МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна магістерська робота

на тему: Дослідження та розроблення програмного забезпечення для аналізу на унікальність текстів та кодів

Виконала: студент групи МгІТ-1-22
спеціальності 122 Комп'ютерні науки
освітньої програми Комп'ютерні науки
Володимир ГЛУЩЕНКО
Керівник:
к.т.н., доц. Тетяна АСТІСТОВА

Рецензент _____
д.т.н., проф. Володимир Щербань

Київ 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій

Кафедра комп'ютерні науки

Спеціальність 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерні науки

_____Володимир ЩЕРБАНЬ

«_____»_____2023_____року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТА

Глущенко Володимира Валерійовича

1.Тема роботи: Дослідження та розроблення програмного забезпечення для аналізу на унікальність текстів та кодів,

науковий керівник роботи: Астістова Тетяна Іванівна, к.т.н., доц.,

затверджені наказом закладу вищої освіти від 12 . 09.2023 року , № 210-уч.

2. Строк подання студентом роботи 12.11. 2023р.

3. Вихідні дані до роботи:

Розробка кафедри комп'ютерних наук

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)

РОЗДІЛ 1. Аналіз предметної області; РОЗДІЛ 2.Алгоритми та методи виявлення плагіату; РОЗДІЛ 3. Розробка та практична реалізація системи. Додатки - програмні коди модулів системи.

5.Консультанти розділів кваліфікаційної роботи

Розділ	Ім’я, прізвище та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Вступ	Тетяна АСТІСТОВА, к.т.н., доц.		
Розділ 1	Тетяна АСТІСТОВА, к.т.н., доц.		
Розділ 2	Тетяна АСТІСТОВА, к.т.н., доц.		
Розділ 3	Тетяна АСТІСТОВА, к.т.н., доц.		
Висновки	Тетяна АСТІСТОВА, к.т.н., доц.		

1. Дата видачі завдання: 08. 2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної магістерської роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	20.08.2023	
2	Розділ 1 Аналіз предметної області	10.09.2023	
3	Розділ 2. Алгоритми та методи виявлення плагіату	5.10.2023	
4	Розділ 3. Розробка та практична реалізація системи	25.10.2023	
5	Висновки	28.10.2023	
6	Оформлення кваліфікаційної магістерської роботи (чистовий варіант)	31.10.2023	
7	Здача кваліфікаційної магістерської роботи на кафедру для рецензування	01.11.2023	
8	Перевірка кваліфікаційної магістерської роботи на наявність ознак плагіату	04.11.2023	
9	Подання кваліфікаційної магістерської роботи на затвердження завідувачу кафедри	07.11.2023	

Студент _____ Володимир ГЛУЩЕНКО

Науковий керівник роботи _____ Тетяна АСТІСТОВА

Директор НМЦУПФ _____ Олена ГРИГОРЕВСЬКА

АНОТАЦІЯ

Глущенко В.В. Дослідження та розроблення програмного забезпечення для аналізу на унікальність текстів та кодів.

Кваліфікаційна робота за спеціальністю 122 - «Комп'ютерні науки». – Київський національний університет технологій та дизайну, Київ, 2023 рік.

Темою кваліфікаційна роботи є розробка системи для аналізу тесту та програмного коду з метою надання інформації про можливість плагіату.

В роботі проведено аналіз існуючих сервісів та технологій порівняння тексту та кодів; розглянуто алгоритми та методи виявлення плагіату як в тестових документах так і в програмних кодах; розглянута класифікація підходів комп'ютерного виявлення схожості контенту; розроблено алгоритми та методи виявлення плагіату відповідно до вибраних критеріїв; створена програмна реалізація роботи цих алгоритмів; розроблено веб-систему для аналізу програмного коду.

Для створення програмного продукту було обрано: середовище програмування Visual Studio Code; мову програмування JavaScript; гіпертекстову розмітку - HTML; стиль сайту було створено при використанні каскадних таблиць стилю CSS.

Розробники програмних додатків та програмного забезпечення, працівники сфери IT, студенти, кампанії, викладачі, працюючи з розробленою веб - системою, зможуть ефективно перевіряти оригінальність свого коду та запобігати випадкам плагіату.

Ключові слова: веб - орієнтована система, регулярні вирази, алгоритми, ASCII, JavaScript, HTML, CSS, Visual Studio Code , Text-based, Metrics-based, Token-based.

ANNOTATION

Glushchenko V.V. Research and development of software for the analysis of the uniqueness of texts and codes.

Qualifying master's thesis in specialty 122 - "Computer science". - Kyiv National University of Technology and Design, Kyiv, 2023.

The topic of the qualifying master's thesis is the development of a system for analyzing the test and program code in order to provide information about the possibility of plagiarism.

The paper analyzes existing services and technologies for comparing text and codes; algorithms and methods of plagiarism detection both in test documents and in program codes are considered; the classification of approaches to computer detection of content similarity is considered; algorithms and methods of plagiarism detection were developed in accordance with the selected criteria; the software implementation of these algorithms was created; a web system for software code analysis was developed.

To create a software product, the following were chosen: Visual Studio Code programming environment; JavaScript programming language; hypertext markup - HTML; The site style was created using cascading CSS style sheets.

Developers of software applications and software, IT workers, students, campaigns, teachers, working with the developed web system, will be able to effectively check the originality of their code and prevent cases of plagiarism.

Keywords: web-oriented system, regular expressions, algorithms, ASCII, , JavaScript, HTML, CSS, Visual Studio Code, Text-based, Metrics-based, Token-base

ЗМІСТ

ВСТУП

Актуальність проблеми. З розвитком Інтернету стрімко зросла потреба в можливості перевірки тексту на унікальність. Виявити схожість частин тексту запозичених в своїй роботі з Інтернету потрібно як науковим робітникам, викладачам, аспірантам, студентам, так і власникам різних видавництв , авторам текстів, щоб переконатися в абсолютної оригінальності своєї роботи.

На сьогодні велику увагу приділяють принципам академічної доброчесності.

Академічна доброчесність - це сукупність етичних принципів та визначених законом правил, якими мають керуватися учасники освітнього процесу під час навчання, викладання та провадження наукової (творчої) діяльності з метою забезпечення довіри до результатів навчання та/або наукових (творчих) досягнень» [«Закон України про освіту (стаття 42)»].

Розвиток інформаційних технологій породив стрімке зростання розробок програмних додатків, веб-сайтів та інших програмних продуктів у всіх галузях нашого життя. Зростання кількості створення програмних кодів породило ще одину проблему в цьому напрямку - це проблема плагіату в програмному коді. Плагіат в програмуванні, це загроза якості програмних продуктів, порушення прав автора і все це може мати серйозні наслідки для розробників і організацій при роботі.

Дослідити існуючі сервіси порівняння тексту і програмного коду, розробити варіанти, які б перейняли переваги та врахували недоліки існуючих методів та алгоритмів пошуку запозичених фрагментів до порівняння є метою роботи.

Мета дослідження. Дослідження та розробка програмного забезпечення для аналізу на унікальність текстів та програмних кодів**.**

Завдання дослідження. Дослідити існуючі сервіси аналізу тексту та програмного коду і розробити варіант, який би перейняв переваги та врахував недоліки існуючих сервісів та програм такого типу на ринку

України ; реалізувати сучасні методи аналізу коду для пошуку ознак плагіату; створити ефективний веб-інтерфейс для взаємодії користувачів з веб – системою..

Об'єкт дослідження. Основним об'єктом дослідження є система опрацювання існуючих сервісів порівняння тексту та розробка алгоритму підготовки тексту до порівняння при максимально допустимих об'ємів даних та розробка веб - системи для аналізу програмного коду на плагіат, яка забезпечує можливості виявити «запозичені частини роботи».

Предмет дослідження кваліфікаційної магістерської роботи є область пошуку системи, яка включає колекцію повних текстів авторефератів і дисертацій, колекцію юридичних , нормативних і документів та огляд аналогів веб-системи для аналізу програмного коду, проведення аналізу засобів розробки та обраного середовища розробки.

Методи дослідження. Одним із методів дослідження системи перевірки контексту є ядро системи, що використовує унікальні алгоритми, який забезпечує швидкий і ефективний пошук запозичених фрагментів.

Система аналізу програмного коду буде використовувати три основні методи аналізу для виявлення плагіату, а саме: метод text-based, метод metrics-based та token-based метод , а розроблені блок - схеми методів та їх програмування, дозволять створити програму у вигляді веб - системи для поставленої задачі .

Елементи наукової новизни. Даний програмний сервіс відрізняється більш точними алгоритмами пошуку співпадінь, що гарантує більш високу якість порівняння.

Наукова новизна та практичне значення. В кваліфікаційній магістерській роботі система для оцінки контексту на плагіат дозволяє оптимізувати шляхи зрівняння, що в свою чергу дасть більш високу швидкість роботи та отримати результат.

При розробки веб - системи для оцінки програмного коду були розроблені програмні коди реалізації всіх трьох методів : text-based, метод

metrics-based та token-based метод. Розробники програмних додатків та програмного забезпечення, працюючи з розробленою веб-системою, зможуть ефективно перевіряти оригінальність свого коду та запобігати випадкам плагіату.

Веб-система завдяки запобіганню поширення плагіату підвищить якість програмного продукту.

Результати роботи були опубліковані в наступних виступах та статтях:

1. Астістова Т. І. Дослідження сервісів для аналізу інформації на унікальність / Т. І. Астістова, В. В. Глущенко // Інформаційні технології в науці, виробництві та підприємництві : збірник наукових праць молодих вчених, аспірантів, магістрів кафедри комп'ютерних наук та технологій / за заг. наук. ред. В. Ю. Щербаня. – Київ : ТОВ "Фастбінд Україна", 2023. – С. 98-101

https://er.knutd.edu.ua/handle/123456789/24119

2. Астістова Т.І, Аналіз сервісів для порівняння тексту та кодів/ Т. І. Астістова, В. В. Глущенко // Тези II Міжнародної науково-практичної Інтернет конференції молодих учених та студентів: «Електромеханічні, інформаційні системи та нанотехнології»- Київ, КНУТД, 20 квітня 2023- С.87-89

https://er.knutd.edu.ua/handle/123456789/20650

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Постановка задачі

Академічна доброчесність - це сукупність етичних принципів та визначених законом правил, якими мають керуватися учасники освітнього процесу під час навчання, викладання та провадження наукової (творчої) діяльності з метою забезпечення довіри до результатів навчання та/або наукових (творчих) досягнень [Закон України про освіту (стаття 42)].

Сьогодні створена велика кількість алгоритмів перевірки текстів на оригінальність. Визначити які з них кращі вкрай важко.

«Виявлення плагіату або визначення схожості вмісту – це процес знаходження місць плагіату чи порушення авторських прав у творі чи документі.»

Поява Інтернету та використання комп'ютерів сприяли поширенню плагіату. Процес знаходження місць плагіату чи порушення авторських прав у творі чи документі називають виявленням плагіату або визначення схожості вмісту.

Унікальність тексту є основним параметром, що визначає його несхожість з іншим текстом і вказується у відсотках. Чим вищі відсотки, тим унікальнішим є текст. Існують різні критерії оцінки унікальності в залежності від типу тексту для публікації. Так для публікації на сайтах унікальність тексту повинна становити не менше 90%. Для наукових робіт (курсових, дипломних) унікальність повинна становити вище 70%. Проте, кожна тема має свої особливості. Наприклад, курсова чи дипломна робота на юридичну тематику апріорі не може бути на 100% унікальною через посилання на закони, які іншими словами переписати не можна.

Для того щоб перевірити роботу на плагіат існує чимало підходів та програм:

1. Вияв плагіату людиною є найбільш традиційною формою його виявлення. Це може бути тривалим і трудомістким завданням для читача [2], а також може призвести до неузгодженості в тому, як ідентифікується плагіат

в організації.

2. Програмне забезпечення для виявлення плагіату або як його ще називають «програмне забезпечення для боротьби з плагіатом», стало широко доступним у формі як програмного забезпечення з відкритим кодом так і комерційно доступних продуктів. Воно фактично не відображає плагіат, але містить певні фрагменти тексту в одному документі, які відповідають тексту в іншому документі.

Антиплагіат — це перевірка текстів за складними алгоритмами.Вони розбивають текст на молекули, розглядають його під мікроскопом і збирають назад. Виявляються збіги за словами, словосполученням, цілим реченням і так далі.[1]/

Алгоритми програм та сервісів побудовані типово:

- Документ транслюється в тексту форматі .txt або doc. та перевіряється. Перевірка та пошук співпадінь виконується пошинглах (методом шингл–розбивки тексту). Шингл, це структурно-логічний фрагмент тексту, що складається з послідовності декількох слів.
- Пошук в Інтернеті здійснюється декількома пошуковими системами.
- В результаті візуалізується відсоток оригінальності тексту та список сайтів з відсотком збігу у відповідному кольорі в залежності від пошукових серверів.

Аналізується також послідовність викладу, структура, загальний хід розповіді. В результаті програма на раз виявляє не тільки повністю скопійований матеріал, але і недбало перероблений. Замінити кілька слів у реченні і переставити їх місцями не вийде. Алгоритм все одно знайде збіги і видасть низький бал. Алгоритми програм антиплагіату постійно вдосконалюються. Будь-яку роботу з перевірки на унікальність тексту, потрібно почати з вибору сервісу та програми перевірки.

Справа в тому, що різні системи можуть видавати абсолютно різні оцінки одному і тому ж матеріалу.

1.2. Огляд сервісів перевірки унікальності текстів.

1.2.1.Коштовні програми.

Unicheck

Популярний сервіс перевірки унікальності текстів (див рис.1), який орієнтований усім хто працює з текстами, — маркетологам,SMM-щикам, копірайтерам тощо. Unicheck популярний у США, Латинській Америці, Іспанії, Бельгії, Австралії, Україні та інших країнах.

