УДК 004.02
DOI https://doi.org/10.32838/2663-5941/2021.6/14

Зайченко І.В.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

АНАЛІЗ МЕТОДІВ ТА ДІЄВИХ СИСТЕМ ВИЯВЛЕННЯ ПЛАГІАТУ НАВЧАЛЬНИХ КОДОВИХ ПРОГРАМ

У статті розглядаються основні методи та програмні засоби для виявлення плагіату навчальних програм. Проведено дослідження та аналіз останніх публікацій на тему методів виявлення плагіату програмних проєктів. Виконано аналіз базових методів та методів програмних систем, що присутні на ринку та виявляють плагіат навчальних кодових програм. Було структуровано та досліджено такі методи: метод текстового аналізу; метод, орієнтований на атрибути коду; структурно-орієнтований метод; метод на основі абстрактного синтаксичного дерева; метод на основі токенів; метод на основі метрик програми; метод на основі орієнтованого графа; метод на основі аналізу стилю програмування; метод на основі ідентифікаційних міток. Проаналізовано методи виявлення плагіату перелічених нижче сервісних програм: SIM, MOSS, JPlag. Детально розібрано алгоритми виявлення плагіату в навчальних кодових програм найпопулярніших сервісів, зокрема MOSS, Jplag та SIM. Було проведене порівняння вказаних сервісів між собою. Серед порівнюваних елементів сервісів було розглянуто кількість підтримуваних мов програмування, основний алгоритм, виключення шаблонного коду, історичні порівняння, підтримка багатофайлового проєкту, відкритість коду та надання зручного користувацького інтерфейсу завантаження студентських робіт для виявлення плагіату. Кожна функція та особливість була оцінена з урахуванням важливості, повноти реалізації та зручності. Проведене дослідження та аналіз діючих програмних систем дає змогу систематизувати методи виявлення плагіату в навчальних кодових програмах. Внаслідок наведеного порівняльного аналізу наявних програмних систем для виявлення плагіату в навчальних кодових програмах було обґрунтовано доцільність розробки нового програмного рішення, орієнтованого на успішне поєднання та покращення методів і функціоналу систем виявлення плагіату

Ключові слова: *плагіат, методи виявлення плагіату, плагіат кодових програм, системи антиплагіату, порівняння антиплагіатів.*

Постановка проблеми. Незалежно від предметної області, плагіат є реальною проблемою для викладачів. Легкість, з якою студенти можуть копіювати комп'ютерний код, відкриває можливості для плагіату в проєктах програмування. Наразі викладачам надається мало інструментів, які допомагають їм у виявленні можливого плагіату в завданні. Тому часто рішенням є втомлива і схильна до помилок ручна перевірка робіт у пошуках схожості між будь-якими двома роботами.

Використання студентами плагіату зумовлено багатьма причинами, наприклад, іноді виникають деякі недоліки під час перевірки завдань викладачем, а іноді студент просто лінується писати власний код. Також гостро стоїть питання «академічної доброчесності» у багатьох вузах. Це поняття полягає в тому, що студенти мають самі виконувати всі свої завдання, а у випадку виявлення ознак свідомого плагіату вони можуть понести покарання (у деяких випадках – навіть відрахування з вищого навчального закладу).

Програмне забезпечення, яке могло б ідентифікувати проєкти, що містять високу ступінь подібності, було б корисним для викладачів, даючи їм змогу зосередити свій час, витрачений на пошук плагіату, на виконання інших запланованих завдань. Однак досить важко найти систему, яка має весь необхідний функціонал і водночас використовує сучасний метод виявлення плагіату в навчальних кодових програмах.

Мета роботи – розширення функціоналу системи виявлення плагіату в навчальних кодових програмах шляхом аналізу й систематизації методів та функціональних можливостей систем-аналогів.

Аналіз останніх досліджень і публікацій. Плагіат у кодуванні – не зовсім нове явище. Це питання раніше обговорювалося і вивчалася науковцями, щоб визначити серйозність проблеми й те, які фактори сприяють актам плагіату. Було проведено багато досліджень на тему методів виявлення плагіату. У статті [1] було винайдено власний алгоритм на основі фільтрації даних. Робота [2] надає інформацію про виявлення

плагіату вихідного коду програмного забезпечення за допомогою математичних метрик складності.

Постановка завдання. Головним завданням є огляд та систематизація методів та функціональних можливостей систем виявлення плагіату в навчальних кодових програмах. Мета аналізу – виявлення необхідного функціоналу для якісної реалізації власної системи антиплагіату кодових програм.

