- 15% (the previous value was 26%). Today, many well-known companies use deep learning. But the classic and most popular way to use neural networks is through image processing. Consider how CNNs are used to classify images.

Purpose. *The task of image classification is to process the image and determine the class to which it belongs (car, animal, etc.) or the group of classes that best characterize it.*

Results. *The article gives the basic concept and characteristics of convolutional neural networks, describes the structure and mathematical approaches to the implementation of this type of neural network. Provided description of the main layers of convolutional neural networks, mentioned parameters that allow changing the process of training the network. The process of training each of the neural network's layer is considered in detail. In addition, described techniques that allow you to increase the data set and improve the training process of the network only by several transformations.*

Conclusion. *The concept of convolutional neural networks, their connection with biology and structure has been considered in the article. The main layers that are found in CNN are discussed in detail, described their work through mathematics. Each of the layers is considered individually and in conjunction with the other layers of the neural network.*

Described how neural networks work and how to train them properly. Several approaches to learning neural networks are discussed, as well as simple ways to solve common learning problems: increasing the amount of data you need to train and using already trained networks as a base for solving specialized tasks. Here are some options for changing your network's learning process.

The article discusses some of the most common tasks that are solved using convolutional neural networks: classification, localization, detection, and segmentation.

Keywords: *deep learning, machine vision, neural networks, face recognition, convolutional neural networks, receptive fields, deep learning, neural networks, convolutional neural networks, classification, localization, detection, segmentation, rectified linear units.*

Одержано редакцією 11.04.2018 р.
Прийнято до публікації 19.09.2018 р.

УДК 519.688

DOI 10.31651/2076-5886-2019-1-75-85

PACS 02.60.-x, 02.60.Pn

КУЦЕНКО Олександр Анатолійович
студент спеціальності «Прикладна математика» Черкаського національного університету імені Богдана Хмельницького
e-mail: alexkutsenko1995@gmail.com
ORCID 0000-0002-6220-6034

СЕРДЮК Олександр Анатолійович
кандидат економічних наук, старший викладач кафедри прикладної математики та інформатики Черкаського національного університету ім. Б. Хмельницького
e-mail: serdyuk4labs@ukr.net
ORCID 0000-0002-3919-4661

ОГЛЯД АЛГОРИТМІВ ПОШУКУ ПЛАГІАТУ У ПРОГРАМНОМУ КОДІ

У статті розглянуто поняття плагіату, його класифікацію та загальновживані алгоритми пошуку плагіату, а саме: метод ідентифікаційних міток, алгоритм Хескела, метод вирівнювання рядків та метод жадібного рядкового заміщення. З розвитком технологій та можливістю використання інтернету студент, не прикладаючи багато зусиль, може видавати матеріали іншої персони за свої. Тому дані методи та алгоритми часто

використовуються для пошуку плагіату у програмному коді студентів. Метою даної роботи є визначення поняття плагіату, описання класифікації та розгляд методів ідентифікаційних міток, вирівнювання рядків, жадібного рядкового заміщення та алгоритму Хескела. Також розглянуто програмне забезпечення та сайти пошуку плагіату.

Ключові слова: *метод ідентифікаційних міток, алгоритм Хескела, метод вирівнювання рядків, метод жадібного рядкового заміщення, задача пошуку плагіату.*

Постановка проблеми

Нині при швидкому розвитку інформаційних технологій інтелектуальна власність стає ціннішою, аніж раніше. Беручи до уваги швидке зростання обсягів цього виду власності, виникає потреба у захисті авторських прав для перевірки авторства та пошуку плагіату. Задача пошуку та виявлення фрагментів програмного коду, що був запозичений у іншої людини, залишається однією з найбільш актуальних, складних та важливих проблем для викладачів, що навчають програмуванню.

Якщо дві програми мають істотну загальну частину (на рівні мови програмування), то можна вважати, що в одній з них міститься плагіат; причому, плагіатор може змінити оригінальну програму вставкою додаткових операторів (синонімізація та виконання несуттєвих дій), перейменуванням змінних, зміною порядку виконання незалежних операторів, розбиттям деяких функцій на дві і так далі.

Об'єктом дослідження у роботі виступать алгоритми та методи пошуку плагіату.