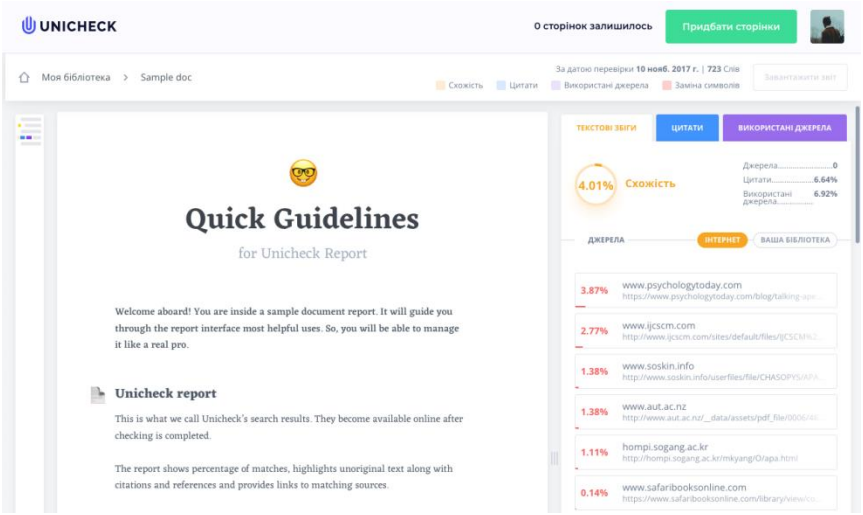


Рисунок 1.1 - Сервіс пошуку плагіату Unicheck

Особливістю Unicheck є бібліотека файлів користувача. Ці файли зберігаються в Особистому кабінеті із збереженими результатами перевірки. Користувач может отримати в будь-який час доступ до них, використовувати пошук за назвами, за датою, та інше.

Великою перевагою Unicheck є можливість додавати файли не тільки копіюванням і вставкою або завантаженням файлу з комп'ютера. Сервіс вміє працювати з хмарними сховищами Google Drive, Dropbox та One Drive.

Copyscape

Сервіс дозволяє перевіряти на унікальність цілі сторінки сайту. Користувачу потрібно тільки вставити URL сторінки, що його цікавить, і натиснути Go(рис.2). Інтерфейс сервісу доволі простий.

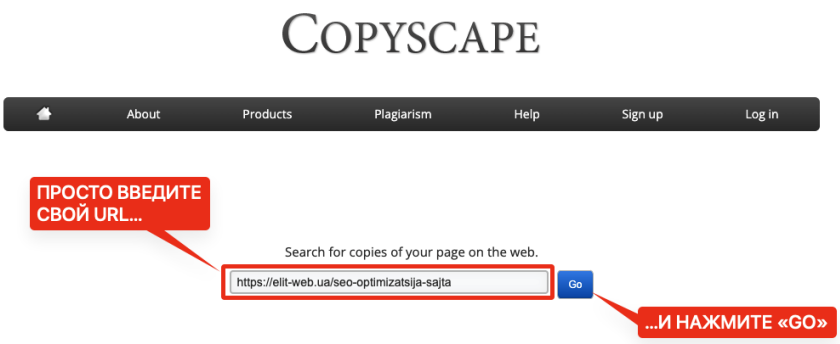


Рисунок 1.2 - Сервіс пошуку плагіату Copyscape

Після натиску Go, через кілька секунд можна буде побачити список результатів пошуку, що нагадує видачу Google. У ній містяться сторінки, де знайдені збіги з сторінками сайту які перевірялись.

Зауважимо, що далеко не завжди результати виявляються коректними. Так, наприклад, під час аналізу сторінки з описом послуги SEO-просування Elit-Web ми побачили у списку наш сайт, сайт іншої компанії зі схожими послугами та кілька абсолютно нерелевантних сторінок — «Добавки до бетону», «Тести на спорідненість» та інше. Ймовірно, варто звертати увагу лише на перші результати.

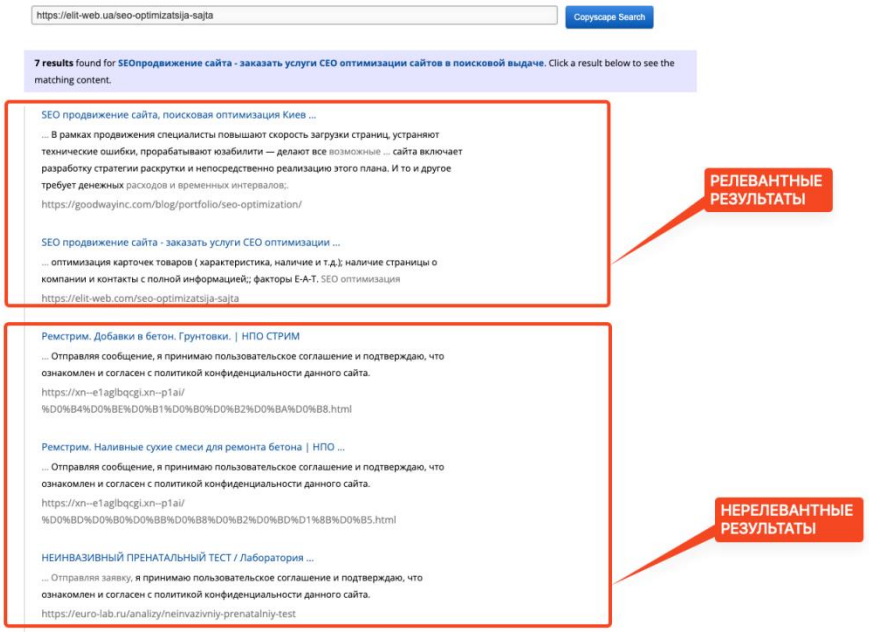


Рисунок 1. 3 - Результат роботи сервісу пошуку плагіату Copyscape

По кожному з результатів (рис.1.3.) ви можете натиснути на відповідний

текст і відкриється відповідна сторінка, на якій кольором будуть виділені частини тексту, які система вважала не унікальною. Тут же вказано кількість слів на сторінці та ступінь схожості. (15% збігів).

Недоліком сервісу Copyscape є перевірка всієї сторінки, а не окремого тексту. Перевіряти можуть пункти меню, контактні дані, футер, та інших елементи сторінки, які насправді не потрібно враховувати

Plagiarismcheck.org.

Цей сервіс пошуку плагіату більш орієнтований на студентів та викладачів. Але й інші користувачі, які працюють з текстом, можуть його застосовувати. Новим користувачам доступна одна безкоштовна перевірка. Для подальшого використання Plagiarismcheck доведеться платити — від $5,99 за 20 сторінок

Процес перевірки є стандартним. Ви копіюєте свій текст у поле для введення або завантажуєте текстовий файл, після чого натискаєте Proceed і система починає перевіряти документ на унікальність.

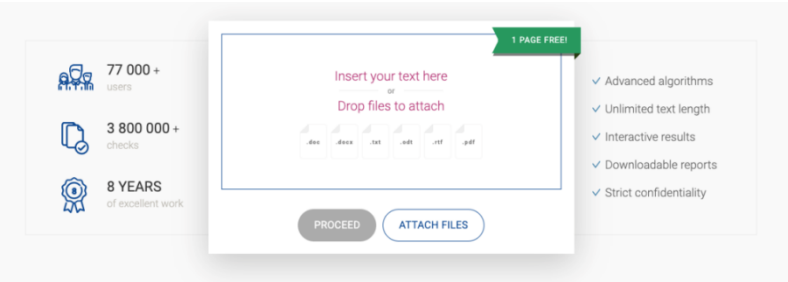


Рисунок 1.4 - Сервіс пошуку плагіату Plagiarismcheck.org

Усі тексти, що перевіряються, зберігаються в бібліотеці вашого облікового запису разом з результатами. Ви можете в будь-який момент повернутися до них.

Результати перевірки можна завантажити у форматі PDF, роздрукувати або одразу надіслати Email. Загалом Plagiarismcheck — непоганий сервіс, але може виявитися досить дорогим, якщо ви регулярно перевіряєте тексти на

унікальність та ще й по кілька разів у міру написання, редагування та коректури.

Plagiarisma

Досить простий сервіс, який проте підтримує перевірку текстів більш ніж 190 мовами. Працює Plagiarisma дуже інтуїтивно. Треба вставити текст або текстовий файл більш ніж у десяти форматах у порожнє поле та натискнути Check Duplicate Content. Можна також перевірити сторінку URL.

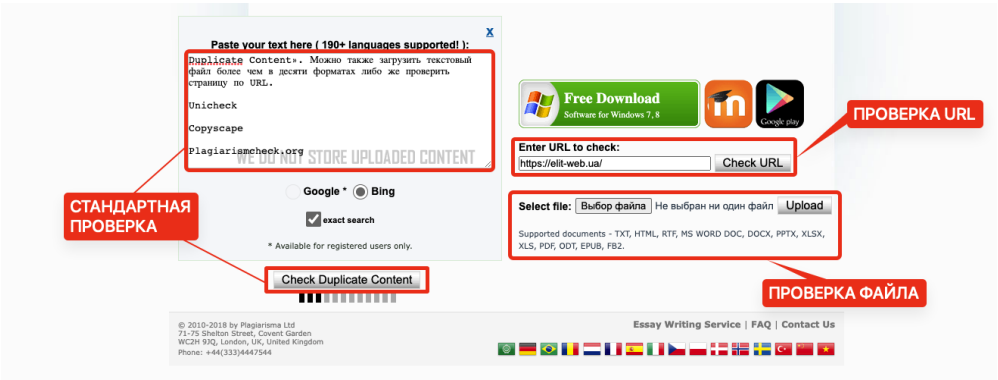


Рисунок 1.5 - Сервіс пошуку Plagiarisma

Переваги Plagiarisma:

• простий та мінімалістичний інтерфейс;

• додаток для Windows та Android;

• можливість шукати збіги лише у Google або Bing на вибір.

Але є кілька причин не рекомендувати Plagiarisma:

1. Сервіс не оновлювався з 2018 року.

2. На сайті не встановлено сертифікат безпеки, що несе загрозу для ваших даних.

3. Під час перевірки тексту Plagiarisma показує унікальність 0 % та пропонує скористатися послугою написання есе.

Можна зробити висновок, що Plagiarisma або більше взагалі не працює, або сервіс перетворили на майданчик для викачування грошей із користувачів.

Plagiarism Checker by EduBirdie

EduBirdie - популярний сервіс, для перевірки ткстів на унікальність. Перевірка працює он-лайн прямо на сайті.

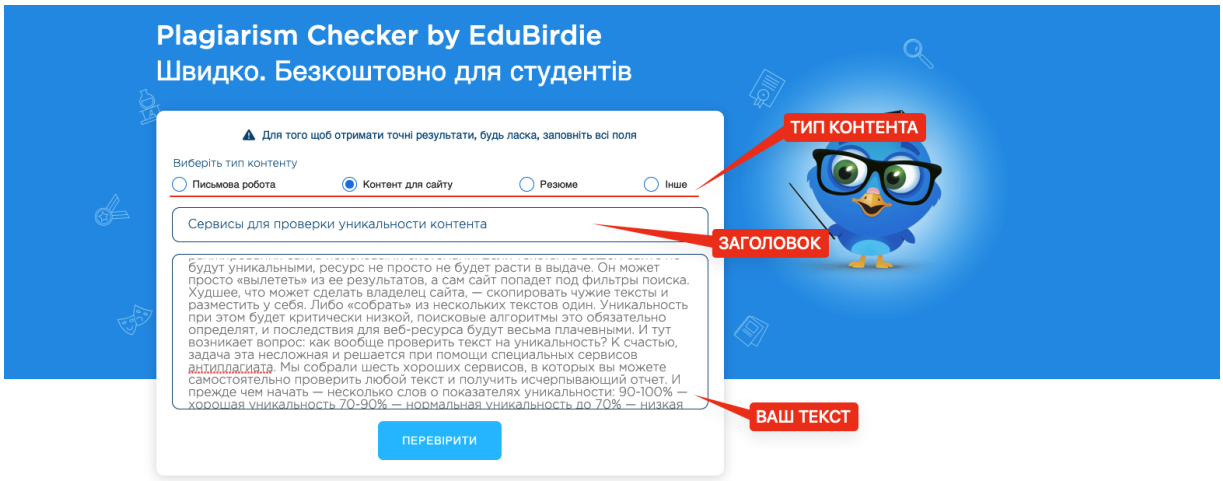
Для роботи з програмою необхідно:

1. Вибрати тип контенту: письмова робота, заповнення для сайту, резюме або інше (необов'язково).

2. Вставити заголовок (у відповідне поле) заголовок. Але це необов'язково.

3. Вставити основний текст (копіювати безпосередньо у текстове поле або завантажити файл у форматі .doc, .docx або .txt.)

4. Натиснути «Перевірити» та отримати результат перевірки.



Інформація про унікальність тексту буде виведена у відсотках, а не унікальні частини будуть виділені кольором.

Plagiarism Checker доступний українською мовою.

Крім оглянутих програм перевірки текстових документів на плагіат, які є коштовними, існують і безкоштовні програми. Розглянемо деякі з них.

.

1.2.2. Безкоштовні програми:

EtxtАнтиплагиат

Програм призначена для пошуку плагіату в мережі та оцінки унікальності текстів; сервіс перевірки текстів на Унікальність доступний як програмне забезпечення для установки на персональний комп'ютер, так і у форматі онлайн ресурсу. Можливо здійснювати перевірку текстових фрагментів,

окремих файлів та пакетів файлів, а також сторінок сайтів.

Передбачено2 функції перевірки тексту: *на плагіат*(наявність дослівних збігів) за допомогою опції «Метод виявлення копій», на *наявність рерайта* (смислових збігів) – опція «Метод виявлення рерайта».

Недоліки: он-лайн перевірка є безкоштовною з обмеженням до 3000 знаків без реєстрації до 5000 знаків після реєстрації

AdvegoPlagiatus

AdvegoPlagiatus -програма пошукув мережі Інтернет часткових або повних копій текстового документа, що показує ступінь унікальності тексту, список джерел тексту, відсоток збігу текстів. Також доступна перевірка унікальності за URL адресою. Найновіша версія AdvegoPlagiatus1.3.0.92 в стадії публічного бета-тестування. Платформа системи розрахована для Windows 8, Windows 7, Windows Vista, Windows XP, Windows 2000.

