Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

РОЗРОБКА ПРОТОТИПУ СИСТЕМИ ДЛЯ АНАЛІЗУ ЗОБРАЖЕНЬ НА ПЛАГІАТ З ВИКОРИСТАННЯМ АЛГОРИТМУ PSNR

Текстова частина до курсової роботи
за спеціальністю „Інженерія програмного забезпечення” 121

Керівник курсової роботи
с.в. Вовк Н.Є.
(прізвище та ініціали)
_____
(підпис)
“____” _____ 2020 р.

Виконав студент
Сахаров А.Ю.
(прізвище та ініціали)
“____” _____ 2020 р.

Київ 2020

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри мультимедійних систем,

доцент, к.ф. – м.н. О. П. Жежерун _____

(підпис)

„____”_____2019 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту__Сахарову А.Ю.__ факультету _інформатики_ _____3-го____курсу

Розробити Комп’ютерну систему порівняння зображення на плагіат, використовуючи алгоритм PSNR та результати Google Images

Зміст ТЧ до курсової роботи:

Індивідуальне завдання
Анотація
Вступ
1 Огляд існуючих інструментів для перевірки на плагіат
2 Алгоритми порівняння зображень та алгоритм PSNR
3 Розробка алгоритму програми
4 Аналіз результатів
Висновки
Список літератури
Додатки

Дата видачі „___” _____ 2019 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Тема: РОЗРОБКА ПРОТОТИПУ СИСТЕМИ ДЛЯ АНАЛІЗУ ЗОБРАЖЕНЬ НА ПЛАГІАТ З ВИКОРИСТАННЯМ АЛГОРИТМУ PSNR

Календарний план виконання роботи

No п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1	Отримання завдання	15.10.2019	
2	Аналіз літератури, пошук джерел	30.11.2019	
3	Аналіз технологій та додатків для створення програми	28.12.2019	
4	Розробка алгоритму, архітектура системи	20.01.2020	
5	Створення програмного коду	26.03.202	
6	Аналіз створеного продукту з керівником	15.04.2020	
7	Написання текстової частини до курсової роботи	03.05.2020	
8	Перегляд текстової частини керівником	06.05.2020	
9	Написання доповіді, слайдів, останні правки	10.05.2020	
10	Завантаження курсової роботи	11.05.2020	
11	Захист курсової роботи	19.05.2020	

Студент _____

Керівник _____

“_____” _____

Зміст

Анотація

Робота націлена на оптимізацію процесу перевірки зображень на плагіат для різних потреб. Програма поєднує у собі кілька ключових компонентів та етапів, щоб видавати необхідний результат будь-якому типу користувачів(-ок) – досвічених та недосвідчених. В основній частині описаний використаний у програмі алгоритм порівняння зображень PSNR, що з високою точністю визначає схожість зображень та який не збивають навіть видозміни зображень, до яких можуть вдаватися плагіатори(-ки). Також описано процес порівняння зображення разом з результатом Google Images, що не менш точно видає схожі зображення до досліджуваного, так само повертає ідентичні. В аналізах результатів описано, що програма надає можливість завантажити зображення в архівах, що користувач(-ка) хотів(-ла) б перевірити, а повертає статистику по схожості зображення зі знайденими в інтернеті та коефіцієнтом схожості.

Ключові слова: унікальність, ідентичність, схожість, зображення, алгоритм перевірки, результати пошуку.

Вступ

Одна з найбільших проблем сучасності, де будь-яка інформація є у вільному доступі для будь-кого з усього світу – брак контролю щодо доступу до цієї інформації. Тому надзвичайно високу актуальність має проблема плагіату та його викриття. Соціальні зрушення вже давно зачепили і цей аспект збереження людської гідності, за який дуже часто можна навіть потрапити до в'язниці або ж, хоча б, отримати великий штраф. Але питання врегулювання не є ключовим об'єктом дослідження цієї роботи.

Насправді, головним таргетом цієї роботи є плагіат у сфері досліджень. На даний час відома велика кількість випадків плагіату під час навчання у навчальних установах різного рівня, або ж під час подачі роботи на публікацію, що досить давно змусило замислитись над способами запобігання цьому явищу. Насправді, як виявилось, найкращим способом запобігання є можливість якісної перевірки на плагіат. Адже якщо автор(-ка) розуміє, що його/її роботу очікує перевірка на плагіат, це стає одним з найвагоміших аргументів проти використання чужої роботи у власних цілях.

З іншого боку, приділяючи величезну увагу різним випадкам плагіату серед текстових фрагментів, на сьогодні існує суттєвий недолік багатьох систем перевірки – відсутність перевірки для візуальних джерел. Саме через те, що це є не менш важливою категорією ресурсів, що потребує аналізу на унікальність, ця робота націлена на розробку простого в користуванні застосунку для подібних потреб.

Робота складається з чотирьох розділів.

Перший розділ присвячено аналізу вже існуючих продуктів для задоволення відповідних потреб. У ньому описано те які застосунки вдалося дослідити, а які, будучи не у вільному доступі, можливо й мають відповідний функціонал, але достеменно пересвідчитися в цьому не вдалося.

Другий розділ описує існуючі алгоритми порівняння зображень, що є ключовим етапом роботи програми, адже саме на перевірці зображень поставлено акцент. Серед описаних алгоритмів є також алгоритм PSNR та обгрунтування його вибору.