Виклад основного матеріалу дослідження. Метод текстового аналізу. Під час визначення того, був текст плагіатом чи ні, може бути проведений аналіз тексту. Статистика використовується для визначення того, чи дійсно текст був плагіатом, на основі підрахунку частот слів і пропозицій. Наприклад, текст порівнюється з іншими текстами в базі даних для пошуку подібності у словах і складі пропозицій. Більш висока частота слів і кількість посилань будуть вказувати на вищу ймовірність плагіату.

Системні підходи, засновані на тексті, не підходять для виявлення плагіату коду, оскільки вони ігнорують синтаксис кодування і, крім того, зміни в скопійованому коді зможуть уникнути виявлення плагіату.

Метод, орієнтований на атрибути коду. Система, орієнтована на атрибути, націлена на ключові властивості коду й оцінює ці ключові властивості. Система, яка пояснюється Дональдсоном, використовує чотири атрибути і вимірює їх для виявлення плагіату:

– кількість унікальних операторів;
– число унікальних операндів;
– кількість входжень операторів;
– кількість входжень операндів.

Дві кодові програми апроксимуються шляхом вимірювання різниці між зазначеними вище атрибутами. Відзначається, що атрибутивно-орієнтовані системи дуже вузькі. Системи, орієнтовані на атрибути, корисні тільки для тих програм, які мають мінімум змін у коді. Найбільш часто використовуваний спосіб обходження цього методу – додавання або видалення непотрібного коду. Оскільки ця система перевіряє код рядок за рядком, існує можливість великої різниці між числом операндів і операторів по всьому коду. Також важко виявити плагіат у програмі з дуже великим кодом.

Структурно-орієнтований метод. Він заснований на комбінації двох методів: пошук подібності в структурі подання з двох частин вихідного коду, а також застосування атрибута методів підрахунку. У цих системах такі елементи, як коментарі, пробіли й імена змінних, ігноруються, тому що вони можуть бути легко змінені.

Існує безліч структурно-орієнтованих підходів, доступних для виявлення плагіату в вихідному коді. Кожен підхід фокусується на певних характеристиках коду. Наприклад, деякі підходи призначені тільки для перевірки на плагіат вихідного коду, написаного на різних мовах програмування. Деякі підходи доступні для перевірки плагіату дуже складних модифікацій коду, але їм потрібно багато часу, щоб виявити схожість. У структурно-орієнтованих системах схожість між двома програмами вимірюється на основі подібності структури двох вихідних кодів. За такого підходу вихідний код порівнюється в два етапи. На першому етапі генерується потік токенів програм, а другий – це етап порівняння потоків токенів з допомогою алгоритмів зіставлення рядків.

Метод на основі абстрактного синтаксичного дерева (далі – АСД). Пошук запозичень у коді на основі побудови абстрактного синтаксичного дерева дає змогу представити код у формі, не враховуючи інформацію, яка не впливає на оригінальність. Представлення коду враховує тільки логіку програми.

АСД – це структурне представлення коду, очищене від елементів синтаксису конкретної мови програмування, де вузлами є оператори, до яких приєднуються їхні аргументи, а вони, своєю чергою, теж можуть бути складовими вузлами. Перевагою використання цього уявлення є велика кількість парсерів для різних мов програмування. До недоліків належать складність реалізації та тривалий час роботи алгоритмів побудови й аналізу АСД.

Метод на основі токенів. Часто вузли АСД виходять із лексем, які виділяються на етапі лексичного аналізу. Тобто код перетворюється на послідовність лексичних одиниць із певним значенням, так званим токеном.

Токенізація здійснюється за таким алгоритмом:

1. Кожному оператору мови програмування або групі операторів, що мають подібне призначення, присвоюється унікальний ідентифікатор. Значення ідентифікаторів призначаються заздалегідь і для всіх класів операторів.

2. За отриманими ідентифікаторами будується рядок. У цьому рядку порядок токенів відповідає порядку прямування їх у вихідному коді.

Такий підхід дає змогу ігнорувати частини коду, які легко піддаються змінам, а також виявляти запозичені фрагменти скопійованого коду, розташованих у різних місцях програми.

Процес токенізації залежить від конкретної мови програмування, тобто підтримка декількох мов можлива тільки за умови використання декількох лексичних аналізаторів.

