое выше понятие базового кода. Он учитывает, что такой код должен быть проигнорирован при выявлении заимствованных фрагментов исходного кода.

Как было сказано выше, MOSS принимает на вход набор файлов для анализа, а на выходе генерирует набор HTML-страниц, которые содержат подробные отчеты по каждой паре решений, содержащих заимствованные коды. Отчет включает в себя количество и процент общих строчек всех файлов, а также таблицы с исходным кодом решений, включающие цветовое обозначение общих фрагментов кода.

Механизм поиска заимствований инструментом MOSS заключается в преобразовании исходного кода решений в наборы токенов и их фильтрации, составлении подгрупп цифровых отпечатков решений на основе выделенных токенов, а также в непосредственном сравнении выбранных отпечатков отдельных решений. Предварительные преобразования исходного кода в токены и их фильтрация позволяют не учитывать при поиске заимствований неспецифичные части решений, как, например, базовый код, комментарии, расстановку пробелов и переносов строк, имена переменных и методов. Кроме того, подход к выбору подгрупп цифровых отпечатков и их сравнению, используемый

[2] Распределенная система версионирования Git. https://git-scm.com/
[3] Платформа разработки GitHub. https://github.com/

инструментом, позволяет находить заимствования с учетом базовых рефакторингов кода, как, например, вынесение переменной или выделение метода.

Несмотря на все преимущества, сервис MOSS обладает рядом недостатков. Во-первых, формат выходных данных HTML не пригоден для эффективной автоматической обработки, поэтому результаты анализа необходимо автоматически разбирать и преобразовывать в другой формат данных. Кроме того, сервис удаляет результаты анализа плагиата спустя 14 суток, в связи с чем результаты его работы требуется полностью выгружать до истечения этого срока.

Важно отметить, что в системе Flaxo архитектурно предусмотрена возможность замены используемого инструмента анализа заимствований. Таким образом, вместо используемого стороннего сервиса для поиска заимствований MOSS могут быть также использованы и другие инструменты.

4. ПРЕДСТАВЛЕНИЕ РЕЗУЛЬТАТОВ АНАЛИЗА ЗАИМСТВОВАНИЙ

4.1. Индикация найденных заимствований

Образовательная система Flaxo формирует сводные таблицы (рис. 1) для отображения результатов автоматизированных проверок решений обучающихся. В качестве первичной индикации найденных заимствований система приводит в соответствие каждому решению пару показателей: количество найденных заимствований и максимальное значение процента общего кода среди всех найденных заимствований. Дополнительно красным цветом выделяются решения, показатель процента общего кода которых превышает некоторый заданный порог, например 80%.

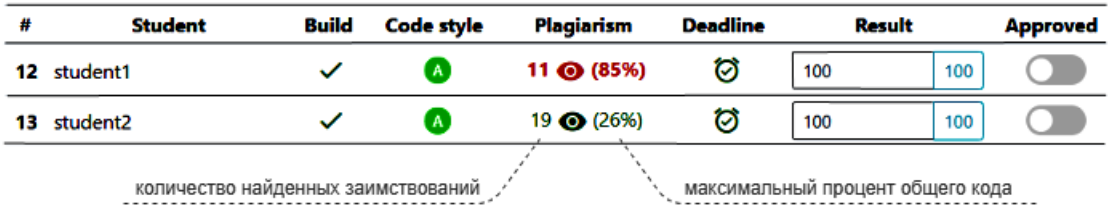

Рис. 1. Первичная индикация найденного плагиата

Описанный способ визуализации позволяет доступным образом указать преподавателю на работы, которые необходимо исследовать в первую очередь, а какие с меньшей вероятностью обладают недобросовестными заимствованиями. Тем не менее, подобная система не позволяет визуализировать связи между работами, особенно если одну работу скопировали несколько раз.

Действительно, механизм анализа плагиата предполагает попарное сравнение всех работ обучающихся, поэтому выявление небинарных зависимостей может оказаться непростой задачей. При этом подобные зависимости могут выявлять, например, группы недобросовестных обучающихся, которые делятся друг с другом своим исходным кодом. При этом, если похожие заимствования имеются более чем у двух людей, то это значительно уменьшает вероятность ложного срабатывания системы анализа плагиата.

4.2. Графовая визуализация

Таким образом, для определения связей между работами обучающихся, в том числе информации о том, кто у кого и сколько заимствовал, необходим инструмент, способный отображать общую картину связей между решениями.