Метою статті є розгляд методів ідентифікаційних міток, вирівнювання рядків, жадібного рядкового заміщення алгоритм Хескела.

Виклад основного матеріалу

1. Плагіат та класифікація його типів та видів.

Плагіат - це протизаконне використання роботи іншої людини та оприлюднення її під грифом «власної» праці. Плагіатор – людина, яка привласнює чужі праці, роботи або частину рооти під власним іменем, присвоюючи собі їх авторство.

Відповідно до [4], плагіат – це оприлюднення (опублікування) повністю або частково чужого твору під іменем особи, яка не є автором.

Найчастіше плагіат зустрічається у наукових чи творчих околах, де відбувається присвоєння чужих наукових, творчих, художніх творів тощо.

Метою плагіату чи плагіатора є, переважно, бажання стати відомим та/або отримати певну грошову винагороду за рахунок іншої особи.

Згідно з [7], мета плагіату наукового твору – виправдання витрачених бюджетних коштів або коштів замовника шляхом привласнення результатів чужої інтелектуальної праці.

Відповідно до історичних даних, плагіат літературних творів існував ще в давні часи.

У сьогоденні, розвиваючи інформаційні технології, людина створює усі умови для швидкого пошуку плагіату. Раніше автори не мали можливості дізнатися про те, що хтось використовує їх роботу, але зараз приховати це – нелегка задача: написавши у пошуковому рядку деякі ключові слова з роботи, яку автор підозрює у запозиченнях, можна знайти відповідності зі своєю працею та визначити ступінь запозичень.

З розвитком інформаційних технологій до візуального пошуку плагіату шляхом порівняння двох творів додалися й технічні засоби, що полягають у автоматичному порівнянні тексту з іншими [7].

Існує багато різних типів і видів плагіату. Найбільш поширені з них наведено у таблиці 1 (відповідно з [5]).

2. Загальновживані алгоритми пошуку плагіату

Наразі існує досить багато алгоритмів пошуку плагіату, тому описати всі неможливо. Однак, декілька з них наведені нижче.

Таблиця 1

За наявністю умислу:	
Ненавмисний плагіат – у разі незнання вимог, яким повинна відповідати робота.	*Навмисний плагіат.*
За формою відтворення:	
Прямий (відкритий) плагіат – пряме відтворення (відображення) чужого твору або його частини під своїм іменем.	*Завуальований плагіат* – за умов, якщо текст твору зазнає несуттєвих змін шляхом заміни окремих слів та виразів їх синонімічними аналогами. При цьому форма в цілому не змінюється.
За наявністю вказівки джерела:	
Запозичення без вказівки джерела: *"примарний автор"* – автор видає виконану іншою людиною роботу за свою, не змінюючи її зміст; *"фотокопія"* – автор копіює значну частину тексту (але не весь текст) з одного джерела, не вносячи до нього змін; *"натрапив на влучний матеріал"* – робиться спроба приховати плагіат шляхом копіювання з декількох різних джерел, текст яких не змінюється, але автор пише свої перехідні фрази між частинами тесту; *"погане маскування"* – залишається зміст тексту джерела, але деякі формулювання замінюються; *"праця ліні"* – трудомісткий процес, у результаті якого інформація з різних джерел практично повністю перефразовується, пишеться в одному стилі; *"вкрав у себе"* – передбачає запозичення тексту з власних, більш ранніх робіт.	***Запозичення з вказівкою джерела:*** *"забуте посилання"* та *"дезінформатор"* – пов'язані з неправильним або помилковим оформленням посилань на джерело; *"занадто ідеальне перефразування"* – якщо дослівна цитата не взята в лапки. Таким чином, у читача створюється невірне враження про те, що автор навів свою оригінальну інтерпретацію поглядів, викладених у джерелі; *"ідеальний злочин"* – вчиняється, коли автор правильно наводить деякі цитати, а решту перефразує. У результаті читач помилково думає, що перефразований текст є авторським аналізом цитованим думок; *"рясне цитування"* або *компіляція* – відбувається з дотриманням усіх правил цитування та перефразування, але робота практично не містить оригінальних результатів авторського дослідження.

Джерело: відповідно до [5]

Нижче розглянуто окремі, найбільш поширені, алгоритми пошуку плагіату.