В третьому розділі роз’яснено принцип роботи застосунку, алгоритм, який проходить зображення та користувач(-ка) разом із ним. Також доволі детально описано усе, що стосується додаткових модулів програми, зокрема Google Vision API, адже саме він здйснює підключення до Google Image Search і надає можливість опрацювати збіги із тим, що вдалося знайти у мережі. Додатково у третьому розділі обгрунтовано корисність системи та зручність у користуванні.

Четвертий розділ присвячено роботі з результатами програми. Саме там стає зрозумілою уся простота та корисність застосунку та наводяться описи різних результатів роботи програми із запропонованими зображеннями.

Створено програму, яка забезпечує легкий в користуванні алгоритм перевірки зображень на плагіат.

Постановка задачі

1. Провести аналіз різних ресурсів перевірки зображень на плагіат.

2. Обрати найбільш ефективний алгоритм для порівняння зображень, що буде використовуватися у програмі.

3. Розробити зручний та послідовний алгоритм для використання будь-яким користувачем(-кою).

4. Виконати аналіз результатів роботи програми та зробити висновок про її ефективність та можливі оптимізації у застосуванні.

Розділ 1: Огляд існуючих інструментів для перевірки на плагіат

1.1 Google.com

Якщо перед будь-якою людиною світу поставити задачу знайти необхідну інформацію, то першою системою, яку вона обере, щоб скористатися, буде пошукова система Google. Ми доволі сильно звикли до того, що будь-яку інформацію можна знайти саме там і, безсумнівно, ця пошукова система одна з найпопулярніших у світі. Тому не дивно, що велика кількість інформації, яку необхідно проглянути для перевірки на плагіат, міститься саме серед результатів пошуку цією системою. При перевірці на плагіат можна керуватися доволі тривіальним алгоритмом, що включає в себе дві дії. Перша з них – пошук досліджуваного фрагменту інформації, другий – самостійний аналіз на унікальність відносно знайдених результатів. Якщо з текстом це не такий швидкий процес і можна легко визначити лише майже ідентичні тексти, то з зображеннями все доволі легко: Google Images надзвичайно точно видає результати пошуку по зображенню, що дає можливість оцінити унікальність досліджуваного файлу на власні очі. У цьому сервісі застосовується технологія Reverse Image Search. Це так само ефективно у ситуаціях, коли ми досліджуємо лише фрагмент зображення, адже Google Images видає не лише ідентичний результат, а і частково ідентичний. Це є актуальним вирішенням проблеми плагіату з використанням зміни розміру чи обрізання зображення, що також вважається за плагіат. Але і у цієї тривіальної системи перевірки на ідентичність є недолік – візуально схожі результати. Справа в тому, що Reverse Image Search включає технологію машинного навчання та надає також результат власного розпізнавання об’єктів на зображенні, що додатково видає як результат пошуку *схожі* зображення, що вже не є гарантією плагіату. Зокрема це стосується ситуацій часткової видозміни характеристик зображень (кольору, яскравості тощо), збіги з оригіналом яких мали б з’являтися саме у категорії візуально

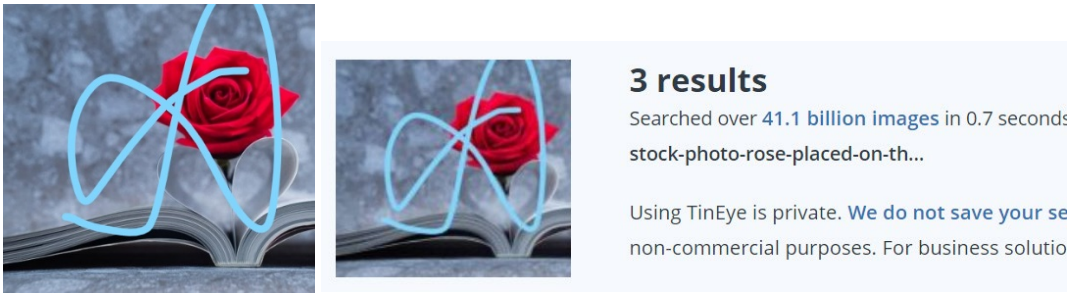
схожих. Через це у користувача виникають очевидні труднощі з ручною оцінкою зображення на унікальність.

1.2 TinEye

Доволі популярним ресурсом для порівняння зображень та пошуку ідентичних є TinEye. Цей сервіс є у вільному доступі та також використовує технологію Reverse Image Search. Роботу цього застосунку можна проаналізувати на конкретних прикладах.

Рисунки 1.2.1, 1.2.2 - При завантаженні звичайного зображення з інтернету ресурс видає 35 збігів.

Рисунки 1.2.3, 1.2.4 - Якщо частково деформувати зображення (обрізати), то кількість результатів зменшується.



Рисунки 1.2.5, 1.2.6 - Намагання знайти проблему при спотворенні зображення, не призвело до нульових збігів.

Таким чином, ресурс TinEye є доволі ефективним при пошуку навіть видозмінених зображень, адже чітко видає кількість збігів, що дає оцінити ймовірність того, що зображення було скопійоване та використане з

порушенням доброчесності. Єдиний недолік полягає в тому, що результатом є саме кількість збігів та перелік посилань, де вони були знайдені, хоча для оцінки плагіату необхідний також відсоток схожості зі збігом, що був знайдений. Цей недолік став одним з ключових компонентів модифікації, що відрізняє створену програму цієї роботи від ресурсу TinEye.