Одним из возможных способов представления данных, позволяющим наблюдать подобные сложные зависимости, является графовая визуализация. На сегодняшний день существует несколько инструментов графовой визуализации результатов анализа плагиата [8, 12]. Как показано в связанных статьях, использование графов для отображения результатов анализа плагиата наравне с гистограммами распределения процента заимствованного кода повышает эффективность применения системы анализа плагиата в целом. Кроме того, использование наложенных графовых представлений нескольких заданий в практических курсах позволяет значительно снизить число ложных срабатываний алгоритма анализа плагиата. При таком подходе становится возможным выявлять группы обучающихся, которые стабильно выделяются на фоне своих коллег и тем самым, вероятнее всего, действительно недобросовестно используют чужой исходный код.

Использование графов для отображения результатов анализа плагиата даже одного задания в отдельности, с точки зрения наглядности, имеет очевидные преимущества перед табличным представлением.

4.2.1. Инструмент графовой визуализации

Описанный в [12] инструмент визуализации плагиата делает большой шаг вперед к простоте интерпретации результатов анализа плагиата. Однако он не позволяет динамически менять параметры, определяющие структуру графа, что ограничивает преподавателя в возможности наглядного рассмотрения зависимостей в различных перспективах и с отличающимся масштабом. По этой причине авторами настоящей работы был создан инструмент[4] интерактивной визуализации графов.

Инструмент рассматривает обучающихся как вершины графа, а случаи заимствования как ребра графа, соединяющие двух различных обучающихся. В качестве веса ребра w используется значение процента соответствующего заимствования, которое лежит в интервале $[0, 100]$. Процент заимствования может составлять разную долю в исходном и скопированном решениях. На практике разница составляет единицы процентов, и для отображения выбирается наибольшее значение.

В связи с тем, что рассчитываемые на основе веса длины ребер не могут быть прямо отображены на плоскость, для визуализации графа заимствований используется силовой алгоритм, который позволяет с определенной точностью отображать на плоскости вершины и ребра с заданной длиной. Это достигается путем формирования сил, действующих на отдельные вершины и стремящихся свести разницу между требуемой и фактической длиной всех ребер к нулю. Далее в статье под термином *длина* подразумевается именно требуемая длина, а не фактическая.

Учитывая, что бóльшие веса соответствуют меньшим расстояниям между двумя решениями в пространстве всех возможных решений некоторой задачи, значения длин ребер l должны быть тем меньше, чем больший вес им соответствует. При таком подходе построенный граф позволит визуально различать кластеры, каждый из которых определяет группу обучающихся, работы которых содержат значительно выделяющееся

[4] Инструмент интерактивной визуализации графов https://github.com/tcibinan/data2graph

количество заимствований друг у друга. В случае отсутствия ярко выделенных кластеров можно говорить о том, что примеров плагиата в работах найти не удалось, или о том, что необходимо изменить параметры отображения графа.

В рамках проведения ряда практических курсов по программированию с использованием образовательной системы Flaxo было отмечено следующее явление: среднее значение процента общего кода всех найденных заимствований может значительно варьироваться от задания к заданию. При этом число подтвержденных случаев заимствований не всегда зависит от выявленного среднего значения процента общего кода.

В таблице 1 приведена статистика найденных заимствований для девяти заданий практического курса по изучению языка программирования Java. В частности, для каждого из заданий указаны число обучающихся, выполнивших задание, суммарное число отправленных ими решений, а также минимальное, максимальное и среднее значение процента общего кода среди всех найденных заимствований. В рамках исследования найденных заимствований в этом практическом курсе были подтверждены несколько случаев плагиата для части заданий. Можно заметить, что в приведенных данных для большинства заданий максимальные проценты общего кода заимствований не превышают 50%, хотя некоторые из обнаруженных заимствований почти полностью копируют оригинальные решения. Такие заниженные показатели найденных заимствований вызваны большой долей базового кода, который не определяется как заимствование, но учитывается в общем количестве кода, относительно которого система MOSS рассчитывает процент заимствований.

Таблица 1. Статистика найденных заимствований курса по Java

№ задания	Кол-во обучающихся	Кол-во решений	Мин, %	Макс, %	Сред, %
1	18	52	1	16	5
2	17	49	1	33	8
3	17	49	1	28	7
4	17	76	1	27	6
5	17	67	5	77	25
6	17	42	1	13	4
7	17	44	1	41	7
8	15	38	3	50	15
9	15	28	2	38	11

Одним из способов увеличения точности находимых заимствований является минимизация базового кода. В таблице 2 приведена статистика найденных заимствований другого практического курса, который направлен на изучение языка программирования C++. В связи со спецификой механизма автоматизированного тестирования решений в этом курсе, количество базового кода в нем сведено к минимуму. Как можно увидеть, величина максимального процента общего кода заимствований в этом курсе значительно выше, чем в курсе по Java, и для большинства заданий выше 80%. Хочется отметить, что абсолютное большинство найденных заимствований с высоким процентом общего кода являются подтвержденными случаями плагиата.

