Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему *Метод семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконав: *студент 4 курсу, група КН-19-2* _____ *М.Б. Шиманський*
Курс, група виконавця | Підпис | Ініціали, прізвище

Керівник: *викладач кафедри КН* _____ *М.О. Молчанова*
Науковий ступінь, посада | Підпис | Ініціали, прізвище

Нормоконтроль: *к.т.н., доцент кафедри КН* _____ *Р.О. Багрій*
Науковий ступінь, посада | Підпис | Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КН, д.т.н., професор _____ *О.В. Бармак*
Підпис | Ініціали, прізвище

05 06 2023 р.

Хмельницький 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *бакалавр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*
Освітня програма *освітньо-професійна програма підготовки бакалавра*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук*

_____
(підпис)
д.т.н., професор О.В. Бармак
«*06*» *03* *2023* року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: *«Метод семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату»*

2. Завдання видано студенту *Шиманському Максиму Борисовичу*
(прізвище, ім'я, по батькові)

3. Керівник роботи *викладач кафедри КН Молчанова Марина Олексіївна*
(посада, прізвище, ім'я, по батькові)

4. Затверджено наказом університету від «*01*» *03* *2023* р. № *5*

5. Дата видачі завдання студенту: «*03*» *03* *2023* р.

6. Зміст пояснювальної записки (перелік задач) та вихідні дані: *виконати аналіз сучасних методів обробки текстів природною мовою в області аналізу текстових даних для пошуку плагіату; виконати аналіз існуючих рішень для пошуку плагіату за семантичним аналізом текстів; розробити метод семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату й спроєктувати структуру відповідної бази даних; розробити інформаційну систему пошуку плагіату за семантичним аналізом текстів з відповідною функціональністю; провести тестування створеної інформаційної системи пошуку плагіату за семантичним аналізом текстів; провести дослідження практичної ефективності розробленого методу.*

7. Календарний план виконання кваліфікаційної роботи бакалавра:

№	Назва етапів (розділів) кваліфікаційної роботи бакалавра	Термін виконання	Примітка
1	*Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи бакалавра з керівником*	*грудень 2022*	виконано
2	*Ознайомлення з предметною областю, формулювання мети та задач дослідження, визначення об'єкта та предмета дослідження*	*січень 2023*	виконано
3	*Робота над розділом 1 – Характеристика предметної області та постановка задачі*	*січень 2023*	виконано
4	*Робота над розділом 2 – Метод семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату*	*березень 2023*	виконано
5	*Робота над розділом 3 – Програмна реалізація інформаційної системи пошуку плагіату за семантичним аналізом текстів*	*квітень 2023*	виконано
6	*Оформлення пояснювальної записки згідно вимог*	*травень 2023*	виконано
7	*Попередній захист кваліфікаційної роботи бакалавра*	*травень 2023*	виконано
8	*Захист кваліфікаційної роботи бакалавра на засіданні Екзаменаційної комісії*	*червень 2023*	виконано

Виконавець: *студент 4 курсу, група КН-19-2* (Курс, група виконавця) _____ (Підпис) *М.Б.Шиманський* (Ініціали, прізвище)

Керівник: *викладач кафедри КН* (Науковий ступінь, посада) _____ (Підпис) *М.О. Молчанова* (Ініціали, прізвище)

Анотація

Тема кваліфікаційної роботи бакалавра: *Метод семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату*

Виконавець кваліфікаційної роботи бакалавра: *студент групи КН-19-2 Шиманський Максим Борисович*

Керівник кваліфікаційної роботи бакалавра: *викладач кафедри КН Молчанова Марина Олексіївна*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
72	42	2	31	4

Метою кваліфікаційної роботи бакалавра є розробка методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату та відповідної інформаційної системи пошуку плагіату за семантичним аналізом текстів, що дозволяє за текстовим контентом визначати числову оцінку встановленого обсягу текстових запозичень шляхом його семантичного аналізу з використанням n-грам.