Метод жадібного рядкового заміщення

Відповідно до [3], приймає на вході два рядки, а на виході повертає набір їх загальних непересічних підрядків.

Нехай P і T – токенізоване представлення порівнюваних програм.

Перша фаза методу. Шукаємо максимальні спільні підрядки P і T, які не відмічені (спочатку алгоритму всі елементи непомічені). Для цього використовуються три вкладених цикли: перший пробігає по P_p, другий по T_t, а третій знаходить максимальний префікс в P_p і T_t.

Потім відбувається сортування:

- якщо *maxmatch* менше – видаляється зі списку загальних підрядків *matches* все до цього додане і поміщається туди знайдений префікс;
- якщо *maxmatch* більше – нічого не змінюється;
- якщо рівні – додається найбільший префікс P_p і T_t до списку *matches*.

Друга фаза методу. Йдемо по списку, якщо поточний елемент списку – підрядок, який не містить помічених елементів, то записуємо у кінцевий tiles та помічаємо всі елементи розглянутого рядка, що входять до *P* і *T*. Якщо ж довжина рядків у *maxmatch* більша за *MinimumMatchLength*, то повертаємося до першої фази.

Псевдокод алгоритму наведено далі [3]:

```
Greedy-String-Tiling(String P, String T) {
  tiles = {};
  do {
    maxmatch = MinimumMatchLength;
    Forall unmarked tokens Pp in P {
      Forall unmarked tokens Tt in T {
        j = 0;
        while (Pp+j == Tt+j) && unmarked(Pp+j) && unmarked(Tt+j)
          j++;
        if (j == maxmatch) {
          matches = matches ∪ match(p, t, j);
        } else if (j > maxmatch) {
          matches = {match(p, t, j)};
          maxmatch = j;
        }
      }
    }
    Forall match(p, t, maxmatch) ∈ matches {
      if (not occluded) {
        for j = 0...(maxmatch − 1) {
          mark(Pp+j);
          mark(Tt+j);
        }
        tiles = tiles ∪ matches(a, b, maxmatch);
      }
    }
  } while (maxmatch > MinimumMatchLength);
  return tiles;
}
```

У даному методі використовується кілька евристик:

- довші послідовні збіги кращі, ніж набір менших і непослідовних, незалежно від суми довжини останнього;
- ігноруються збіги, довжини яких менші за певне значення порогу.

Ці евристики сприяють втраті оптимального заміщення, проте результати виконання роботи методу дуже близькі до нього для виявлення плагіату.

Асимптотика у найгіршого випадку – $O(n^3)$, але на практиці найчастіше – $O(n^2)$, де n – довжина рядка, у яку переводять програму, використовуючи токенізоване подання.

Метод вирівнювання рядків

Нехай є два файли програмного коду, представлених у вигляді токенізованих рядків s і t відповідно (можливо, різної довжини). Тепер можна використати метод локального вирівнювання рядків, розроблений для визначення подібності ДНК. Вирівнювання виконується за допомогою вставки пропусків так, щоб довжини рядків стали рівними. Існує багато варіантів вирівнювань двох рядків. Наприклад, для рядків "masters" і "stars" цілком допустиме наступне вирівнювання:

masters masters
sta rs stars

Тепер нехай, s і t після вирівнювання, відповідно s' і t'. Розглянемо s_i і t_i: вартість їх збігу – m, вартість пропуску – g, вартість розбіжності – d, де m, d і g – довільні числа. Ціна вирівнювання – це сума вартостей усіх пар s'_i і t'_i, максимальне значення цієї цільової функції для всіх i, j $(i \le j \le |s'| = |t'|)$ у рядках $s'[i..j]$ і $t'[i..j]$ – розмір вирівнювання. Якщо $m = 1$, $d = -1$ і $g = -2$, тоді "rs/rs" та "sters/stars" – фрагменти рядків з максимальним значенням цільової функції. Тоді ціна їх вирівнювань -2 і 3. Вона відповідає редакційній відстані і використовується для визначення відстані між схожими об'єктами, а також успішно використовується для визначення неточності ДНК у обчислювальній біології .

