МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

дискретного аналізу та інтелектуальних систем
(повна назва кафедри)

Дипломна робота

АВТОМАТИЗОВАНА СИСТЕМА ПЕРЕВІРКИ ТЕКСТІВ НА ПЛАГІАТ

Виконав: студент групи ПМІ-45
спеціальності
122 «Комп'ютерні науки»
(шифр і назва спеціальності)

Петах Н. В.
(підпис) (прізвище та ініціали)

Керівник проф. Притула М. М.,
конс. ас. Прядко О. Я.
(підпис) (прізвище та ініціали)

Рецензент
(підпис) (прізвище та ініціали)

2023

АНОТАЦІЯ

Ця дипломна робота описує створення вебзастосунку для виявлення плагіату у тексті з використанням широкого спектру технологій, таких як обробка природної мови (NLP), машинне навчання(ML) та інформаційний пошук (IR). Інструмент розроблений на базі бібліотеки NLTK та API пошукової системи Google, що дозволяє попередньо обробляти та токенізувати текст, видаляти стоп-слова та визначати найбільш релевантні терміни та фрази. Крім того, інструмент розбиває введений текст на менші фрагменти та запускає кожен з них через API пошукової системи Google окремо, що дозволяє отримати більш точну оцінку плагіату.

Даний інструмент виявлення плагіату має декілька переваг порівняно з наявними інструментами, такими як PlagAware, iThenticate та PlagScan. Вебзастосунок розпізнавання плагіату у тексті використовує потужні можливості обробки природної мови та інформаційного пошуку, а також може обробляти великі обсяги тексту, що дозволяє отримувати більш точні результати. Крім того, вебінтерфейс інструменту є зручним для користувачів, що дозволяє легко вводити текст та переглядати результати. Узагальнюючи, ця дипломна робота демонструє ефективність використання широкого спектру технологій для створення інструменту для виявлення плагіату.

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет Прикладної математики та інформатики
Кафедра Дискретного аналізу та інтелектуальних систем
Спеціальність 122 «Комп'ютерні науки»
(шифр і назва)

«ЗАТВЕРДЖУЮ»
Завідувач кафедри
проф. Притула М. М.,

"31" серпня 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНУ У РОБОТУ СТУДЕНТУ

Петах Норберт Вікторович
(прізвище, ім'я, по батькові)

1 .Тема роботи: Автоматизована система перевірки тексту на плагіат

керівник роботи: проф. Притула М. М., конс. ас. Прядко О. Я.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені Вченою радою факультету від **"13" вересня 2022 року № 15**

2. Строк подання студентом роботи **13.06.2023р.**

3. Вихідні дані до роботи: Статті та публікації про плагіат, його види та методи його уникнення. Онлайн сервіси перевірки текстів на плагіат.

4. Зміст дипломної роботи (перелік питань, які потрібно розробити): Розробити вебсайт перевірки тексту на плагіат, який має можливість вводу тексту напряму та з файлу. Показувати відсоток та джерела плагіату.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Графічне порівняння контролю плагіату за допомогою PlagAware, iThenticate та PlagScan. Макети сторінок перевірки тексту напряму та з файлу.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання **31 серпня 2022 р.**

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Ознайомлення з видами плагіату, методами його уникнення та розпізнавання.	20/09/2022 – 10/10/2022	Виконано
2	Дослідження та аналіз існуючих онлайн платформ перевірки тексту на плагіат.	11/10/2022 – 16/10/2022	Виконано
3	Вибір середовища розробки та стеку технологій.	16/10/2022 – 08/11/2022	Виконано
4	Вибір API та потрібних бібліотек.	09/11/2022 – 25/12/2022	Виконано
5	Інтеграція API пошукової системи Google та розробка backend частини.	26/12/2022 – 05/02/2023	Виконано
6	Тестування backend частини.	06/02/2023 – 23/02/2023	Виконано
7	Розробка frontend частини.	24/02/2023 – 20/03/2023	Виконано
8	Поєднання backend та frontend частини.	21/03/2023 – 10/04/2023	Виконано
10	Тестування функціоналу розробленого застостунку.	11/04/2023 – 21/04/2023	Виконано
11	Написання тексту роботи.	22/04/2023 – 13/06/2023	Виконано

Студент _____ **Петах Н. В.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

Плагіат став повсюдною проблемою в сучасному суспільстві. Легкість, з якою можна отримати доступ до інформації та скопіювати її, зросла в геометричній прогресії з розвитком інтернету, і, як наслідок, плагіат став більш поширеним, ніж будь-коли раніше. Згідно з дослідженням, проведеним Turnitin, провідним постачальником послуг з виявлення плагіату, понад 50% старшокласників зізналися, що копіювали текст безпосередньо з інтернету, а 36% повідомили, що купували роботу з онлайн-джерела [1]. Ця тенденція не обмежується лише студентами; вчені, фахівці та журналісти були визнані винними у плагіаті в останні роки.

З розвитком інтернету людям стало легше, ніж будь-коли, отримувати доступ до інформації з різних джерел і копіювати її, що призвело до поширення плагіату як в академічному, так і в професійному середовищі. Інструменти виявлення плагіату стають все більш важливими для виявлення випадків плагіату та збереження цілісності письмових робіт.

У зв’язку з цим, метою моєї дипломної роботи є створення сервісу перевірки текстів на плагіат. Вебзастосунок реалізований за допомогою мови високого рівня програмування Python із використанням фреймворку Django та інструментом для обробки природної мови NLTK.

Завдання дипломної роботи:

- Ознайомлення з видами плагіату, методами його уникнення та розпізнавання;
- Дослідження та аналіз існуючих онлайн платформ перевірки тексту на плагіат;
- Інтегрування API пошукової системи Google;
- Використання біліотеки nltk для обробки природної мови;
- Написання алгоритму для перевірки схожості текстів;
- Створення дизайну та інтерфейсу застосунку;

- Реалізація можливості введення тексту зручними для користувача методами;
- Тестування створеного застосунку для перевірки його якості, функціональності та надійності.

Також у дипломній роботі розглянуто різні методи і способи, що використовуються для створення детекторів плагіату, з акцентом на машинному навчанні та обробці природної мови. Розглянуто виклики, пов'язані зі створенням ефективного детектора плагіату, включаючи питання попередньої обробки даних, функціональної інженерії та вибору алгоритму.

Ця дипломна робота має велике значення та практичне застосування в сучасному інформаційному суспільстві для боротьби з плагіатом, підвищення якості освіти, захисту прав інтелектуальної власності та розвитку технологій.

Результати роботи можуть бути корисні для ефективно виявляти випадків плагіату в текстових матеріалах, забезпечення дотримання науково-етичних норм у наукових та академічних роботах, вдосконаленню технологій обробки природної мови та аналізу даних.

1. ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1.1 Загальні положення про плагіат

Плагіат — це неетична практика використання чужої інтелектуальної власності без належного визнання або згоди. Він є порушенням академічної доброчесності і може підірвати автентичність досліджень і творчих робіт. Плагіат включає не лише пряме копіювання, але й перефразування, використання неповних цитат та само-плагіат[2]. Це серйозна проблема, яка вимагає уваги і дій для підтримки якості та оригінальності наукової роботи.

Плагіат — це серйозна проблема в академічній та творчій сферах, яка охоплює не лише пряме копіювання чужої роботи, але й неналежне посилання на джерела. Важливо уникати плагіату з кількох причин.

По-перше, принципово нечесно видавати чужу роботу за власну.

По-друге, плагіат підриває процес навчання, оскільки позбавляє можливості розвивати критичне мислення та навички письма. Натомість він сприяє розвитку культури лінощів та обману.

По-третє, плагіат є неповагою до оригінального автора, який доклав зусиль для створення твору. Він позбавляє його визнання та пошани до його праці. Варто зазначити, що плагіат поширюється не лише на письмові роботи, а й на інші форми творчої та академічної діяльності. Це форма крадіжки інтелектуальної власності, яка є кримінальним злочином, якщо здійснюється з метою отримання прибутку. Тому вкрай важливо підтримувати академічну доброчесність і належним чином посилатися на першоджерела інформації, щоб забезпечити автентичність і якість своєї роботи.

1.1.1 Види плагіату

Для того, щоб зрозуміти різні види плагіату, важливо зазначити, що існує сім найпоширеніших форм плагіату [3]:

- Повний плагіат — це відкритий тип плагіату відбувається, коли автор подає чужу роботу від свого імені. Заплатити комусь за написання роботи,

а потім здати її з вашим ім'ям є актом повного плагіату, так само як і крадіжка або «запозичення» чужої роботи та подання її як власної. Прикладом повного плагіату є подання на заняття з англійської мови наукової роботи, яку написала і подала ваша старша сестра, коли відвідувала цей курс п'ять років тому.

- Прямий плагіат — схожий на повний плагіат тим, що він також є відкритою подачею слів іншого автора за свої власні. Різниця між ними полягає в тому, яка частина роботи є плагіатом. При повному плагіаті копіюється вся робота. При прямому плагіаті включаються окремі розділи або абзаци без зазначення (або навіть визнання) автора. Прикладом прямого плагіату є вставка одного-двох рядків з вашого джерела безпосередньо у вашу роботу без цитування або посилання на джерело.
- Перефразування — це форма плагіату, що відбувається, коли автор повторно використовує чужу роботу і змінює кілька слів або фраз. Це поширений тип плагіату, і багато студентів навіть не усвідомлюють, що це форма плагіату. Але якщо ви представляєте чиюсь оригінальну ідею у своїй роботі, не посилаючись на неї, навіть якщо ви представляєте її своїми словами, це є плагіатом.
- Само-плагіат: повторне використання контенту з власної роботи у іншій роботі буде актом само-плагіату. Ми можемо використовувати ті ж самі джерела, і якщо правильно їх процитуємо, то не доведеться турбуватися про те, що вас звинуватять у плагіаті. Само-плагіат може бути проблемою, якщо хтось пише на замовлення. Коли вам доручають написати роботу для клієнта, він є власником цієї роботи. Повторне використання ваших власних слів для наступних клієнтів є плагіатом вашої власної роботи і може зашкодити вашій професійній репутації (а також виставити ваших клієнтів у поганому світлі).
- Шматковий плагіат (також відомий як мозаїчний плагіат) — стосується випадків, коли плагіат переплітається з оригінальною роботою автора. Цей

вид плагіату може бути малопомітним, його легко не помітити, і він може відбуватися в поєднанні з прямим плагіатом. Прикладом шматкового плагіату є взяття речення з джерела і вбудовування його у власне речення.

- Плагіат на основі джерела — це вид плагіату, при якому автор може правильно цитувати свої джерела, але представляти їх у неправдивий спосіб. Наприклад, автор може посилатися на вторинне джерело у своїй роботі, але посилатися лише на першоджерело, з якого походить це вторинне джерело. Інші приклади включають посилання на неправильне джерело і навіть вигадування джерел.
- Випадковий плагіат — це, мабуть, найпоширеніший тип плагіату, оскільки він трапляється, коли автор не усвідомлює, що плагіатує чужу роботу. Випадковий плагіат включає наступне: забуття посилань на джерела у вашій роботі, неправильне цитування джерел, невикористання лапок навколо цитованого матеріалу.

1.1.2 Методи уникнення плагіату у написаному тексті

Плагіат — це серйозне правопорушення, яке може мати значні наслідки для авторів і дослідників. Навіть ненавмисний плагіат, коли хтось несвідомо використовує чужу роботу без належного зазначення авторства, може призвести до академічних і професійних санкцій. У наукових дослідженнях, де точність і доброчесність є надзвичайно важливими, уникнення ненавмисного плагіату має вирішальне значення. У цій частині ми обговоримо деякі практичні поради та стратегії, які автори можуть використовувати, щоб уникнути ненавмисного плагіату у своїй роботі. Дотримуючись цих рекомендацій, дослідники можуть гарантувати, що їхня робота є оригінальною, етичною та підтримує цілісність наукової спільноти.

