3. Edublog. [Ел. ресурс]. − Режим доступу: http:// https://edublog.com.ua/blog/id1306488505/posts/moi-publikatsii-2f746b46-46e4-4d5d-917b-a2c7c36a3b41/istoriya-zastosuvannya-smart-tekhnolohiy-v-osvitnomu-protsesi
4. Yesina, O., Smart-tehnologiyi yak zasib vdoskonalennya osvitnogo procesu LypskaLarysa [Smart technologies as a means of improving the educational process], Кафедра економічної кібернетики та інформаційних технологій, 2021

Зеленкевич С.П.,
здобувач другого (магістерського) рівня вищої освіти
Тернопільський національний педагогічний університет ім. В. Гнатюка
zelenkevych_sp@fizmat.tnpu.edu.ua
Ящик О. Б.,
Кандидат педагогічних наук, доцент кафедри комп'ютерних технологій
Тернопільський національний педагогічний університет ім. В. Гнатюка,
м. Тернопіль, Україна

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МОВОЮ C# ДЛЯ ПЕРЕВІРКИ НАУКОВИХ РОБІТ НА ПЛАГІАТ

Зростаюча кількість академічних та наукових робіт в сучасному світі створює потребу в ефективних засобах контролю за оригінальністю текстового матеріалу. Тим більше часто виникає складність щей через те, що популярні сервіси для перевірок працюють на власних серверах, що спричиняє достатньо великі черги та затримки перевірок, а відповідно і результатів. Також якщо університет має власну базу наукових робіт то для того щоб уникнути розповсюдження цих матеріалів на просторах інтернету використовуються програми, а не онлайн сервіси які здебільшого є платними. У зв'язку з цим, розробка програмного забезпечення для виявлення плагіату стає надзвичайно актуальною задачею. У даній анотації розглянуто написання програми антиплагіату мовою програмування C#.

На сьогоднішній день існують різноманітні антиплагіатні системи, такі як Turnitin, Plagscan, Unicheck та інші. Багато з цих систем пропонують платні плани для користувачів і обмеженим функціоналом у безкоштовній версії таким як невеликий обсяг тексту який може бути перевірений з одного документу, або обмежену кількість слів на тиждень або місяць. Такі системи мають свої переваги, проте їхні вартості можуть бути недосяжними для окремих користувачів, або коли навчальний заклад не має змоги платити за численні перевірки від кожного студента в такий важкий період як зараз. Також варто враховувати розповсюдженість російських аналогів антиплагіат сервісів та програм які були більш доступні як у ціні так і у локалізації, часто використовувались для попередніх перевірок робіт. Зараз ж це являється недопустимим адже такі програми можуть дуже ефективно збирати персональні дані студентів які реєструватимуться для перевірок своїх робіт або просто в процесі перевірок ці програми будуть зберігати копії робіти на віддалених серверах.

Звісно не можна не згадати Антиплагіат програми які існують на українському ринку також, такі як "Antiplagiat", "eTXT Antiplagiat" та інші. Вони часто спеціалізуються на потреби українських користувачів та можуть мати свої унікальні особливості, такі як підтримка української мови та адаптація до особливостей української наукової спільноти.

Які ж все таки переваги можна отримати, створивши власну антиплагіат програму. По-перше, це дає можливість повного контролю над функціоналом та алгоритмами програми. Крім того, власна програма може бути адаптована до конкретних потреб, включаючи специфічні особливості мови чи предметної області. Власна програма також може бути економічно вигідною альтернативою, оскільки уникне необхідності платити за підписку на комерційні антиплагіатні системи. Достатньо буде просто доручити відповідним викладачам займатись підтримкою даної програми, виправленням багів та удосконаленнню перевіряючих алгоритмів.

Отже для початку ми вирішили створити програму яка може відкривати та зчитувати текст з *.pdf та *.docx форматних файлів. Я провів невеликі дослідження і знайшов декілька

безкоштовних бібліотек для такої роботи які дають можливість витягувати посторінково текст з документів, бо більшість програм дають урізаний функціонал або вимагають використання ключа для роботи з їхніми наборами із засобів розробки (SDK). Тому я продовжу пошук оптимальних рішень та покращення цієї частини роботи програми.