Также для некоторых практических заданий проведение анализа плагиата может и вовсе являться избыточным, например, если код ожидаемого решения представляется достаточно стереотипичным или даже де-факто стандартным, чтобы ожидать больших содержательных расхождений между решениями обучающихся. В каждом частном случае преподаватель должен принимать решение о целесообразности проведения анализа

Таблица 2. Статистика найденных заимствований курса по C++

№ задания	Кол-во обучающихся	Кол-во решений	Мин, %	Макс, %	Сред, %
1	46	162	6	73	16
2	43	140	6	86	15
3	45	208	8	86	17
4	44	191	5	82	14
5	45	185	3	90	15
6	44	124	7	85	16
7	42	137	7	75	16
8	42	88	3	85	15
9	39	81	5	90	19
10	33	54	4	86	21

плагиата и интерпретации его результатов.

С учетом того, что в некоторых случаях минимизация базового кода заданий не представляется возможной, а опираться на абсолютные значения весов ребер не всегда эффективно, необходимо использовать нормализованные относительные значения, сформированные с учетом минимальных и максимальных значений весов.

Для получения более наглядной картины в некоторых других случаях удобно использовать возможность *пороговой нормализации*, которая позволяет объединять в кластеры обучающихся, обладающих заимствованиями с весом, превышающим некоторое заданное пороговое значение r. Кластеризация в данном случае обусловлена механизмом работы силового алгоритма отображения графа на плоскость, которому на вход приходят заимствования с длиной ребер, стремящейся к нулю.

Помимо прочего, существует ряд случаев, ведущих к необходимости как относительного, так и абсолютного масштабирования длин ребер в графе. К примеру, при отключенной нормализации длин ребер задания со смещенным средним значением процента заимствованного кода влево будут представлены графом с очень длинными ребрами, которые могут выходить за границы экрана. В этом случае существование относительного масштабирования позволит сохранить пропорции длин и при этом уменьшить отображаемый граф до необходимого размера. В другом случае, при очень высоких значениях весов части ребер, их длины стремятся к нулю, отчего вершины графа могут накладываться друг на друга и становиться трудноразличимыми. Возможность абсолютного масштабирования позволяет справиться с этой ситуацией путем добавления некоторой фиксированной базовой длины всем ребрам в графе.

Итоговая формула длины ребра в графе представляется в следующем виде:

$$l_i = (1 - \eta(w_i))s + h \text{ для всех i} \in 1, \ldots, n, \tag{1}$$

где l_i — длина i-ого ребра, w_i — вес i-ого ребра, n — число ребер в графе, η — выбранная функция нормализации, s — параметр относительного масштабирования, h — параметр абсолютного масштабирования.

Перечень функций нормализации представлен в таблице 3.

Пример отображения графа заимствований приведен на рис. 2. При необходимости преподаватель может менять значения параметров масштабирования длин ребер графа, пороговое значения веса отображаемых ребер, а также он может выбирать применяемую функцию нормализации. В ответ на каждое действие пользователя граф меняет параметры отображения и плавно переходит в необходимое состояние.

Таблица 3. Функции нормализации

Стандартная	$\eta(w) = w/100$
Адаптивная	$\eta(w) = w/\max\limits_{1\le i\le n} w_i$
Пороговая	$\eta(w) = \begin{cases} 1 & \text{при } w \ge r, \\ 0 & \text{при } w < r \end{cases}$