На щастя, не все так страшно. Уникнути плагіату насправді легко, якщо існує добре розуміння, що це таке. Для запобігання плагіату у вашому тексті варто звернути увагу на наступні рекомендації [4]:

- Необхідно посилатися на джерело: посилаючись на ідею або формулювання, які не належать вам, додайте в тексті посилання, яке вказує повну назву джерела, дату його публікації та будь-який інший елемент цитування, який вимагається стилем, якого ви дотримуєтесь.
- Варто включати цитати: якщо ви дослівно цитуєте слова джерела у своєму тексті, одним із найпростіших і водночас найочевидніших способів уникнути плагіату є використання лапок навколо тексту, щоб позначити, що ці слова не належать вам. Пряма цитата також повинна містити посилання на джерело, щоб читачі знали, звідки вона взята.
- Перефразування думки — це переписування ідей або інформації джерела своїми словами, не змінюючи його змісту. Але варто бути обережним, оскільки неправильне перефразування може перетворитися на плагіат.

Успішне перефразування без плагіату вимагає певних зусиль. Щоб уникнути ненавмисного плагіату під час перефразування, потрібно виконати обережну і вправну роботу над осмисленням та подачею ідеї, що була взята з оригінального тексту. Це вимагає від вас не лише унікального перефразування змісту, але й уникнення використання надто великої кількості фраз чи слів, схожих на оригінал. Однак, роблячи це, необхідно бути обережним, щоб не змінити основний зміст оригінального тексту.

Важливо пам'ятати, що навіть якщо ви висловлюєте ідею своїми словами, сама ідея належить комусь іншому, а отже, необхідно належним чином віддати належне джерелу, процитувавши його належним чином. Дотримуючись цих рекомендацій, можна успішно перефразувати, не вдаючись до плагіату.

1.2 Засоби розпізнавання плагіату

Виявлення плагіату стає все більш важливим у сучасну цифрову епоху, коли інформація є широкодоступною та легкодоступною. Плагіат може бути навмисним або ненавмисним, але незалежно від наміру, він вважається серйозним академічним та етичним порушенням.

Інструменти виявлення плагіату допомагають виявити випадки плагіату в письмовій роботі, порівнюючи її з базою даних раніше опублікованих робіт, а також з іншими онлайн-джерелами. Ці інструменти використовують передові алгоритми та методи обробки природної мови для виявлення випадків плагіату, зокрема прямого копіювання, перефразування та клаптикового плагіату.

Використовуючи інструменти виявлення плагіату, викладачі та установи можуть забезпечити академічну доброчесність і підтримувати стандарти чесності та оригінальності в науковій роботі. Студенти та дослідники також можуть використовувати ці інструменти для перевірки власної роботи на ненавмисний плагіат, що дає їм можливість виправити будь-які потенційні проблеми до її подання.

1.2.1 Підходи виявлення плагіату

Існує три основні підходи до розпізнавання плагіату: ручне виявлення, виявлення на основі тексту та програмне забезпечення для виявлення плагіату.

Ручна перевірка

Традиційним підходом до виявлення плагіату є ручне виявлення. Він передбачає залучення людини-читача, яка читає текст і виявляє будь-які випадки плагіату.

Ручна перевірка забирає багато часу, і за її допомогою важко виявити малопомітні випадки плагіату. Крім того, людина-читач не завжди володіє необхідними знаннями в галузі, щоб розпізнати плагіат.

Виявлення на основі тексту

Виявлення плагіату на основі тексту використовує алгоритми зіставлення тексту для виявлення випадків плагіату. Він передбачає порівняння тексту, який перевіряється, з базою даних текстів для виявлення будь-яких збігів. Цей підхід ефективний для виявлення дослівного плагіату, коли текст копіюється слово в слово. Однак він не є ефективним для виявлення випадків, коли текст було перефразовано або переказано.

Програмне забезпечення для виявлення плагіату

Найефективнішим підходом до виявлення плагіату є використання програмного забезпечення для виявлення плагіату. Ці програмні інструменти використовують комбінацію текстового виявлення та алгоритмів машинного навчання для виявлення випадків плагіату. Обробка природної мови (NLP) та алгоритми машинного навчання допомагають програмному забезпеченню виявляти схожість у тексті, навіть якщо текст було перефразовано або переказано. Ці програмні інструменти також надають звіт, в якому висвітлюються випадки плагіату, що полегшує користувачеві їх виявлення та виправлення.

1.2.2 Методи та засоби розпізнавання плагіату

Існує великий спектр методів та засобів для виявлення плагіату. Як правило, їх розподіляють за галуззю застосування. За таким критерієм методи виявлення плагіату поділяються на:

- Внутрішнє виявлення плагіату — виявлення плагіату в документі без доступу до потенційного оригінального тексту. Також називається внутрішнім виявленням плагіату.
- Виявлення зовнішнього плагіату — полягає в порівнянні підозрілих на плагіат документів з потенційними оригінальними документами.
- Виявити плагіат у вихідному коді — відносно легше, ніж виявити плагіат у природній мові. Тому що в мовах програмування немає ні двозначності, ні інтерференції між словами. Але в природній мові кожне слово може мати багато синонімів і різних значень. Деякі методи виявлення плагіату не залежать від мови, а деякі залежать від мови.
- Виявлення плагіату в письмових документах — цей метод можна розділити на дві категорії, які називаються незалежне від мови виявлення плагіату та залежне від мови виявлення плагіату.

Мовно-незалежні методи базуються на оцінці характеристик тексту, які є спільними для всіх мов. Наприклад, кількість спеціальних символів і середня

довжина речення. Щоб ввести в оману незалежні від мови системи, можна використовувати прийоми перефразування.

Мовно-залежні методи виявлення плагіату ґрунтуються на оцінці характеристик тексту, характерних для однієї мови. Наприклад, підрахунок частоти вживання певного слова в конкретній мові. Мовно-залежне виявлення плагіату ефективніше, ніж мовно-незалежне.

Нас цікавлять методи, що працюють напряму з текстом в письмових документах та різних типах робіт. У цій галузі існують три ключові засоби реалізації методів розпізнання, що можуть лягти у основу розробки майбутнього вебзастосунку, а саме: методи, які реалізовані на основі стилометрії, методи, які орієнтуються на контекст тексту та обробка природної мови із застосуванням машинного навчання.

Методи, засновані на стилометрії

Стилометрія — це статистичний підхід, який використовується для визначення авторства. Він натхненний методами атрибуції авторства і в основному складається з класифікації стилів письма авторів для виявлення схожості. Він базується на припущенні, що кожен автор має унікальний стиль. Стиль письма можна проаналізувати, використовуючи фактори в межах одного документа або порівнюючи два документи одного автора. Для цього документи розбивають на частини, наприклад, на абзаци та речення. Потім виокремлюють і аналізують стильові особливості. Основними лінгвістичними стилометричними ознаками є статистика тексту, яка працює на рівні символів:

- синтаксичні ознаки для вимірювання стилю письма на рівні речень (довжина речень, використання службових слів тощо);
- набори слів закритого класу для підрахунку спеціальних слів (кількість стоп-слів, іноземних слів, складних слів тощо);
- структурні ознаки, що відображають організацію тексту (довжина абзаців, глав тощо).

На основі цих ознак можна вивести формули для визначення стилю письма автора. Методи, засновані на стилометрії, можна використовувати для внутрішнього та зовнішнього виявлення плагіату.

Методи, що базуються на змісті

Аналіз специфікацій текстів з точки зору логічної структури та виявлення схожості. Контент-орієнтовані методи можна використовувати лише при зовнішньому виявленні плагіату.

Методи виявлення плагіату на основі контенту стають дедалі популярнішими останніми роками, оскільки поширеність онлайн-контенту та легкість доступу до інформації полегшують копіювання та вставлення чужого контенту без належного зазначення авторства. Ці методи використовують алгоритми для аналізу змісту документа і порівняння його з корпусом існуючих документів, щоб визначити, чи є в тексті збіги або схожість.

Один із поширених методів виявлення плагіату на основі контенту називається «шинглінг». Шинглінг передбачає поділ документа на набір суміжних фраз або «шинглів», а потім порівняння цих шинглів з базою даних існуючих шинглів, щоб визначити, чи існують якісь збіги. Цей підхід ефективний для виявлення прямого копіювання та перефразування тексту, але він може бути не настільки ефективним для виявлення більш витончених форм плагіату, таких як переформування або зміна порядку речень.

Інший метод виявлення плагіату на основі контенту називається «дактилоскопіювання». Він передбачає створення унікального «відбитка пальця» документа на основі його основних характеристик, таких як частота вживання слів, довжина слів і структура речень. Потім цей відбиток можна порівняти з базою даних існуючих відбитків, щоб визначити, чи є схожість між документами. Цей підхід ефективніший для виявлення більш витончених форм плагіату, але він може вимагати більшої обчислювальної потужності і може бути менш точним для довгих документів.

Деякі методи виявлення плагіату на основі контенту також включають алгоритми машинного навчання, які можуть навчитися виявляти закономірності та схожість у тексті з плином часу. Це може допомогти підвищити точність виявлення плагіату, особливо для довших документів або тих, що містять складнішу мову чи ідеї.

Хоча методи виявлення плагіату на основі контенту можуть бути ефективними для виявлення випадків плагіату, важливо зазначити, що вони не є безвідмовними. Існує багато способів маніпулювання текстом, щоб уникнути виявлення, наприклад, зміна порядку слів, перефразування речень або використання синонімів. Крім того, деякі методи виявлення плагіату можуть давати хибно-позитивні або хибно-негативні результати, що може призвести до неправильних або оманливих результатів.

Обробка природної мови та машинне навчання для виявлення плагіату

NLP — це підгалузь комп'ютерних наук, яка займається взаємодією між комп'ютерами та природними людськими мовами. Вона використовує статистичні та обчислювальні методи для обробки, аналізу та розуміння текстів природною мовою. З іншого боку, ML — це галузь штучного інтелекту (ШІ), яка фокусується на розробці алгоритмів, здатних навчатися на даних і покращувати свою продуктивність з часом.

Методи виявлення плагіату на основі контенту, які використовують методи NLP і ML, стають дедалі популярнішими завдяки своїй високій точності та ефективності. Ці методи порівнюють текст, про який йде мова, з великим корпусом існуючих текстів і виявляють схожість, що вказує на плагіат. Вони працюють, витягуючи з тексту особливості, такі як n-грами, частота слів і синтаксичні структури, і використовуючи ці особливості для навчання моделей машинного навчання, які потім можуть класифікувати текст як оригінальний або плагіат.

Одним із поширених підходів до виявлення плагіату за допомогою NLP і ML є використання стило-метричного аналізу. Цей метод передбачає аналіз

стилю написання тексту, про який йде мова, і порівняння його зі стилями написання інших текстів у тому ж жанрі або сфері. Стилометричний аналіз розглядає такі особливості, як довжина речення, вибір слів і пунктуація, і використовує алгоритми машинного навчання для виявлення шаблонів, що вказують на плагіат.

Інший підхід полягає у використанні семантичного аналізу, який передбачає вивчення значення тексту, що розглядається, і порівняння його з іншими текстами. Цей метод шукає схожість в ідеях, висловлених у тексті, а також у словах, використаних для вираження цих ідей. Семантичний аналіз можна використовувати для виявлення перефразування та інших форм витонченого плагіату, які можуть бути пропущені іншими методами.

Отже, використання методів обробки природньої мови та машинного навчання для виявлення плагіату революціонізувало наш підхід до проблеми академічної доброчесності. Ці методи пропонують високий ступінь точності та ефективності і можуть бути використані для виявлення як явних, так і прихованих форм плагіату. Оскільки сфера НЛП і ML продовжує розвиватися, ми можемо очікувати появи ще більш досконалих і ефективних методів виявлення плагіату.