Недоліки: вимагає ручних налаштувань, обмеженість до 100000 символів.

Текст, який перевіряється у AdvegoPlagiatus, ділиться на фрази, а потім відбувається пошук подібних фраз в Інтернеті через пошукові сервери. Якщо фраза знайдена в Інтернеті, то вона вважається неоригінальною й підсвічується жовтим кольором.

Praide Unique Content Analyser II

Praide Unique Content Analyser II (http://best-soft.npps.biz/files.php?f=259) програма для перевірки тексту на унікальність в мережі Інтернет щляхом розбору вихідних даних на пасажі або шингли. Але збільшення параметрів, розмірів тексту і менше розмір шинглу, що задають користувач, тим довший час пошуку.

Програма має наступні переваги: три варіанти введення вихідних даних(по url, з файлу або фрагмент тексту з буфера обміну), розбір вихідних даних на пасажі або шингли, а також можливість вибору пошукових машин користувачем.

Он-лайн перевірка не можлива, програму треба скачати і установити на комп'ютер**.** Підсумковий звіт не містить статистичних даних, а лише список

сайтів, де знайдено подібні пасажі.

Недоліки: занадто довгий час на виконання перевірки.

Cognitive Text Analyzer

У систему вводиться досліджуваний текст. Далі Cognitive Text Analyzer проводить синтаксичний, семантичний і структурний розбір і аналіз тексту, усуває синтаксичні і семантичні неоднозначності і виявляє близько сотні характеристик, властивих даному тексту. Точна кількість характеристик залежить від обсягу тексту і різноманітності конструкцій, що зустрічаються в тексті. В порівняні з іншими програмами, ця програма працює дуже швидко.

Недоліки: вимагає «ідеального» інтернет-з'єднання, працює адекватніше з англомовними, ніж з україномовними текстами. Cognitive Text Analyzer враховує структурні зв'язки, підлеглість одних конструкцій іншим в рамках пропозиції. Виділені зв'язки інваріантні до перестановки абзаців і пропозицій усередині тексту, контекстній заміні слів.

Саме ці характеристики демонструють високу статистичну стійкість в застосуванні до творів конкретного автора. Це дозволяє порівнювати текстові твори, отримувати кількісні оцінки їх близькості і з високою статистичною достовірністю судити про приналежність перу того або іншого автора; застосовується в англомовній журналістиці.

PlagiarismChecker (http://www.plagiarismchecker.com) –позиціонується як безкоштовний он-лайн ресурс для викладачів, призначений для виявлення плагіату в наукових роботах студентів шляхом перевірки коротких фраз або окремих речень. Пошук здійснюється по мережі Інтернет за допомогою пошукових серверів Googleand. Додатково передбачена можливість перевірки веб-сторінок. Введення e-mail адреси на сайті надає користувачеві ряд преференцій, в тому числі, дозволяє отримувати сповіщення від Google про випадки плагіату його тексту у майбутньому.

Недоліки: значні обмеження по об'єму введеного тексту, що не задовольняє об'ємам наукової літератури.

Висновки до першого розділу.

В розділі проведено аналіз антиплагіатних он-лайн Інтернет – ресурсів, висвітлені недоліки оглянутих програм. При перевірки одного й того ж самого тексту на різних програмах, можна побачимо різні дані про унікальність. Це відбувається тому, що різні програми використовують різні алгоритми пошуку та налаштування аргументу перевірки, а це може суттєво вплинути на результат. При довжині шинглу = 1 , всі тексти будуть мати оригінальність 0%; а одна й та ж сама програма, що використовує для пошуку в Інтернет (приклад: Адвего, ETXT), з ідентичними налаштуваннями, може показувати різний % (у межах 1-3%, але іноді й більше). Це залежіть від: IP адреси (країна, місто) користувача, потужності комп'ютера, наявністю вірусів, підключеного майнінгу криптовалют, використання своєї локальні бази. Процеси, які споживають багато ресурсів, не дають можливість програмам перевірки плагіату працювати на повну потугу і за відсутності обробленої відповіді серверу перериває ряд запитів і в результаті оригінальність збільшується.

Підсумуємо, унікальність, - це результат роботи певної програми, що використовує певні бази, з певним налаштуваннями, в певних умовах. Тому вона не є непорушною константою і може суттєво варіюватися.

Щоб отримати достовірну інформацію перевірки тексту на плагіат, слід перевіряти роботу різними програмами антиплагіату.

Однак слід також пам'ятати , що ви надаєте всю інформацію про проведені дослідження третім, юридично не відповідальним особам, що є надзвичайно ризикованим заходом. Матеріали перевірки повинні бути додані до бази даних, а це може порушувати права інтелектуальної власності.

РОЗДІЛ 2. АЛГОРИТМИ ТА МЕТОДИ ВИЯВЛЕННЯ ПЛАГІАТУ

2. 1. Класифікація підходів та алгоритми комп'ютерного виявлення схожості контенту

Комп'ютерне виявлення плагіату (CaPD) - це інформаційно-пошукове (IR) завдання, що підтримується спеціалізованими інформаційно-пошуковими системами, які називаються системами виявлення плагіату (СПД) або системами виявлення схожості документів[4].

Еталонна колекція, яка є набором документів, які вважаються справжніми, порівнюється зовнішніми системи з підозрілим документом. На основі заздалегідь визначених критеріїв схожості, завданням виявлення плагіату є пошук всіх документів, які містять текст, схожий за ступенем вище обраного порогу з текстом у підозрілому документі. Внутрішні системи аналізують виключно текст, що підлягає оцінці, не проводячи порівняння із зовнішніми документами. Такий підхід спрямований на розпізнавання змін в унікальному стилі письма автора як індикатора потенційного плагіату.. Схожість та особливості стилю написання обчислюються за допомогою заздалегідь визначених моделей документів і можуть давати помилкові спрацьовування.

Сервіси для виявлення плагіату використовують два вида алгоритмів: шингл і кореляцію.

1. Шингл.

Шингл – це частина тексту довжиною від 2 до 10 слів, яка використовується для перевірки унікальності. Якщо сервіс перевіряє фрагменти тексту із 5 послідовних слів, то значення шинглу дорівнює 5.
Буде складніше домогтися 100 % унікальності тексту , якщо шингл має невелике значення.

2. Кореляційні алгоритми.

Кореляційні алгоритми перевіряють текст на наявність синонімів і перестановок слів.

Алгоритм працює наступним чином:

- розділяють текст на декілька частин;

- вираховується частота вживання кожного слова для певної частини тексту;
- в перевірку включають найпоширеніші слова.

Якщо на сторонках вебресурсів будуть знайдені подібні частини тексту, то це буде вважатися плагіатом або рерайтом.

Розглянемо методи та класифікацію підходів, які використовують для комп'ютерного виявлення схожості контенту

На рисунку 2.1 представлено класифікацію підходів, які наразі використовуються для комп'ютерного виявлення схожості контенту.

Підходи можна поділити за типом оцінки схожості:

- глобальні підходи - використовують характеристики, взяті з більших частин тексту або документа в цілому;
- локальні методи - досліджують лише попередньо вибрані сегменти тексту як вхідні дані.

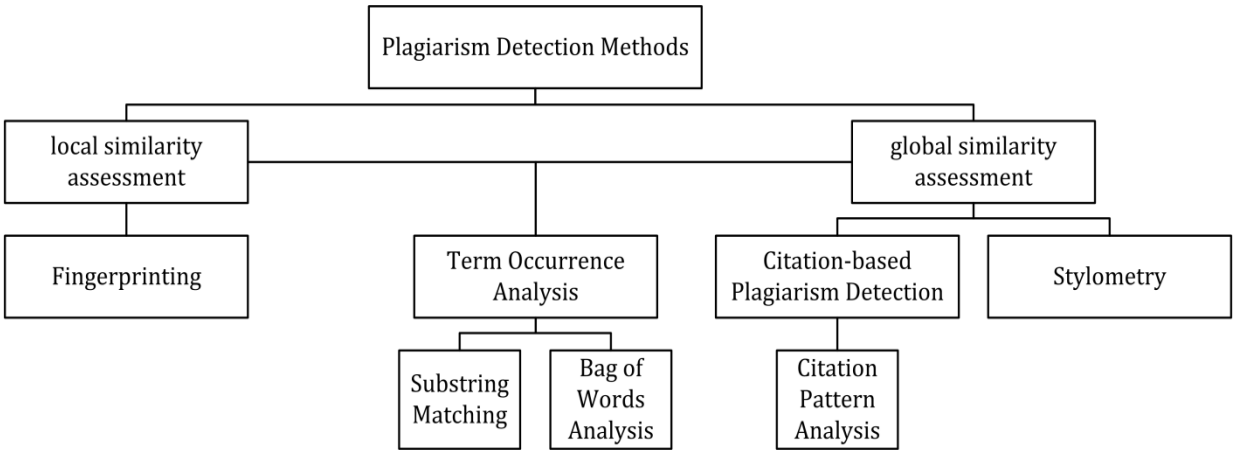


Рисунок 2.1 -Найсучасні методи виявлення плагіату контенту

Класифікація методів комп’ютерного виявлення плагіату

2.1.1. Зняття відбитків пальців.

Дактилоскопічний метод є на сьогодні Найбільш поширеним є діактилоскопічний метод до виявлення схожості змісту. Він формує

репрезентативні фрагменти документів. З набору множинних підрядків (n-грам) виділяють фрагменти для перевірки.

N – грама або Q-грама - це послідовність *з n* елементів з певної вибірки тексту. *N*-грам використовується для виявлення плагіату.

Алгоритм метода N-грама:

1. розділити текст на кілька невеликих фрагментів(N-грамами);
2. порівняти один фрагмент з іншим ;
3. отримати ступінь подібності контрольованих документів.

Використовуючи N-грами, можна ефективно знайти кандидатів, щоб замінити слова з помилками правопису.

1. Приклад роботи алгоритму:

P = P (щастя) * P (є | щастя) * P (задоволення | щастя є) * P (без | щастя є задоволення) * P (каяття | щастя є задоволення без)

Ймовірність P (щастя) можна розрахувати, для цього потрібно підрахувати кількість разів це слово зустрілося в тексті та поділити його на загальне число слів.

2. Приклад, де ймовірність слова в тексті залежить тільки від попереднього слова:

P (каяття | щастя є задоволення без). Формула для розрахунку фрази прийме вигляд:

P = P (щастя) * P (є | щастя) * P (задоволення | є) * P (без | задоволення) * P (каяття | без)

Умовну ймовірність P (є | щастя) розраховується наступним чином: потрібно кількість пар 'щастя є' поділити на кількість в тексті слова 'щастя'. Коли порахуємо всі пари слів в деякому тексті, то отримуємо вірогідність довільної фрази. Цей набір буде біграмною моделлю.

Набори представляють собою відбитки пальців, а їх елементи називаються мініатюрами (minutiae). Підозрілий документ перевіряється на плагіат шляхом обчислення його "відбитка" та запиту мініатюр за

попередньо розрахованим індексом "відбитків" для всіх документів репрезентативної колекції.

Збіг реквізитів з реквізитами інших документів вказує на спільні сегменти тексту і свідчить про потенційний плагіат, якщо вони перевищують обраний поріг схожості. Для прискорення обчислення та забезпечення перевірки у дуже великих колекціях, наприклад Інтернет, порівнють лише підмножину мініатюр.

2.1.2 Зіставлення рядків.

Порівняння рядків, це метод в якому документи порівнюються на наявність дослівних текстових збігів. Цей метод дуже широко використовується в комп'ютерних науках. Для вирішення проблеми цього методу використовують численні методи, які повинні адаптувати для зовнішнього виявлення плагіату. Перевірка підозрілого документа в таких умовах вимагає обчислення та зберігання ефективно порівнянних представлень для всіх документів у довідковій колекції для їх подальшого попарного порівняння. Для цього використовуються суфіксальні моделі : суфіксальні дерева або суфіксальні вектори.

Суфіксне дерево знаходить застосування в алгоритмах на рядках. Воно основано на дереві структури даних.

Алгоритм метода суфіксного дерева.

Суфіксне дерево T для рядка S довжини *m* це орієнтоване дерево, що має *m* листів пронумерованих від 1 до *m*. Кожна вершина, окрім кореня, має щонайменше два відгалуження, кожне ребро марковане не порожнім підрядком з рядка *S*. Жодна вершина не може мати ребра, якщо маркування починається з однакової літери. (див.рис.2.1)

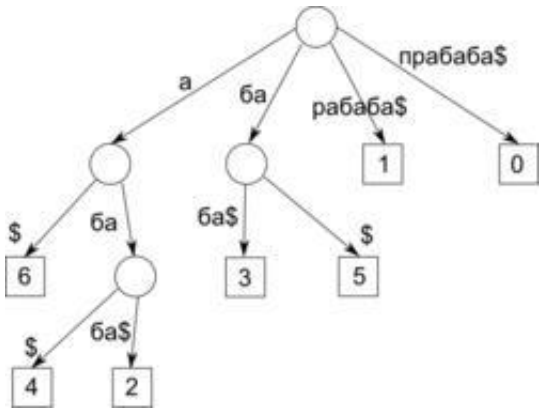


Рисунок 2.1- Суфіксне дерево для слова прабаба

Розглянемо роботу алгоритму Суфіксне дерево на прикладі слова прабаба.

Кожен підрядок завершується спеціальним символом $.(рис.2.1). Сім шляхів від кореня до листів (показані квадратами) відповідають семи суфіксам.

Конкатенація маркувань ребер на шляху від кореня до листа дає суфікс рядка S починаючи з і-ї літери. Саме це і є головною особливістю цього алгоритму. По суті суфікс не дерево, це граф.

Слід зазначити, що суфіксне дерево існує не для кожного рядка. Якщо суфікс рядка збігається з його префіксом (наприклад, рядок *xabxa*), то шлях для першого суфікса не зможе бути завершений і в графі з'явиться цикл. Цієї проблеми можна уникнути, Якщо до рядка додати літеру яка не зустрічається в тексті, наприклад ($),то проблеми не буде.