Метод на основі метрик програми. Подання ґрунтується на оцінці різних метрик програми, в якості метрики можуть використовуватися, наприклад, кількість викликів функцій, використовуваних змінних, циклів. Далі дві програми порівнюються з відповідними значеннями метрик, якщо вони близькі або збігаються, то виявляється плагіат.

Названий метод не прив'язаний до мови програмування, має високу продуктивність і хорошу масштабованість, але в невеликих програмах, якими найчастіше і є навчальні завдання, метод видає велику кількість неправильних даних для всіх типів запозичення.

Метод на основі орієнтованого графа. Запозичення виявляються на основі графа залежностей програми (далі – ГЗП) – орієнтованого графа.

Вершини ГЗП представляють операції програми, а ребра представляють залежності між ними. Ребра поділяють на залежності за даними і за управлінням. Після складання ГЗП аналізованих програм для них здійснюється пошук схожих підграфів. Підграфи визнаються подібними, якщо вони пов'язані і мають збіги багатьох ребер. Цей метод вирізняється тим, що граф зберігає інформацію і про семантику, і про структуру, сприяючи високій точності виявлення запозичень програмного коду. Недоліком подання є погана масштабованість графа і тривалий час його побудови та аналізу.

Метод на основі аналізу стилю програмування. В аналізі стилю програмування часто згадується родимі плями. Родимі плями – це певні властивості коду програми, які використовуються з самого початку написання програми, а потім тісно пов'язані в подальшій розробці програмного забезпечення. Вказана сутність вважається об'єктивною, тому вона ідеально підходить для підтвердження авторства та факту плагіату.

Кожному програмісту, який розробляє програмне забезпечення, пише код, властивий індивідуальний підхід та стиль написання. Ось чому в програмному коді часто можна простежити велику кількість родимих плям, які будуть характерні для конкретного розробника. Якщо розглядати на прикладі, то типовим кейсом може виступати ситуація, коли будуть використані шаблони для рішення типової проблематики, такі шаблони з'являються в розробників за роки написання програмного забезпечення.

Помилки також є складовою родимих плям автора вихідного тексту. Повторення унікальної помилки також може предстати доказом плагіату, адже ймовірність отримання однакових специфічних помилок дуже мала.

Метод на основі ідентифікаційних міток. Ідентифікаційна мітка – це певна інформація, яка належить автору та неочевидна для випадкового розпізнання. Вона допомагає підтвердити авторство своєю присутністю. Зазвичай ідентифікаційні мітки кодуються способом, відомим лише самому автору.

Несподіване розташування та невизначеність – головна складова ефективності та доцільності використання ідентифікаційних міток. Присутність таких міток повинна бути проінформована. Це забезпечує додатковий захист проти зловмисника, який завершить спробу запозичення тексту з названими елементами. Збереження вихідного тексту визначає справжнього автора, незважаючи на спроби зловмисника додати додаткові ідентифікаційні мітки.

Основну увагу потрібно приділити принципу побудови ідентифікаційних міток. Адже за правильної розробки вони будуть добре захищені від різноманітних модифікацій та видалення. Якщо зловмисник буде знати принцип побудови, то зникне весь сенс використання міток.

Слід зауважити, що важливим аспектом проблеми, виявлення та видалення особливих тегів автора є знання зловмисником будь-якої конфіденційної інформації. Щоб уникнути цієї проблеми, рекомендується шифрувати інформацію з мітками автора. Однак не варто зациклюватись на цьому, оскільки складне шифрування не забезпечує коректне розшифрування даних, тому що чим більша складність алгоритму шифрування, тим більша вірогідність невірного розшифрування даних.

Метод, який використовує сервіс SIM [3]. SIM використовується для виявлення плагіату коду, написаного на мові програмування Java, C, Pascal, Lisp та тексту на природній мові.

Щоб виявити плагіат, SIM проводить токенізацію вихідного коду, а потім створює таблицю символів (для збереження динамічно призначених ідентифікаторів токенів), після цього порівнює токенізовані кодові рядки програм, використовуючи алгоритм локального вирівнювання рядків.

SIM використовується для перевірки подібності між звичайними текстовими файлами. Система перетворює вихідний код рядка, а потім порівнює їх за допомогою методу вирівнювання рядків динамічного програмування. Названий метод використовується для зіставлення рядків ДНК. Вирівнювання є дорогим та вичерпним з точки зору обчислення для всіх систем, адже для великих текстів коду SIM не масштабується.