1.3 Ресурси з категорії посередників

При перевірці на плагіат зображень на ресурсах у вільному доступі можна натрапити на такі, що не являють собою якісь унікальні розробки, а є відкритими посередниками між користувачем та пошуковими системами. Наприклад, один з перших ресурсів, що можна знайти, це smallseotools. При завантаженні зображення користувач(-ка) все ще перебуває на початковому сайті, але після завантаження процесу пропонуються результати пошуку різних пошукових систем, при чому результати ті ж самі, що і результати виконання запиту пошуковою системою наодинці.

1.4 Платні ресурси

Такі ресурси не вдалося дослідити, адже їх функціонал обмежений, подекуди на них безкоштовно навіть не можна перевірити текст на плагіат.

Розділ 2: Алгоритми порівняння зображень та алгоритм PSNR

Під час дослідження зображення на унікальність виникла проблема вибору алгоритму, що необхідно застосовувати для вирахування відсотка схожості із зображеннями. Це необхідний крок, адже це суттєво пришвидшує роботу алгоритму відносно перевірки унікальності зображення вручну (людським оком). Для конкретного розгляду варіантів алгоритмів, щоб застосувати при порівнянні зображення, була обрана відома бібліотека OpenCV, що надає вичерпні можливості як для image processing загалом, так і для порівняння зображень між собою задля знаходження спільних рис.

2.1 Histogram comparison

Одним з найелементарніших методів порівнянння зображень у комп'ютерній графіці загалом є порівняння за допомогою методу гістограм. Суть цього методу полягає в тому, що для досліджуваних зображень в залежності від кольорової гами створюються, так би мовити, статистичні графіки [1]. Якщо досліджуване зображення є чорно-білим, то для нього створюється лише один графік, що зображує статистику кольорів по градації сірого для цього зображення. У випадку з кольоровими зображеннями виникають певні обмеження, адже потрібно чітко встановити яку систему розбиття на кольори необхідно використовувати (RGB, CMY, YUV тощо). Фактично, суть гістограм полягає в тому, щоб розбити усі пікселі на категорії по кольорам та порахувати кількість пікселів кожного з цих кольорів. Тим самим алгоритм встановлює комбінацію кількостей пікселей кожного кольору для досліджуваного зображення. Таким чином, однакові зображення точно будуть співставлені як такі, що утворюють однакові гістограми. Але, в цього алгоритму є суттєвий недолік: якщо використовувати ображення однакових кольорових гам, то можна назвати ідентичними зображення зовсім різних предметів. Наприклад, алгоритм чітко розпізнає відмінність між зеленим лісом та синім морем, але у нього виникнуть проблеми з жовтими бананами та жовтим піском на пляжах. Через що

цей алгоритм не може впоратися на високому рівні точності із порівнянням візуально схожих зображень, а це не є оптимальним розв'язком поставленої задачі.

2.2 Template matching

Наступний алгоритм, який було розглянуто, це алгоритм зіставлення із шаблоном. Цей алгоритм є прикладом такого, що акцентує увагу не на кольорі, а на відношенні між сусідніми пікселями. Метод має в основі пошук такої ділянки на зображенні, що буде максимально схожою з тією, що намагаються знайти або ж розпізнати [2]. Конкретніше, цей алгоритм є доволі ефективним для пошуку визначеного об'єкту на зображенні, а не порівнянні зображень між собою, адже це не є раціональним використанням підрахунків, що виконує програма. Алгоритм може базуватися на розрахунках різних коефіцієнтів, серед яких квадратичний, кореляційний та нормалізований [5].

Рисунки 2.2.1, 2.2.2 – розпізнавання обличчя, знаходження за шаблоном на великому зображенні з використанням нормалізації [5]

Метод з високою точністю можна використовувати для розпізнавання об'єктів (наприклад, обличь), але це нераціонально для аналізу на плагіат, адже робота з пошуком цілого зображення на інших зображеннях є ресурсомісткою та працює з низькою точністю для хоча б трохи видозмінених зображень (цей

алгоритм не буде видавати ефективних результатів, якщо змінити розміри зображення або відношення його вимірів).

2.3 Feature matching

Один з найефективніших способів порівняння двох зображень – порівняння об’єктів, що на ньому знаходяться. Саме в цьому полягає суть алгоритму Feature matching. За принципом цього алгоритму, на початковому зображенні знаходяться деякі об’єкти, що достатньо чітко виділені та мають доволі виразне положення. Система не розуміє, що саме за об’єкт вона виділила, але вона сприймає її як набір точок, що становлять контури вибраного об’єкту. В основі алгоритму використовують гіпотезу так званих distinguishing points, що, конкретніше, можна назвати keypoints або feature points [6]. Саме ці точки і визначають об’єкти та їх контури на зображенні. Одним з найпопулярніших прикладів у комп’ютерній графіці є використання кутів, знайдених за допомогою алгоритму Гаріса. Для кожної точки на зображенні знаходяться такі точки в її околі, що мають характерний колір чи інтенсивність, що можуть вказувати, що саме в їх напрямку відбувся поворот контуру. Саме так визначається врешті-решт і контур всього зображення. Алгоритм Гаріса був перетворений у різні інші алгоритми OpenCV (FAST, SURF, SIFT тощо), а також є важливою частиною feature matching. Загалом, віднайшовши ключові об’єкти на одному з зображень, алгоритм переходить до того, щоб знайти ті ж самі на іншому. Зокрема, метод дозволяє знаходити змінений у розмірі або повернутий елемент, що додатково дає можливості для ефективної перевірки на унікальність. Але і у цього методу є суттєвий недолік – час. Алгоритм працює дуже довго і не є достатньо ефективним для поставленої задачі – перевірити на унікальність певне зображення, порівнюючи його далеко не з одним знайденим в інтернеті.