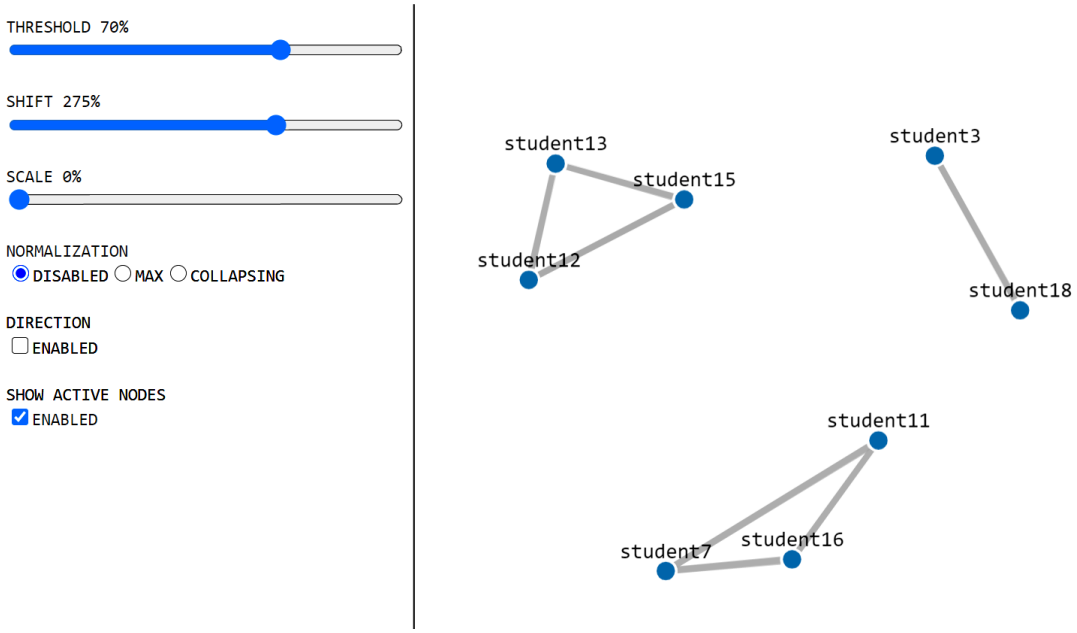

Рис. 2. Граф заимствований

Пример применения пороговой нормализации при пороге отображения, равного 50 %, приведен на рис. 3.

Чтобы изменить вид графа таким образом, понадобилось выполнить несложные действия с интерфейсом на рис. 2 — выбрать опцию COLLAPSING для настройки нормализации и отрегулировать масштаб, чтобы скомпенсировать эффект нормализации. Можно заметить, что эти изменения позволяют визуально выделить кластеры связанных обучающихся, однако также не позволяют различать отдельные заимствования. В связи с этим, использование разных видов нормализации и параметров масштабирования должно сочетаться между собой для наиболее эффективного исследования результатов анализа плагиата.

4.3. Детальные отчеты

Ручная проверка заимствований является неотъемлемым элементом исследования результатов анализа плагиата. В качестве отображения найденных заимствований приводятся попарные листинги связанных решений с цветовым выделением заимствованных участков. На рис. 4 можно наблюдать пример отображения трех найденных заимствований, каждому из которых соответствует определенный цвет. Заметно, что связанные решения отличаются несколькими простыми операциями рефакторинга.

