УДК 004.03:001.9

ДО ПИТАННЯ ВИЯВЛЕННЯ ПЛАГІАТУ КОДОВИХ ПРОГРАМ

Азаренков В.І., *доцент, кафедра САІТ НТУ«ХПІ»*
Заболотний О.С., *студент, САІТ НТУ «ХПІ»*

Анотація. *В роботі розглянуто основні методи та програмні засоби для виявлення навчальних плагіатів програм. Проведено дослідження та аналіз останніх публікацій на тему методів виявлення плагіату програмних проектів.*

Ключові слова*: ПЛАГІАТ; ЗАСОБИ АВТОМАТИЧНОГО ПОШУКУ ПЛАГІАТУ; ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.*

Використання деякими здобувачами освіти плагіату кодових програм зумовлено багатьма причинами. Однією з них є те, що іноді студент просто лінується писати власний код. Однак розпізнати плагіат проблематично, тому що досить важко знайти систему, яка має весь необхідний функціонал і водночас сучасний метод його виявлення в навчальних кодових програмах [1]. Системні підходи, засновані на тексті, не актуальні для цього, оскільки вони ігнорують синтаксис кодування і, крім того, зміни у скопійованому коді зможуть уникнути виявлення плагіату. Завданням системи оцінки ідентичності програмного коду є автоматичне визначення того, чи була використана у програмі чужа ідея. На практиці певним чином задаються функція близькості та поріг, якими можна визначити наскільки ймовірно, що певна частина коду була запозичена [2].

Система, орієнтована на атрибути коду, націлена на ключові його властивості і оцінює їх. Вона використовує чотири атрибути і вимірює їх для виявлення плагіату: кількість унікальних операторів; число унікальних операндів; кількість входжень операторів і операндів. Дві кодові програми апроксимуються шляхом вимірювання різниці між зазначеними атрибутами. Відзначається, що атрибутивно-орієнтовані системи дуже вузькі. Тому вони орієнтовані на атрибути, корисні тільки для тих програм, які мають мінімум змін у коді. Найбільш частий 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 [1].

Структурно-орієнтований метод заснований на комбінації двох методик: пошуку подібності у структурі подання з двох частин вихідного коду, а також у застосування атрибута методів підрахунку. За такого підходу вихідний код порівнюється в два етапи. На першому етапі генерується потік токенів програм, а другий – це етап порівняння потоків токенів за допомогою алгоритмів зіставлення рядків. У цих системах такі елементи, як коментарі, пробіли й імена змінних, ігноруються, тому що вони можуть бути легко змінені. Кожен підхід фокусується на певних характеристиках коду. Деякі підходи призначені тільки для перевірки на плагіат вихідного коду, написаного на різних мовах програмування.

Часто вузли абстрактного синтаксичного дерева виходять із лексем, які виділяються на етапі лексичного аналізу. Тобто код перетворюється на послідовність лексичних одиниць із певним значенням, так званим токеном. Процес токенізації залежить від конкретної мови програмування і підтримка декількох мов можлива тільки за умови визначення декількох лексичних аналізаторів.

В аналізі стилю програмування часто згадуються родимі плями – певні властивості коду програми, які використовуються з самого початку її написання, а потім тісно пов'язані у подальшій розробці програмного забезпечення. Вказана сутність вважається об'єктивною, тому вона ідеально підходить для підтвердження авторства та наявності факту плагіату. Кожному програмісту властивий індивідуальний підхід та стиль написання. Ось чому у програмному коді часто можна простежити велику кількість родимих плям, які будуть характерні для конкретного розробника. Помилки також є складовою родимих плям автора вихідного тексту. Тому повторення унікальної помилки може стати доказом плагіату, адже ймовірність отримання однакових специфічних помилок дуже низька [2]. Також слід зауважити, що важливим аспектом проблеми виявлення та видалення особливих тегів автора є знання зловмисником будь-якої конфіденційної інформації. Щоб уникнути цього, 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.

Таким чином, серед наявних сервісів наразі досить складно знайти комплексне рішення, що якісно поєднує в собі всі наведені вимоги: підтримання великої кількості мов програмування; ефективність основного алгоритму; виключення шаблонного коду; можливість історичних порівнянь; підтримка багатофайлового проєкту; відкритість коду та надання зручного користувацького інтерфейсу для завантаження робіт. Деякі підходи доступні для перевірки плагіату дуже складних модифікацій коду, але їм потрібно багато часу, щоб виявити схожість. Отже, необхідне подальше вдосконалення цих сервісів, а також створення нового програмного забезпечення, орієнтованого на поєднання та розширення функціоналу систем для виявлення плагіату кодових програм і в якому будуть вирішені наявні проблеми та реалізовані додаткові функціональні можливості. Основна ж мета використання даного класу програмних засобів – максимально загальмувати тенденцію до стрімкого поширення плагіату та вивести вітчизняну науку на новий якісний рівень.

Література.

1. Зайченко, І.В. (2021). Аналіз методів та дієвих систем виявлення плагіату навчальних кодових програм. Вчені записки ТНУ імені В. І. Вернадського. Серія: Технічні науки, 32(71(6)), 85-90. https://doi.org/10.32838/2663-5941/2021.6/14.

2. Киричек, А.А., Амонс, А.А., Киричек, Г.Г. (2013). Алгоритм фільтрації для системи визначення плагіату в програмному коді. Вісник Національного технічного університету «ХПІ». Серія: Нові рішення в сучасних технологіях, (16), 76-82. http://nbuv.gov.ua/UJRN/vcpinrct_2013_16_20.