Оптимальне вирівнювання двох рядків – це максимальне значення цільової функції всіх комбінацій конкретного вирівнювання.

Алгоритм у системі SIM є таким:

1. Отримуються токенізовані рядки A1 і A2 програм.

2. A2 ділиться на певні секції, що являє собою модуль програми.

3. Для кожної секції A1 обчислюються значення оптимального вирівнювання.

4. Результати між собою комбінуються.

Даний підхід дає змогу SIM коректно обробляти перестановки модулів програми.

Вихідний код SIM публічно доступний, але він більш активно не підтримується.

Метод, який використовує сервіс MOSS [4]. MOSS доступний для використання в академічних колах і в якості онлайн-сервісу. Сервіс підтримує програми мов Ada, C, C++, Java, звичайний текст, Паскаль та інші. MOSS підтримує операційні системи UNIX та Windows.

Згаданий вебсервіс використовує алгоритм відбитків документа – алгоритм просіювання. У цьому алгоритмі відбитків токенізована програма надається у вигляді набору міток, щоб набори для подібних програм перетиналися.

Алгоритм методу відбитків:

1. Один за одним хешуються підрядки токенізованого проєкту A1 довжини n.

2. Виділяється підмножина хеш-значень, що характеризує A1. Ті самі кроки виконуються і для інших програм, їхні хеш-значення зберігаються у таблицю.

3. За допомогою таблиці отримується набір ділянок коду P, підозрілих на плагіат.

4. Відбувається аналіз отриманих даних.

Важлива частина алгоритму – це вибір n та 2 крок. n обмежує довжину підрядка, з якої працює алгоритм. Мале його значення сприяє ігноруванню шумів, що не дає алгоритму пропустити випадок плагіату.

Метод, який використовує сервіс Jplag [5]. JPlag доступний як безкоштовний сервіс. Сервіс використовується для перевірки на плагіат коду, написаного на Java, C, C++, Scheme та інші. Надається проєкт готових файлів програм у якості вхідних даних у JPlag.

Jplag, оскільки є десктопним додатком та має відкритий код, надає можливість використання його в розробці проєкту для взаємодії інших систем для виявлення плагіату. Система перетворює код у рядки токенів, що представляють структуру коду. Далі йде порівняння рядків токенів алгоритмом жадібного рядкового замощення.

Цей алгоритм використовує дані евристики:

1. Довгі послідовні збіги для порівняння рядків-токенів – це набагато краще, чим набір менших і непослідовних.

2. Алгоритм ігнорує збіги, менші за вказане значення.

Друга евристика фільтрує шуми – невеликі частини коду, що випадково збігаються в кодах.

Кінцевим результатом жадібного алгоритму буде набір спільних підрядків, які не перетинаються. Підрядок, який входить у набір, є тайлом. Чим більша нормалізована сума довжин тайлів, тим більша схожість цих підрядків.

Рабіна-Карпа оптимізував алгоритм жадібного рядкового замощення на порівнянні підрядків за хеш-значеннями:

1. Хеш обчислюються для підрядків довжини min в A1 і A2.

2. Кожен хеш рядка A1 порівнюється з кожним відповідним A2. Якщо значення однакові, значить, знайдено збіг підрядків для цього токена.

3. Хеш-таблиця розміщує підрядки A2, які мають однакові значення хешу з A1, що зменшує обчислення аж до лінійної кількості кроків алгоритму.

Порівняння систем виявлення програмного плагіату. Порівняння буде відбуватися за такими критеріями (див. табл. 1):

Таблиця 1

Порівняння за критеріями систем виявлення плагіату

	SIM	MOSS	JPlag
Кількість підтримуваних мов програмування	5	23	7
Основний алгоритм	Токенізація, вирівнювання рядків	Токенізація, вирівнювання рядків, алгоритм відбитків	Токенізація, оптимізоване жадібне рядкове заміщення
Виключення шаблонного коду	–	–	+
Історичні порівняння	–	+	–
Підтримка багатофайлового проєкту	–	+	+
Відкритість коду	+	–	+
Зручний користувацький інтерфейс	–	–	–

1. Кількість підтримуваних мов програмування.

2. Основний алгоритм методу роботи системи.

3. Виключення шаблонного коду – наявність ігнорування системою певної частини коду, спільної для всіх.

4. Історичні порівняння – наявність порівняння нового програмного коду з попередніми.

5. Підтримка багатофайлового проєкту – оцінка системою на плагіат у межах одного чи багатьох файлів проєкту.