Для досягнення поставленої мети було розроблено інформаційну систему пошуку плагіату за семантичним аналізом текстів, яка дозволяє визначати відсоток збігів між текстами, що порівнюються між собою, або текстами, що були збереженні у базі даних. Також інформаційна система забезпечує пошук n-грам у цифровому тексті та редагування збережених текстів. Програмна реалізація виконана з використанням мови програмування C# та бібліотеки OpenNLP.

Ключові слова: плагіат, семантичний аналіз текстів, n-грами, семантичний аналіз, вектор важливих слів, інформаційна система.

Виконавець: *студент 4 курсу, група КН-19-2* (Курс, група виконавця) _____ (Підпис) *М.Б.Шиманський* (Ініціали, прізвище)

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
NLP	Natural language processing
OOV	Out of Vocabulary
POS	Part of speech
SERP	Search Engine Ranking Pages
TF-IDF	Term frequency – inverse document frequency
TF	Term frequency
YAKE	Yet another keyword extractor
LM	Language models
ML-SOM	Machine learning - self-organizing map
MSRP	Microsoft Research Paraphrase Corpus
PAN	Plagiarism Analysis, Authorship Identification, Near-Duplicate Detection
БД	База даних
КРБ	Кваліфікаційна робота бакалавра
КН	Комп’ютерні науки

Вступ

Кваліфікаційна робота бакалавра присвячена розробці методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату та відповідної інформаційної системи пошуку плагіату за семантичним аналізом текстів, що дозволяє за текстовим контентом визначати числову оцінку встановленого обсягу текстових запозичень шляхом його семантичного аналізу з використанням n-грам.

Актуальність. Інтелектуальний аналіз тексту, як один із напрямів ШІ є одним із значимих напрямів сучасних досліджень. Перевірка тексту на плагіат популяризована у закладах освіти, особливо коли потрібно перевірити письмову роботу студента на оригінальність, у «paper mill» компаніях, що допомагають студентам з написанням курсових робіт, а також видавництвам та журналістам, у написанні статей або стрічки новин.

За допомогою n-грам та семантичного аналізу користувач матиме здатність порівнювати документи та тексти, так як n-грами дозволяють поєднувати ряд слів та шукати аналоги у документі або тексті, що перевіряється. Використавши NLP можна полегшити задачу, так як це дозволить краще взаємодіяти з текстом, як людині так і комп’ютеру, що призведе до економії часу, тому що перевірка тексту на плагіат є складною роботою, що вимагає багато часу.

Тому робота над методом семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату є актуальною.

Об’єкт дослідження – процес семантичного аналізу текстів для пошуку плагіату.

Предмет дослідження – моделі, методи, алгоритми та засоби для визначення обсягу текстових запозичень шляхом семантичного аналізу тексту.

Мета кваліфікаційної роботи бакалавра – розробка методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату та відповідної інформаційної системи пошуку плагіату за семантичним аналізом

текстів, що дозволяє за текстовим контентом визначати числову оцінку встановленого обсягу текстових запозичень шляхом його семантичного аналізу з використанням n-грам..

Завдання кваліфікаційної роботи бакалавра – виконати аналіз сучасних методів обробки текстів природною мовою в області аналізу текстових даних для пошуку плагіату; виконати аналіз існуючих рішень для пошуку плагіату за семантичним аналізом текстів; розробити метод семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату й структуру відповідної бази даних; розробити інформаційну систему пошуку плагіату за семантичним аналізом текстів з відповідною функціональністю; провести тестування створеної інформаційної системи пошуку плагіату за семантичним аналізом текстів; провести дослідження практичної ефективності розробленого методу за використання створеної інформаційної системи.

Розділ 1 Характеристика предметної області та постановка задачі

1.1 Аналіз предметної області семантичного аналізу текстів та пошуку плагіату

У мережі є достатньо сайтів, що допомагають користувачам перевірити свої роботи на плагіат. Для виявлення плагіату в семантичних текстах використовуються різні методи. Такий засіб, як NLP (Natural language processing) передбачає здатність комп'ютерів розуміти природні мови за допомогою технологій штучного інтелекту та машинного навчання. Це включає створення комп'ютерних моделей людської мови та засобів обробки природної мови, таких як словники чи семантичні мережі.