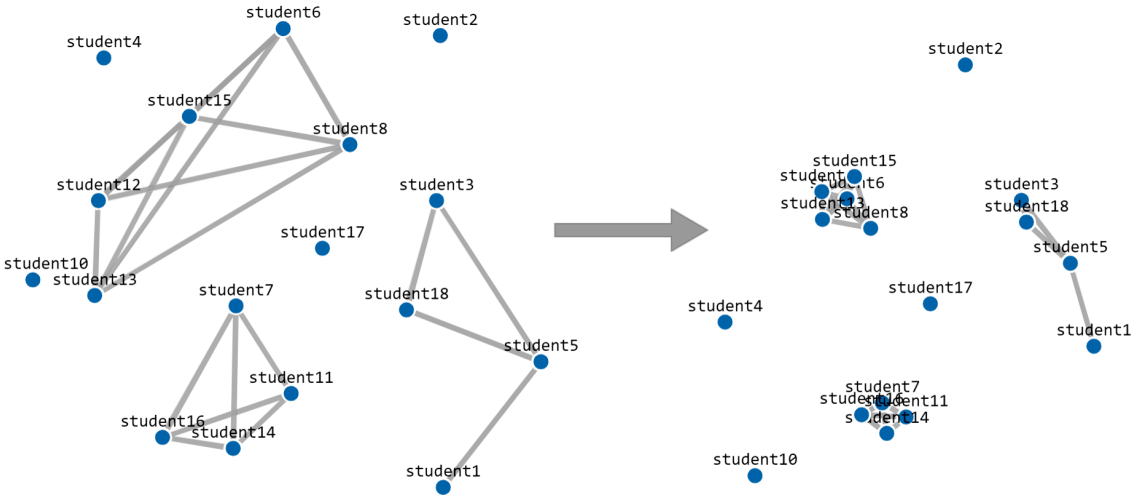

Рис. 3. Пример использования пороговой нормализации

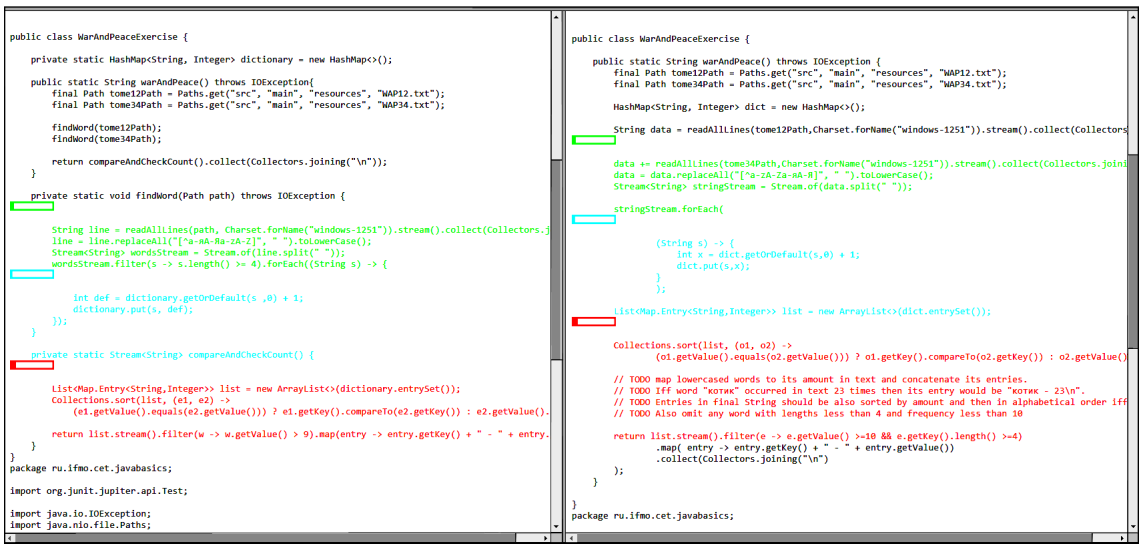

Рис. 4. Детальное отображение заимствований

5. ПРОЦЕСС АНАЛИЗА ПЛАГИАТА

Целостное представление модуля анализа плагиата, описанного в настоящей работе, схематически представлено на рис. 5. Как можно увидеть, построенный процесс анализа плагиата решений обучающихся в рамках практических курсов по программированию включает в себя следующие шаги:

1) агрегацию задания и решений,
2) загрузку задания и решений на сервер MOSS,
3) выгрузку всех результатов, сгенерированных MOSS,
4) составление отчета по результатам анализа плагиата,
5) генерирование графа заимствований.

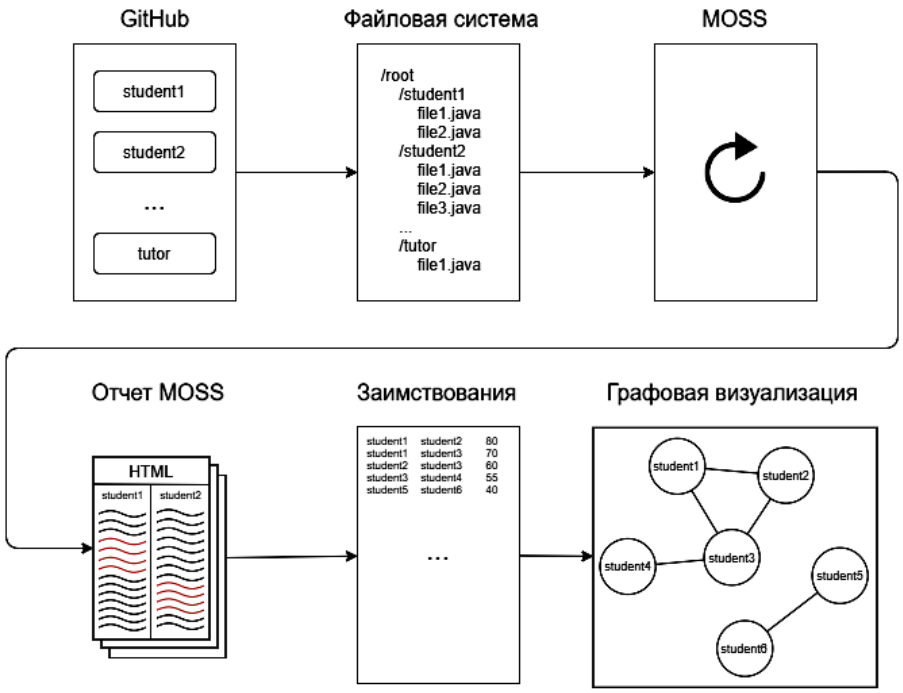

Рис. 5. Стадии анализа плагиата

Результатом работы построенного процесса являются адаптированные результаты анализа плагиата, которые можно исследовать несколькими способами, в том числе используя сводные таблицы результатов, интерактивный инструмент графовой визуализации, а также детальные отчеты для каждого из найденных заимствований.