Всі внутрішні вершини суфіксного дерева мають щонайменше два відгалуження, значить кількість вершин не може перевищувати $N - 1$; а загальна кількість вершин не перевищує $n + (n - 1) + 1 = 2n$, із них: n листів, $n - 1$ внутрішніх вершин окрім кореня, 1 корінь.

Суфіксні дерева та суфіксні масиви знаходять широке застосування для розв'язання задач на рядках. Наприклад, задача з'ясувати, чи входить взірець P до рядка S із побудованим суфіксним деревом, може бути розв'язана за час $O(|P|)$. Це буде найбільше загальне підстроювання двох рядків, лінеаризація циклічного рядка, завдання про підстроку для бази зразків.

2.1.**3. Мішок слів.**

Мішок слів або аналіз пакетів слів, являє собою векторний пошук, традиційної концепції IP до області виявлення схожості контенту.

Документ у цій моделі розглядається як невпорядкована множина термів. Термами в інформаційному пошуку називають слова, з яких складається текст.

Алгоритм метода аналізу пакетів

Різними способами можна визначити вагу терма в документі — «важливість» слова для ідентифікації цього тексту. Всі терми, які трапляються в документах оброблюваної колекції, можна впорядкувати. Якщо тепер для деякого документа виписати по порядку ваги всіх термів, включаючи ті, яких немає в цьому документі, вийде вектор, який і буде представленням цього документа у векторному просторі. Розмірність цього вектора, як і розмірність простору, дорівнює кількості різних термів у всій колекції, і є однаковою для всіх документів.

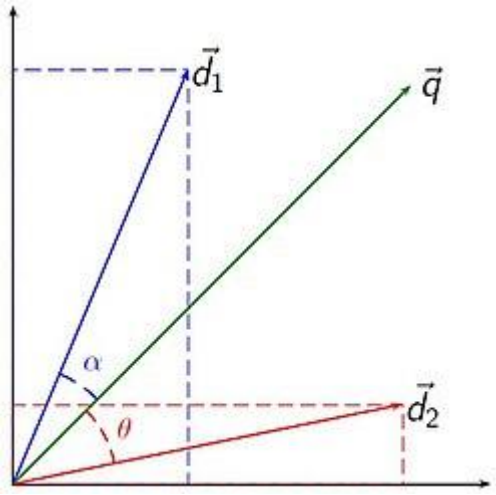


Рисунок 2.2 –Представлення документ у векторному просторі,

Де:

d_j— векторне уявлення j-го документа, α — кут d_i -го терма в j-му документі, θ— кут q_i-го терма в j-му документі,

Маючи таке подання для всіх документів, можна, знаходити відстань між

точками простору і вирішувати задачу подоби документів. Чим ближче розташовані точки, тим більше схожі відповідні документи. У разі пошуку документа за запитом, запит теж представляється як вектор того ж простору і можна обчислювати відповідність документів запиту.

Рейтинг релевантності документів у пошуку за ключовими словами можна обчислити, використовуючи припущення про теорію спільних рис документа, порівнявши відхилення кутів між кожним вектором документу та оригінальним вектором запиту, де запит представлений як той же самий вид вектора як документа.

Документи представляються у вигляді одного або декількох векторів, наприклад, для різних частин документа, які використовуються для попарних обчислень подібності. Обчислення подібності може базуватися на традиційній косинусоїдальній мірі подібності або на більш складних мірах подібності.

2.1.4. **Аналіз цитування.**

Виявлення плагіату на основі цитування (CbPD) ґрунтується на аналізі цитування. Цей метод не спирається на текстову схожість. CbPD досліджує інформацію про цитування та посилання в текстах для виявлення подібних закономірностей у послідовності цитування.

Алгоритм метода аналізу цитування

Цей підхід підходить для текстів, які містять цитати. Це відносно молодою концепцією. Вона не була прийнята комерційним програмним забезпеченням, але існує перший прототип системи виявлення плагіату на основі цитування. Близькість цитувань у досліджуваних документах є основними критеріями, що використовуються для обчислення схожості шаблонів цитування. Шаблони цитування являють собою послідовності для кількісної оцінки ступеня схожості *патернів.* В роботі цього методу враховуються такі фактори, як абсолютна кількість або відносна частка спільних цитувань у патерні, а також імовірність того, що цитати зустрічаються в одному документі.

2.1.5. Стилометрія.

Стилометрія - це статистичні методи кількісної оцінки унікального стилю письма автора. Використовуються для встановлення авторства або виявлення внутрішнього плагіату. Внутрішнє виявлення плагіату у підозрілому документі без порівняння його з іншими документами здійснюється шляхом побудови та порівняння стилістичних моделей для різних текстових сегментів підозрілого документа. Уривки, які стилістично відрізняються від інших, позначаються як потенційно плагіат/порушення. Незважаючи на простоту отримання, символьні n-грами виявилися одними з найкращих стилометричних ознак для виявлення плагіату за його суттю.

2.1.6. Нейронні мережі.

Більш сучасні підходи до оцінки схожості контенту з використанням нейронних мереж досягли значно більшої точності, але потребують великих обчислювальних витрат. Традиційні нейромережеві підходи вбудовують обидві частини контенту в семантичні векторні вставки для обчислення їхньої схожості, яка часто є їхньою косинусоїдальною схожістю. Більш просунуті методи виконують наскрізне прогнозування схожості або класифікації з використанням архітектури трансформатора. Особливо виявлення парафраз виграє від високопараметризованих попередньо навчених моделей.

2 .2. Продуктивність методів

На основі порівняльних оцінок систем виявлення схожості контенту можемо бачити , що ефективність залежить від типу наявного плагіату (див. рис. 2.3). Всі підходи до виявлення плагіату ґрунтуються на текстовій схожості, крім аналізу структури цитування. Точість виявлення знижується тим більше, чим більше випадків плагіату завуальовано.

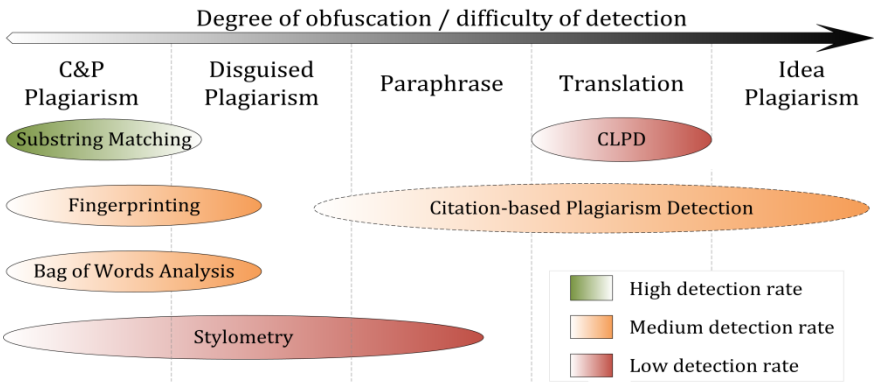


Рисунок 2.3-Ефективність виявлення підходів CaPD в залежності від типу наявного плагіату

Плагіат копіювання та вставки (c&p), або грубе порушення авторських прав, або скромно замасковані випадки плагіату можуть бути виявлені з високою точністю за допомогою сучасних зовнішніх СППР, якщо джерело є доступним для програмного забезпечення.

а). Процедури зіставлення підрядків досягають хороших результатів для виявлення плагіату c&p. Вони використовують моделі документів без втрат, (суфіксальні дерева).

б). Ефективність дактилоскопічного аналізу або аналізу пакетів слів для виявлення копій, залежить від втрат інформації, яких зазнає модель документа, що використовується. Використовуючи стратегії розбиття на частини та відбору, ці методи краще виявляють помірні форми замаскованого плагіату порівняно з процедурами зіставлення підрядків.

в). Виявлення внутрішнього плагіату за допомогою стилометрії може певною мірою подолати межі текстової схожості, порівнюючи лінгвістичну схожість. Стилістичні відмінності між плагіатними та оригінальними сегментами є значними і можуть бути достовірно ідентифіковані, стиліметрія може допомогти у виявленні замаскованого та перефразованого плагіату. Стилометричні порівняння не дадуть результатів, коли сегменти сильно перефразовані і нагадують особистий стиль письма плагіатора, або текст був складений кількома авторами. Стилометричний аналіз надійно працює лише для документів обсягом у кілька тисяч або десятків тисяч слів, що обмежує

застосовність методу до умов CaPD.

Зростає кількість досліджень, присвячених методам і системам, здатним виявляти перекладений плагіат. Наразі технології виявляти перекладений плагіат або міжмовне виявлення плагіату (ММВП) не змогли досягти задовільних результатів виявлення на практиці.

г). Аналіз структури цитування дає змогу виявляти перекази та переклади з вищими показниками успішності порівняно з іншими підходами до виявлення плагіату. Цей метод не залежить від текстових характеристик. Однак, він поступається текстовим підходам у виявленні коротших плагіатних уривків, які характерні для випадків плагіату типу "скопіювати і вставити" або "струснути і вставити", тобто змішати дещо змінені фрагменти з різних джерел.

2.3. Програмне забезпечення засобів виявлення схожості

Проєктування програмних засобів виявлення схожості змісту для роботи з текстовими документами характеризується низкою факторів, які представлені в таблиці 1.

Таблиця 1- Фактори схожості в текстових документах

Фактор	Опис та альтернативи
Сфера пошуку	У загальнодоступному Інтернеті, використовуючи пошукові системи / Інституційні бази даних / Локальні, специфічні для системи бази даних.
Час аналізу	Затримка між подачею документів і часом, коли результати стають доступними.
Документообіг / Пакетна обробка	Кількість документів, які система може обробити за одиницю часу.
Перевірка інтенсивності	Як часто і за якими типами фрагментів документа (абзаци, речення, послідовності слів фіксованої довжини) система запитує зовнішні ресурси, наприклад, пошукові системи.
Тип алгоритму порівняння	Алгоритми, які визначають спосіб, яким система використовує для порівняння документів між собою.
Точність і відкликання	Кількість документів, які були правильно позначені як плагіат, порівняно із загальною кількістю позначених документів, а також із загальною кількістю документів,

	які насправді були плагіатом. Висока точність означає, що було знайдено мало хибнопозитивних результатів, а високий відгук означає, що мало хибнонегативних результатів залишилися невиявленими.

Більшість масштабних систем виявлення плагіату використовують великі внутрішні бази даних (на додаток до інших ресурсів), які зростають з кожним додатковим документом, поданим на аналіз. Однак ця особливість розглядається деякими як порушення авторських прав студентів.

2.4. Методи та алгоритми виявлення плагіату у програмному коді

Методи виявлення плагіату в комп'ютерному вихідному коді вимагають інших інструментів, ніж для порівняння текстів.

В цьому виду плагіату не існує бази текстів наукових робіт, рефератів, статей, які можна знайти в традиційному плагіаті. Більшість завдань з програмування пишуться під свої вимоги і знайти існуючі програми, які б відповідали їм, доволі складно.

Алгоритми класифіковані по таким означенням .

- Рядки - пошук точних текстових збігів сегментів, наприклад, рядків з п'яти слів. Швидко, але можна заплутатися через перейменування ідентифікаторів.
- Токени - як і у випадку з рядками, але з використанням лексеми для перетворення програми в токени. Це дозволяє відкинути пробіли, коментарі та імена ідентифікаторів, що робить систему більш стійкою до простих замін тексту. Більшість систем виявлення академічного плагіату працюють на цьому рівні, використовуючи різні алгоритми для вимірювання схожості між послідовностями токенів.
- Дерева розбору - побудова та порівняння дерев розбору. Це дозволяє виявити схожість більш високого рівня. Наприклад, порівняння дерев може нормалізувати умовні оператори і виявити еквівалентні конструкції як схожі між собою.

- Графіки програмних залежностей (PDG) - PDG відображає фактичний потік управління в програмі і дозволяє знаходити еквіваленти набагато більш високого рівня, з більшими витратами на складність і час обчислень.
- Метрики - метрики фіксують "оцінки" сегментів коду відповідно до певних критеріїв; наприклад, "кількість циклів та умовних операторів" або "кількість різних змінних, що використовуються". Метрики легко обчислюються і можуть бути швидко порівняні, але також можуть призвести до помилкових спрацьовувань: два фрагменти з однаковими оцінками за набором метрик можуть робити абсолютно різні речі.
- Гібридні підходи - наприклад, дерева розбору + суфіксні дерева можуть поєднувати можливості виявлення дерев розбору зі швидкістю, яку надають суфіксні дерева, тип структури даних, що відповідає рядкам.

Ця класифікація була розроблена для рефакторингу коду (дублюючого коду або клону коду), а не для виявлення академічного плагіату. Ці підходи ефективні для різних рівнів схожості: низький рівень схожості стосується ідентичного тексту, а високий рівень схожості може бути зумовлений схожими специфікаціями. В академічному середовищі, Всі студенти на своїх заняттях повинні писати код за однаковими специфікаціями і функціонально еквівалентний код (з високим рівнем схожості) цілком очікуваний, а ось низький рівень схожості розглядається як доказ шахрайства.

2.4.2 Метод та алгоритм методу Text-based

Text-based - метод доводі швидкий, відрізняється простотою, дає змогу отримати повноту даних про наявність плагіату.

Text-based метод полягає у порівнянні текстових уявлень програм на основі відстані Левенштейна або Джаро-Віклера.

Алгоритм відстані Левенштейна.

Відстань Левенште́йна (алгоритм Левенштейна або відстань редагування) , це міра відмінності двох послідовностей символів (рядків),

яка обчислюється, як мінімальна кількість операцій, необхідних для перетворення однієї послідовності в іншу (видалення, вставки і заміни).

Для розрахунку відстані Левенштейна використовується матриця розміром (n + 1) * (m + 1), де n і m - довжини порівнюваних рядків. Вартість операцій вилучення, заміни та вставки вважається однаковою

Для опису алгоритму будемо використовувати як неформальний запис алгоритму, тобто псевдокод. В алгоритмі використовується структура мов програмування, але не береться до уваги деталі коду, які є неістотні для розуміння алгоритму. Це може бути опис типів, виклик підпрограм.