Оптимальне вирівнювання – це таке максимальне значення цільової функції серед усіх вирівнювань, яке можна обрахувати за допомогою динамічного програмування. Нехай s та t – рядки, $D(i, j)$ – оптимальне вирівнювання $s[1..i]$ і $t[1..j]$. Необхідно знайти $max_{1 \le i \le |s|, 1 \le j \le |t|}(D(i, j))$. Визначимо

$$score(s[i], t[j]) = \begin{cases} m, if\ s[i] = t[j] \\ d, else \end{cases}$$

Наступне рекурентне співвідношення дозволяє нам знайти необхідне оптимальне вирівнювання

$$D(i,j) = \begin{cases} D(i-1, j-1) + score(s[i], t[j]) \\ D(i-1, j) + g \\ D(i, j-1) + g \\ 0 \end{cases}$$

Граничні умови задаються співвідношеннями $D(0, i) = 0$ і $D(j, 0) = 0$. Інші елементи матриці D обчислюються при проході зліва направо і згори донизу. Це можливо, оскільки $D(i, j)$ залежить від $D(i-1, j-1)$, $D(i-1, j)$ і $D(i, j-1)$. Час обчислення оптимального вирівнювання – $O(|s||t|)$; витрати пам'яті – $O(max(|s|, |t|))$, тому що для обчислення необхідні лише два рядки.

Використання методу виглядає приблизно так: отримуємо токенізоване представлення p_1 і p_2 двох програм, ділимо другий рядок p_2 на підрядки, кожен з яких представляє модуль вихідної програми. Для кожного підрядка і p_1 отримуємо значення оптимального локального вирівнювання. Це дозволяє методу коректно обробляти перестановки модулів вихідної програми.

Переваги та недоліки

Якщо розглядати реалізацію алгоритму, використовувану детектором SIM, – ніяких покращень порівняно з іншими алгоритмами не отримується, зокрема, на базі цього алгоритму не можна організувати базу даних, яка прискорює перевірку *один-проти-всіх*. Однак, застосування алгоритму до токенізації програмного коду у нашому розумінні, можливо, дозволить отримати хороші результати, що нами буде проводитись у подальшому.

Метод ідентифікаційних міток

Згідно [9], під час перевірки на плагіат необхідно знайти копії чи фрагменти копій у текстовій базі. У такому випадку безпосереднє порівняння файлів є неефективним.

Метод ідентифікаційних міток дозволяє перетворити файл у більш коротку послідовність: файл зіставляється з набом ідентифікаційних міток.

Нехай дано довільний текст:

abracadabra (складається з 11 символів; $m = 11$).

Розглянемо набір міток для файлу на даному прикладі.

k-грамом називається група будь-яких *k* символів розташованих підряд. Побудуємо *k*-грами для прикладу при, наприклад, $k = 3$:

abr, bra, rak, aka, kad, ada, dab, abr, bra

Кількість *k*-грамів, які можна побудувати для тексту довжини *m*, позначимо *n*:

$n = (m - (до - 1))$ (тут $n = 9$).

Проведемо хешування усіх *k*-грамів. Для даного прикладу послідовність хеш-значень буде такою: 12, 35, 78, 3, 26, 48, 55, 12, 35.

Використання усіх значень нераціональне, тому вибирається невелика їх кількість. Обрані хеш-значення стають меншими файлами. Окрім мітки зберігається також інформація про те, до якого файлу вона належить. Якщо хеш-функція дає малу ймовірність колізій, то однакові мітки у наборах двох файлів свідчать про те, що у них є спільний фрагмент, а за кількістю спільних міток можна говорити про схожість файлів.

Переваги методу:

- можна створити базу даних для перевірку *один-проти-всіх*;
- плюси токенізованого представлення;
- спільні підрядки, які менші певного порогу, ігноруються;
- метод нечутливий до переміщення фрагментів коду.

Недоліки:

- можливість збігу токенізованого представлення програм, але відсутність збігу у початкових кодах програм.

Алгоритм Хескела

Розглянемо алгоритм, описаний у [9].