1.3 Приклади програмних рішень розпізнавання плагіату у тексті

Існує багато програмних систем, які припускають, що вони можуть достовірно визначити, чи є надісланий текст або онлайн-документ плагіатом, чи ні. Програмне забезпечення може лише порівнювати синтаксис на рівні символів або слів і визначати схожість між текстами. У сфері семантичного розпізнавання проводиться певна експериментальна робота. Але це здається успішним лише в області високо-структурованих текстів, таких як код програмної мови [5].

PlagAware — це пошукова система, яка є основним елементом у виявленні типового змісту заданих текстів (рис 1.1). Класична пошукова система використовується для виявлення та сканування плагіату і надає різні типи звітів,

які допомагають користувачеві або власнику документа вирішити, чи є його документ плагіатом, чи ні.

Рис 1.1. Графічне порівняння контролю плагіату за допомогою PlagAware.

Як і будь-яке інше програмне забезпечення, PlagAware має свої плюси і мінуси, які користувачі повинні враховувати, перш ніж вирішити, використовувати його.

Плюси PlagAware:

– Точні результати: PlagAware використовує передові алгоритми для аналізу текстів і порівняння їх з великою базою даних джерел. Це робить його надійним інструментом для точного виявлення плагіату.
– Простота у використанні: PlagAware має зручний інтерфейс, що робить його простим у використанні. Вам не потрібні спеціальні навички, щоб орієнтуватися в програмному забезпеченні. Просто завантажте документ і натисніть кнопку "перевірити плагіат", а програма зробить все інше.

- Підтримка різних форматів файлів: PlagAware підтримує різні формати файлів, зокрема Word, PDF і TXT, що полегшує перевірку на плагіат різних типів документів.
- Швидко та ефективно: PlagAware розроблено для швидкого та ефективного виявлення плагіату. Він може аналізувати великі обсяги тексту за лічені секунди, що робить його чудовим інструментом для приватних осіб та організацій, яким потрібно регулярно перевіряти документи на плагіат.
- Економічна ефективність: PlagAware пропонує доступні тарифні плани, які відповідають різним потребам. Користувачі можуть вибрати план, який відповідає їхньому бюджету та вимогам.

Мінуси PlagAware:

- Обмежена кількість джерел: Хоча PlagAware має велику базу даних джерел, вона може не охоплювати всі джерела в інтернеті. Це означає, що існує ймовірність пропустити деякі випадки плагіату.
- Хибні спрацьовування: Алгоритми PlagAware іноді можуть позначати текст як плагіат, навіть якщо він ним не є. Це може статися, коли текст має схожість з іншими джерелами, які не обов'язково є скопійованими.
- Обмежені звіти: Функції звітності PlagAware обмежені, і вона не може забезпечити всебічний аналіз тексту. Користувачам може знадобитися використання додаткових інструментів для більш поглибленого аналізу тексту.
- Обмежені можливості налаштування: PlagAware не дозволяє користувачам налаштовувати параметри виявлення плагіату. Це означає, що користувачі не можуть встановити конкретні параметри для виявлення плагіату відповідно до своїх уподобань.

Отже, PlagAware є надійним та ефективним інструментом виявлення плагіату, який має низку переваг, серед яких точність, простота використання та доступність. Однак, як і будь-який інший програмний інструмент, він має свої

недоліки, зокрема обмежену кількість джерел, хибні спрацьовування, обмеженість звітів та обмеженість налаштувань.

iThenticate розроблено спеціально для дослідників, видавців авторських прав та інших (рис. 1.2). Він призначений для використання установами, а не приватними особами, але, нарешті, вони надали лімітовану послугу з виявлення плагіату для окремих користувачів, таких як магістри та докторанти, а також дозволили їм перевіряти один документ обсягом до 25 000 слів. Таким чином, вони можуть скористатися цією послугою, щоб застрахуватися або перевірити свій проект дисертації на предмет правильності цитування та оригінальності змісту.

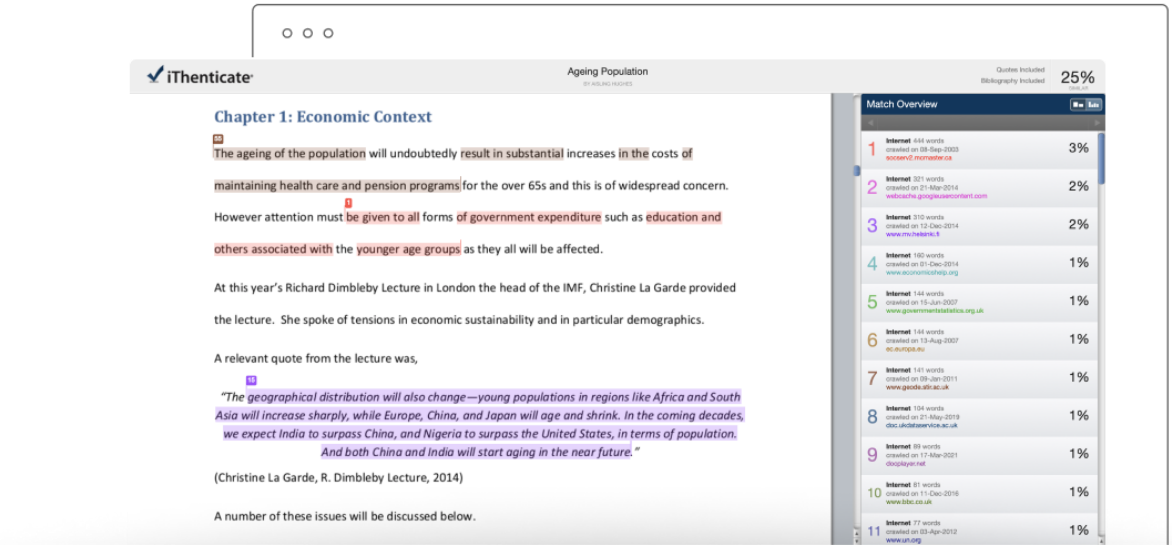

Рис 1.2. Графічне порівняння контролю плагіату за допомогою iThenticate.

Плюси iThenticate:

– Висока точність: iThenticate використовує передові алгоритми та велику базу даних джерел, щоб забезпечити високоточні результати виявлення плагіату.
– Комплексні звіти: iThenticate надає комплексні звіти, які аналізують текст і виділяють будь-які випадки плагіату. Звіти включають відсоток тексту, який вважається плагіатом, і джерела, з яких було скопійовано текст.
– Настроюванні параметри: iThenticate дозволяє користувачам налаштовувати параметри виявлення плагіату відповідно до їхніх потреб.

Користувачі можуть встановити певні параметри для виявлення плагіату відповідно до своїх уподобань.

- Підтримка різних форматів файлів: iThenticate підтримує різні формати файлів, зокрема Word, PDF і TXT, що полегшує перевірку на плагіат різних типів документів.
- Відмінна клієнтська підтримка: iThenticate пропонує відмінну клієнтську підтримку з командою експертів, готових допомогти користувачам з будь-якими питаннями або проблемами, які можуть у них виникнути.

Мінуси iThenticate:

- Висока вартість: iThenticate відносно дорогий інструмент для виявлення плагіату, і він може бути доступним не для всіх користувачів.
- Складність: iThenticate може бути складним у використанні, і користувачам може знадобитися певне навчання, щоб орієнтуватися у функціях та налаштуваннях програмного забезпечення.
- Обмеженість джерел: база даних джерел iThenticate може не охоплювати всі джерела в інтернеті. Це означає, що існує ймовірність пропустити деякі випадки плагіату.
- Хибні спрацьовування: алгоритми iThenticate іноді можуть позначати текст як плагіат, навіть якщо він ним не є. Це може статися, якщо текст має схожість з іншими джерелами, які не обов'язково скопійовані.

У підсумку, iThenticate — це високоточний інструмент виявлення плагіату з можливістю налаштування, який надає вичерпні звіти. Однак він є відносно дорогим і може бути занадто складним для деяких користувачів. Крім того, обмежена безкоштовна пробна версія, можливість хибних спрацьовувань та обмеженість джерел важливі фактори, які слід враховувати при прийнятті рішення про використання iThenticate.

PlagScan — це онлайн-програма для перевірки текстів на плагіат (рис 1.3). Його часто використовують навчальні заклади для надання різних типів облікових записів з різними функціями. PlagScan використовує складні

алгоритми для перевірки та аналізу завантаженого документа на предмет виявлення плагіату, засновані на сучасних лінгвістичних дослідженнях. Унікальний підпис витягується зі структури документа, який потім порівнюється з базою даних PlagScan і мільйонами документів в інтернеті.

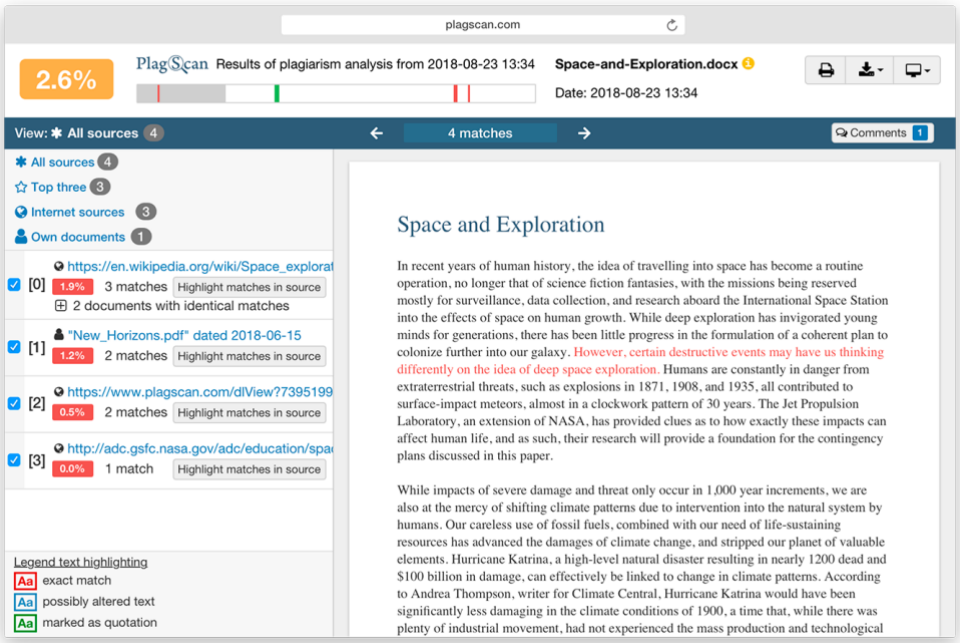

Рис 1.3. Графічне порівняння контролю плагіату за допомогою PlagScan.

Плюси PlagScan:

– Точні результати: PlagScan надає високоточні результати виявлення плагіату. Алгоритми програми можуть виявити навіть ледь помітну схожість між текстом та іншими джерелами.
– Вичерпні звіти: PlagScan надає вичерпні звіти, які аналізують текст і виділяють будь-які випадки плагіату. Звіти включають відсоток тексту, який вважається плагіатом, і джерела, з яких було скопійовано текст.
– Настроюванні параметри: PlagScan дозволяє користувачам налаштовувати параметри виявлення плагіату відповідно до своїх потреб. Користувачі можуть встановити певні параметри для виявлення плагіату відповідно до своїх уподобань.

- Підтримка різних форматів файлів: PlagScan підтримує різні формати файлів, зокрема Word, PDF і TXT, що полегшує перевірку на плагіат різних типів документів.
- Простота у використанні: PlagScan має зручний інтерфейс, що робить його простим у використанні. Вам не потрібні спеціальні навички, щоб орієнтуватися в програмному забезпеченні. Просто завантажте документ і натисніть кнопку "перевірити плагіат", а програма зробить все інше.