У псевдокоді алгоритм виглядає так:

```
int LevenshteinDistance(char str1[1..lenStr1], char str2[1..lenStr2])
  // d таблиця кількість рядків = lenStr1+1 та кількість стовпців = lenStr2+1
  declare int d[0..lenStr1, 0..lenStr2]
  // i та j використовуються для індексування позиції у str1 та у str2
  declare int i, j, cost
  for i from 0 to lenStr1
     d[i, 0] := i
  for j from 0 to lenStr2
     d[0, j] := j
  for i from 1 to lenStr1
     for j from 1 to lenStr2
        if str1[i] = str2[j] then cost := 0       //однакові
                             else cost := 1       //заміна
        d[i, j] := minimum(
                             d[i-1, j  ] + 1,     // вилучення
                             d[i  , j-1] + 1,     // вставка
                             d[i-1, j-1] + cost   // заміна або однакові
                           )
  return d[lenStr1, lenStr2] //значення відстані Левенштайна в останній клітинці матриці.
```

У мові програмування PHP цей алгоритм реалізований функцією levenshtein.

Приклад реалізації оптимізованого алгоритму пошуку відстані Левенштейна на мові Python 2.x:

```
def levenstein(s1,s2):
    n = range(0,len(s1)+1)
    for  y in xrange(1,len(s2)+1):
        l,n = n,[y]
        for x in xrange(1,len(s1)+1):
            n.append(min(l[x]+1,n[-1]+1,l[x-1]+((s2[y-1]!=s1[x-1]) and 1 or 0)))
    return n[-1]
```

Розглянемо роботу метод text-based на блок –схемі.

Метод text-based базується на зчитуванні фрагментів словесних символів та сканування за допомогою функціоналу javascript . Створюється база двох масивів даних фрагментів програмного коду. У базі будуть слова та символи мови програмування (фігурні , звичайні дужки; точки з крапками та всі символи , крім символів відступів та табуляції).

Обидва масиви даних порівнюються між собою способом перевірки слова в слово. Слід зауважити, що в даній схемі символи мови програмування також являються словами. Кількість співпадінь буде конвертуватися в числове значення і виводитись в блок виводу результату.

Для надання більшої інформації показана кількість слів в кожному з програмних кодів. Щоб вирішити проблему покидання меж масиву, потрібно словам для порівняння за межами масиву призначити empty.

Блок - схему алгоритму роботи text-based методу можна побачити на рисунку 2.4.

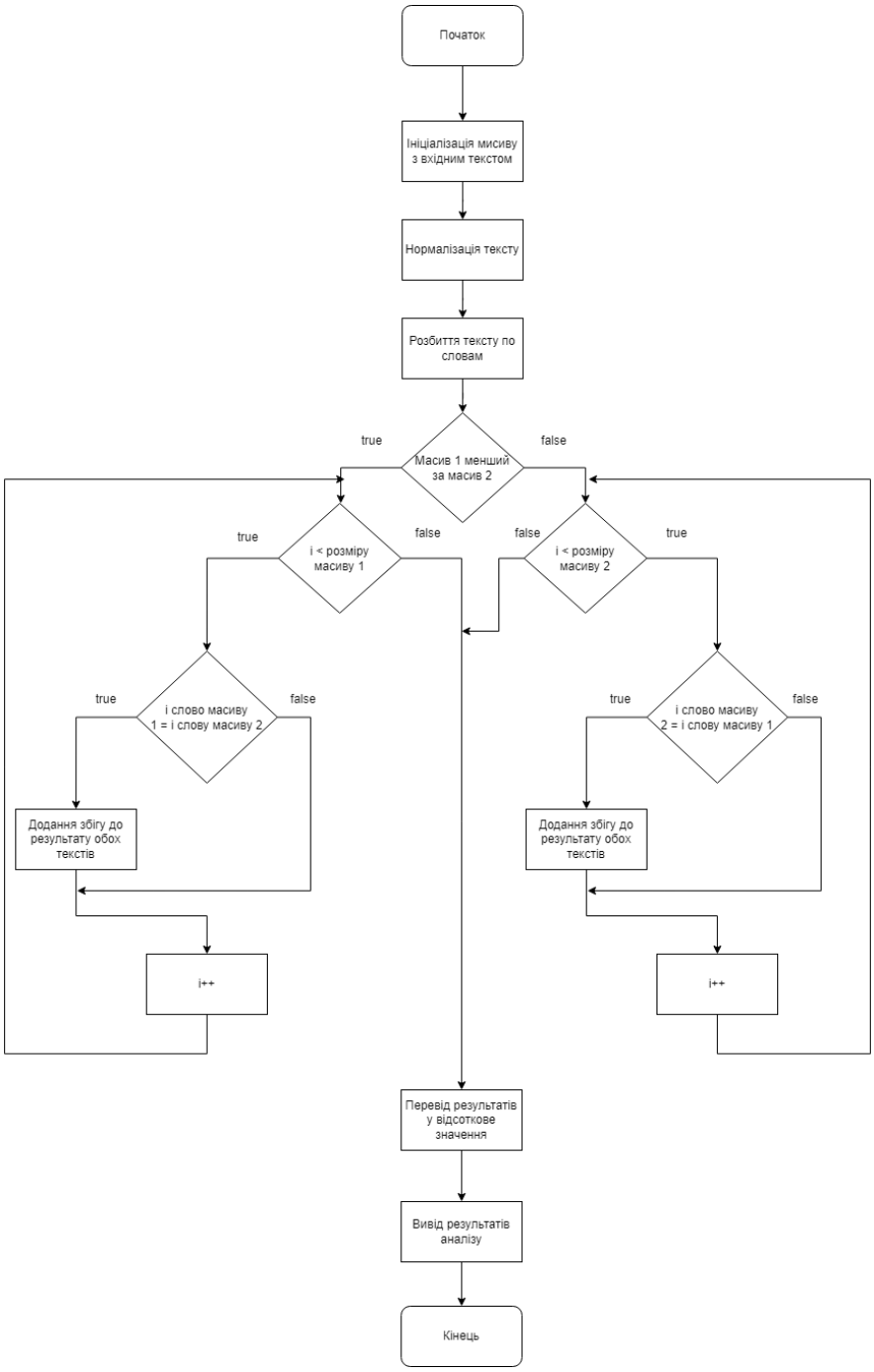


Рисунок 2.4- Блок - схему алгоритму роботи метода text-based

Фрагмент коду метод text-based:

```
<script>
function Perevirka()
{
	//змінні

	var plagword1 = 0;
	var plagword2 = 0;
	var plagbloc1 = 0;
	var plagbloc2 = 0;
	var ezkey1 = -1;
	var ezkey2 = -1;
	var hk = 0;
	var hardkey2 = 0;
	var superhardkey1 = 0;
	var superhardkey2 = 0;

	//перевірка по словам

   var text1 = document.getElementById("txt1").value;
   var text2 = document.getElementById("txt2").value;
   var Text1 = text1.toLowerCase();
   var Text2 = text2.toLowerCase();
   var Text1word = Text1.split(" ");
   var Text2word = Text2.split(" ");
   if(Text1word.length<=Text2word.length)
   {
   var limit = Text1word.length;
      for (i = 0; i < limit; i++)
      {
                  if (Text1word[i]==Text2word[i])
                  {
                        var plagword1 = plagword1 + 1;
                        var plagword2 = plagword2 + 1;
                  }
            }
   }
      else
      {
      var limit = Text2word.length;
      for (i = 0; i < limit; i++)
      {
                  if (Text2word[i]==Text1word[i])
                  {
```

```
                    var plagword1 = plagword1 + 1;
                    var plagword2 = plagword2 + 1;
                }
            }
        }

    // вивід перевірки по словам
```

2.4.2 Метод та алгоритм Token-based методу

Token-based - цей метод доповнює варіант першого методу і надає можливість збільшити точність вихідних результатів. Його бажано комбінувати з іншими методами для отримання надійної і продуктивної системи аналізу програмного коду.

Метод Token-based заснований на перетворенні ключових слів програми в послідовність лексем мови програмування (токенів), які і будуть використані для аналізу тексту програмного коду. Тобто,в цьому методі буде відтворюватись заміна ключових слів на їх символьне. Отримані токени порівнюються будь-яким доступним способом. Цей алгоритм точніший за попередній, однак залишається проблема зі структурними змінними.

Token-based - метод перетворює вхідний текст програмного коду в новий текст, який являється набором уніфікованих значень, токенів. Створюючи масив значень ключових слів ми можемо розпізнати слова, які будуть перенесени до нового текстового масиву після токенізації. Перед тим як це використовувати , потрібно проаналізувати до якого типу відноситься дане слово, щоб надати йому відповідний токен . Для цієї процедури використовують таблиці відповідності ключових слів . Ця процедура залишає умовний скелет коду. Якщо використовувати цю процедуру «очищення» в комбінації з повторним аналізом через Text-based метод то можна збільшити точність результатів. Це досягається за рахунок значного зменшення залежності перевірки від текстової зміни значень коду.

Слід зауважити, що метод порівнює лише програмну суть коду.

Розглянемо, як буде виглядати класифікація слів, символів на прикладі

мови програмування C#;

Класифікація слів, символів, що являються ключовими:

- символи для виклику і опису функцій, виділення умовних блоків завершення дій. Це «;», «:», « (», «)», «{» , «}», « [», «]»;
- слова для створення та використання циклів, наприклад в мові програмування C#;
- числові значення в тексті;
- слова, що означають створення змінних в мові програмування , програмний код якої аналізуємо;
- не виділяються подвійні оператори мови C# не виділяються, вони просто записані як один за одним. як два оператора

Блок-схема алгоритму token-based методу показана на рисунку 2.5. Для спрощення ознайомлення і запису, блок-схема не має повтору для другого тексту процесів.

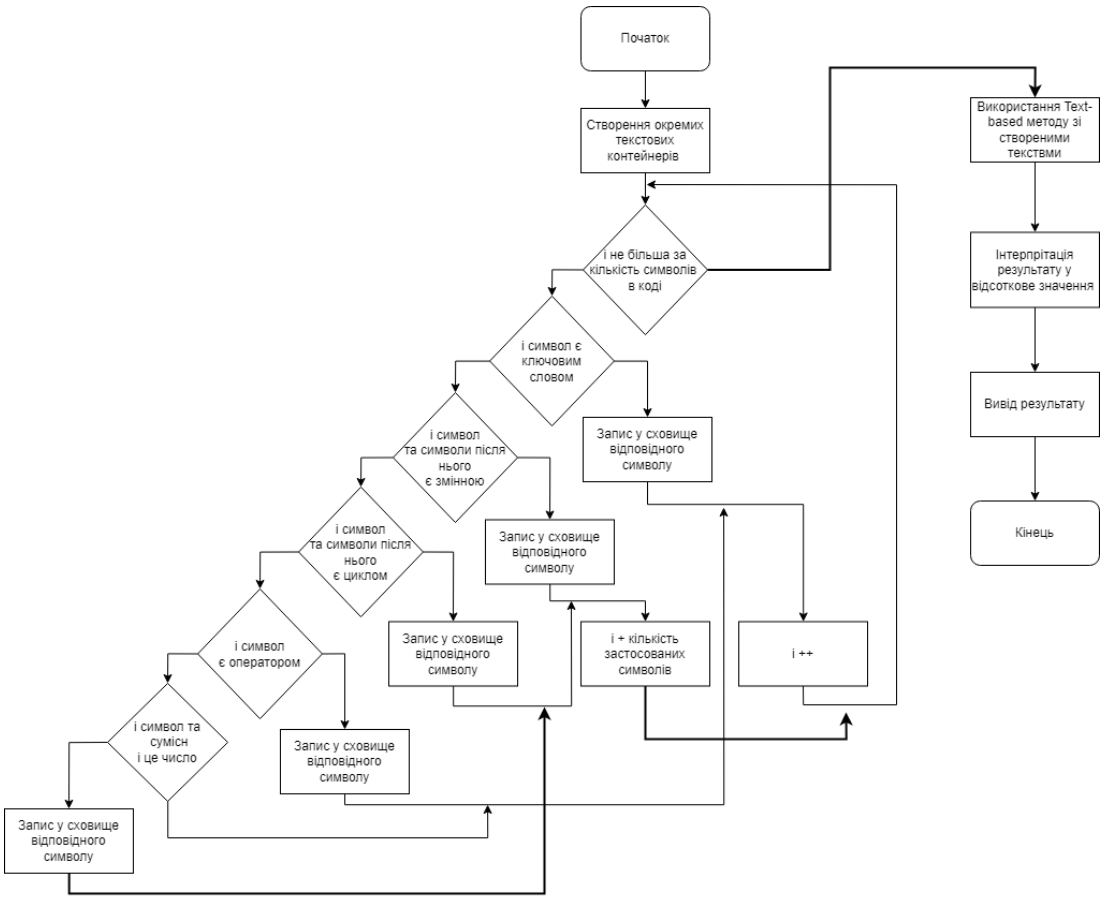


Рисунок 2.5– Блок-схема алгоритму роботи методу Token-based .

Фрагмент коду :

```
<meta charset="UTF-8">
<title>PerevirkaPlagiatu</title>
<link rel="stylesheet" type="text/css" href="style.css">
<script src="script.js"></script>
<meta name="viewport" content="width-device-widht, initial-scale=1.0">
</head>

<body>
    <h1 style="position: relative;text-align: center;"><i>Перевірка коду на плагіат</i></h1>
    <div id="leftSidebar">

    <p><textarea id="txt1" style="width: -webkit-fill-available;" rows="40" cols="70" name="text1"></textarea></p>
    </div>
    <div id="rightSidebar">
    <p><textarea id="txt2" style="width: -webkit-fill-available;" rows="40" cols="70" name="text2"></textarea></p>
    </div>
    <div id ="perevirka">
    <button onclick="Perevirka()"><div class="left"></div>
  Перевірка
 <div class="right"></div></button>
    <p id="word1"></p>
    <p id="word2"></p>
    <p id="key1id"></p>
    <p id="key2id"></p>
    <p id="ckl1id"></p>
    <p id="ckl2id"></p>
    <p id="var1id"></p>
    <p id="var2id"></p>
    <p id="oper1id"></p>
    <p id="oper2id"></p>
    <p id="doper1id"></p>
    <p id="doper2id"></p>
    <p id="num1id"></p>
    <p id="num2id"></p>
    <p id="tok1id"></p>
    <p id="tok2id"></p>
  </div>

<script>
function Perevirka()
```

```
{
    //змінні

    var plagword1 = 0;
    var plagword2 = 0;
    var plagbloc1 = 0;
    var plagbloc2 = 0;
    var ezkey1 = -1;
    var ezkey2 = -1;
    var hk = 0;
    var hardkey2 = 0;
    var superhardkey1 = 0;
    var superhardkey2 = 0;

    //перевірка по словам
```

2.4.3. Метод та алгоритм Metrics-based методу

Metrics-based метод допомагає в виявлені плагіату другого рівня. Це досягається завдяки комбінації з методом Text-base.