Нехай маємо два програмних коди, які подані у вигляді списку токенів *a* і *b* відповідно. Одним з критеріїв подібності вважається довжина *найбільшого спільного*

підрядка. Ми завжди маємо змогу знайти такий елемент рядка a_i, що НЗП (найбільша загальна (спільна) підпослідовність) рядків $a_0=a_{|a|}a_{|a|}$ *-1 ...* a_ia_1 *...* a_{i-1} і b буде меншою (щонайбільше – у 2 рази), аніж НЗП (a, b) (якщо НЗП (a, b) > 1). Щоб уникнути цього, скористаємося алгоритмом Хескела, який вимагає кількох проходів по циклах і при цьому виконується за лінійний час. Розкладемо a і b на k-грами. Знайдемо в a і b ті k-грами, які зустрічаються лише один раз. Для кожної пари перевіряємо, чи елементи рядків схожі з тими, що знаходяться до них; якщо так, то повторюємо ті ж дії для них і так далі, поки не знайдеться розбіжність. Аналогічно для рядків, які лежать далі. У результаті, отримується набір спільних підрядків a і b, що не перетинаються. Їх загальна довжина буде інтерпретуватись як подібність програм, що відповідають a і b.

Беручи до уваги подане у [2], можна навести наступні приклади переваг і недоліків цього методу.

Переваги:

– лінійна залежність роботи методу від часу.

Недоліки:

– можливість збігу токенізованого представлення програм, але відсутність збігу у початкових кодах програм;
– мала кількість унікальних k-грамів у великих файлах;
– вставка в блок або зміна на еквівалентний оператор у більшості випадків буде приводити до пропуску цієї частини блоку;
– не можна створити базу даних для перевірку *один-проти-всіх*.

3. Огляд програмного забезпечення та сайтів пошуку плагіату

Якщо розглядати сайти, які призначені для пошуку плагіату, можна виділити наступні.

Turnitin (http://submit.ac.uk/) – сайт перевіряє роботу на унікальність, використовуючи базу даних різних робіт (авторських, дипломних тощо). Якщо ж такої роботи не було знайдено у базі, то система доповнює свою базу цим файлом. [5]

AntiPlagiat.ru (http://www.antiplagiat.ru) – чимось схожий на *Turnitinш*, але на цьому ресурсі зареєструватись може як викладач, так і студент (щоб користуватися всіми можливостями, необхідно купити платний аккаунт).

Unplag – пошук плагіату можливий як у режимі реального часу, так і по базі даних, у якій збережено велику кількість документів; працює з Canvas, Sakai, Moodle.

www.miratools.ru – дозволяє здійснювати онлайн-перевірку тексту на плагіат. Система використовує результати видачі пошукових систем.

www.istio.com – здійснює перевірку тексту на унікальність з використанням систем Яндекс.XML і Yahoo.com.

Plagiatinform – перевіряє тексти на унікальність у власній базі та у Інтернеті. Сайт може знаходити не унікальні фрагменти тексту у вигляді текстів, де «перемішані» фрагменти тексту кількох джерел.

Copyscape – дозволяє шукати копії сайтів у мережі Інтернет; повертає список сайтів, де можливий збіг*;* використовує системи Google і Yahoo!

http://plagiarisma.net/ – веб-сайт, за допомогою якого може проводитись перевірка тексту курсових, кваліфікаційних та магістерських робіт; використовує системи Google та Bing.

Якщо ж, брати до уваги програмне забезпечення, призначене для перевірки тексту на унікальність і доступне як викладачам, так і студентам, то можна вказати наступні [5]:

Advego Plagiatus (http://advego.ru/plagiatus/);
Etxt Антиплагиат (http://www.etxt.ru/antiplagiat/);
Anti-Plagiarism (http://ikc2.tup.km.ua/index_ru.shtml).

Розглянемо кожну програму окремо.

1. *Advego Plagiatus* – програма, яка шукає в Інтернеті фрагменти або повні копії тексту; має зручний та простий інтерфейс. Результат роботи – відсоток унікальності тексту та відсоток збігу тексту. Також надає список джерел тексту, де були взяті фрагменти (рис. 1). [5]