Далі ми почали створення алгоритм для пошуку та порівняння схожостей у тексті. Провівши невеликі пошуки в інтернеті стало зрозуміло що кожна антиплагіат система має власні інноваційні алгоритми для якісного, швидкого та правильно пошуку схожостей і навіть виявлення побуквенного проценту співвідношення в тих чи інших групах слів але деталі роботи цих алгоритмів є власністю компанії і не розголошуються.

Тому ми почали з основ, звичайної перевірки тексту по заданому діапазону слів підряд, що витрачає дуже багато часу. Після чого цей алгоритм було трішки покращено і тепер перевірка відбувається з урахуванням перевірок на присутність лапок в тексті адже так позначаються цитати. Також було додано на перевірку завершення речення, тобто якщо у для прикладу 7 словах було знайдено крапку після 3 слова то береться не тих 7 слів, а 7 слів до крапки щоб перевірити чи в цьому закінченні речення є присутні певні збіжності (це звісно теж трішки зробило перевірку трішки довшою адже її складність збільшилась).

Далі для знаходження співвідношення схожості було додано лічильники які визначають який процент від загального обсягу слів документа є присутнім в тому з яким було порівняно його і який процент від загального обсягу роботи це становить. Це дало певну зрозумілість у тому який процент роботи складає плагіат, а який ні. Після обробки ці дані записуються у звичайний текстовий документ який зберігається з тією ж назвою що і документ який було завантажено для перевірки.

Після чого було прийнято рішення зробити так щоб програма після запуску по таймеру кожні 5 хвилин перевіряла вхідну папку на те чи не з'явилися нові документи для перевірки. А самі перевірки запускати в потоках для кращої оптимізації та розпаралелення роботи.

Далі ми повернулись до оптимізації роботи самої перевірки на схожість і використали систему хеш-сум для скорочення часу обробки тексту, для цього потрібно взяти частину тексту заданої довжини слів, видалити з неї всі розділювальні знаки і перевести її у нижній регістр, після чого перевести у набір байт, та вирахувати хеш-суму обох стрічок і порівняти її, у випадку збігу цих сум, йде безпосередня перевірка самих слів на співпадіння адже у рідкісних випадках хеш-сума може збігатись але текст буде різним.

Для контролю перевірених документів, програма має файли регістру в яких збережено назви документів які були оброблені, підрахунок часу який було витрачено та чи не було тих чи інших проблем під час перевірок файлів, щоб уникнути падіння та зберігати помилки програми, важливі частини коду обернено try - catch обгортками;

Тепер перейдемо безпосередньо до огляду програми яку було створено нами в процесі написання магістерської роботи. Сама програма має доволі простий інтерфейс:

З самого інтерфейсу можна зрозуміти що можна вказувати такі налаштування як кількість слів підряд для пришвидшення перевірок, чи потрібно враховувати та перевіряти текст у лапках, також вказується шлях до папки у яку будуть завантажені файли робіт які в свою чергу студенти будуть завантажувати у мудлі по відповідному посиланні. Та вихідний шлях куди будуть збережені папки з вихідними файлами та Log.txt файл.

Далі слідує група “Список папок з якими перевіряти” де можна вказати усі папки з базою робіт з якими будуть порівнюватись студенські роботи, та завдяки чекбоксам біля шляху до папок, можна вмикати та вимикати ті чи інші папки, щоб покращити доступність і вам не було потреби постійно шукати папки і вказувати ті чи інші.

Також є присутній попередній перегляд вмісту вибраної папки, під час вибору тієї чи іншої папки зі списку у полі праворуч можна переглянути що ця папка містить, зокрема там відображаються файли з розширенням *.pdf, *.docx. адже з такими файлами і відбувається перевірка.