6. ДАЛЬНЕЙШЕЕ РАЗВИТИЕ

Представленный модуль анализа плагиата применялся на протяжении последних 4 семестров для проведения 10 практических курсов по программированию со средней численностью обучающихся порядка 40 человек, общим числом заданий порядка 50 и числом решений больше 2200. Как показала практика применения, модуль обладает значительным потенциалом с точки зрения наглядности представления результатов анализа плагиата и может быть использован для любых практических курсов по программированию сравнимого размера.

Тем не менее, построенный процесс анализа плагиата обладает рядом ограничений. Во-первых, визуализация на текущий момент учитывает данные об анализе плагиата только одного единственного задания. При этом, как было отмечено ранее в настоящей работе, наложение результатов анализа плагиата нескольких заданий положительно влияет на общее качество визуализации. Во-вторых, на текущий момент визуализация результатов анализа плагиата представляется лишь одним из возможных подходов — графовым отображением. Однако поддержка иных видов визуализации может улучшить наглядность результатов работы модуля анализа плагиата в целом.

Приведенные приемы и средства организации процесса анализа плагиата описаны с точки зрения поиска заимствований среди решений обучающихся одного курса, тем не менее при определенных условиях они могут быть применены и на расширенных

наборах решений, включающих, например, решения предыдущих запусков курса или решения из известных открытых источников.

7. ЗАКЛЮЧЕНИЕ

Построенный процесс анализа плагиата для практических курсов по программированию с использованием разработанного инструмента графовой визуализации позволяет отвечать на новые вызовы в современном образовании, появляющиеся в связи с активным развитием электронного обучения, массового образования и связанных с ними автоматизированных инструментов контроля результатов обучения.

При этом крайне важно понимать, что никакой автоматизированный процесс анализа плагиата не дает результатов, которые можно было бы слепо использовать для выделения недобросовестных обучающихся. Каждый отдельный случай должен индивидуально исследоваться преподавателем. Стоит также отметить, что проблемой могут быть не только заимствования, но и риск представления решения, выполненного не самим обучающимся, а кем-то еще, кто не является слушателем курса. Именно поэтому в системе Flaxo результаты всех автоматизированных проверок используются лишь в качестве рекомендаций, а ключевые решения об оценивании работ обучающихся осуществляются преподавателем, в том числе после очного собеседования.

Описанный ранее процесс значительно упрощает проведение анализа плагиата и интерпретацию его результатов. В итоге, преподаватель имеет возможность интерактивно изучать все автоматически найденные заимствования и эффективно классифицировать решения обучающихся как оригинальные и заимствованные.

Список литературы

1. *Бывальцев В. А., Степанов И. А., Белых Е. Г., Калинин А. А., Бардонова Л. А.* Плагиат и академическая добросовестность в науке // Вестник Российской академии медицинских наук. 2017. № 72 (4). С. 299–304. doi: 10.15690/vramn788
2. *Цибин А. И.* Разработка информационной системы создания, сдачи и проверки знаний по дисциплине «Программирование» https://github.com/tcibinan/flaxo/blob/dev/papers/paper_rus.pdf, 2018. [Электронный ресурс] // GitHub: сайт.
3. *Ефимчик Е. А., Цибин А. И.* Распределенная интегрированная информационная система организации сдачи и проверки заданий по программированию: применение и особенности технической реализации // Сборник тезисов докладов конгресса молодых учёных. [Электронное издание]. СПб.: Университет ИТМО, 2018.
4. *Степочкин Н. А.* Разработка информационной системы анализа схожести исходного кода решений задач по программированию в открытых образовательных репозиториях. https://github.com/gitplag/gitplag/raw/master/thesis/thesis.pdf, 2019. [Электронный ресурс] // GitHub: сайт.
5. *Aiken A.* Moss, a system for detecting software plagiarism. University of Berkeley, 2002. http://www.cs.berkeley.edu/~moss/ (date: 20.12.2020).
6. *Bonifacio R.* Effects of plagiarism intervention program in the research papers of central mindanao university students. International Journal of Scientific & Technology Research. 2020. Vol. 9, № 2. P. 5412–5418.
7. *Franclinton R., Karnalim O.* A language-independent library for observing source code plagiarism // Journal of Information Systems Engineering and Business Intelligence, 2019. Vol. 5, № 10. P. 110–119.
8. *Gniazdowski Z.* Detection of a source code plagiarism in a student programming competition // Zeszyty Naukowe. 2019. Vol. 13, № 12. P. 74–94.