Інтелектуальний аналіз тексту та NLP використовують комп'ютерні алгоритми та програми для розуміння та аналізу текстів. Це важливий напрямок досліджень у сфері штучного інтелекту, який забезпечує значні результати в автоматичному абстрагуванні, автоматичних відповідях на запитання та розпізнаванні тексту [1].

Навчальні програми, які використовують машинне навчання, автоматично фокусуються на найпоширеніших випадках, тоді як при ручному написанні правил часто незрозуміло, на чому зосередити зусилля [2].

Програми машинного навчання використовують статистичні алгоритми логічного висновку, які стійкі до незнайомих та помилкових вхідних даних. Обробка таких даних з використанням рукописних правил є складною, схильною до помилок та часозатратною [2].

Важливою підзадачею NLP і рушієм інструментів машинного навчання є семантичний аналіз, який є процесом вилучення значення з тексту. Це дозволяє комп'ютерам розуміти та інтерпретувати речення, абзаци або цілі документи, аналізуючи їхню граматичну структуру та визначаючи зв'язок між окремими словами в конкретному контексті [3].

Основні задачі семантичного аналізу мови включають наступне [4, 5].

1. Розуміння семантичного змісту: це включає в себе розуміння значень слів та їх взаємозв'язків, а також семантичного контексту, що визначає, яке значення має вживане слово.

2. Визначення семантичних відношень: це включає в себе визначення відношень між словами, які допомагають зрозуміти їх значення та взаємозв'язки. Такі відношення можуть бути синонімічними, антонімічними, гіперонімічними (загальними поняттями) та гіпонімічними (частинами поняття).

3. Аналіз дискурсу: це включає в себе аналіз семантики речень та текстів, щоб зрозуміти їх зміст та мету. Це може включати аналіз теми, тези, структури, стилю та інших елементів тексту.

4. Виявлення семантичних помилок: це включає в себе виявлення неправильного вживання слів, невідповідності між семантичним контекстом та значенням слів, неправильної інтерпретації висловлювань та інших помилок.

5. Семантичний аналіз мови для машинного навчання: це включає в себе розробку алгоритмів та моделей, які можуть автоматично розуміти та обробляти семантику мови. Це може включати розробку моделей класифікації, кластеризації, пошуку та інших задач машинного навчання, які вимагають розуміння семантики мови.

6. Аналіз емоційного вмісту: це включає в себе аналіз відтінків емоцій, що містяться в текстах, а також виявлення сильно навантажених слів та виразів.

7. Аналіз контексту: це включає в себе аналіз контексту, у якому вживається слово або речення, щоб зрозуміти його семантику та інтенцію. Наприклад, використання одного й того ж слова в різних контекстах може мати різне значення.

8. Аналіз смислової близькості: це включає в себе визначення ступеня схожості між словами та фразами на основі їх семантичних властивостей. Це допомагає виявляти синоніми, антоніми та інші відношення між словами.

9. Розв'язання задач машинного перекладу: це включає в себе переклад текстів з однієї мови на іншу, з урахуванням семантичних особливостей кожної мови та контексту, в якому вживаються слова та речення. Для цього можуть

використовуватися різні алгоритми та методи, які базуються на семантичному аналізі мови.

Плагіат – це подання думок, ідей чи слів іншої людини як власних [6]. Це вважається формою неетичної поведінки та може призвести до покарання на основі законів про авторське право. Окрім літературних і наукових творів, плагіат поширений у музиці, маркетингу та інших сферах творчості [6]. Крім того, технологічний прогрес робить плагіат легшим, ніж будь-коли.

Плагіат має багато форм, деякі серйозніші, ніж інші, починаючи від перефразування чиїхось ідей до викрадення цілих статей. Ось найпоширеніші види плагіату [6].

1. Повний плагіат – це коли особа копіює повністю роботу іншої людини, таку як наукові роботи, статті, зображення тощо, і представляє її як свою власну. Це є найбільш екстремальною формою плагіату і можна порівняти з крадіжкою особистих даних або крадіжкою.

2. Прямий плагіат – відкрите видавання чужої роботи без зазначення автора та без цитування, але у відміну від повного плагіату, злодій викрадає лише окремі частини або уривки роботи.