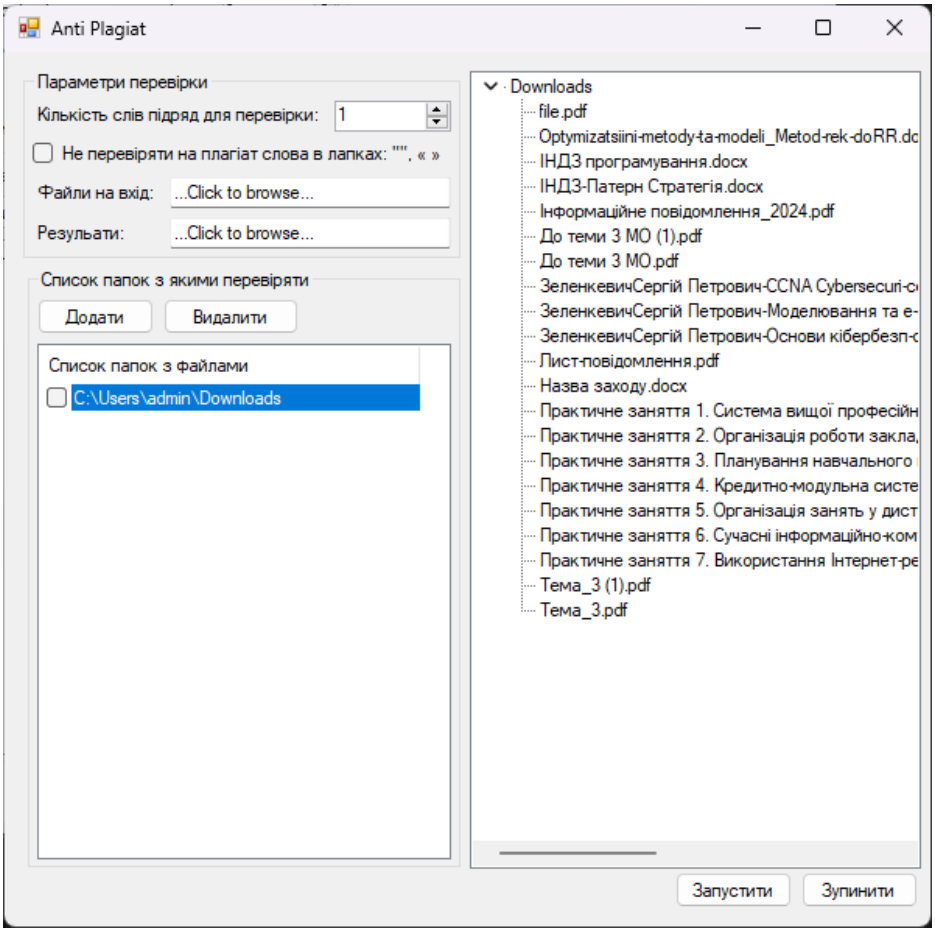

Рисунок 1. Інтерфейс програми.

Ну і звісно основні кнопки “Запустити” та “Зупинити” адже сама програма працює як фонова, тобто її можна запусти на основній машині-сервері де знаходиться безпосередньо база робіт і вона у фоновому режимі буде перевіряти вказану директорію на наявність нових файлів, щоб почати перевірку, це дає змогу одноразово її запустити під час початку процесу перевірки і по завершенню вимкнути її.

Отже розробка власної антиплагіатної програми є актуальною та перспективною задачею в контексті потреб сучасного освітнього середовища. Як було сказано вище зростаюча кількість студентів та науковців збільшує потребу в ефективних інструментах для контролю за оригінальністю їхніх робіт, а існуючі антиплагіатні системи часто мають високі вартості або обмежений функціонал, що робить їх важко доступними. Також важливо враховувати особливості української мови та наукового середовища, для яких може бути потрібна спеціалізована підтримка. Таким чином, розробка власної антиплагіатної програми може забезпечити ефективний інструмент контролю за плагіатом, адаптований до потреб українських користувачів, знижуючи вартість та збільшуючи доступність такого програмного забезпечення.

Список використаних джерел:

1. Учасники проектів Вікімедіа. Хеш-функція – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Хеш-функція.
2. GitHub: Let’s build from here. GitHub. URL: https://github.com/.
3. Stack Overflow - Where Developers Learn, Share, & Build Careers. Stack Overflow. URL: https://stackoverflow.com/.
4. W3Schools Online Web Tutorials. W3Schools Online Web Tutorials. URL: https://www.w3schools.com/.