2.4 PSNR

Peak signal-to-noise ratio (PSNR) – сам по собі, інженерний термін. Але у комп’ютерній графіці він так само застосовується. Він застосовується як

коефіцієнт для двох зображень, що використовує значення пікселів по усіх кольорових каналах [7]. Для застосування цього алгоритму необхідно перевести зображення в іншу кольорову систему, наприклад, HSL. Коефіцієнт PSNR вимірюється у децибелах. Зазвичай, до цього методу звертаються тоді, коли хочуть перевірити реально однакові зображення, одне з яких є реконструкцією або розархівованою версією іншого. Справа в тому, що при таких операціях можуть відбутися певні втрати у якості зображень, і сама суть PSNR у тому, щоб порівняти кожен піксель двох зображень на предмет появи шумів. Через це, цей коефіцієнт є також дуже ефективним і для порівняння двох зображень на предмет унікальності, адже навіть при деяких змінах у кольорі, яскравості чи інших параметрах, коефіцієнт доволі точно передає (не)ідентичність двох пікселів, а отже і двох зображень. Саме його було вирішено обрати для даної роботи, адже він є доволі зручним у реалізації (вбудовано в OpenCV) та швидким у видачі результатів.

Розділ 3: Розробка алгоритму програми

3.1 Користувацький інтерфейс

Враховуючи те, що для роботи з OpenCV однією з найзручніших мов програмування є python, було вирішено обрати саме її для розробки прототипу. Внаслідок продумування архітектури застосунку постало питання до створення графічного інтерефейсу, який би полегшив використання застосунку для пересічних користувачів, які не дуже розуміються на програмуванні. Перебравши кілька варіантів бібліотек для створення інтерфейсу у python вибір було зупинено на вбудованому плагіні для PyCharm – PySimpleGUI [3]. Навіть не опановуючи до кінця увесь функціонал даної бібліотеки, вдалося використати деякі приклади застосування методів з документації самої бібліотеки. Загалом вдалося створити доволі дружній та зрозумілий інтерфейс для роботи із застосунком.

Перше вікно призначене для основного процесу роботи програми – вибору файлу для обробки. За допомогою кнопки “Browse” можна перейти у файлову систему ПК та обрати файл, що буде використано для аналізу на плагіат. Було вирішено, що для використання системи необхідно завантажувати до неї архів із зображеннями, які будуть перевірятися на плагіат. Доволі часто при подачі будь-якої наукової роботи/статі вимагається окремо надавати архів і використаними зображеннями. Саме тому і в розробленому застосунку обраний цей вид файлозавантаження. Система підтримує .ZIP та .TAR. В середины кожного архыву опрацювуються лише зображення. Крім того, є можливість завантажити зображення напряму, не в архіві.

Рисунок 3.1.1 – початкове вікно, тут користувач(-ка) обирає файл архіву

Наступне вікно – фінальне, воно дозволяє отримати результат. Внаслідок роботи алгоритму пошуку та порівняння зображень зі знайденими, на екран виводяться посилання на зображення, з якими система знайшла збіги та відсоток цих збігів. Зображення відсортовані за відсотком збігів і таким чином на вікні виводиться інформація про всі зображення одразу. Крім того, система зберігає результати порівняння у CSV файлі для зручного проглядання поза системою. Наприклад, у Microsoft Excel.