3. Перефразування плагіату – це коли автор запозичує чужу роботу змінюючи кілька слів чи фраз. Це поширений тип плагіату, яким грішать студенти навіть не підозрюючи про це. Однак якщо подаючи чужу оригінальну ідею у своєму тексті без посилання на авторство, навіть якщо подача відбувається з власних слів, це плагіат.

4. Самоплагіат – форма плагіату коли автор запозичує з власної попередньої роботи певні висловлювання. Це несе вагомий вплив лише на професійну діяльність, так як коли працюючи з клієнтом, те що написано в подальшому буде власністю клієнта. Повторне використання власних слів для наступних клієнтів є плагіатом власної роботи та завдає шкоди професійній репутації

5. Patchwork плагіат також відомий, як мозаїчний або клаптиковий плагіат – це ситуація, коли плагіат переплітається з оригінальною роботою

автора. Цей тип плагіату може бути непомітним і його легко пропустити, і він може поєднуватися із повним плагіатом.

6. З плагіатом на основі джерел боротися важко. При цьому типі плагіату автор може правильно цитувати джерело, але спотворюючи його. Тобто згадувати лише вторинне джерело уникаючи первинне, цитування неправдивих джерел або навіть фальсифікованих джерел.

7. Випадковий плагіат є найпоширенішим типом плагіату, через те що автори навіть не усвідомлюють, що вони копіюють чиюсь роботу.

Випадковий плагіат включає наступне [6]:

– забуття вказати джерела у своїй роботі;

– неправильне посилання на джерело;

– відмовитися від цитування цитованого матеріалу.

Виявлення плагіату в текстових документах з високою точністю є складним завданням. За останні два десятиліття дослідники повідомили про численні підходи до цієї проблеми. Ці методи можна згрупувати в одинадцять різних категорій. Деякі добре відомі методи оглянуто нижче [7].

– Символьний метод порівнює текст документа запиту з іншими документами, використовуючи порівняння символів, синтаксису та слів. Більшість методів виявлення плагіату базуються на цьому підході та використовують пошук подібності n-грамів для знаходження точних збігів. Деякі дослідники віддають перевагу приблизній відповідності рядків, яка використовує фрази розміром 8-грамів або менше.

– Векторний підхід – це спосіб вимірювання подібності текстів, що базується на класифікації слів і речень як векторів. Цей підхід використовує різні формули для обчислення показників векторної подібності, найпопулярнішими є: косинус і коефіцієнт Жаккара, оскільки вони враховують як лексичні, так і синтаксичні особливості текстів. Векторний підхід дозволяє виявляти частковий плагіат, а також забезпечує конфіденційність документів.

– Методи на основі синтаксису виявляють плагіат за допомогою аналізу частин мови, таких як дієслова, займенники, прикметники, прислівники

та сполучники, а також вставних слів. Вони використовують синтаксичні POS-теги для представлення текстових структур та подальшого порівняння текстів.

– Семантичний підхід вимірює подібність фраз на основі семантики. Для цього використовуються словники, такі як WordNet і Resnick, які допомагають визначити значення кожного слова і семантичну подібність між ними. Автор також використовує кількість вузлів на найкоротшому шляху між двома словами для визначення семантичної подібності.

– Методи на основі нечіткості використовують неоднозначні слова та їх взаємозв'язки для вимірювання подібності тексту. Вони використовують коефіцієнти кореляції для визначення схожості між словами та реченнями, що дозволяє досліднику знайти схожість між документами, яка не вказана в них напряму. Інфрачервоний метод нечіткості використовується для визначення подібності між двома документами чи веб-сайтами.

– Підходи на основі структури визначають схожість між документами на основі контекстуальної подібності, використовуючи деревоподібні представлення функцій, як в ML-SOM(Machine learning - self-organizing map). Контекстна інформація обробляється за допомогою деревоподібних представлень функцій, а плагіат виявляється в два етапи: на першому етапі використовується деревоподібне представлення ознак для кластеризації документів і пошуку кандидатів, а на другому етапі ML-SOM використовується для виявлення збігів.