Після отримання в попередніх методах деяких метрик, таких, як умовні конструкцій, кількість циклів , йде оцінка подібності програм на основі кількості метрик, що збігаються.

Metrics-based,- метод базується на виявлені і підрахунку змінних або функції, що є об'єктами в програмних кодах. Ймовірність однакової кількості даних об'єктів в двох різних кодах неймовірна мала. Якщо з'являється їх збіг, то це свідчить про плагіат.

Метод також дає можливість проводити очищення коду від пробільних символів та символів відступу. Ключові слова виявляються за рахунок по символьного аналізу коду для ідентифікації символів. Такі слова для будь-якої програми вже будуть вважатися новими. Для уникнення цього створюється масив небезпечних для виявлення плагіату символів. До цих символів відносяться: коми, крапки, двокрапки, крапки з комою. Ці символи виявляються в вхідних кодах, але їх не видаляють, тому що вони також є ключовими символами. Їх просто замінюють символами відступу, щоб розділити сусідні слова

Ключові слова розподіляють по категоріях залежно від їх типу: ключові слова, цикли, змінні, числа, оператори, подвійні оператори. Це дозволяє підрахувати ключові слова та вивести результати в відповідний блок.

Розглянемо ці вказані категорії:

- Ключові слова: « asm, else, new, this, auto, enum, operator, throw, explicit, private, true, break, export, protected, try, case, extern, public, typedef, catch, false, register, typeid, reinterpret_cast, typename, class, return, union, const , friend, unsigned, const_cast, goto, signed, using, continue, if, sizeof, virtual, default, inline , static, void, delete, static_cast, volatile, struct, wchar_t, mutable, switch, dynamic_cast, namespace, template»;
- Цикли: for, do, while;
- Змінні: bool, char, float, short, int, long, double, string;
- Числа – будь-які числа;
- Оператори: +, =, -, *,<, >, /;
- Подвійні оператори: ==, <=, =<, >=, =>, =*, *=, =+, +=,=-, -=, =/, /=, &&, ||.

Схема алгоритму роботи metrics-based методу показана на рисунку 2.6. Варто зауважити, що ідентичні дії проводяться для обох фрагментів, але опис ідентичних дій були опущені в схемі алгоритму для спрощення його розуміння і легкості читання.

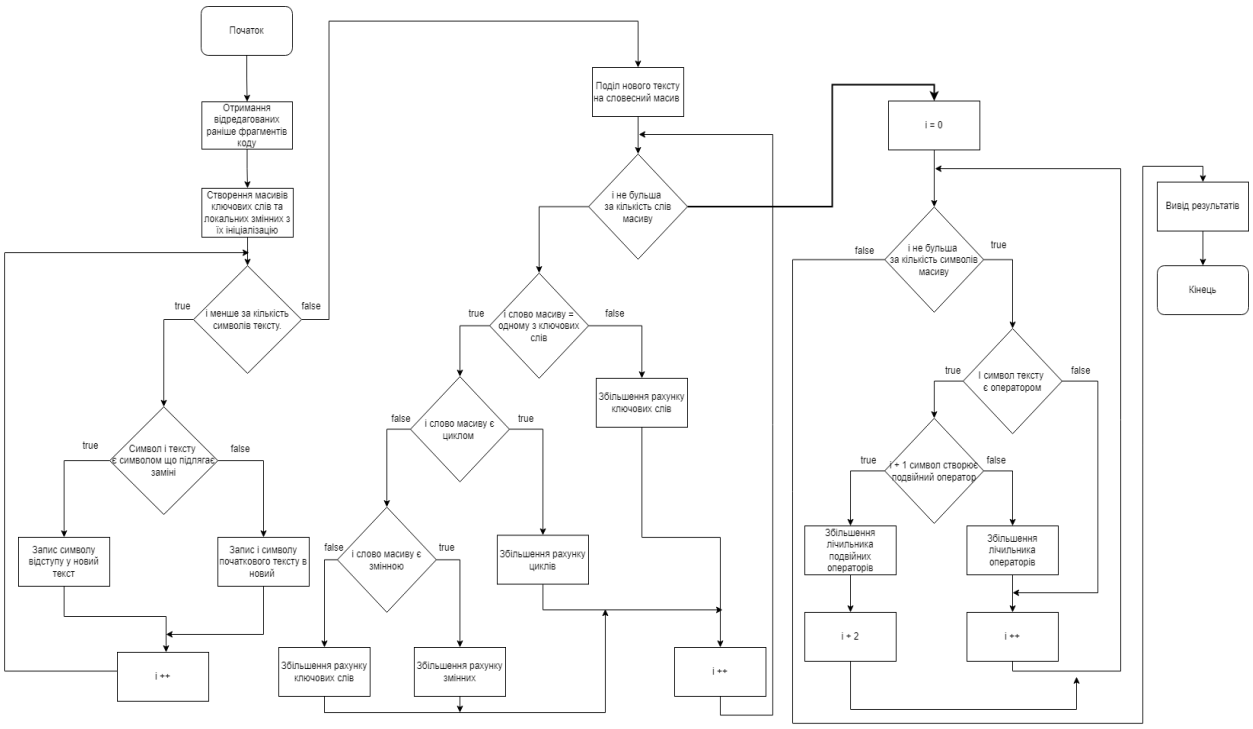


Рисунок 2.6 – Схема алгоритму роботи методу Metrics-based методу

Фрагмент коду метода роботи metrics-based:

```
if(Text2[i] == "=" && Text2[i+1] == "=")
        {
                tokenText2 += "o";
                tokenText2 += " ";
        }
        if(Text2[i] == "=" && Text2[i+1] == ">")
        {
                tokenText2 += "o";
                tokenText2 += " ";
        }
        if(Text2[i] == "=" && Text2[i+1] == "<")
        {
                tokenText2 += "o";
                tokenText2 += " ";
        }
        if(Text2[i] == "&" && Text2[i+1] == "&")
        {
                tokenText2 += "o";
```

```
            tokenText2 += " ";
        }
        if(Text2[i] == "|" && Text2[i+1] == "|")
        {
            tokenText2 += "o";
            tokenText2 += " ";
        }
        if(Text2[i] == "=" && Text2[i+1] == "*")
        {
            tokenText2 += "o";
            tokenText2 += " ";
        }
        if(Text2[i] == "*" && Text2[i+1] == "=")
        {
            tokenText2 += "o";
            tokenText2 += " ";
        }
        if(Text2[i] == "=" && Text2[i+1] == "+")
        {
            tokenText2 += "o";
            tokenText2 += " ";
        }


    if(Text2[i] == "=" && Text2[i+1] == "=")
        {
            tokenText2 += "o
    if(Text2[i] == "+" && Text2[i+1] == "=")
        {
            tokenText2 += "o";
            tokenText2 += " ";
        }
        if(Text2[i] == "=" && Text2[i+1] == "-")
        {
            tokenText2 += "o";
            tokenText2 += " ";
        }
        if(Text2[i] == "-" && Text2[i+1] == "=")
        {
            tokenText2 += "o";
            tokenText2 += " ";
        }
        if(Text2[i] == "=" && Text2[i+1] == "/")
        {
            tokenText2 += "o";
```

```
            tokenText2 += " ";
        }
        if(Text2[i] == "/" && Text2[i+1] == "=")
        {
            tokenText2 += "o";
            tokenText2 += " ";
        }
        if(Text2[i] == '=' && Text2[i+1]!='=' && Text2[i-1]!='=' &&
Text2[i+1]!='<' && Text2[i-1]!='<' && Text2[i+1]!='*' && Text2[i-1]!='*' &&
Text2[i+1]!='/' && Text2[i-1]!='/' && Text2[i+1]!='+' && Text2[i-1]!='+' &&
Text2[i+1]!='-' && Text2[i-1]!='-')



}
```

Висновки до другого розділу.

В другому розділі роботи розглянута класифікація підходів та алгоритми комп'ютерного виявлення схожості тексту. Розглянуто шість видів підходів до виявлення плагіату. Всі алгоритми ґрунтуються на текстовій схожості, крім одного - аналізу структури цитування.

Була проаналізована статистика роботи алгоритмів і на основі порівняльних оцінок систем та методів виявлення схожості контенту, зроблено висновок , що ефективність виявлення плагіату залежить від типу наявного плагіату. Чим більше випадків плагіату завуальовано, тим точність виявлення плагіату знижується

Для проєктування програмних засобів виявлення схожості змісту для роботи з текстовими документами слід виділяти низку факторів, які були представлені в цьому розділі.

В розділі також розглянути найбільш відомі методи та алгоритми виявлення плагіату у програмному коді. В роботі було розглянуто три метода та розроблено блок - схеми алгоритмів до них.

РОЗДІЛ 3. Розробка та практична реалізація системи

3.1.Сруктура системи для аналізу програмного коду

WEB-орієнтована система для аналізу коду, це сайт, який має ієрархічну структуру та складається з чотирьох сторінок (рис. 2.2).

Рисунок 3.1. – Ієрархічна схема сайту для аналізу коду

Середовищем для розробки веб - системи було обрано Visual Studio Code. Це потужний редактор вихідного коду, який поставляється з вбудованою підтримкою JavaScript, TypeScript і Node.js і має систему розширень для інших мов: C++, C#, Java, Python, PHP, Go, .NET та має гнучкий API. Visual Studio Code поставляється з IntelliSense для JavaScript, TypeScript, JSON, CSS і HTML і підтримує налагодження Node.js.

Веб - орієнтована система для аналізу програмного коду має ієрархічну структуру і включає 4 файли типу html. На рисунку 3.2 представлена файлова структура веб - орієнтованої системи для аналізу програмного коду.

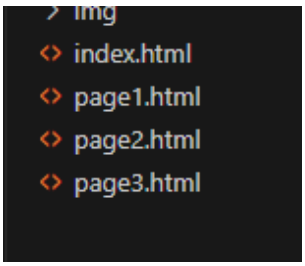


Рисунок 3.2. – Файлова структура системи

Розглянемо призначення кожного файлу.

1. Першою і головною («гостьовою») сторінкою сайту є файл index.html (рис 3.3)

Головна сторінка має три пункти меню - посилання: «Перевірка коду на плагіат, «Про C#», «Довідка про роботу програми».

Пункт меню «Довідка про роботу програми» - надає інформацію про методи аналізи коду та виявлення плагіату.

Рисунок 3.3. – Головна сторінка сайту

2. З інформаційної сторінки page1.html бачимо відомості про доступну мову програмування для перевірки на плагіат.

3. З інформаційної сторінки page2.html – відомості про методи, що використовує система.

4. Page3.html – надає доступ до основного функціоналу системи та є робочою сторінкою сайту. Це робоча сторінка програми.

На робочій сторінці сайту є панель виводу результатів перевірки коду, поля для вводу фрагментів коду і кнопка, яка запускає програмний

код javascript з функціональними можливостями системи. На рисунку 3.4 представлено код ініціалізації.

```
<a href="index.html"><div class="close">&times;</div></a>
<div class="head"><i>Перевірка коду на плагіат</i></div>
<div id="leftSidebar">

<p><textarea id="txt1" style="width: -webkit-fill-available;" rows="40" cols="70" name="text1"></textarea></p>
</div>
<div id="rightSidebar">
        <p><textarea id="txt2" style="width: -webkit-fill-available;" rows="40" cols="70" name="text2"></textarea></p>
</div>
<div id ="perevirka">
```

Рисунок 3.4. – Код ініціалізації

Далі розглянемо процес написання програм для трьох методів аналізу коду. Алгоритми цих методів представлені в розділі 2.

3.2. **Розробка програми метода text-based**

Метод text-based ініціалізується першим при запуску функції системи аналізу. Створюємо не типізовані глобальні змінні зі сталими величинами. (скріншот рис. 3.5).

```
var plagword1 = 0;
var plagword2 = 0;
var plagbloc1 = 0;
var plagbloc2 = 0;
var ezkey1 = -1;
var ezkey2 = -1;
var hk = 0;
var hardkey2 = 0;
var superhardkey1 = 0;
var superhardkey2 = 0;
```

Рисунок 3.5. – Створення глобальних змінних

Далі потрібно було розробити код для отримання фрагментів коду у вигляді рядкових текстових змінних. Відомо, що при написані сайтів, html та javascript, код розділяється. Він надає об’єкти html, які створились за допомогою тегу <textarea> та ті, що були наділені особистим id для подальшого з’єднення з javascript при запуску методу text-based.

На рисунку 3.6 показан скріншот цього фрагменту коду.

```
<p><textarea id="txt1" style="width: -webkit-fill-available;" rows="40" cols="70" name="text1"></textarea></p>
</div>
<div id="rightSidebar">
        <p><textarea id="txt2" style="width: -webkit-fill-available;" rows="40" cols="70" name="text2"></textarea></p>
</div>
```

Рисунок 3.6. – Створення полей для вводу даних

Метод text-based з’єднується з елементами вводу html за допомогою персональних id . А за допомогою операції копіювання відбувається запис даних до нових строкових змінних. За допомогою функціонала js при записі змінних відсортовуються символи відступів.