9. Karnalim O., Kurniawati G. Programming style on source code plagiarism and collusion detection // International Journal of Computing. 2020. Vol. 19, № 1. P. 27–38.
10. *Lesner B., Brixtel R., Bazin C., Bagan G.* A novel framework to detect source code plagiarism: now, students have to work for real // Proc. of the 2010 ACM Symposium on Applied Computing, Sierre. Switzerland. 2010. P. 57–58. doi: 10.1145/1774088.1774101
11. *Martins V., Fonte D., Henriques P. R., Cruz D. Da.* Plagiarism detection: A tool survey and comparison // OpenAccess Series in Informatics, 38: 143–158, 01 2014.
12. *Obaido G., Ranchod P., Klein R.* Constructing and analysing plagiarism in student programs using graphs. 2017. URL: https://www.researchgate.net/publication/319629298_Constructing_and_Analysing_Plagiarism_in_Student_Programs_Using_Graphs (date: 20.12.2020).
13. *Pawelczak D.* Effects of plagiarism in introductory programming courses on the learning outcomes // Proc. 5th International Conference on Higher Education Advances (HEAd'19). Universitat Politecnica de Valencia. Valencia. Spain, 2019. P. 623–631. doi: 10.4995/HEAd19.2019.9297
14. *Portillo O., Ayala-Rivera V., Murphy E., Murphy J.* A unified approach to automate the usage of plagiarism detection tools in programming courses // Proc. 12th International Conference on Computer Science and Education (ICCSE). Houston, TX. USA. 2017. P. 18–23. doi: 10.1109/ICCSE.2017.8085456
15. *Wang C., Liu Z., Liu D.* Preventing and detecting plagiarism in programming course // International Journal of Security and Its Applications. 2013. Vol. 7, № 9. P. 269–278. doi: 10.14257/ijsia.2013.7.5.25

Поступила в редакцию 27.05.2020, окончательный вариант — 21.11.2020.

Ефимчик Евгений Александрович, кандидат технических наук, доцент факультета программной инженерии и компьютерной техники Национального исследовательского университета ИТМО, ✉ eugene.efimchick@gmail.com

Цибин Андрей Игоревич, инженер-программист ООО «ЭПАМ Систэмз», ✉ tsibin.andr@gmail.com

Computer tools in education, 2020
№ 4: 79–92
http://cte.eltech.ru
doi:10.32603/2071-2340-2020-4-79-92

Source Code Plagiarism Analysis and Visualization for Programming Courses

Efimchik E. A.[1], PhD, ✉ eugene.efimchick@gmail.com
Tsibin A. I.[2], Software Engineer, ✉ tsibin.andr@gmail.com

[1]ITMO University, 49 Kronverksky, bldg. A, 197101, Saint Petersburg, Russia
[2]EPAM Systems, Ltd, 41, Chyornoy rechki embankment, 197342, Saint Petersburg, Russia

Abstract

The problem of unfair borrowing in the academic environment is still relevant. Unfair borrowing, or plagiarism, is found today in various forms of academic activity, ranging from semester papers of students to dissertations of scientists. The development of

communications and the global nature of interaction have led to a wide availability of materials that are easy to copy. This leads to the fact that it becomes easier for students to find a solution than to compose it. A separate problem is the unfair borrowing in the works of students of educational institutions, which they perform in the framework of practical programming courses. As in the case of text, it is possible to detect plagiarism manually only in the smallest subsamples of data. Fortunately, today there are quite a large number of systems that allow you to automatically identify the similarity of the source code. Moreover, there are tools that allow you to aggregate the results of the search for plagiarism by several different systems, which also increases the likelihood of detecting cases of unfair borrowing. At the same time, the use of these tools is still not so widespread in educational institutions.

This article describes the process of plagiarism analysis built for use in practical programming courses, as well as a tool for interactive graph visualization of the results of plagiarism analysis.