– Стилометричні методи визначають схожість між стилями написання для виявлення плагіату. Ці методи аналізують конкретні фрази та абзаци для знаходження спільних рис між джерелами. Якщо джерела мають подібний стиль, то одне з них може бути копією іншого. Спосіб читання тексту визначає його стиль, і аналіз викидів може допомогти виявити плагіат, проаналізувавши методи читання.

– Класифікація та кластерні методи включають контрольоване чи неконтрольоване впорядкування документів для зменшення кількості потенційних результатів для пошуку. Групуючи подібні розділи документа, це

допомагає скоротити час, необхідний для порівняння документів під час виконання слідчої роботи. Деякі методи слідчої роботи використовують ключові або конкретні слова для групування подібних розділів.

– Метод виявлення плагіату за допомогою методів аналізу цитування. Ця нова ідея використовує невикористані частини документів, які були проаналізовані, але на які не було посилань. Методи, які виявляють семантичний плагіат, використовують текст документа, який цитується в їх аналізі. Ці методи використовують систему аналізу цитування, яка порівнює тексти двох документів, щоб знайти схожість між ними.

Отже, використання семантичного аналізу є комплементарним методом для виявлення випадків плагіату, оскільки він може ефективно порівнювати тексти та виявляти повний плагіат. Семантичний аналіз тексту та виявлення плагіату мають вирішальне значення для забезпечення наукової цілісності та підтримки легітимності наукових статей і публікацій. Аналізуючи семантичні зв’язки, можна точно визначити схожість між текстами. Застосування методу семантичного аналізу може значно підвищити якість і точність виявлення плагіату, що робить її особливо корисною в академічних і дослідницьких умовах.

1.2 Аналіз методів і засобів семантичного аналізу текстів для пошуку плагіату

Концепція ключових слів допомагає знайти найкращі слова для націлювання та побудови контенту, що підвищує видимість сайту в пошукових системах. Це передбачає вибір важливих слів, пов'язаних з темою пошуку. SEO інструменти допомагають дослідити ключові слова, які цікавлять цільову аудиторію та мають велику кількість пошукових запитів [8].

Методи пошуку ключових слів:

– TextRank – використовує систему ранжирування на основі графіків для визначення найбільш відповідних речень у тексті [9]. Його також можна

використовувати для пошуку ключових слів у тексті. Щоб знайти найбільш відповідні речення в документі, будується графік, де вершини представляють кожне речення в документі, а межі представляють спільні слова між ними [9]. Кількість країв визначає збіги кожного слова.

– TF-IDF (Term frequency – inverse document frequency) та TF (Term frequency) – це числова статистика, яка використовується для вимірювання ключових слів в документі [10]. Він використовується для пошуку інформації та аналізу тексту. TF-IDF точно відображає кількість разів, коли слово з’являється в документі.

– YAKE (Yet Another Keyword Extractor) – це автоматичний метод вилучення ключових слів, який не потребує навчання чи зовнішніх ресурсів [11]. Він використовує статистичні характеристики тексту для автоматичного збору найважливіших ключових слів. YAKE не потребує певного словника, мови чи домену.

Скористатись інструментами (бази даних з тезаурусом або інтелектуальну карту) для дослідження ключових слів, які допоможуть визначити популярні теми серед цільової аудиторії [12]. Такі інструменти надають можливість знайти ключові слова з великою кількістю пошукових запитів, які можуть бути використані для планування та створення вмісту, та допоможуть спланувати дослідження, записати ключові слова, імена авторів та ідеї в ході дослідження [12].

Для того щоб утворювати ключові слова та фрази підчас пошуку плагіату використовують n-грами. N-грами – це безперервні послідовності слів, символів або токенів у документі [13]. З технічної точки зору їх можна визначити як безперервні послідовності елементів у документі. Вони вступають у гру, коли користувач працює з текстовими даними в завданнях NLP.

Збільшення прогностичної здатності програми можна оцінити за допомогою зовнішньої або внутрішньої оцінки методу n-грам, що використовує метрику perplexity для визначення загальної ймовірності мови [14]. Для покращення ефективності n-грамових моделей можна використовувати

спеціальні функції, але при виборі більших n потрібно враховувати важливість певних ознак, тому інші мовні моделі, такі як тематична LM і кеш-LM, можуть допомогти захопити контекст більш ефективно [14].