На рисунок 3.7 показано програмну реалізацію цього процесу.

```
var text1 = document.getElementById("txt1").value;
var text2 = document.getElementById("txt2").value;
```

а)

```
Text1 = Text1.replace(/(\r\n|\n|\r)/gm, "");
Text2 = Text2.replace(/(\r\n|\n|\r)/gm, "");
```

б)

Рисунок 3.7. – Отримання даних: а) перезапис тексту з html об'єкту по id;
б) перезапис тексту без символів відступу

Для перетворення строкового типу даних в повноцінний масив слів використовуємо метод «split(" "). Цей метод розділяє текст по символу, який вказаний у дужках, та ігнорує запис подібних символі при їх безперервному використанні. Програмна реалізації методу показана на рисунку 3.8.

```
var Text1 = text1.toLowerCase();
var Text2 = text2.toLowerCase();
Text1 = Text1.replace(/(\r\n|\n|\r)/gm, "");
Text2 = Text2.replace(/(\r\n|\n|\r)/gm, "");
var Text1word1 = Text1.split(" ");
var Text2word1 = Text2.split(" ");
```

Рисунок 3.8. – Програмна реалізація функції «split»

Далі, за допомогою створення масивів слів, будемо відтворювати

порівняння слово в слово . Визначаємо довжину самого довшого фрагменту коду. З ним і будемо проводити порівнювати з другими фрагментами. Проходячи цикл з кількості слів та порівнюючи фрагменти коду будемо отримувати фіксацію збігів.(рис 3.9)

```
for (var i = 0; i < Text1word1.length; i++)
{
    Text1word += Text1word1[i];
}
    for (var i = 0; i < Text2word1.length; i++)
{
    Text2word += Text2word1[i];
}

if(Text1word.length<=Text2word.length)
{
var limit = Text2word.length;
    for (i = 0; i < Text2word.length; i++)
    {
        if (Text1word[i]==Text2word[i])
        {
            var plagword1 = plagword1 + 1;
            var plagword2 = plagword2 + 1;
        }
    }
}
else
{
var limit = Text1word.length;
    for (i = 0; i < limit; i++)
    {
        if (Text2word[i]==Text1word[i])
        {
            var plagword1 = plagword1 + 1;
            var plagword2 = plagword2 + 1;
        }
    }
}
```

Рисунок 3.9. – Програма порівняння вхідніх фрагментів

Запис отриманих результатів збігів перевірених кодів у вигляді відсоткового значення та вивід результатів користувачу, відбувається надсиланням значеня змінних і в html за допомогою особистих ір абзаців (рис. 3.10).

```
var plagword1 = plagword1 * 100 / Text1word.length;
var plagword2 = plagword2 * 100 / Text2word.length;
var procentplagiatword1 = "Перевірка слово в слово код1 = " + plagword1 + '%';
var procentplagiatword2 = "Перевірка слово в слово код2 = " + plagword2 + '%';

document.getElementById("word1").innerHTML = "Перевірка слово в слово код1 = " + plagword1 + '%';
document.getElementById("word2").innerHTML = procentplagiatword2 ;
```

Рисунок 3.10. – Програмна реалізіція надання результатів text-based методу

3.3. Реалізація методу Metrics-based

Метод metrics-based автоматично запускається після закінчення роботи методу text-based. Як відомо, метод базується на пошуку і фіксації ключових слів мови програмування та проводить очищення від символів, що можуть заважати в виявленні ключових слів («{» , «}», « (», «)», «;», «:», « [», «]»). Проводячи аналіз вхідного коду ці символи не видаляються а замінюються символом пробілу. Програмна вилучення зайвих символів показана на рисунку 3.11.

```
for (var i = 0; i < Text1.length; i++)
{
    var lol = Text1[i];
    if (lol =='{' || lol =='}' || lol =='(' || lol ==')' || lol ==';' || lol ==':' || lol =='[' || lol ==']')
    {
        normText1 += " ";
    }
    else
    {
        normText1 += lol;
    }
}
var normText2 = '';
for (var i = 0; i < Text2.length; i++)
{
    var kek = Text2[i];
    if (kek =='{' || kek =='}' || kek =='(' || kek ==')' || kek ==';' || kek ==':' || kek =='[' || kek ==']')
    {
        normText2 += " ";
    }
    else
    {
        normText2 += kek;
    }
}
```

Рисунок 3.11. – Вилучення зайвих символів

Для виявлених ключових слів було створено три масиви відповідно до мови програмування C#:

- масив ckl – зберігає текстове значення циклів;
  масив varibl –зберігає текстове значення змінних в мові C#; ;
- масив key – масив всіх іних ключових слів, які не потрапляють під параметри інших двох масивів.

Створення масивів ключових слів та локальних змінних показана на рисунку 3.12.

```
var key = ['asm', 'else','new', 'this', 'auto', 'enum', 'operator', 'throw', 'explicit','private','true','break','export','protected','try'
var ckl = ['while', 'for', 'do'];
var varibl = ['bool','char','float','short','int','long','double', 'string'];
var numkey1 = 0;
var numkey2 = 0;
var numckl1 = 0;
var numckl2 = 0;
var numvar1 = 0;
var numvar2 = 0;
var operato1=0;
var operator2=0;
var doperator1=0;
var doperator2=0;
var numb1=0;
var numb2=0;
var snormText1 = normText1.split(" ");
var snormText2 = normText2.split(" ");
```

Рисунок 3.12. – Розробка масивів ключових слів та локальних змінних

Відредагований текст аналізуємого програмного коду знову розбивають на масиви. Фрагмент розробленої реалізації пошуку ключів у вхідному тексті зображено на рисунку 3.13.

```
var procentplagiatkey1 = "Кількість ключових слів код1 = " + numkey1;
var procentplagiatkey2 = "Кількість ключових слів код2 = " + numkey2;

document.getElementById("key1id").innerHTML = procentplagiatkey1;
document.getElementById("key2id").innerHTML = procentplagiatkey2;

 for (var i = 0; i < snormText1.length; i++)
{
    for (var y = 0; y <= ckl.length; y++)
    {
        if (snormText1[i] == ckl[y])
        {
            numckl1++;
        }
    }
}
    for (var i = 0; i < snormText2.length; i++)
{
    for (var y = 0; y <= ckl.length; y++)
    {
        if (snormText2[i] == ckl[y])
        {
            numckl2++;
        }
    }
}

var procentplagiatckl1 = "Кількість циклів код1 = " + numckl1;
var procentplagiatckl2 = "Кількість циклів код2 = " + numckl2;

document.getElementById("ckl1id").innerHTML = procentplagiatckl1;
document.getElementById("ckl2id").innerHTML = procentplagiatckl2;

 for (var i = 0; i < snormText1.length; i++)
{
    for (var y = 0; y <= varibl.length; y++)
    {
```

Рисунок 3.13 – Фрагмент реалізації пошуку ключів у вхідному тексті

Вище було зазначено , що було реалізовано пошук ключів які можна виявити методом послідовного аналізу. Слід зауважити ,що зустрічаються в програмному коді операторі та подвійні оператори.

Для виявлення операторів потрібно посимвольно проаналізувати перевіряємий програмний код. Приклад реалізації пошуку операторів показано на рисунку 3.14.

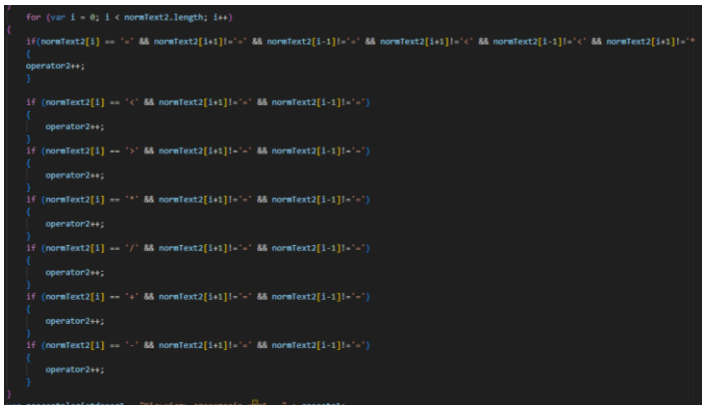


Рисунок 3.14. – Пошук операторів

Крім операторів та подвійних операторів ключовим словом являються числа. Лічильник чисел буде збільшуватись, коли буде знайдено число (рис. 3.15).

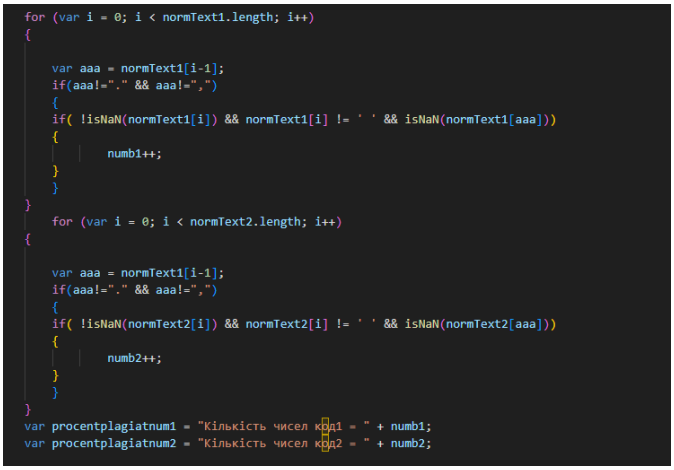


Рисунок 3.15. – Код виявлення чисел

3.4 Реалізація методу Token-based

Текст перетворюється в послідовність лексем, яка потім сканується для

пошуку дублікатів, а дві порожні строкові змінні являються сховищем, що має бути заповнене послідовністю лексем (рис.3.16).

```
var tokenText1 = '';
var tokenText2 = '';
```

Рисунок 3.16. – Сховища лексем

За зразком методів metrics-based необхідно проаналізувати надані користувачем фрагменти коду та виявити в них ключові символи і відтворити їх у вигляді символьних позначень.

В новий текст при по символьному аналізу коду переписується символьне значення, відносно типу знайденого фрагменту.

Правила запису символів:

- «/» –при ваиявленні будь-якої дужки, крапки з комою, та двокрапки;
- «f» –при виявленні циклу «for»;
- «w» –при виявленні циклу «while»;
- «o» –при виявленні операторів;
- «n» – записується при виявленні числа;

Розроблена програмна реалізація виявленя ключових слів показана на рисунку 3.17.

```
        tokenText2 += " ";
    }
    if(Text2[i] == 's' && Text2[i+1] == 't' && Text2[i+2] == 'r' && Text2[i+2] == 'i' && Text2[i+3] == 'n' && Text2[i+3] == 'g')
    {
        tokenText2 += "var";
        tokenText2 += " ";
    }
    if(Text2[i] == "<" && Text2[i+1] == "=")
    {
        tokenText2 += "o";
        tokenText2 += " ";
    }
    if(Text2[i] == ">" && Text2[i+1] == "=")
    {
        tokenText2 += "o";
        tokenText2 += " ";
    }
    if(Text2[i] == "=" && Text2[i+1] == "=")
    {
        tokenText2 += "o";
        tokenText2 += " ";
    }
    if(Text2[i] == "=" && Text2[i+1] == ">")
    {
        tokenText2 += "o";
        tokenText2 += " ";
    }
    if(Text2[i] == "=" && Text2[i+1] == "<")
    {
        tokenText2 += "o";
        tokenText2 += " ";
    }
    if(Text2[i] == "&" && Text2[i+1] == "&")
    {
        tokenText2 += "o";
        tokenText2 += " ";
    }
    if(Text2[i] == "|" && Text2[i+1] == "|")
    {
```

Рисунок 3.17 – Фрагмент коду Заповнення строкової змінної послідовністю лексем

Спрощений текст розбивається за допомогою методу split(). Далі порівняння проводиться слово в слово (рис. 3.17).

```
}
var tokenTextword1 = tokenText1.split(" ");
var tokenTextword2 = tokenText2.split(" ");
var tokenplag1=0;
var tokenplag2=0;
if(tokenTextword1.length<=tokenTextword2.length)
{
var limit = tokenTextword1.length;
    for (i = 0; i < limit; i++)
    {
        if (tokenTextword1[i]==tokenTextword2[i])
        {
            var tokenplag1 = tokenplag1 + 1;
            var tokenplag2 = tokenplag2 + 1;
        }
    }
}
else
{
var limit = tokenTextword2.length;
    for (i = 0; i < limit; i++)
    {
        if (tokenTextword2[i]==tokenTextword1[i])
        {
            var tokenplag1 = tokenplag1 + 1;
            var tokenplag2 = tokenplag2 + 1;
        }
    }
}
```

Рисунок 3.18. – Порівняння Послідовностей лексем.

Результат перевірки програмного коду перетворено у відсоткове значення (рис. 3.19)

```
var tokenplag1 = tokenplag1 * 100 / tokenTextword1.length;
var tokenplag2 = tokenplag2 * 100 / tokenTextword2.length;
var procenttokenplag1 = "Результат токенізації код1 " + tokenplag1 + "%";
var procenttokenplag2 = "Результат токенізації код2 " + tokenplag2 + "%";

document.getElementById("tok1id").innerHTML = procenttokenplag1;
document.getElementById("tok2id").innerHTML = procenttokenplag2;
```

Рисунок 3.19. – Результат роботи порівняння.

3.5. Результати роботи системи

Для можливості користування веб-сайтом на ПК повинен бути підключений Інтернет та будь-який браузер (Opera, Firefoxі т. д.). Якщо розмістити сайт на хостінгу в відкритому доступі, то у сайта автоматично

створюється URL- адреса на основі назви вашого сайту. За цією URL-адресою користувачі зможуть отримати доступ до сайту через Інтернет.

Для запуску веб-сайту потрібно відкрити браузер та перейти по відповідній адресі. Коли відкриється сторінка веб-сайту, потрібно заповнити поля кодами, які бажаєте перевірити на плагіат (рис. 3.20).