Мінуси PlagScan:

- Обмежена кількість джерел: База даних джерел PlagScan може не охоплювати всі джерела в інтернеті. Це означає, що існує ймовірність пропустити деякі випадки плагіату.
- Хибні спрацьовування: Алгоритми PlagScan іноді можуть позначати текст як плагіат, навіть якщо він ним не є. Це може статися, коли текст має схожість з іншими джерелами, які не обов'язково є скопійованими.
- Обмежені звіти: Можливості PlagScan у створенні звітів обмежені, і він не може забезпечити всебічний аналіз тексту. Користувачам може знадобитися використання додаткових інструментів для більш поглибленого аналізу тексту.
- Вартість: PlagScan є відносно дорогим порівняно з іншими інструментами виявлення плагіату, що може бути доступним не для всіх користувачів.

З усього сказаного вище можна зробити висновки, що PlagScan — це надійний і простий у використанні інструмент для виявлення плагіату, який надає точні результати та налаштовуванні параметри. Однак обмежена кількість джерел, помилкові спрацьовування, обмеженість звітів, вартість і обмежена безкоштовна пробна версія є значним мінусом для його застосування.

PlagiarismDetection.org — це онлайн-сервіс, який забезпечує високий рівень точності виявлення плагіату (рис.1.4). Він призначений для того, щоб допомогти викладачам і студентам підтримувати або запобігати та виявляти

плагіат у своїх академічних документах. Плагіат може бути виявлений з високим рівнем точності.

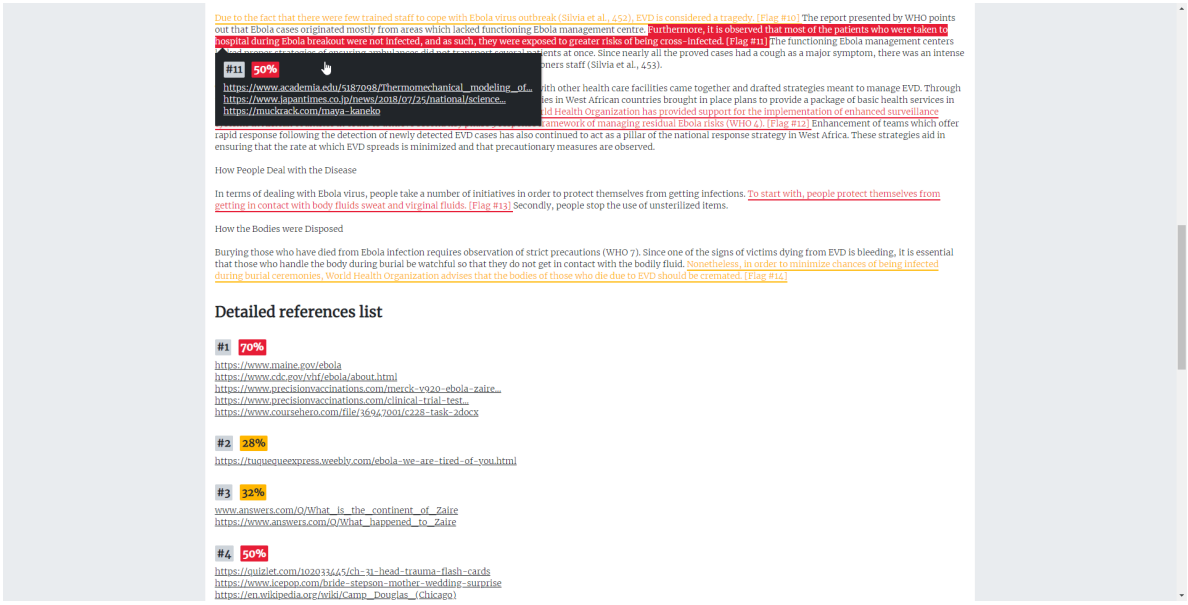

Рис 1.4. Демонстрація розпізнавання плагіату за допомогою PlagiarismDetection.org.

Плюси PlagiarismDetection.org:

- Висока точність: PlagiarismDetection.org надає високоточні результати виявлення плагіату. Алгоритми програми можуть виявити навіть ледь помітну схожість між текстом та іншими джерелами.
- Вичерпні звіти: PlagiarismDetection.org надає вичерпні звіти, які аналізують текст і виділяють будь-які випадки плагіату. Звіти включають відсоток тексту, який вважається плагіатом, і джерела, з яких було скопійовано текст.
- Налаштовуванні параметри: PlagiarismDetection.org дозволяє користувачам налаштовувати параметри виявлення плагіату відповідно до своїх потреб. Користувачі можуть встановити конкретні параметри для виявлення плагіату відповідно до своїх уподобань.
- Підтримка декількох форматів файлів: PlagiarismDetection.org підтримує кілька форматів файлів, зокрема Word, PDF і TXT, що полегшує перевірку на плагіат різних типів документів.

- Простота використання: PlagiarismDetection.org має зручний інтерфейс, що робить його простим у використанні. Користувачам не потрібні спеціальні навички, щоб орієнтуватися в програмному забезпеченні. Просто завантажте документ і натисніть кнопку "перевірити плагіат", а програма зробить все інше.

Мінуси PlagiarismDetection.org:

- Обмежена кількість джерел: База даних джерел PlagiarismDetection.org може не охоплювати всі джерела в інтернеті. Це означає, що існує ймовірність пропустити деякі випадки плагіату.
- Помилкові спрацьовування: Алгоритми PlagiarismDetection.org іноді можуть позначати текст як плагіат, навіть якщо він ним не є. Це може статися, коли текст має схожість з іншими джерелами, які не обов'язково скопійовані.
- Обмежена звітність: Функції звітування PlagiarismDetection.org обмежені, і він не може забезпечити всебічний аналіз тексту. Користувачам може знадобитися використання додаткових інструментів для більш поглибленого аналізу тексту.
- Вартість: PlagiarismDetection.org є відносно дорогим порівняно з іншими інструментами виявлення плагіату, що може бути доступним не для всіх користувачів.

Отже, PlagiarismDetection.org є надійним і простим у використанні інструментом для виявлення плагіату, який надає точні результати і параметри, що налаштовуються. Однак обмежена кількість джерел, помилкові спрацьовування, обмежена кількість звітів, вартість та обмежена кількість безкоштовних пробних версій важливі фактори, які слід враховувати при прийнятті рішення про використання PlagiarismDetection.org.

2. ЗАСОБИ РЕАЛІЗАЦІЇ ІНСТРУМЕНТУ ПЕРЕВІРКИ ТЕКСТІВ НА ПЛАГІАТ

2.1 Середовище розробки вебзастосунку перевірки на плагіат

При розробці вебзастосунку із використанням різноманітних функціональних особливостей, таких як перевірка на плагіат, важливо випрати правильні інструменти та середовище розробки. Нижче наведені мови програмування [6], які надають можливість реалізувати програмними засобами проект для даної дипломної роботи.

Однією з найпопулярніших мов програмування для веброзробки є JavaScript. Ця мова широко використовується для створення інтерактивних вебдодатків і може бути використана для створення детектора плагіату. JavaScript має багато бібліотек, які можна використовувати для обробки природної мови, що є важливим для виявлення плагіату. Однак JavaScript не найкращий вибір для розробки складних алгоритмів, які потрібні для більш просунутих методів виявлення плагіату.

Python — популярна мова, яку часто використовують для обробки природної мови. Python має багато бібліотек, таких як NLTK (Natural Language Toolkit) і spaCy, які можна використовувати для обробки та аналізу тексту. Крім того, Python має велику спільноту розробників, що дозволяє легко знайти підтримку та рішення будь-яких питань, які можуть виникнути під час розробки. Python також добре підходить для алгоритмів машинного навчання, які все частіше використовуються для виявлення плагіату.

Java — ще одна популярна мова програмування, яку можна використовувати для розробки програмного забезпечення для виявлення плагіату. Java являє собою високопродуктивну мову програмування, яку можна використовувати для складних алгоритмів і великомасштабних додатків. Java також має багато бібліотек для обробки природної мови та машинного навчання, що робить її підходящим вибором для розробки детектора плагіату.

Нарешті, Ruby — мова програмування, яку можна використовувати для розробки програмного забезпечення для виявлення плагіату. Ruby має чіткий і лаконічний синтаксис, що робить її легкою для читання і написання. Ruby також має багато бібліотек для обробки природної мови, але вона може бути не настільки придатною для складних алгоритмів, як Java або Python.

Отже, найкраща мова програмування для розробки вебдетектора плагіату залежить від конкретних вимог проекту. Для простіших проектів можуть підійти JavaScript або Ruby, тоді як для складніших проектів може знадобитися використання Python або Java. Зрештою, найважливішим фактором є вибір мови, яка має бібліотеки для обробки природної мови та машинного навчання, оскільки вони є важливими компонентами виявлення плагіату. Серед продемонстрованих тут мов найкраще для вирішення поставленого завдання підходять мови Python та Java, а отже необхідно детальніше порівняти їх та вибрати ключову мову для розробки вебзастосунку розпізнавання плагіату у тексті.

2.1.1 Порівняння Java та Python як засобів розробки вебсайтів

У веброзробці вибір мови програмування має вирішальне значення, оскільки він повністю змінює підхід розробників до конкретного вебпроекту. Коли мова йде про основні мови програмування, такі як Python та Java, вибір мови для веброзробки може бути складним. З одного боку, Java, універсальна, об'єктно-орієнтована мова програмування, що базується на класах, має високу продуктивність і швидкість. З іншого боку, Python, як мова програмування високого рівня, що інтерпретується та типізується у реальному часі, вирізняється простотою, масштабованістю та читабельністю.

Java для веброзробки: переваги та недоліки

Порівнюючи Java та Python для веброзробки, варто почати з переваг та недоліків кожної з них. Java має багато переваг, які широко застосовуються для веброзробки. До них відносяться характеристики, які описані нижче.

Java — це сильно типізована мова. Це означає, що тип даних, які може зберігати змінна, явно оголошується під час її створення, що дозволяє розробникам виявляти помилки на ранніх стадіях процесу розробки [8].

Масштабованість: Java спроектована так, щоб її можна було масштабувати, що робить її популярним вибором для великомасштабних вебдодатків. Модель успадкування на основі класів у поєднанні з підтримкою багато потоковості та динамічного зв'язування полегшує створення складних і масштабованих вебдодатків.

Велика спільнота та потужна підтримка: Java має велику та активну спільноту розробників, входить до трійки найкращих мов програмування, поряд з Python та Javascript, які шукали рекрутери у 2022 році. Вона також має багату екосистему бібліотек і фреймворків, які дозволяють швидко і ефективно створювати вебдодатки.

Хоча переваги більш-менш очевидні, давайте також згадаємо про існуючі виклики. Вони можуть стати бар'єрами, які допоможуть вам обрати фаворита у битві між Python та Java для веброзробки. Основними труднощами використання Java для веброзробки є:

Складність і багатослівність: Java відома своєю багатослівністю і може бути складною для читання, особливо для великих проектів з великою кількістю класів і методів. Крім того, незважаючи на те, що можливості розробки на Java величезні, може бути складно орієнтуватися в застарілому коді або працювати зі складною інфраструктурою.

Проблеми з продуктивністю у великомасштабних додатках: у випадках, коли вебрішення стають ресурс ємними, продуктивність може постраждати, оскільки додатки стають складними. Це може стати проблемою для великомасштабних вебпроектів.

Хоча Java має свої переваги, вона також пов'язана зі специфічними проблемами, які можуть змусити розробників розглянути можливість створення вебрішень за допомогою Python.

Python для веброзробки: переваги та недоліки

Також слід зазначити, що Python використовується для створення різноманітних програм, у тому числі й вебрішень. Давайте також розглянемо ключові переваги, які змушують розробників обирати Python, а не Java для веброзробки. До основних переваг Python для веброзробки можна віднести наведені нижче характеристики [7].