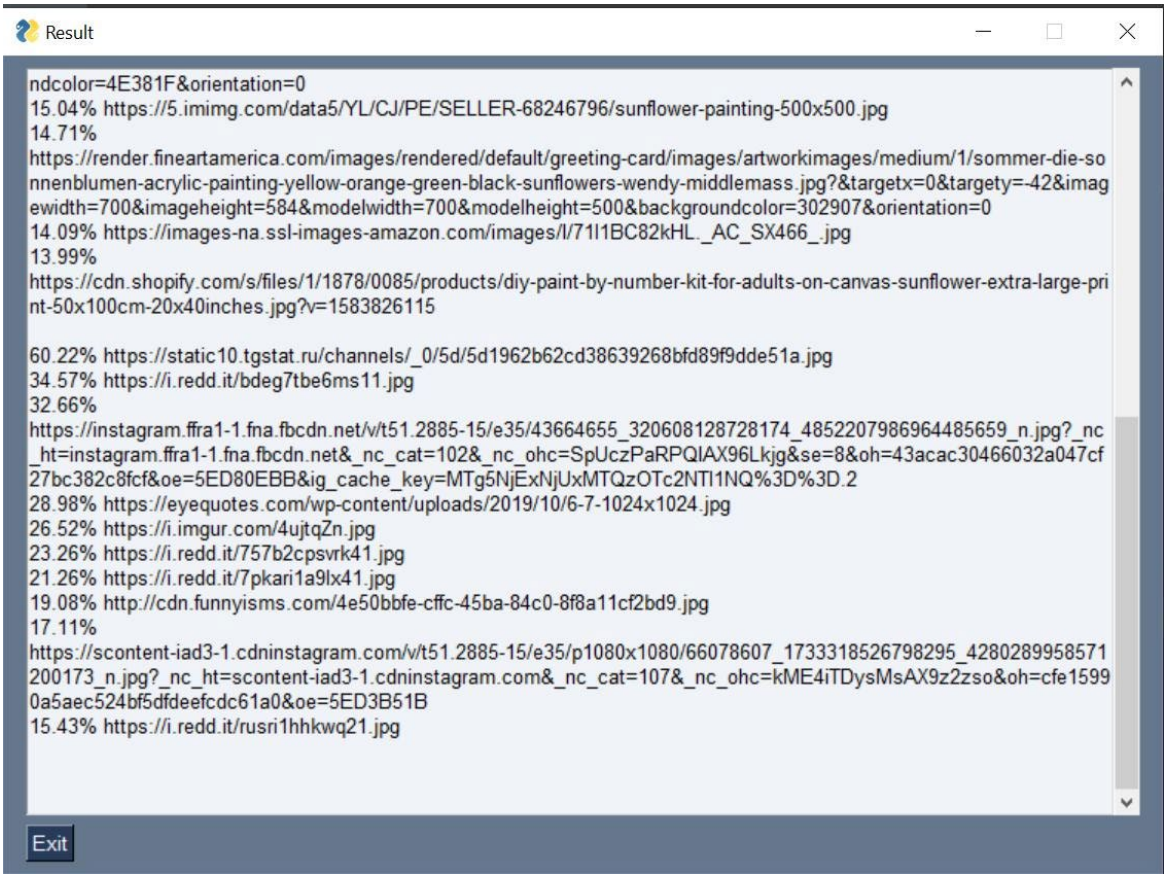
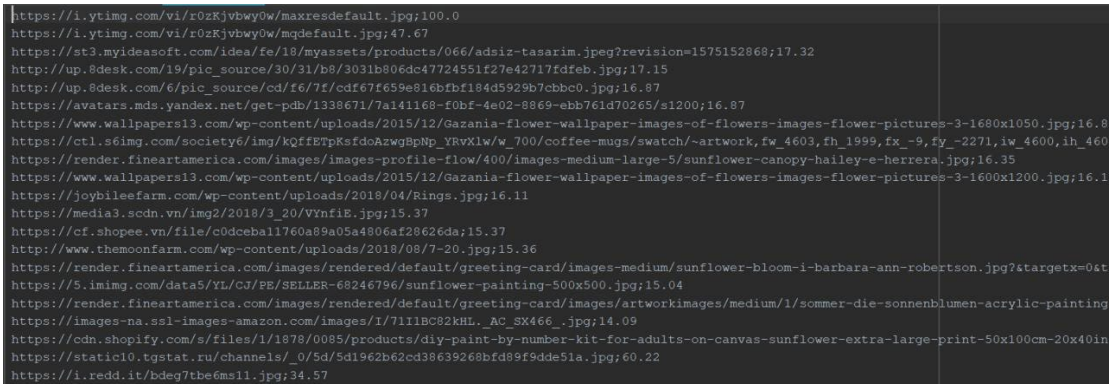


Рисунок 3.1.2 – фінальний екран з виведеними результатами по кількох зображеннях архіву



```
https://i.ytimg.com/vi/r0zKjvbwy0w/maxresdefault.jpg;100.0
https://i.ytimg.com/vi/r0zKjvbwy0w/mqdefault.jpg;47.67
https://st3.myideasoft.com/idea/fe/18/myassets/products/066/adsiz-tasarim.jpeg?revision=1575152868;17.32
http://up.8desk.com/19/pic_source/30/31/b8/3031b806dc47724551f27e42717fdfeb.jpg;17.15
http://up.8desk.com/6/pic_source/cd/f6/7f/cdf67f659e816bfbf184d5929b7cbbc0.jpg;16.87
https://avatars.mds.yandex.net/get-pdb/1338671/7a141168-f0bf-4e02-8869-ebb761d70265/s1200;16.87
https://www.wallpapers13.com/wp-content/uploads/2015/12/Gazania-flower-wallpaper-images-of-flowers-images-flower-pictures-3-1680x1050.jpg;16.8
https://ctl.s6img.com/society6/img/kQffETpKsfdoAzwgBpNp_YRvXlw/w_700/coffee-mugs/swatch/~artwork,fw_4603,fh_1999,fx_-9,fy_-2271,iw_4600,ih_460
https://render.fineartamerica.com/images/images-profile-flow/400/images-medium-large-5/sunflower-canopy-hailey-e-herrera.jpg;16.35
https://www.wallpapers13.com/wp-content/uploads/2015/12/Gazania-flower-wallpaper-images-of-flowers-images-flower-pictures-3-1600x1200.jpg;16.1
https://joybileefarm.com/wp-content/uploads/2018/04/Rings.jpg;16.11
https://media3.scdn.vn/img2/2018/3_20/VYnfiE.jpg;15.37
https://cf.shopee.vn/file/c0dceba11760a89a05a4806af28626da;15.37
http://www.themoonfarm.com/wp-content/uploads/2018/08/7-20.jpg;15.36
https://render.fineartamerica.com/images/rendered/default/greeting-card/images-medium/sunflower-bloom-i-barbara-ann-robertson.jpg?&targetx=0&t
https://5.imimg.com/data5/YL/CJ/PE/SELLER-68246796/sunflower-painting-500x500.jpg;15.04
https://render.fineartamerica.com/images/rendered/default/greeting-card/images/artworkimages/medium/1/sommer-die-sonnenblumen-acrylic-painting
https://images-na.ssl-images-amazon.com/images/I/71I1BC82kHL._AC_SX466_.jpg;14.09
https://cdn.shopify.com/s/files/1/1878/0085/products/diy-paint-by-number-kit-for-adults-on-canvas-sunflower-extra-large-print-50x100cm-20x40in
https://static10.tgstat.ru/channels/_0/5d/5d1962b62cd38639268bfd89f9dde51a.jpg;60.22
https://i.redd.it/bdeg7tbe6ms11.jpg;34.57
```