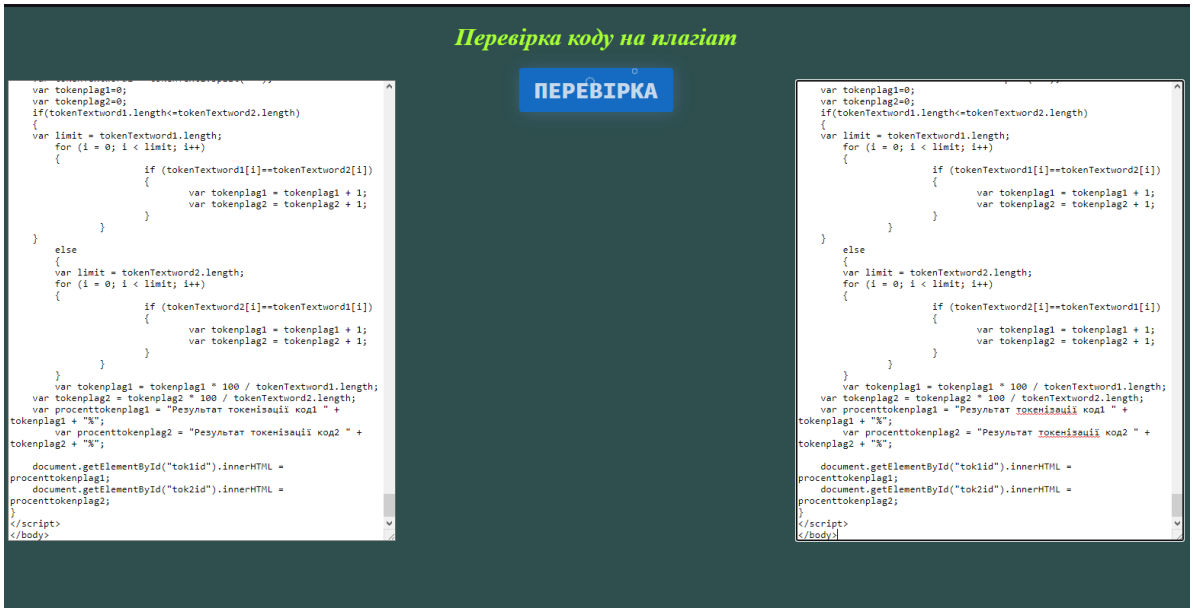


Рисунок 3.20– Заповнені поля для вводу коду

Після натискання на кнопку «Перевірка», програма видасть результати перевірки коду на плагіат (рис.3.21) .

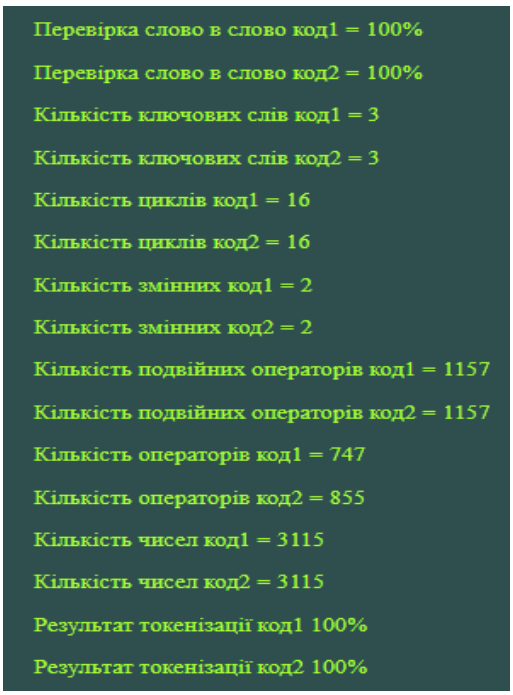


Рисунок 3.21– Результати роботи системи

Розробка програмних кодів веб - орієнтованої системи представлена в додатку А.

Висновки до третього розділу.

В третьому розділі кваліфікаційної роботи розглядається програмна реалізації веб - орієнтованої системи для перевірки плагіату в програмних кодах. Перевірка коду на плагіат,- це аналіз двох кодів на виявлення їх ідентичності. Під плагіатом програмного коду вважають копіювання (відтворення) вихідного коду без подальшого підтвердження справжності. Сюди також можна віднести адаптацію поміркованого відтворення чужої роботи або включення фрагментів вихідного коду у ваш власний код.

Для виконання поставлених задач було проаналізовано три методи – три рівня плагіату та розроблені програмні коди та система в цілому. WEB - орієнтована система для аналізу коду - це сайт, який має ієрархічну структуру та складається з 4 файли типу html (чотирьох сторінок - головної та інформаційних).

Всі методи аналізу на плагіат та т сама веб- система реалізовані в середовищі Visual Studio Code та з вбудованою підтримкою JavaScript, TypeScript і Node.js і має систему розширень для інших мов: C++, C#, Java, Python, PHP, Go, .NET та має гнучкий API. Visual Studio Code поставляється з IntelliSense для JavaScript, TypeScript, JSON, CSS і HTML і підтримує налагодження Node.js.

ВИСНОВКИ

Результатом виконання кваліфікаційної магістерської роботи є дослідження та розробка програмного забезпечення для аналізу на унікальність текстів та кодів. Робота має три розділи.

В першому розділі було проведено дослідження предметної області, а саме проведено аналіз он-лайн Інтернет – сервісів для виявлення плагіату в текстових документах та програмних кодах, висвітлені недоліки оглянутих програм. Перевірка одного й того ж самого тексту на різних програмах, дає різні дані про унікальність. Унікальність роботи при перевірки на плагіат, є - результатом роботи певної програми, яка працює з певною базою, з певним налаштуваннями та в певних умовах. Тому вона не є непорушною константою і може суттєво варіюватися.

В другому розділі роботи розглянута загальна класифікація підходів до виявлення плагіату та алгоритми комп'ютерного виявлення схожості тексту. Всі алгоритми ґрунтуються на текстовій схожості, крім одного - аналізу структури цитування, було зазначено ряд факторів для виявлення плагіату.

В розділі було розглянуто методи виявлення плагіату у програмному коді. Перевірка коду на плагіат,- це аналіз двох кодів на виявлення їх ідентичності. Під плагіатом програмного коду вважають копіювання (відтворення) вихідного коду без подальшого підтвердження справжності.
Сюди також можна віднести адаптацію поміркованого відтворення чужої роботи або включення фрагментів вихідного коду у ваш власний код. В розділі було розглянуто три найбільш відомих метода та алгоритму виявлення плагіату у програмному коді, розроблено блок - схеми алгоритмів до них.

В третьому розділі кваліфікаційної роботи розглядається програмна реалізації веб - орієнтованої системи для перевірки плагіату в програмних кодах, описані етапи розробки веб - системи для аналізу програмного коду, описуючи етапи створення основних методів роботи системи.

Для створення програмного продукту було обрано:середовище

програмування Visual Studio Code; мову програмування JavaScript; гіпертекстову розмітку - HTML; стиль сайту було створено при використанні каскадних таблиць стилю CSS.

Програма являє собою систему, що допомагає проаналізувати код та надати дані для виявлення плагіату.

Розробники програмних додатків та програмного забезпечення, працівники сфери IT, студенти, кампанії, викладачі, працюючи з розробленою вебсистемою, зможуть ефективно перевіряти оригінальність свого коду та запобігати випадкам плагіату.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Roy C. K. and Cordy J. R.. A survey on software clone detection research, Tech.Rep. 2007-541, School of Computing, Queen's University, Kingston, Ontario,Canada, 2013., pages 43-59.
2. ГОСТ 19.001-77 ЄСПД. Загальні положення.
3. JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/JavaScript
4. WhatIs [Електронний ресурс] – Режим доступу до ресурсу: https://www.techtarget.com/whatis/definition/browser
5. Oxford learners dictionaries [Електронний ресурс] – Режим доступу до ресурсу: https://www.oxfordlearnersdictionaries.com/definition/english/website?q=website
6. Sitesaga [Електронний ресурс] – Режим доступу до ресурсу: https://www.sitesaga.com/what-is-a-website/
7. Hostinger tutorilals [Електронний ресурс] – Режим доступу до ресурсу: https://www.hostinger.com/tutorials/what-is-html
8. Careerfoundry [Електронний ресурс] – Режим доступу до ресурсу: https://careerfoundry.com/en/blog/web-development/what-is-html-a-beginners-guide/
9. TechTarget [Електронний ресурс] – Режим доступу: https://www.theserverside.com/definition/HTML-Hypertext-Markup-Language
10. Investopedia [Електронний ресурс] – Режим доступу до ресурсу: https://www.investopedia.com/terms/h/html.asp
11. Welcome to aiogram’s documentation [Електорнний ресурс] – Режим доступу до ресурсу: https://docs.aiogram.dev/en/latest/
12. Geeksforgeeks [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/javascript/

13. Javascript.info [Електронний ресурс] – Режим доступу до ресурсу: https://uk.javascript.info/intro
14. Microsoft [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/dotnet/csharp/
15. Офіційний сайт КНУТД [Електронний ресурс] – Режим доступу до ресурсу: https://www.knutd.edu.ua/
16. Javatpoint [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/how-does-javascript-work
17. Noodle Printing Machine [Електронний ресурс] – Режим доступу до ресурсу: https://www.npmjs.com/package/typescript
18. Alexsoft [Електронний ресурс] – Режим доступу до ресурсу: https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation/
19. Archlinux [Електронний ресурс] – Режим доступу до ресурсу: https://wiki.archlinux.org/title/Visual_Studio_Code
20. Flutter [Електронний ресурс] – Режим доступу до ресурсу: https://docs.flutter.dev/tools/vs-code
21. Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/docs
22. Berkeley Extension [Електронний ресурс] – Режим доступу до ресурсу: https://bootcamp.berkeley.edu/resources/coding/learn-css/how-does-css-work/
23. W3schoolsUa [Електронний ресурс] – Режим доступу до ресурсу: https://w3schoolsua.github.io/css/index.html#gsc.tab=0
24. Конспект онлайн марафону GoIT [Електронний ресурс] – Режим доступу до ресурсу: https://www.edu.goit.global/uk/learn/14010952/11452717/11452728/textbook
25. Офіційний сайт IBM [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/topics/web-hosting

26. Namecheap [Електронний ресурс] – Режим доступу до ресурсу: https://www.namecheap.com/hosting/what-is-web-hosting-definition/
27. Netlifly [Електронний ресурс] – Режим доступу до ресурсу: https://www.netlify.com/p/enterprise-trial/?utm_content=eyebrow
28. ResearchGate [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/publication/228949390_A_token-based_code_clone_detection_tool-ccfinder_and_its_empirical_evaluation
29. Поняття, структура та різновиди веб-сайтів. Автоматизоване розроблення веб-сайтів[Електронний ресурс] – Режим доступу до ресурсу: http://www.ndu.edu.ua/liceum/web.pdf
30. Techopedia [Електронний ресурс] – Режим доступу до ресурсу: https://www.techopedia.com/definition/5411/website
31. Jeffrey E.F. Friedl Mastering Regular Expressions, 3rd Edition - O'Reilly Media, 2016. - 544
32. Gregory S. PostgreSQL 9.0: High Performance - Packt Publishing Ltd, 2010. - 442 c.
33. Alan McKee: Textual Analysis: A Beginner's Guide. - SAGE Publications Ltd, 2003. - 160 c. - ISBN: 978-0761949930
34. Martin Kay The Proper Place of Men and Machines in Language Translation. Research report CSL-80-11// Xerox Palo Alto Research Center, Palo Alto, CA./ Передруковано у 1997 році в Machine Translation 12: 3–23, 1997.(PDF)
35. Claudio Fantinuoli, Specialized Corpora from the Web and Term Extraction for Simultaneous Interpreters. // Baroni, Bernardini (ред.), WaCky! Working Papers on the Web as Corpus, С. 173-190, Wackybook.
36. Українська радянська енциклопедія : у 12-ти т. / гол. ред. М. П. Бажан ; редкол.: О. К. Антонов та ін. — 2-ге вид. — К. : Головна редакція УРЕ, 1974–1985.

37. Перебийніс В.С.. Математична лінгвістика . Українська мова : енциклопедія./ В. С. Перебийніс /— К. : Українська енциклопедія, 2014. — ISBN 966-7492-07-9
38. Бук С. Основи статистичної лінгвістики: Навчально-методичний посібник / Відп. ред. проф. Ф. С. Бацевич./— Львів: Видавничий центр ЛНУ імені Івана Франка, 2018.— 124 с.
39. Дарчук Н. П. Комп'ютерна лінгвістика (автоматичне опрацювання тексту):/ Н.П.Дарчук/ підручник.— К.: Видавничо-поліграфічний центр “Київський університет”, 2012.— 351 с.
40. Карпіловська Є. А. Вступ до комп’ютерної лінгвістики/ Є. А. Карпіловська/ — Донецьк: Юго-Восток, 2003.— 184 с.
41. Ланде Д. В. Елементи комп’ютерної лінгвістики в правовій інформатиці.// Д.В.Ланде //— К.: НДІІП НАПрН України, 2014. — 351 с. — ISBN 978-966-2344-33-2.
42. Пасічника В. В. Математична лінгвістика .Квантитативна лінгвістика / В. В. Пасічник, Ю. М. Щербина, В. А. Висоцька, Т. В. Шестакевич / навч. посіб. Кн.1 :: Новий Світ-2010, 2012. – 359 с.
43. Астістова Т.І Розробка автоматизованої системи аналізу текстів «Антиплагіат» / Т.І. Астістова, В.О.Керіб. //Інформаційні технології в науці, виробництві та підприємництві: зб. наук. праць молодих вчених, аспірантів, магістрів кафедри інформаційних технологій проектуванн. - К. : КНУТД, 2017. С. 118-121 – ISBN 978-966.
44. Астістова Т. І. Дослідження сервісів для аналізу інформації на унікальність / Т. І. Астістова, В. В. Глущенко // Інформаційні технології в науці, виробництві та підприємництві : збірник наукових праць молодих вчених, аспірантів, магістрів кафедри комп’ютерних наук та технологій / за заг. наук. ред. В. Ю. Щербаня. – Київ : ТОВ "Фастбінд Україна", 2023. – С. 98-10 https://er.knutd.edu.ua/handle/123456789/24119
45. Астістова Т.І, Аналіз сервісів для порівняння тексту та кодів/

Т. І. Астістова, В. В. Глущенко // Тези ІІ Міжнародної науково-практичної Інтернет конференції молодих учених та студентів: «Електромеханічні, інформаційні системи та нанотехнології»- Київ, КНУТД, 20 квітня 2023- С.87-89 https://er.knutd.edu.ua/handle/123456789/20650