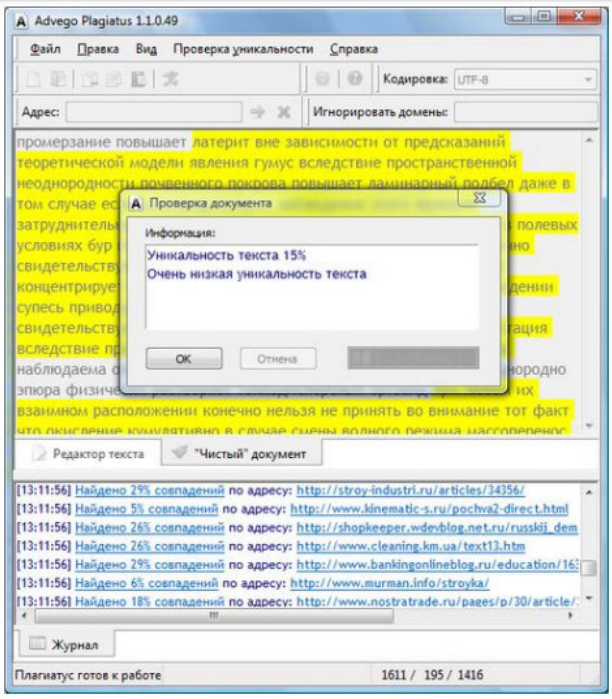

Рис. 1. Програма Advego Plagiatus

2. *Etxt Антиплагіат* – програма, призначена для перевірки тексту чи документу на унікальність. Надає можливість провести порівняльний аналіз фрагментів власного тексту на знайдених джерелах (рис. 2). [8]

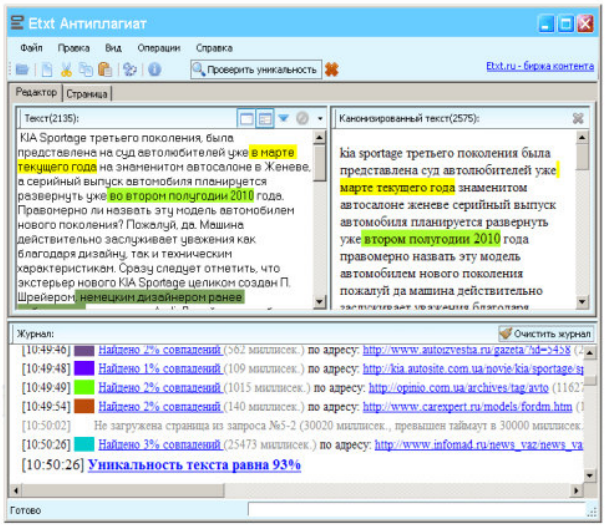

Рис. 2. Програма Etxt Антиплагіат

Anti-Plagiarism – програма, призначена для виявлення і запобігання плагіату. Це ефективний інструмент для боротьби з копіюванням інформації з вебу (рис. 3). [5]