	1	2	3	4	5	6	7
1	https://i.y	100.0					
2	https://i.y	47.67					
3	https://st3	17.32					
4	http://up.	17.15					
5	http://up.	16.87					
6	https://av	16.87					
7	https://w	16.86					
8	https://ctl	16.59					
9	https://re	16.35					
10	https://w	16.19					
11	https://jo	16.0					
12	https://m	15.37					
13	https://cf.	15.37					
14	http://ww	15.36					
15	https://re	15.21					
16	https://5.i	15.0					
17	https://re	14.71					
18	https://im	14.0					
19	https://cd	13.99					
20	https://st	60.22					
21	https://i.r	34.57					
22	https://in	32.66					
23	https://ey	28.98					
24	https://i.i	26.52					

Рисунки 3.1.3 та 3.1.4 – результати зображень у csv файлі, вигляд у PyCharm та Microsoft Excel

3.2 Google Vision API

Для реалізації даного прототипу виникло одне дуже складне питання, яке необхідно було вирішити, - яким чином знаходити зображення, з якими порівнювати ті, унікальність яких хочуть перевірити користувачі(-чки). Для того,

щоб порівнювати зображення з іншими ефективно, необхідно знайти такі зображення, які є хоч трохи схожими на те, що є досліджуваним. Оскільки перевірка будь-чого на унікальність найчастіше ставить за мету перед собою перевірити відсутність/наявність збігів з тим, що є в вільному доступі в Інтернеті, то і для даного завдання було вирішено шукати схожі зображення саме там. Для цього найзручнішим способом виявилось використовувати одну з найефективніих систем інформаційного пошуку по зображенням – image retrieval у Google Images. Найскладнішим завданням цього проекту було забезпечення доступу від програми до Google Images, але рішення було знайдене – використання Google Vision API [9]. Саме цей інструмент є ефективним способом поєднання програми разом з системою пошуку по зображеннях Google. Суть полягає в тому, що за допомогою машинного навчання API дозволяє робити різні висновки щодо конкретного зображення, функціонал є дуже широким. Але для даного застосунку було використано функціонал пошуку в Google зображень, що є схожими на досліджуване.

```
def principal(link):
    os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = r'vision_key.json'
    client = vision.ImageAnnotatorClient()

    with io.open(link, 'rb') as image_file:
        content = image_file.read()
    image = vision.types.Image(content=content)
    response = client.web_detection(image=image)
    print(response)
    response = json.loads(MessageToJson(response, preserving_proto_field_name=True))
    webdetection = response["web_detection"]

    urls = []
    try:
        full_matching = webdetection["full_matching_images"]
        for obj in full_matching:
            urls.append(obj['url'])
    except:
        print("no full_matching")

    try:
        partial_matching = webdetection["partial_matching_images"]
        for obj in partial_matching:
            urls.append(obj['url'])
    except:
        print("no partial_matching")

    try:
        visually_similar = webdetection["visually_similar_images"]
        for obj in visually_similar:
            urls.append(obj['url'])
```

Рисунок 3.2.1 – метод principal, що повертаэ результати роботи Google Vision API

Google Vision API надає результати пошуку у трьох категоріях: fully matching, partially marching та visually similar. Серед цих трьох категорій завжди знаходяться і повністю ідентичні зображення, і частково змінені (якщо, наприклад, відбулися зміни характеристик зображень, розмір чи орієнтація), а також ті зображення, що є візуально схожими (такі, що могли бути змінені за кольором, яскравістю або ж, наприклад, перемальовані звідкись, тобто, візуально скопійовані). Саме тому API є високоефективним для того, щоб надати користувачу(-чці) доступ до зображень, що будуть схожими за різними принципами на ті, унікальність яких вони намагаються перевірити.