Легкий для вивчення синтаксис: Завдяки простому синтаксису Python є чудовим вибором для написання, підтримки та налагодження коду, особливо для великих проектів. До речі, Python також може бути хорошим вибором для розробників початкового рівня, які можуть максимально використати простоту синтаксису цієї мови програмування.

Динамічна типізація та інтерпретована мова: Python має динамічну типізацію, що означає, що тип даних, які може зберігати змінна, визначається під час виконання програми. Це полегшує швидке написання коду і зменшує час, витрачений на перевірку типів. Python також є інтерпретованою мовою, що означає, що код може бути виконаний одразу після написання, не потребуючи етапу компіляції, який має Java.

Багата бібліотека та екосистема: Python має велику та активну спільноту розробників, тому розробники можуть легко знайти підтримку, ресурси та навчальні посібники в інтернеті. Python також має багату екосистему фреймворків, таких як Flask та Django, які дозволяють швидко та ефективно створювати вебдодатки.

Завдяки широкій підтримці спільноти, легкому для вивчення синтаксису та швидкості виконання, Python має низку переваг для веброзробки. Проте, важливо також перерахувати декілька викликів, а саме

Потенційні помилки під час виконання: хоча динамічна типізація полегшує швидке написання коду, вона також може призвести до помилок під час виконання, якщо тип даних змінної не відповідає очікуваному. Це може бути

проблемою для великих проектів з великою кількістю змінних і вимагає ретельного тестування та налагодження, щоб уникнути цього.

Обмеження продуктивності: Python — це інтерпретована мова, і її продуктивність не така висока, як у скомпільованої мови, наприклад, Java. Це може стати серйозною проблемою для великомасштабних вебдодатків.

Перш ніж перейти до висновків, давайте також розглянемо варіанти використання, які допоможуть зробити вибір мови програмування для розробки нашого вебзастосунку.

Java є гідним вибором для вебдодатків, що вимагають швидкої роботи, оскільки це компільована мова з кращою продуктивністю. Крім того, що стосується великомасштабних рішень, Java широко використовується для створення вебдодатків корпоративного рівня завдяки своїй надійності, стабільності та функціям безпеки.

На противагу цьому, Python широко використовується в науці про дані та машинному навчанні, а його розгалужена екосистема бібліотек та фреймворків дозволяє легко створювати вебдодатки, що включають ці технології. Отже, Python краще, якщо ваш вебпроект орієнтований на ці технології.

Остаточний вибір засобу веброзробки на Python чи Java слід робити, враховуючи конкретні потреби мого вебпроекту, а саме — наявність засобів розробки функціоналу розпізнавання плагіату.

2.1.2 Порівняння Java та Python як засобів розробки вебсайту з функціоналом розпізнавання плагіату

У сучасну цифрову епоху плагіат став серйозною проблемою в академічному та професійному світі. Для вирішення цієї проблеми були розроблені інструменти виявлення плагіату з використанням різних мов програмування. Серед найпоширеніших мов програмування для розробки інструментів виявлення плагіату виділяють Python та Java. Хоча обидві мови мають свої сильні та слабкі сторони, розуміння їхніх відмінностей має вирішальне значення для прийняття рішення про те, яка мова найкраще

підходить для виявлення плагіату завдяки API. У цьому пункті ми порівняємо Python і Java у виявленні плагіату завдяки API і визначимо, яка мова краще підходить для цього завдання [9].

Python — мова програмування високого рівня, відома своєю простотою та читабельністю. Вона має велику спільноту користувачів, яка активно розробляє бібліотеки та інструменти для різних застосувань, у тому числі для виявлення плагіату. Простота і читабельність Python полегшують написання і налагодження коду, а також мають велику кількість бібліотек для аналізу даних і обробки природної мови (NLP). Ці бібліотеки можна використовувати для створення ефективних інструментів виявлення плагіату.

Java, з іншого боку, є популярною мовою, що використовується для розробки додатків корпоративного рівня. Вона відома своєю масштабованістю, надійністю та безпекою, що робить її кращим вибором для великих організацій. Java має велику кількість бібліотек для NLP, машинного навчання та аналізу даних, які можна використовувати для створення інструментів виявлення плагіату.

Коли справа доходить до виявлення плагіату завдяки API, як Python, так і Java мають свої сильні та слабкі сторони. Однією з ключових відмінностей між цими двома мовами є рівень абстракції. Python — це мова високого рівня, тоді як Java — мова низького рівня. Це означає, що код на Python загалом легше писати і розуміти, але він може бути не таким ефективним, як код на Java.

Простота використання та гнучкість Python роблять її ідеальною для розробки прототипів та експериментів з різними алгоритмами. Його велика бібліотека інструментів NLP та машинного навчання може бути використана для створення ефективних алгоритмів виявлення плагіату. Крім того, лаконічний синтаксис Python та динамічна типізація полегшують написання та швидке налагодження коду.

Java, з іншого боку, є скомпільованою мовою, а це означає, що вона, як правило, швидша та ефективніша за Python. Це робить її ідеальним вибором для

розробки складних і масштабованих додатків. Об'єктно-орієнтований дизайн Java також полегшує створення великомасштабних додатків та їх підтримку з часом. Крім того, статична типізація Java полегшує виявлення помилок до того, як код буде виконано, зменшуючи ймовірність помилок у кінцевому продукті.

Що стосується виявлення плагіату завдяки API, то і Python, і Java мають бібліотеки, які можна використовувати для побудови ефективних алгоритмів. Python має такі бібліотеки, як NLTK, Spacy та Gensim, які можна використовувати для обробки природної мови та завдань машинного навчання. Java має такі бібліотеки, як Stanford CoreNLP, OpenNLP та Apache Lucene, які можна використовувати для тих же цілей.

Що стосується продуктивності, то Java, як правило, швидша та ефективніша за Python. Це пов'язано з тим, що Java є скомпільованою мовою і може працювати на різних платформах. З іншого боку, Python — це інтерпретована мова, а це означає, що вона, як правило, повільніша за Java. Однак простота використання та гнучкість Python роблять її ідеальним вибором для розробки прототипів та експериментів з різними алгоритмами.

Насамкінець, і Python, і Java мають свої сильні та слабкі сторони у виявленні плагіату завдяки API. Простота, гнучкість і зручність використання Python роблять її ідеальним вибором для розробки прототипів і експериментів з різними алгоритмами. Його велика бібліотека інструментів NLP і машинного навчання також може бути використана для створення ефективних алгоритмів виявлення плагіату.

2.2 Реалізація алгоритмів розпізнавання плагіату за допомогою API

2.2.1 Види API для реалізації розпізнавання плагіату

Під час розробки детектора плагіату одним із найважливіших рішень є вибір правильної мови програмування та інструментів для роботи. API, або інтерфейси прикладного програмування, є важливим компонентом будь-якої системи виявлення плагіату, оскільки вони дозволяють розробникам отримати доступ до готової функціональності та інтегрувати її з власним кодом.

У випадку з Python є кілька доступних API, які можна використовувати для розробки системи виявлення плагіату. Ці API можуть допомогти у вирішенні таких завдань, як обробка тексту, машинне навчання та обробка природної мови, полегшуючи аналіз великих обсягів тексту та виявлення випадків плагіату.

Отже, якщо ви хочете розробити детектор плагіату на Python, вибір API відіграватиме вирішальну роль у визначенні точності та ефективності вашої системи. У наступних розділах ми розглянемо деякі з найпопулярніших API для виявлення плагіату в Python і те, як їх можна використовувати для створення надійної та ефективної системи.

Існує кілька API, за допомогою яких можна створити інструмент для виявлення плагіату в Python. Ось деякі з найпопулярніших [10]:

- The Natural Language Toolkit (NLTK) — це бібліотека Python, яка надає інструменти для роботи з даними людської мови. Її можна використовувати для таких завдань, як токенізація, стеммінг, тегування частин мови тощо. NLTK можна використовувати для попередньої обробки текстових даних перед застосуванням алгоритмів виявлення плагіату.
- Gensim — це бібліотека Python для моделювання тем, аналізу схожості документів та обробки природної мови. Вона надає функції для створення та навчання моделей, таких як латентний семантичний аналіз (LSA) та латентний розподіл Діріхле (LDA), які можуть бути використані для виявлення плагіату.
- Scikit-learn — це популярна бібліотека машинного навчання на Python, яку можна використовувати для різних завдань, таких як класифікація, кластеризація та регресія. Вона надає різноманітні алгоритми для класифікації тексту, включаючи наївний Байєс, дерева рішень та машини опорних векторів (SVM).
- PyTorch — це бібліотека машинного навчання на мові Python, яка надає інструменти для побудови та навчання нейронних мереж. Її можна

використовувати для таких завдань, як класифікація текстів і моделювання мови, які корисні для виявлення плагіату.

2.2.2 Реалізація розпізнавання плагіату за допомогою бібліотеки мови Python NLTK та пошукової системи Google

Існують різні інструменти для виявлення плагіату в письмових роботах. У цій частині ми обговоримо використання бібліотеки Natural Language Toolkit (NLTK) та API пошукової системи Google для виявлення плагіату.

Бібліотека NLTK є популярним інструментом для обробки природної мови мовою Python. Вона пропонує різноманітні функції для обробки тексту, включаючи токенізацію, тегування частин мови та класифікацію тексту. Ці функції корисні для виявлення плагіату шляхом порівняння схожості між двома текстами. Бібліотеку NLTK можна використовувати для токенізації як вихідного тексту, так і документа, що аналізується на схожість [11]. Потім можна порівняти токенізовані документи, щоб визначити, наскільки вони збігаються. Якщо документи мають високий рівень схожості, ймовірно, що один з них був скопійований з іншого.

API пошукової системи Google є ще один інструмент, який можна використовувати для виявлення плагіату. Google — одна з найпоширеніших пошукових систем у світі, а її API надає розробникам доступ до результатів пошуку [12]. Використовуючи API пошукової системи Google, можна визначити, чи був даний документ скопійований з інтернету. Це робиться шляхом порівняння тексту документа з текстом результатів пошуку. Якщо в документі міститься значна кількість тексту, схожого на результати пошуку, ймовірно, що документ є плагіатом.

Поєднання NLTK та API пошукової системи Google пропонує потужний підхід до виявлення плагіату. Токенізуючи вихідний текст і документ, що аналізується, можна отримати числову міру схожості між ними. Потім, використовуючи API пошукової системи Google, можна визначити, чи був

документ скопійований з інтернету. Цей підхід пропонує комплексний спосіб виявлення плагіату в письмових роботах.

Для реалізації цього підходу можна почати з імпорту бібліотеки NLTK в Python. Потім можна токенізувати вихідний текст і документ, що аналізується, за допомогою функцій NLTK [11]. Після цього токенізовані документи можна порівняти за допомогою мір схожості, таких як косинусна схожість або схожість Жаккара. Ці міри подібності надають числове значення, яке відображає ступінь подібності між документами.

Далі можна скористатися API пошукової системи Google, щоб визначити, чи був документ плагіатом. Для цього за допомогою API здійснюється пошук тексту, схожого на документ, що аналізується. Якщо результати пошуку містять значну кількість тексту, схожого на документ, ймовірно, що документ було скопійовано з інтернету.

Використання NLTK та API пошукової системи Google для виявлення плагіату пов'язане з певними труднощами. Однією з головних проблем є проблема хибних спрацьовувань. Хибні спрацьовування виникають, коли документ помилково позначається як плагіат. Це може статися, коли документ містить текст, схожий на інші документи, але текст не скопійовано з цих документів. Щоб зменшити кількість хибних спрацьовувань, важливо використовувати міру схожості, яка відповідає типу документів, що аналізуються.

Ще однією проблемою є помилкові негативні результати. Хибно-негативні результати виникають тоді, коли документ не позначається як плагіат, хоча він містить скопійований текст. Це може статися, коли скопійований текст відсутній у результатах пошуку, отриманих за допомогою API пошукової системи Google. Щоб зменшити кількість помилкових спрацьовувань, важливо використовувати повноцінну та сучасну пошукову систему.