6. Відкритість коду – вихідний код системи доступний для користувачів.

7. Зручний користувацький інтерфейс для завантаження робіт, які потім будуть проходити виявлення плагіату.

Системи JPlag та MOSS дають користувачу подібну функціональність, Алгоритми, що лежать у їхній основі, надають подібні результати виявлення плагіату.

Лише MOSS із методом просіювання надає змогу організувати базу коду, в якій можна шукати збіги в іншому програмному коді за деякий вказаний час, а не перевіряти знову на плагіат із кожним елементом бази. Це завдяки тому, що MOSS використовує хеш-таблицю.

JPlag є десктопним сервісом та його код відкритий для користувача. Це дає можливість змінювати його для власних потреб та використовувати у своєму кодові й не бути залежним від інших вебсервісів.

SIM використовує алгоритм, який в основі має вирівнювання рядків, що на невеликих частинах коду може надавати хибні збіги.

Використовуючи дослідження методів та порівняльний аналіз популярних сервісів для виявлення програмного плагіату, можна зробити висновок про функціональні можливості методів і сервісів, а також і про їхні переваги й недоліки.

Висновки. Було розглянуто та систематизовано методи виявлення плагіату вихідного коду. Проведений аналіз популярних сервісів, призначених для виявлення плагіату в навчальних кодових програмах, систематизовано функціонал цих сервісів та виявлено їхні переваги та недоліки. Серед наявних сервісів наразі досить складно знайти комплексне рішення, що якісно поєднує в собі всі наведені вимоги: підтримання великої кількості підтримуваних мов програмування; ефективність основного алгоритму; виключення шаблонного коду; можливість історичних порівнянь; підтримка багатофайлового проєкту; відкритість коду та надання зручного користувацького інтерфейсу для завантаження студентських робіт. Результати аналізу статті обґрунтовують необхідність розробки нового програмного забезпечення, орієнтованого на поєднання та розширення функціоналу систем для виявлення плагіату в кодових програмах. Подальші дослідження будуть спрямовані на реалізацію такого програмного забезпечення.

Список літератури:

1. Кірічек А.А., Амонс А.А., Кірічек Г.Г. Алгоритм фільтрації для системи визначення плагіату у програмному коді. *Вісник Національного технічного унівперситету «Харківський політехнічний інститут»*. 2013. Вип. 16(989). Ч. 1. С. 76–82.

2. Виявлення плагіату вихідного коду програмного забезпечення за допомогою математичних метрик складності. 2017. URL: http://ceur-ws.org/Vol-1965/paper7.pdf

3. SIM : вебсайт. URL: https://www.simge.edu.sg

4. MOSS : вебсайт. URL: https://yangdanny97.github.io/blog/2019/05/03/MOSS

5. JPlag : вебсайт. URL: https://github.com/jplag/jplag

Zaichenko I.V. ANALYSIS OF METHODS FOR DETECTING PLAGIARISM OF EDUCATIONAL CODE PROGRAMS

This article discusses the main methods and software tools for detecting curriculum plagiarism. Research and analysis of recent publications on the methods of detecting plagiarism of software projects. The analysis of basic methods and methods of software systems that are present on the market and detect plagiarism of educational code programs is performed. The following methods were structured and investigated: text analysis method; the method focuses on code attributes; structurally-oriented method; method based on the abstract syntactic tree; token-based method; method based on program metrics; method based on the oriented graph; method based on analysis of programming style; method based on identification labels. Methods of detecting plagiarism of data of service programs are analyzed: SIM, MOSS, JPlag. Algorithms for detecting plagiarism in educational code programs of the most popular services, including MOSS, Jplag, and SIM, are analyzed in detail. A comparison of these services with each other. Among the compared elements of the services were considered the number of supported programming languages, the main algorithm, template code exclusion, historical comparisons,

multifile project support, code openness, and providing a user-friendly interface for downloading student work to detect plagiarism. Each function and feature were evaluated taking into account the importance, completeness of implementation, and convenience. The conducted research and analysis of existing software systems allows systematizing the methods of plagiarism detection in educational code programs. As a result of the given comparative analysis of the existing software systems for plagiarism detection in educational code programs the expediency of development of the new software decision focused on successful combination and improvement of methods and functionality of plagiarism detection systems was substantiated.

Key words: *plagiarism, methods of plagiarism detection, plagiarism of code programs, antiplagiarism systems, comparison of antiplagiarism.*