3.3 Порівняння зображень

Знайшовши схожі зображення, які доцільно перевірити на предмет збігу з тими, що досліджуються, постала проблема того, як саме порівнювати ці зображення. В розділі 2 було описано процес вибору алгоритму за яким необхідно порівнювати зображення на унікальність. Внаслідок цього аналізу було обрано використання вбудованого методу PSNR для бібліотеки OpenCV. Завдяки цьому способу можна доволі точно визначити схожість двох зображень. Крім того, після проведених експериментів, стало зрозуміло, що система здатна доволі адекватно оцінити ті зображення, що справді є ідентичними або фрагментарно-ідентичними, а також ті, що є лише візуально схожими, адже на них зображені ті самі предмети, але, все ж, зображення відрізняється.

```
def psnr(link1, link2):
    img1 = cv2.imread(link1)
    img2 = cv2.imread(link2)
    (height, width, c) = img1.shape
    img2 = cv2.resize(img2, (width, height))
    score = peak_signal_noise_ratio(img1, img2)
    if score == float("inf"):
        return 1.0
    else:
        return score / 51
```

Рисунок 3.2.2 – метод PSNR, що порівнює два зображення

Це є дуже важливим, адже Vision API на основі машинного навчання надає можливість отримати зображення, що містять той же об’єкт, що і початкове зображення, але аж ніяк не гарантує того, що зображення справді є ідентичним. Наприклад, це може бути фотографія того ж самого предмету, але зроблена з іншого ракурсу, при іншому освітленні. За таких умов, якщо система аналізуватиме такі дві фотографії, то коефіцієнт схожості буде невисоким і можна не вважати використання такого зображення плагіатом.

Розділ 4: Аналіз результатів

Результатом виконання роботи системи над певним зображенням чи архівом зображень є csv-таблиця, що містить інформацію про усі зображення та збіги з тими, що вдалося знайти через Google Vision API. Для користувачів(-ок) найкращим способом проглядання результатів програми є використання файлу csv, що можна відкрити у Microsoft Excel (Рисунок 3.1.4). В цьому файлі надається таблиця зі збігів, що відсортовані для кожного зображення, та коефіцієнту схожості, що пораховано через PSNR. Користувачі(-чки) можуть самостійно проаналізувати цей файл на предмет підозрілості певних збігів, адже відсоток унікальності, що викликає підозру, кожен може визначити для себе сам. Результати проведених досліджень показали, що система чудово знаходить 100% плагіат із завантаженими зображеннями, а також не плутає оригінальні зображення зі схожими, видаючи не дуже високий відсоток. Саме тому, система надає доволі точні результати та змогу пересвідчитись по посиланням на знайдені зображення, чи є вони унікальними в архіві, чи ні.

Висновки

При виконанні даної курсової роботи, можна з впевненістю сказати, що найскладнішим етапом розробки прототипу було налагодження зв’язку між алгоритмом та пошуком зображеннями в Інтернеті. Тому Google Vision API є одним з ключових знахідок під час виконання досліджень для даної роботи. Найцікавішою частиною дослідження був аналіз існуючих алгоритмів для порівняння двох зображень, доцільності їх використання для потреб даної системи, співвідношенні їх ефективності до часу виконання тощо. Вдалося ускладнити програму, наблизивши її до реалій аналізу на плагіат, адже просто зображення рідко насправді аналізують, а от щодо архівів – частий випадок, який навіть доводилося зустрічати у реальному житті. Також важливим було максимальне спрощення використання результатів, що надаються програмою, а саме створення csv-файлу з результатами, що надає можливість зручного перегляду знайдених збігів для недосвідчених користувачів(-ок). З нереалізованого: не вдалося реалізувати алгоритм перевірки .RAR архівів, який також є доволі часто використовуваним у питанні архівації загалом. Крім того, не вдалося ще краще підвищити дружність системи, адже у результуючому csv-файлі не вдалося створити автоматичне виділення результатів за певною шкалою (наприклад, понаднормовий відсоток плагіату). Загалом, система надає доволі точні результати, які дають змогу оцінити увесь архів на плагіат, адже аналізує усі зображення. Хоча б одне зображення може видати високий відсоток збігів. Далі, слово за тим, хто користується системою – чи вважати автора плагіатором, чи ні. Але, в будь-якому разі, система надає доволі широке розуміння унікальності досліджуваного архіву зображень та робить це достатньо швидко, це залежить від розміру архіву.

Додатки

1. Лістинг коду

Клас interface.py

```
import PySimpleGUI as sg
import vision
import csv

layout = [
    [sg.Text('Choose file'), sg.InputText(key="file"), sg.FileBrowse()],
    [sg.Submit(), sg.Cancel()]
]
window = sg.Window('Plagiarism check', layout)
while True:
    event, values = window.read()
    if event in (None, 'Exit', 'Cancel'):
        break
    if event == 'Submit' and (str(values['file']).endswith(".zip") or
str(values['file']).endswith(".jpg") or
                              str(values['file']).endswith(".tar") or
str(values['file']).endswith(".png") or
                              str(values['file']).endswith(".JPG") or
str(values['file']).endswith(".jpeg") or
                              str(values['file']).endswith(".JPEG") or
str(values['file']).endswith(".PNG")):
        layout = [[sg.MLine(key='-ML1-' + sg.WRITE_ONLY_KEY, size=(100, 30))],
                  [sg.Button('Exit')]]
        window = sg.Window('Result', layout, finalize=True)
        if str(values['file']).endswith(".zip"):
            vision.unpack_zip(values['file'])
            window['-ML1-' + sg.WRITE_ONLY_KEY].print(vision.check_img())

        elif str(values['file']).endswith(".tar"):
            vision.unpack_tar(values['file'])
            window['-ML1-' + sg.WRITE_ONLY_KEY].print(vision.check_img())

        else:
            f = open('result.csv', 'w')
            writer = csv.writer(f)
            ss = vision.principal(values['file'])
            s = ""
            for obj in ss:
                s += str(obj[1]) + "% " + str(obj[0]) + "\n"
                writer.writerow(obj)
            s += "\n"
            window['-ML1-' + sg.WRITE_ONLY_KEY].print(s)
```