Однак використання API пошукової системи Google для виявлення плагіату пов'язане з певними труднощами. Першою проблемою є обмежена

кількість запитів, які можна зробити за добу. Безкоштовна версія API дозволяє лише 100 запитів на день, що недостатньо для масштабного виявлення плагіату [12]. Однак це обмеження можна подолати, використовуючи кілька акаунтів Google або перейшовши на платну версію API.

Другою проблемою є точність результатів. API пошукової системи Google не розроблений спеціально для виявлення плагіату і тому не завжди може надавати точні результати. Це пов'язано з тим, що API може повертати результати пошуку, які не пов'язані з вхідним текстом, або може пропускати деякі релевантні результати. Однак цю проблему можна вирішити, використовуючи методи обробки природної мови для фільтрації результатів пошуку та визначення найбільш релевантних.

Третя проблема пов'язана з характером вхідного тексту. API пошукової системи Google найкраще працює з довгими текстами, такими як статті, наукові роботи та есе[12]. Він може бути не таким ефективним у виявленні плагіату в коротких текстах, таких як речення або абзаци. Однак цю проблему можна вирішити, використовуючи бібліотеку NLTK для розбиття вхідного тексту на менші одиниці, такі як речення або фрази, і порівняння їх з результатами пошуку.

Використання API, таких як API пошукової системи Google, може бути економічно вигідним і ефективним способом виявлення плагіату у великих обсягах текстів. У поєднанні з методами обробки природної мови та бібліотеками, такими як NLTK, точність і надійність виявлення плагіату можна підвищити. Однак важливо усвідомлювати виклики, пов'язані з використанням цих інструментів, і вживати відповідних заходів для їх подолання.

У моєму випадку використання API пошукової системи Google є оптимальним рішенням, оскільки воно дає мені можливість застосування безкоштовного ресурсу пошуку плагіату у мережі та створити унікальний продукт, який , наразі, не має аналогів на ринку перевірки текстів на плагіат.

У цій частині ми розібрались із основним функціоналом вебзастосунку розпізнавання плагіату, однак також варто вибрати вебфреймворк мови високого рівня програмування Python для реалізації інтерфейсу програми.

2.3 Засоби реалізації вебзастосунку за допомогою Python

2.3.1 Порівняння вебфреймворків Flask та Django

Flask та Django — це фреймворки для веброзробки на основі Python з відкритим вихідним кодом, розроблені для полегшення складності коду для розробників.

Flask відомий як мікро-фреймворк, оскільки він майже не залежить від зовнішніх бібліотек [13]. Використовуючи його, розробники мають гнучкість у виборі шаблонів проектування, інструментів та баз даних. Таким чином, гнучкість є головною особливістю цього фреймворку python.

Оскільки індустрія веброзробки зараз більше схиляється до мікро сервісів та без серверних платформ, популярність Flask постійно зростає. Він широко використовується для створення масштабованих вебдодатків без особливих зусиль. Все завдяки його унікальним можливостям.

Запущений у 2005 році, Django — це фреймворк, розроблений з основною метою полегшити розробку складних вебсайтів, керованих базами даних [14]. Він сприяє безпечній та швидкій розробці, звільняючи розробників від виконання повторюваних завдань веброзробки. Завдяки багатьом чудовим функціям, він дозволяє розробникам створювати надійні та високопродуктивні додатки.

Ще одна перевага фреймворку Django полягає в тому, що він легко масштабується і є безпечним вебфреймворком, тому ідеально підходить для розробки додатків корпоративного рівня. Підтримуючи безліч форматів (XML, HTML, JSON тощо), Django дозволяє розробникам зосередитися на розробці бізнес-логіки вебдодатків і позбавляє від необхідності створювати програми з нуля.

Хоча Flask і Django — це фреймворки Python, придатні для розробки високопродуктивних вебдодатків, які легко масштабуються, вони дуже відрізняються один від одного. Обидва фреймворки підходять для швидкої розробки додатків, але різняться у випадках їх використання. Отже, метою їх порівняння має бути аналіз ключових моментів, які дозволять вам прийняти обґрунтоване рішення в залежності від ваших бізнес-вимог. Детальний порівняльний аналіз фреймворків Django і Flask за ключовими показниками наведений нижче.

Швидкість та продуктивність розробки

Django являє собою вебфреймворк для перфекціоністів з жорсткими термінами. Щоб вкластися в стислі терміни, важливо створювати вебдодатки швидко та ефективно. А веброзробка з Django відбувається швидко і без зайвих зусиль.

На офіційному сайті Django сказано: «Django був розроблений, щоб допомогти розробникам якнайшвидше пройти шлях від концепції до завершення роботи над додатком». Здатність Django давати швидкі результати робить його ідеальним рішенням для створення великих додатків [14].

Офіційний сайт Flask стверджує, що це «мікро фреймворк для Python, заснований на Werkzeug, Jinja 2 і добрих намірах» [13]. Під мікро-фреймворком мається на увазі, що Flask прагне бути максимально простим, але при цьому розширюваним.

Його простота, гнучкість і легкість прокладають шлях для розробників до створення менших додатків у короткі терміни.

Ще одна причина, чому Flask у Python краще за Django з точки зору швидкості, полягає в тому, що можна написати 10k рядків коду за допомогою Flask для того, що в Django зайняло б 24k рядків. Таким чином, у битві продуктивності Flask проти Django, Flask перевершує Django в залежності від вимог проекту.

Гнучкість розробки

Як вже було сказано вище, Flask популярний завдяки мінімалізму та простоті в роботі. Flask не має жодних обмежень, що дозволяє розробнику створювати надійні додатки з можливістю використовувати все, що він вважає за потрібне — це широкий спектр зовнішніх бібліотек та надбудов.

У передмові до документації Flask сказано, що «Flask не буде приймати багато рішень за вас, наприклад, яку базу даних використовувати. Ті рішення, які вона приймає, наприклад, який шаблонізатор використовувати, легко змінити. Все інше залежить від вас, так що Flask може бути всім, що вам потрібно, і нічим зайвим».

З іншого боку, Django постачається з власним набором модулів і вбудованих функцій, що обмежує свободу експериментів і контролю. Батареї, що входять до складу Django, дозволяють розробникам створювати різноманітні вебдодатки без використання сторонніх інструментів та бібліотек.

Масштабованість

Якщо ви хочете розробити високо-масштабований вебдодаток, Django є найкращим варіантом для вас, оскільки він надає повні можливості масштабування. Це компонентний фреймворк, який працює за принципом «Don’t repeat anything» («Нічого не повторювати»).

Кожен рівень додатку не залежить від іншого, а це означає, що ви можете масштабувати додаток на будь-якому рівні. Крім того, він використовує балансування навантаження і кластеризацію для запуску програми на різних серверах. Таким чином, ви можете масштабувати свій вебдодаток в Django без особливих зусиль, зберігаючи бездоганну продуктивність і час завантаження.

Flask також має високу масштабованість, оскільки може обробляти велику кількість запитів щодня. Цей мікро фреймворк модулює весь код і дозволяє розробникам працювати над незалежними частинами і використовувати їх по мірі зростання кодової бази. Однак, у порівнянні з Django, масштабованість Flask

обмежена. Наприклад, він не може масштабуватися, коли сервери підтримують як глобальні, так і локальні проксі-сервери.

Наявність адміністрування

Django постачається з системою адміністрування, яка дозволяє вам швидко створити внутрішній інструмент для керування даними з ваших моделей даних. Він працює на основі ORM (Object Relational Mapper) системи баз даних і структури каталогів. У випадку Django, декілька проектів мають однакову структуру каталогів. Таким чином, розробники відчувають, що це всеохоплюючий досвід.

Якщо ви шукаєте вебдодаток для швидкого запуску, Django Admin вважається легким способом забезпечити простий інтерфейс для управління даними додатку. Лише за допомогою кількох рядків коду ви можете створити додаток, зовнішній вигляд та інтерфейс якого можна налаштувати відповідно до ваших уподобань.

З іншого боку, Flask не постачається з будь-яким інтерфейсом адміністратора. Якщо ви хочете мати систему адміністрування або використовувати ORM, вам потрібно буде встановити кастомні модулі. Розширення Flask-Admin надає те, що дозволяє розробникам мати такий самий досвід роботи, як і з Django. Але у порівнянні з Django, у випадку Flask його налаштування трохи складніше, оскільки вам доведеться інтегрувати його з будь-якою схемою автентифікації, яку ви реалізуєте.

2.3.2 Вибір фреймворку для вебінтерфейсу

Django надає багато вбудованих функцій для веброзробки, таких як маршрутизація URL-адрес, управління базами даних, автентифікація користувачів тощо. Він також відповідає архітектурі Model-View-Controller (MVC), що дозволяє легко розділяти завдання і тримати кодову базу впорядкованою.

Використання Django для створення інтерфейсу виявлення плагіату має багато переваг. По-перше, він забезпечує чисту та організовану структуру для внутрішнього коду, що полегшує його керування та підтримку. По-друге, механізм шаблонів Django дозволяє створювати динамічні та адаптивні користувацькі інтерфейси, які можуть відображати звіти про плагіат у зручному для сприйняття форматі. Крім того, вбудована в Django система автентифікації користувачів може бути використана для захисту програми та обмеження доступу до конфіденційної інформації.

Ще однією перевагою Django є його сумісність з різними базами даних, включаючи SQLite, MySQL, PostgreSQL та Oracle [14]. Це дозволяє розробникам обирати базу даних, яка найкраще відповідає їхнім потребам, і легко переключатися між ними у разі потреби.

Крім того, Django — це фреймворк з відкритим вихідним кодом з великою та активною спільнотою розробників. Це означає, що в інтернеті доступна велика кількість документації та підтримки, що дозволяє легко знайти рішення для будь-яких питань, які можуть виникнути під час розробки.

Коли справа доходить до розробки інструменту виявлення плагіату, можливості обробки природної мови (NLP) Python роблять її ідеальним вибором. Крім того, бібліотека NLTK, яка побудована на основі Python, пропонує багато корисних функцій NLP. Також API пошукової системи Google легко інтегрується з Python і може допомогти виявити плагіат, порівнюючи введені користувачем дані з наявним онлайн-контентом.

Однією з головних переваг використання Python для виявлення плагіату є те, що бібліотеку NLTK та API пошукової системи Google легко впровадити в проект Django. Django є потужним і популярним вебфреймворком для Python, який пропонує безліч готових функцій і модулів, що можуть бути використані для створення зручного інтерфейсу для інструменту виявлення плагіату.

Бібліотека NLTK надає широкий спектр інструментів для обробки тексту, включаючи токенізацію, стиммінг і лематизацію. Ці інструменти можна

використовувати для попередньої обробки та аналізу текстових даних, що полегшує виявлення плагіату. З іншого боку, API пошукової системи Google можна використовувати для пошуку схожого контенту в інтернеті. Порівнюючи введені користувачем дані з наявним онлайн-контентом, інструмент виявлення плагіату може виявити будь-які випадки скопійованого контенту.

Інтегрувати бібліотеку NLTK та API пошукової системи Google у проект Django можна відносно легко завдяки широкому спектру сторонніх бібліотек та модулів, доступних для Python. Наприклад, Django REST framework — це популярна бібліотека, яка надає набір інструментів для побудови RESTful API, які можна використовувати для взаємодії з бібліотекою NLTK та API пошукової системи Google. Система шаблонів Django, з іншого боку, може бути використана для створення зручного інтерфейсу для інструменту виявлення плагіату.

Коли йдеться про створення інструменту для виявлення плагіату, Python з бібліотекою NLTK та API пошукової системи Google пропонує потужне та гнучке рішення. Використовуючи можливості Python для обробки природної мови та величезні ресурси інтернету, ця комбінація забезпечує ефективний спосіб боротьби з плагіатом. Крім того, простота інтеграції цих інструментів у проект Django дозволяє створити зручний інтерфейс інструменту, що робить його доступним для освітян, роботодавців та інших користувачів, які потребують виявлення плагіату.