Keywords: *distance learning, e-learning, blended learning, plagiarism, academic dishonesty.*

Citation: E. A. Efimchik and A. I. Tsibin, “Source Code Plagiarism Analysis and Visualization for Programming Courses,” *Computer tools in education*, no. 4, pp. 79–92, 2020 (in Russian); doi: 10.32603/2071-2340-2020-4-79-92

References

1. V. A. Byvaltsev, I. A. Stepanov, E. G. Belykh, A. A. Kalinin, and L. A. Bardonova, “Plagiarism and academic integrity in science,” *Annals of the Russian academy of medical sciences*, vol. 72, no. 4, pp. 299–304, 2017 (in Russian); doi:10.15690/vramn788
2. A. I. Tsibin, *Razrabotka informatsionnoi sistemy sozdaniya, sdachi i proverki zadanii po po distsipline ”Programmirovanie”*, St. Petersburg, Russia, ITMO, 2019 (in Russian). [Online]. Available: https://github.com/tcibinan/flaxo/blob/dev/papers/paper_rus.pdf
3. E. A. Efimchik and A. I. Tsibin, “Raspredelennaya integrirovannaya informatsionnaya sistema organizatsii sdachi i proverki zadanii po programmirovaniyu: primenenie i osobennosti tekhnicheskoi realizatsii,” in *Proc. of VII Congress of young scientists, ITMO, 2018*, St. Peterburg, Russia, 2018 (in Russian). [Online]. Available: https://openbooks.itmo.ru/ru/file/7344/7344.pdf
4. N. A. Stepochkin, *Razrabotka informatsionnoi sistemy analiza skhozhesti iskhodnogo koda reshenii zadach po programmirovaniyu v otkrytykh obrazovatel’nykh repozitoriyakh*, St. Petersburg, Russia, ITMO, 2019 (in Russian). [Online]. Available: https://github.com/gitplag/gitplag/raw/master/thesis/thesis.pdf
5. A. Aiken, “Moss (for a Measure Of Software Similarity), a system for detecting software plagiarism,” [Soft], University of Berkeley, 2002. [Online]. Available: http://www.cs.berkeley.edu/~moss/
6. R. Bonifacio, “Effects of plagiarism intervention program in the research papers of central mindanao university students,” *International Journal of Scientific & Technology Research*, vol. 9, no. 2, pp. 5412–5418, 2020.
7. R. Franclinton and O. Karnalim, “A language-independent library for observing source code plagiarism,” *Journal of Information Systems Engineering and Business Intelligence*, vol. 5, no. 10, pp. 110–119, 2019; doi: 10.20473/jisebi.5.2.110-119
8. Z. Gniazdowski, “Detection of a source code plagiarism in a student programming competition,” *Zeszyty Naukowe*, vol. 13, no. 21, pp. 74–94, 2019; doi: 10.26348/znwwsi.21.74
9. O. Karnalim and G. Kurniawati, “Programming style on source code plagiarism and collusion detection,” *International Journal of Computing*, vol. 19, no. 1, pp. 27–38, 2020.
10. B. Lesner, R. Brixtel, C. Bazin, and G. Bagan, “A novel framework to detect source code plagiarism: now, students have to work for real!” in *Proc. of the 2010 ACM Symposium on Applied Computing*, Sierre, Switzerland, 2010, pp. 57–58; doi: 10.1145/1774088.1774101

11. V. Martins, D. Fonte, P. R. Henriques, and D. Da Cruz, “Plagiarism detection: A tool survey and comparison,” in *3rd SLATE. OASIcs*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, vol. 38, 2014, pp. 143–158. [Online]; doi: 10.4230/OASIcs.SLATE.2014.143
12. G. Obaido, P. Ranchod, and R. Klein, *Constructing and analysing plagiarism in student programs using graphs*, 2017. [Online]. Available: https://www.researchgate.net/publication/319629298_Constructing_and_Analysing_Plagiarism_in_Student_Programs_Using_Graphs
13. D. Pawelczak, “Effects of plagiarism in introductory programming courses on the learning outcomes,” in *Proc. 5th International Conference on Higher Education Advances (HEAd’19)*, Universitat Politecnica de Valencia, Valencia, Spain, 2019, pp. 623-631; doi: 10.4995/HEAd19.2019.9297
14. A. O. Portillo-Dominguez, V. Ayala-Rivera, E. Murphy, and J. Murphy, “A unified approach to automate the usage of plagiarism detection tools in programming courses,” in *Proc. 12th International Conference on Computer Science and Education (ICCSE)*, Houston, TX, USA, 2017, pp. 18–23; doi: 10.1109/ICCSE.2017.8085456.
15. C. Wang, Z. Liu, and D. Liu, “Preventing and detecting plagiarism in programming course,” *International Journal of Security and Its Applications*, vol. 7, no. 5, pp. 269–278, 2013; doi: 10.14257/ijsia.2013.7.5.25

Received 27.05.2020, the final version — 21.11.2020.

Eugene Efimchik, PhD, Associate Professor of the Program Engineering and Computer Technics Faculty, ITMO University, ✉ eugene.efimchick@gmail.com

Andrey Tsibin, Software Engineer, EPAM Systems, Ltd, ✉ tsibin.andr@gmail.com