Клас vision.py:

```
import os, io
import tarfile
import zipfile
from google.cloud import vision
from google.protobuf.json_format import MessageToJson
from skimage.metrics import peak_signal_noise_ratio
import cv2
import json
import requests
import glob
import csv


def alg_check(images, filepath):
    data = {}
    counter = 1
    for link in images:
        try:
            path = "C:/Users/Andrew/Desktop/UKMA/3 course/course/images/f" +
str(counter) + ".jpg"
            r = requests.get(link)
            with open(path, 'wb') as outfile:
                outfile.write(r.content)
            data[link] = round(psnr(filepath, path) * 100, 2)
            counter += 1
            print(counter)
        except:
            continue
    return data


def psnr(link1, link2):
    img1 = cv2.imread(link1)
    img2 = cv2.imread(link2)
    (height, width, c) = img1.shape
    img2 = cv2.resize(img2, (width, height))
    score = peak_signal_noise_ratio(img1, img2)
    if score == float("inf"):
        return 1.0
    else:
        return score / 51


def principal(link):
    os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = r'vision_key.json'
    client = vision.ImageAnnotatorClient()

    with io.open(link, 'rb') as image_file:
        content = image_file.read()
    image = vision.types.Image(content=content)
    response = client.web_detection(image=image)
    print(response)
    response = json.loads(MessageToJson(response,
preserving_proto_field_name=True))
    webdetection = response["web_detection"]

    urls = []
    try:
        full_matching = webdetection["full_matching_images"]
        for obj in full_matching:
```

```
            urls.append(obj['url'])
    except:
        print("no full_matching")

    try:
        partial_matching = webdetection["partial_matching_images"]
        for obj in partial_matching:
            urls.append(obj['url'])
    except:
        print("no partial_matching")

    try:
        visually_similar = webdetection["visually_similar_images"]
        for obj in visually_similar:
            urls.append(obj['url'])
    except:
        print("no visually_similar")

    files = glob.glob("images/*.*")
    for f in files:
        os.remove(f)

    stats = alg_check(urls, link)
    result = sorted(stats.items(), key=lambda x: x[1], reverse=True)
    return result


def check_img():
    s = ""
    f = open('result.csv', 'w')
    writer = csv.writer(f, delimiter=';', lineterminator='\n')

    files = glob.glob('archive_images/*.png') +
glob.glob('archive_images/*.jpg')+glob.glob('archive_images/*.jpeg')
+glob.glob('archive_images/*.tif')
    for file in files:
        res = principal(file)
        for obj in res:
            s += str(obj[1]) + "% " + str(obj[0]) + "\n"
            writer.writerow(obj)
        s += "\n"

    files = glob.glob('archive_images/*.*')
    for f in files:
        os.remove(f)
    return s


def unpack_zip(filepath):
    with zipfile.ZipFile(filepath, 'r') as zip_ref:
        zip_ref.extractall("archive_images/")


def unpack_tar(filepath):
    tar = tarfile.open(filepath, mode='r')
    tar.extractall("archive_images/")
```

Список використаних джерел

1. K. Dawson-Howe A Practical Introduction to Computer Vision with OpenCV, Enhanced Edition – с. 38-44
2. G. Bradski Learning OpenCV: Computer Vision with the OpenCV library / G.Bradski, A. Kaehler – с. 214-218
3. Документація бібліотеки PySimple GUI. Режим доступу: https://pysimplegui.readthedocs.io/en/latest/cookbook/
4. Документація opencv та приклад використання алгоритму PSNR. Режим доступу: https://docs.opencv.org/2.4/doc/tutorials/gpu/gpu-basics-similarity/gpu-basics-similarity.html
5. Документація opencv та роз'яснення щодо використання різних коефіцієнтів для зіставлення із шаблоном. Режим доступу: https://docs.opencv.org/master/d4/dc6/tutorial_py_template_matching.html
6. I. Culjak A brief introduction to OpenCV / I. Culjak, D. Abram, T. Pribanic, H.Dzapo, M. Cifrek. Режим доступу: https://mipro-proceedings.com/sites/mipro-proceedings.com/files/upload/sp/sp_008.pdf
7. J. Chao Performance Comparison of Various Feature Detector-Descriptor Combinations for Content-based Image Retrieval with JPEG-encoded Query Images / J. Chao, A. Al-Nuaimi, G. Schroth, E. Steinbach. Режим доступу: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.675.3707&rep=rep1&type=pdf
8. U. Sara Image Quality Assessment through FSIM, SSIM, MSE and PSNR—A Comparative Study / U. Sara, M. Akter, M. Shorif Uddin. Режим доступу: https://www.scirp.org/journal/paperinformation.aspx?paperid=90911
9. Google Vision API. Режим доступу: https://cloud.google.com/vision