3. ДЕМОНСТРАЦІЯ ВЕБЗАСТОСУНКУ ПЕРЕВІРКИ ТЕКСТІВ НА ПЛАГІАТ

3.1 Запуск локального середовища для демонстрації проекту

Оскільки вебзастосунок перевірки на плагіат реалізований за допомогою мови високого рівня програмування Python із використанням фреймворку Django та інструментом для обробки природної мови NLTK, необхідно перед запуском сервера вебдодатку встановити пакет необхідних для роботи компонентів.

При розробці вебдодатків важливо мати чітке та організоване налаштування для всіх пакетів і їх версій. Саме тут на допомогу приходять віртуальні середовища Python.

Віртуальні середовища Python (venv) - це інструмент для створення ізольованих середовищ для проектів Python. Це означає, що всі пакунки та залежності, необхідні для конкретного проекту, можна встановити та керувати ними окремо від глобального середовища на комп'ютері. Це полегшує управління та відстеження залежності і дозволяє уникнути конфліктів з іншими проектами.

У випадку нашого вебдодатку для виявлення плагіату, створеного за допомогою Python Django і бібліотеки NLTK, використання віртуального середовища гарантує, що всі необхідні пакети і залежності встановлені і доступні тільки в межах середовища проекту. Це дозволяє краще організувати роботу і зменшує ймовірність конфліктів з іншими проектами Python або глобальним оточенням системи.

Використання віртуального середовища також полегшує спільне використання проекту з іншими, оскільки вимоги до проекту чітко визначені і можуть бути легко встановлені в окремому віртуальному середовищі.

Щоб створити віртуальне середовище, нам спочатку потрібно встановити Python у нашій системі. Після встановлення Python ми можемо створити віртуальне середовище для нашого проекту за допомогою команди python -m

venv venv. Це створить нову папку з назвою venv у каталозі нашого проекту з необхідними файлами для віртуального середовища.

Після створення віртуального середовища ми можемо активувати його за допомогою команди venv\Scripts\activate (на Windows). Після активації всі пакунки, які ми встановлюємо за допомогою pip, будуть встановлені у віртуальному середовищі, а не у глобальному.

Щоб встановити необхідні пакунки для нашого вебдодатку для виявлення плагіату, ми можемо використовувати pip для встановлення їх безпосередньо з файлу requirements.txt або встановити кожен пакунок окремо. Бібліотеку NLTK та API пошукової системи Google можна встановити за допомогою pip3 install nltk та pip3 install google-api-python-client відповідно.

```
user@MacBook-Pro Plagiarsim-Checker % pip3 install -r requirements.txt

Defaulting to user installation because normal site-packages is not writeable
Collecting apiclient==1.0.4 (from -r requirements.txt (line 1))
  Using cached apiclient-1.0.4-py3-none-any.whl
Collecting asgiref==3.2.10 (from -r requirements.txt (line 2))
  Using cached asgiref-3.2.10-py3-none-any.whl (19 kB)
Collecting cachetools==4.1.1 (from -r requirements.txt (line 3))
  Using cached cachetools-4.1.1-py3-none-any.whl (10 kB)
Collecting certifi==2020.6.20 (from -r requirements.txt (line 4))
  Using cached certifi-2020.6.20-py2.py3-none-any.whl (156 kB)
Collecting chardet==3.0.4 (from -r requirements.txt (line 5))
  Using cached chardet-3.0.4-py2.py3-none-any.whl (133 kB)
Collecting click==7.1.2 (from -r requirements.txt (line 6))
  Using cached click-7.1.2-py2.py3-none-any.whl (82 kB)
Collecting Django==3.1.2 (from -r requirements.txt (line 7))
  Using cached Django-3.1.2-py3-none-any.whl (7.8 MB)
Collecting Flask==1.1.2 (from -r requirements.txt (line 8))
  Using cached Flask-1.1.2-py2.py3-none-any.whl (94 kB)
Collecting google-api-core==1.23.0 (from -r requirements.txt (line 9))
  Using cached google_api_core-1.23.0-py2.py3-none-any.whl (91 kB)
```

Рис. 3.1. Демонстрація встановлення необхідного ПЗ.

Таким чином, використання віртуального середовища для нашого вебдодатку для виявлення плагіату, створеного за допомогою Python Django та бібліотеки NLTK, забезпечує кращу організацію, зменшує ймовірність конфліктів з іншими проектами або глобальним середовищем системи, а також

полегшує спільне використання проекту з іншими. Використовуючи віртуальне середовище, ми можемо легко керувати та відстежувати залежності для нашого проекту і гарантувати, що він працює за призначенням.