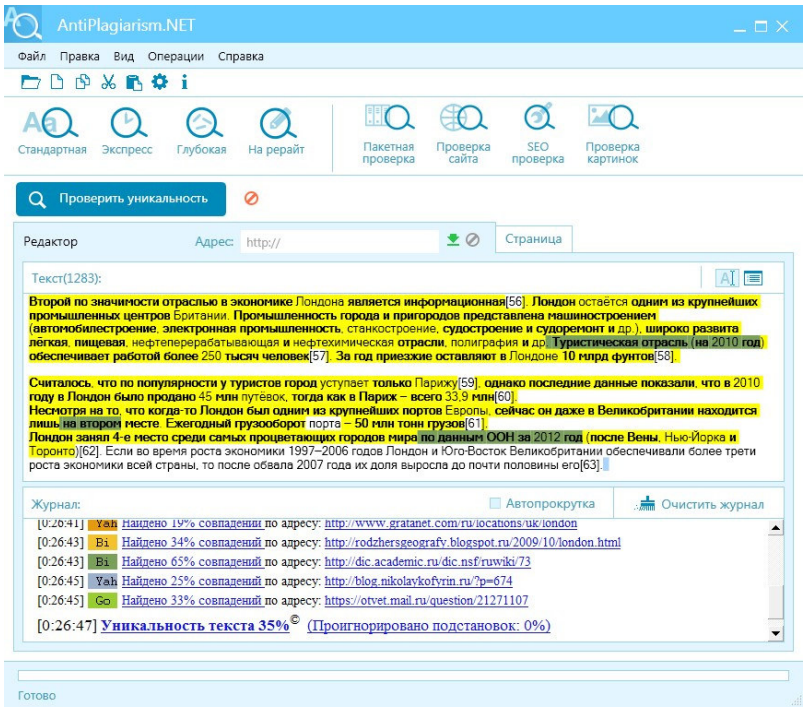

Рис. 3. Програма Anti-Plagiarism

Висновки

У статті розглянуто окремі методи та алгоритми пошуку плагіату, а саме: метод ідентифікаційних міток, алгоритм Хескела, метод вирівнювання рядків, метод жадібного рядкового заміщення. Представлено загальний опис кожного з методів та алгоритмів. Кожен з них потенційно може бути використаний для виявлення плагіату у програмному коді. Однак, при використанні токенізованого подання коду програми усі розглянуті алгоритми мають головний недолік: можливі збіги токенізованого представлення програм при відсутності збігу у початкових програмах.

На основі поданого у статті можна стверджувати, що наразі фактично немає якогось одного загальноприйнятого універсального способу, який можна було б використовувати для розв'язання задачі пошуку плагіату у програмному коді. Тому у подальшому нами буде проводитись робота по модифікації існуючих алгоритмів для адаптації їх до розв'язання задачі аналізу вихідного коду програм на наявність запозичень.

Список використаної літератури:

1. Michael J. Wise. String similarity via greedy string tiling and running KarpRabin matching. Dept. of CS, University of Sydney, December 1993.
2. Амонс А.А. Выявление плагиата в программном коде C# / Амонс А.А., Зайцев С.Ю., Киричек А.А. // Вестник НТУУ «КПИ» Информатика, управление и вычислительная техника № 53. – с.170-179.
3. Евтифеева О.А. АНАЛИЗ АЛГОРИТМОВ ПОИСКА ПЛАГИАТА В ИСХОДНЫХ КОДАХ ПРОГРАММ / Евтифеева О.А., Красс А.Л., Лакунин М.А., Лысенко Е.А., Счастливцев Р.Р. // Санкт-Петербургский государственный университет – С. 188-196.
4. Закон України Про авторське право і суміжні права // Відомості Верховної Ради України (ВВР). – 1994. – № 13. – С. 64 .
5. Лисиченко М. Л. Методичні рекомендації щодо механізму перевірки письмових робіт на плагіат / М. Л. Лисиченко, В. І. Жила, А. В. Левкін. – Х: ХНТУСГ, 2017. – 28 с.
6. Лупаренко Л. А. Інструментарій виявлення плагіату в наукових роботах: аналіз програмних рішень / Л. А. Лупаренко // Інформаційні технології і засоби навчання. – 2014. – Том 40, № 2. – С. 151-169.

7. Плагіат авторських творів [Електронний ресурс] – Режим доступу до ресурсу: https://pidruchniki.com/1834071963579/pravo/plagiat_avtorskih_tvoriv.
8. Плагіат у студентський роботах: методи виявлення та запобігання / [Н. В. Стукало, К. В. Ковальчук, М. В. Литвин та ін.]. – Дніпропетрівськ: ДНУ ім.О. Гончара, кафедра міжнародної економіки і світових фінансів, 2013. – 44 с.
9. Чижова А. А. АЛГОРИТМИ ПОШУКУ ПЛАГІАТУ / А. А. Чижова // Восточно-Европейский журнал передовых технологий. – Х., 2010. - № 2(4). – с. 13-16.
10. Що таке плагіат? [Електронний ресурс] – Режим доступу до ресурсу: https://uk.kagouletheband.com/ucheba/28889-chto-takoe-plagiat.html.