Після отримання звіту, як на рисунку 3.2, необхідно ввести команду python3 manage.py runserver, яка запустить локальний сервер та відправить у командному рядку посилання та локальний вебзастосунок.

```
user@MacBook-Pro Plagiarsim-Checker % python3 manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
May 12, 2023 - 15:18:03
Django version 3.1.2, using settings 'Plagiarism_Checker.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Рис. 3.2. Демонстрація запуску локального середовища вебзастосунку.

3.2 Демонстрація розробленого вебзастосунку

Вебзастосунок сканування тексту на плагіат має мінімалістичний вигляд та включає функціонал перевірки тексту на плагіат на пряму, за допомогою вкладки «Перевірка тексту» (рис. 3.3а) та перевірки тексту з файлу, за допомогою вкладки «Перевірка файлу» (рис. 3.3б).

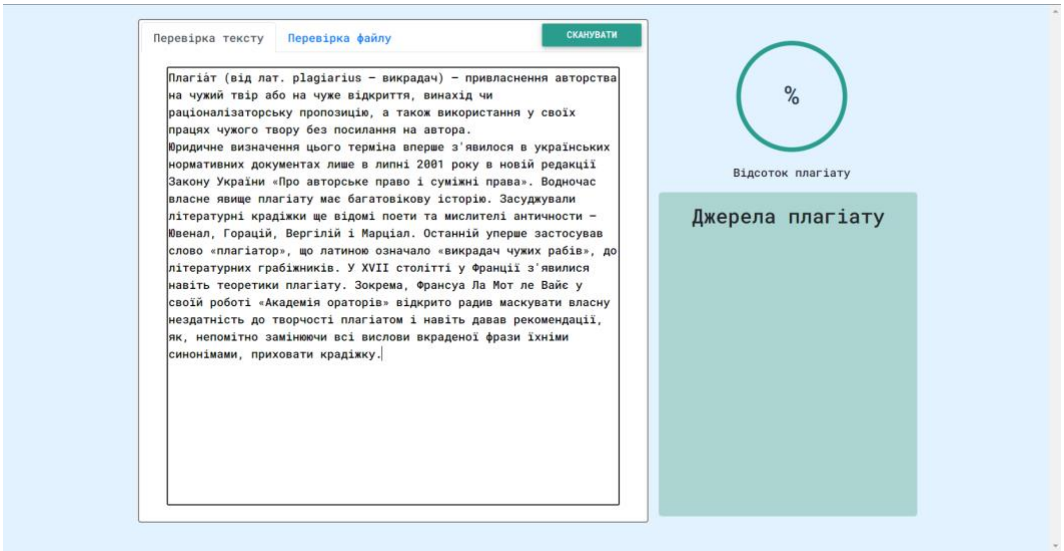

Рис. 3.3а. Вкладка перевірки тексту напряму.

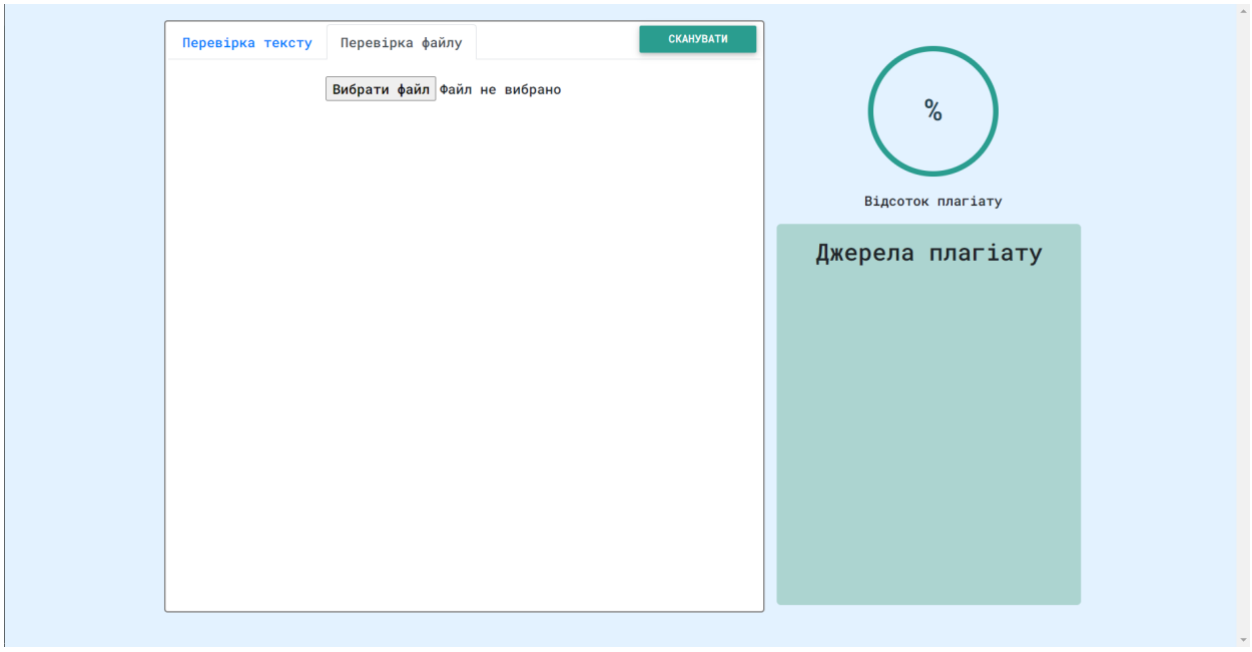

Рис. 3.3б. Вкладка перевірки тексту з файлу.

Для запуску сканування на плагіат необхідно натиснути кнопку «Сканувати», після закінчення сканування, реалізований алгоритм розпізнавання плагіату видає результат, подібний до результату на рисунку 3.4.

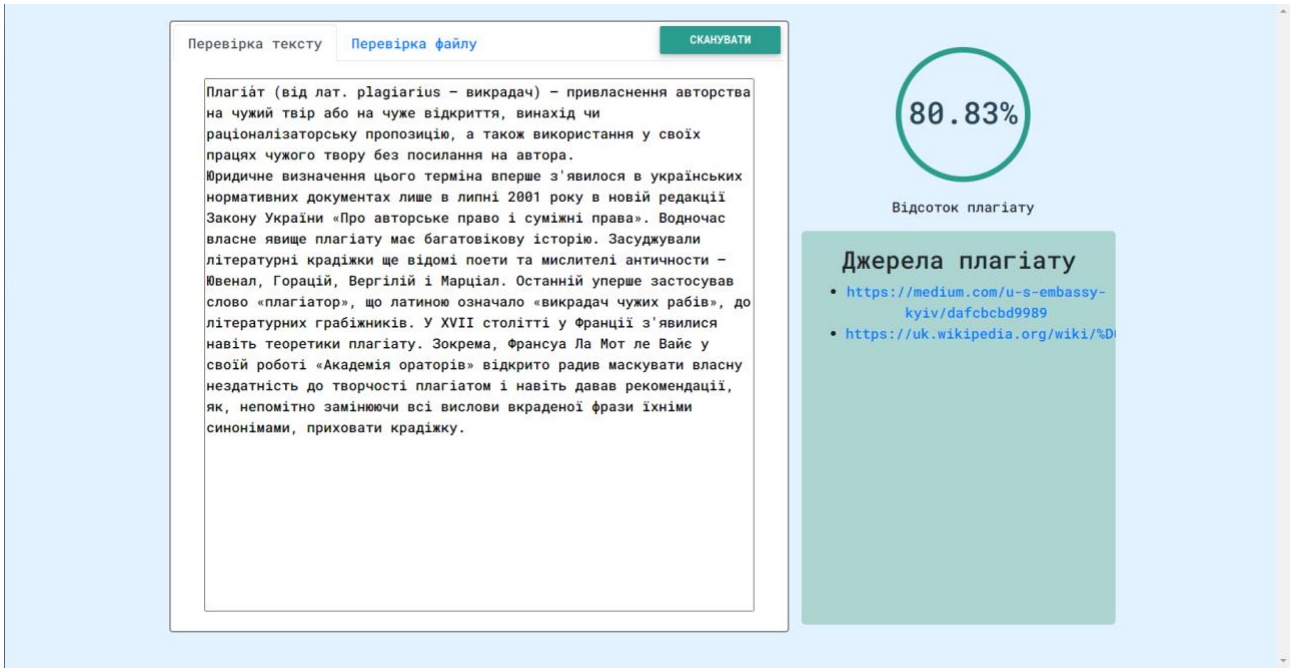

Рис. 3.4. Результат сканування на плагіат.

Детектор плагіату, який ми створили за допомогою бібліотеки NLTK та API пошукової системи Google у проекті Django, має кілька компонентів, які працюють разом, щоб виявити плагіат у заданому тексті. Тут ми пояснимо, як

працює кожен компонент і як вони працюють разом, щоб забезпечити кінцевий відсоток плагіату.

Спочатку текст, який потрібно перевірити на плагіат, завантажується в систему. Потім цей текст попередньо обробляється за допомогою бібліотеки NLTK. Етап попередньої обробки включає кілька процесів, таких як токенізація (детальніше у частині 2.1), видалення стоп-слів, розбиття на частини та лематизація (рис. 3.5).

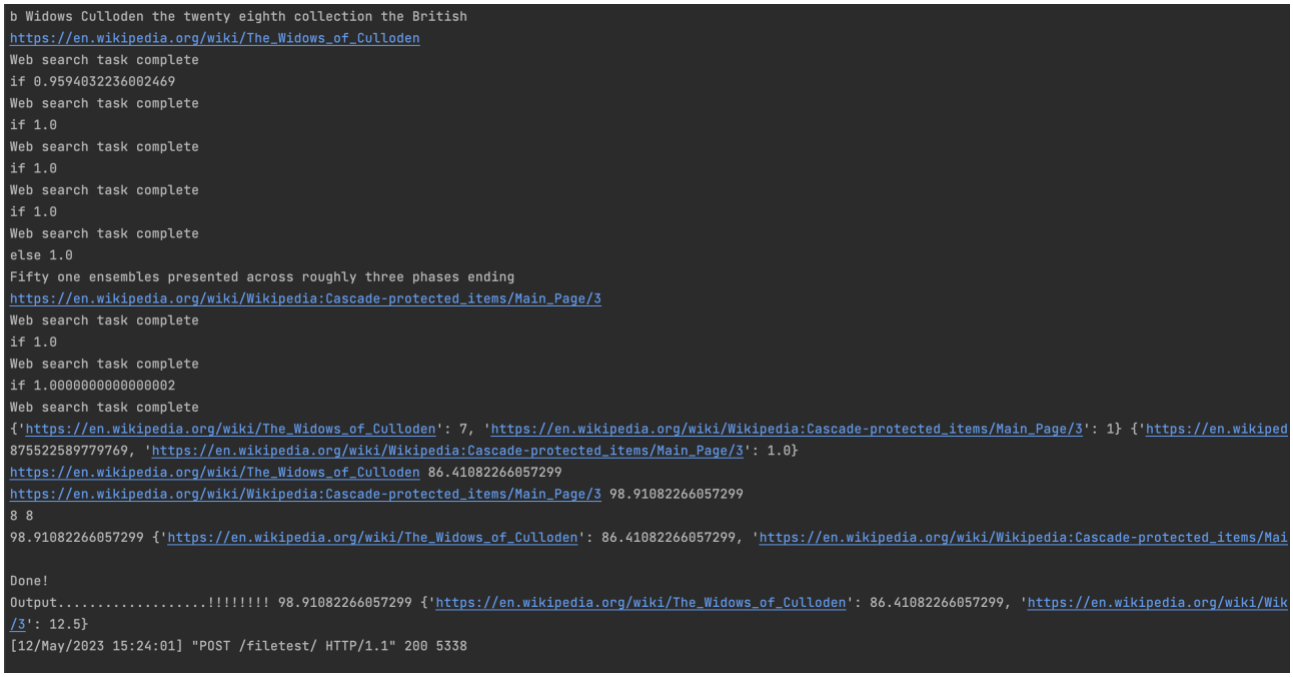

Рис. 3.5. Демонстрація роботи сервера з текстовим файлом.

Після попередньої обробки текст розбивається на n-грами. N-грами - це нерозривні послідовності з n слів із заданого тексту. Ці n-грами потім використовуються для пошуку в API пошукової системи Google. API пошукової системи повертає список URL-адрес, які містять ті ж n-грами, що і заданий текст.

Далі ми відфільтровуємо нерелевантні URL-адреси, які не містять релевантного контенту, пов'язаного із заданим текстом. Це робиться за допомогою регулярних виразів для зіставлення URL-адрес зі списком відомих нерелевантних доменів, таких як вебсайти соціальних мереж або форуми.

Потім завантажуються решта URL-адрес, а їхній текстовий вміст попередньо обробляється за допомогою тих самих процесів бібліотеки NLTK, про які згадувалося раніше. Потім цей попередньо оброблений текст

порівнюється з оригінальним текстом за допомогою декількох методів, таких як косинусоїдальна схожість і схожість Жаккара.

Косинусна подібність вимірює косинус кута між двома векторами. У цьому випадку два вектори представляють частоту появи кожного слова в попередньо обробленому тексті. Подібність Жаккара вимірює схожість двох множин шляхом обчислення відношення кількості елементів в обох множинах до кількості елементів в кожній з них. Ці методи дають нам змогу оцінити, наскільки попередньо оброблений текст схожий на оригінал.

Нарешті, відсоток плагіату обчислюється шляхом ділення загальної кількості n-грам, знайдених у URL-адресах, на загальну кількість n-грам в оригінальному тексті та множення на 100. Це дає нам результат у відсотках, який показує, скільки оригінальних текстів було знайдено в перевірених URL-адресах.

Детектор плагіату використовує кілька методів для виявлення плагіату в заданому тексті. Етап попередньої обробки за допомогою бібліотеки NLTK є важливим для підготовки тексту до порівняння. Розбиття тексту на n-грами та пошук за допомогою API пошукової системи Google дозволяє знайти схожий контент в інтернеті. Фільтрація нерелевантних URL-адрес і попередня обробка текстового вмісту решти URL-адрес допомагає точно порівняти текст. Нарешті, обчислення відсотка плагіату дає нам можливість оцінити, скільки оригінального тексту було знайдено в URL-адресах. Детектор плагіату працює однаково для прямого вставлення та файлів, тільки з файлів текст витягується.

ВИСНОВОК

У процесі написання дипломної роботи було створено інструмент для виявлення плагіату, який використовує бібліотеку NLTK та API пошукової системи Google у вебінтерфейсі Python Django. Вебзастосунок розпізнавання плагіату у тексті був розроблений з використанням широкого спектра технологій і методів, таких як обробка природної мови (NLP), машинне навчання (ML) та інформаційний пошук (IR).

Використання бібліотеки NLTK та API пошукової системи Google у інструменті для виявлення плагіату дає кілька переваг. NLTK надає потужні можливості NLP, які дозволяють попередньо обробляти і токенізувати текст, видаляти стоп-слова і визначати найбільш релевантні терміни і фрази в даному тексті. API пошукової системи Google дозволяє отримувати відповідні вебсторінки для заданого тексту і отримувати оцінку схожості на основі кількості слів, що збігаються між текстом і знайденими сторінками. Django надає простий у використанні вебінтерфейс, який дозволяє користувачам вводити текст, запускати алгоритм виявлення плагіату та переглядати результати. Однією з унікальних особливостей створеного інструменту виявлення плагіату є його здатність розбивати введений текст на менші фрагменти і запускати кожен фрагмент через API пошукової системи Google окремо. Такий підхід дозволяє отримати більш точну оцінку плагіату, враховуючи схожість кожного фрагмента зі знайденими вебсторінками.

Створений інструмент виявлення плагіату має низку переваг над наявними інструментами виявлення плагіату, такими як PlagAware, iThenticate та PlagScan. На відміну від цих інструментів, створений інструмент використовує бібліотеку NLTK та API пошукової системи Google, які надають потужні можливості для NLP та IR відповідно. Крім того, створений інструмент має зручний вебінтерфейс, який дозволяє користувачам легко вводити текст і переглядати результати. Доступ до ресурсу - https://diplom-plagiarism-checker-deploy.herokuapp.com

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. The Plagiarism Spectrum: Instructor Insights into the 10 Types of Plagiarism. [Електронний ресурс] // Turnitin. – 2019. – Режим доступу до ресурсу: https://www.turnitin.com/images/plagiarism_spectrum_10_types_051813.pdf.
2. Sutherland-Smith W. Plagiarism, the Internet, and Student Learning / Wendy Sutherland-Smith., 2008. – 167 с.
3. Bloch J. Plagiarism, Intellectual Property and the Teaching of L2 Writing / Joel Bloch., 2012. – 388 с
4. Lipson C. Doing Honest Work in College: How to Prepare Citations, Avoid Plagiarism, and Achieve Real Academic Success / Charles Lipson., 2008. – 258 с.
5. Overview and Comparison of Plagiarism Detection Tools [Електронний ресурс] / [M. Asim, A. El Tahir, M. Hussam та ін.] // Department of Computer Science, Germany & UC Berkeley. – 2011. – Режим доступу до ресурсу: https://ceur-ws.org/Vol-706/poster22.pdf.
6. Best Programming Languages for Web Development [Електронний ресурс] // MOOC BLOG TEAM. – 2021. – Режим доступу до ресурсу: https://www.mooc.org/blog/best-programming-languages-for-web-development.
7. Romano F. Learn Web Development with Python: Get hands-on with Python Programming and Django web development / F. Romano, G. Hillar, A. Ravindran., 2018. – 796 с.
8. Layka V. Learn Java for Web Development: Modern Java Web Development / Vishal Layka., 2014. – 489 с.
9. Ravoof S. Python vs Java: Pick What's Best for Your Project [Електронний ресурс] / Salman Ravoof. – 2023. – Режим доступу до ресурсу: https://kinsta.com/blog/python-vs-java/.
10. Duggal N. Top 20 Python Libraries for Data Science for 2023 [Електронний ресурс] / Nikita Duggal. – 2023. – Режим доступу до ресурсу: https://www.simplilearn.com/top-python-libraries-for-data-science-article.

11. Natural Language Toolkit [Електронний ресурс] – Режим доступу до ресурсу: https://www.nltk.org/.
12. Programmable Search Engine [Електронний ресурс] // Google – Режим доступу до ресурсу: https://developers.google.com/custom-search.
13. Django documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.djangoproject.com/en/4.2/.
14. Flask Documentation (2.3.x) [Електронний ресурс] – Режим доступу до ресурсу: https://flask.palletsprojects.com/en/2.3.x/.
15. Sanderson D. Programming Google App Engine: Build & Run Scalable Web Applications on Google's Infrastructure / Dan Sanderson., 2012. – 534 с.
16. Sanderson D. Programming Google App Engine with Python: Build and Run Scalable Python Apps on Google's Infrastructure / Dan Sanderson., 2015. – 464 с.