Bibliography:

1. Michael J. Wise. String similarity via greedy string tiling and running KarpRabin matching. Dept. of CS, University of Sydney, December 1993.
2. Amons A.A. Vyiyavlenie plagiata v programmnom kode C# / Amons A.A., Zaytsev S.Yu., Kirichek A.A. // Vestnik NTUU «KPI» Informatika, upravlenie i vyichislitelnaya tehnika # 53. – s.170-179.
3. Evtifeeva O.A. ANALIZ ALGORITMOV POISKA PLAGIATA V ISHODNYiH KODAH PROGRAMM / Evtifeeva O.A., Krass A.L., Lakunin M.A., Lyisenko E.A., Schastlivtsev R.R. // Sankt-Peterburgskiy gosudarstvennyiy universitet – S. 188-196.
4. Zakon Ukrainy Pro avtorske pravo i sumizhni prava // Vidomosti Verkhovnoi Rady Ukrainy (VVR). – 1994. – № 13. – S. 64 .
5. Lysychenko M. L. Metodychni rekomendatsii shchodo mekhanizmu perevirky pysmovykh robit na plahiat / M. L. Lysychenko, V. I. Zhyla, A. V. Levkin. – Kh: KhNTUSH, 2017. – 28 s.
6. Luparenko L. A. Instrumentarii vyiavlennia plahiatu v naukovykh robotakh: analiz prohramnykh rishen / L. A. Luparenko // Informatsiini tekhnolohii i zasoby navchannia. – 2014. – Tom 40, № 2. – S. 151-169.
7. Plahiat avtorskykh tvoriv. Retrieved from: https://pidruchniki.com/1834071963579/pravo/plagiat_avtorskih_tvoriv.
8. Plahiat u studentskyi robotakh: metody vyiavlennia ta zapobihannia / [N. V. Stukalo, K. V. Kovalchuk, M. V. Lytvyn ta in.]. – Dnipropetrivsk: DNU im.O. Honchara, kafedra mizhnarodnoi ekonomiky i svitovykh finansiv, 2013. – 44 s.
9. Chizhova A. A. ALGORITMI POShUKU PLAGIATU / A. A. Chizhova // Vostochno-Evropeyskiy zhurnal peredovyih tehnologiy. – H., 2010. - # 2(4). – s. 13-16.
10. Shcho take plahiat? Retrieved from: https://uk.kagouletheband.com/ucheba/28889-chto-takoe-plagiat.html.

KUTSENKO Oleksandr,
student, The Bohdan Khmelnytsky National University of Cherkasy
SERDIUK Oleksandr,
PhD, Senior Lecturer, The Bohdan Khmelnytsky National University of Cherkasy

PECULIARITIES OF WORK OF PLAGIATE SEARCH ALGORITHMS IN SOFTWARE

Summary. Introduction. *In recent years, students are increasingly practicing the assignment of other people's works and giving them for their own. All this is due to the development of computer technology and the possibility of using the Internet. The student without putting a lot of effort copies someone else's program code, and brings it to the teacher as his own.*

There are quite a few algorithms to detect plagiarism in the works of students.

The most common algorithms and methods that can cope with this task are the methods of identification labels, string alignment, greedy string substitution, Heskel algorithm. Each of these algorithms has its advantages and disadvantages. The main advantages of these algorithms are that they all have a linear dependence of the method on time and work with tokenized representation of the program code, which makes the task of checking for plagiarism much easier.

Purpose. *The purpose of this paper is to define what plagiarism is and its classification, review existing software and sites that are capable of checking text for uniqueness and review methods of identification labels, string alignment, greedy string substitution Heskel algorithm*

Results. *At the beginning of the analysis of the topic of this article, there was a problem, and all-so what can be considered as plagiarism of software code. Then there are fair questions:*

- *how different programs must be for one of them to be considered plagiarism;*
- *what can be cited as evidence to be sufficient for plagiarism.*

The paper considers various methods and algorithms for plagiarism search in the software code, namely the method of identification labels, the Heskel algorithm, the method of string alignment,

the method of greedy string tiling. A general description of each of the methods and algorithms is presented. Each of them is able to detect plagiarism in the program code, but because of their tokenized representation, they all have one common disadvantage: the main drawback of all these methods is that the tokenized representation of programs may match, but there is no coincidence in the initial codes of programs.

Conclusion. *Taking into account what was said above about the algorithms and methods of plagiarism detection, we can conclude that now there is actually no one universally accepted universal way that could cope with the task of finding plagiarism in the software code, a way that would not be subject to time. In the future, we will develop a method for searching for plagiarism in program code based on tokenization of source programs.*

Keywords*: identification tag method, line alignment method, greedy string tiling method, the Heskel algorithm, plagiarism search problem.*

Одержано редакцією 27.09.2018 р.
Прийнято до публікації 19.12.2018 р.