Отже, у роботі над методом семантичного аналізу текстів для задач пошуку плагіату потрібно використати такі методи, як пошук плагіату за ключовими словами та n-грами. Завдяки n-грамам можна визначити ключові словосполучення, що у свою чергу поліпшить якість перевірки тексту.

1.3 Аналіз існуючих рішень для пошуку плагіату за семантичним аналізом текстів

На сьогоднішній день є ряд програм, відповідних темі дослідження. Однією з таких програм є Plagiarisma – зручна, комплексна програма виявлення плагіату, якою користуються студенти, письменники, викладачі та представники мистецької спільноти (рисунок 1.1) [19].

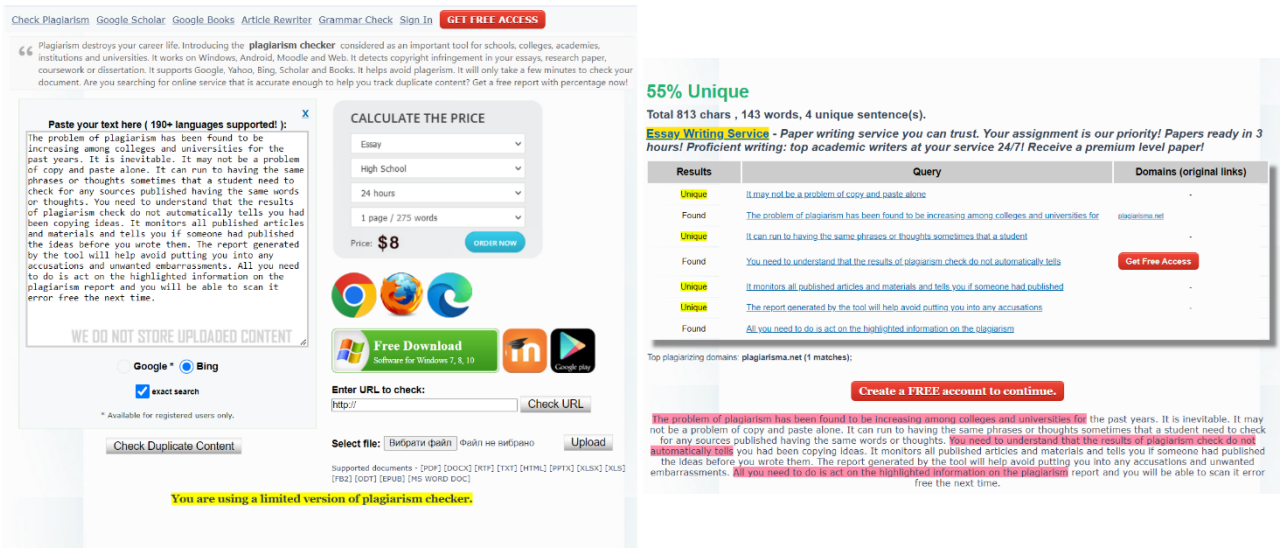


Рисунок 1.1 – Головна сторінка Plagiarisma та результат перевірки тексту на плагіат [16]

Переваги використання цього веб-сайту набагато переважують його недоліки [19].

– Додаток підтримує понад 190 мов;

– Текст можна вводити в поля шляхом копіювання та вставки, введення або вставлення з URL-адреси. Файли можна завантажувати з комп’ютера у форматах DOCX, DOC, ODT, FB2, XLS, XLSX, TXT, HTML і RTF. Крім того, у ці поля можна вводити файли EPUB і PDB;

– Завантажте доповнення для Firefox і Chrome для швидкої перевірки тексту.

Недоліки [19]:

– необхідні щомісячні перевірки на обмежений плагіат.

– Plagiarisma має безкоштовну версію з обмеженим доступом, або можна придбати пакет PRO за 5 доларів на місяць або вище.

У Writer плюсами є те що завдяки мінімалістичній естетиці цей електронний продукт має широку привабливість [17]. У веб-версії можна безкоштовно написати до 2000 слів на перевірку (рисунок 1.2).

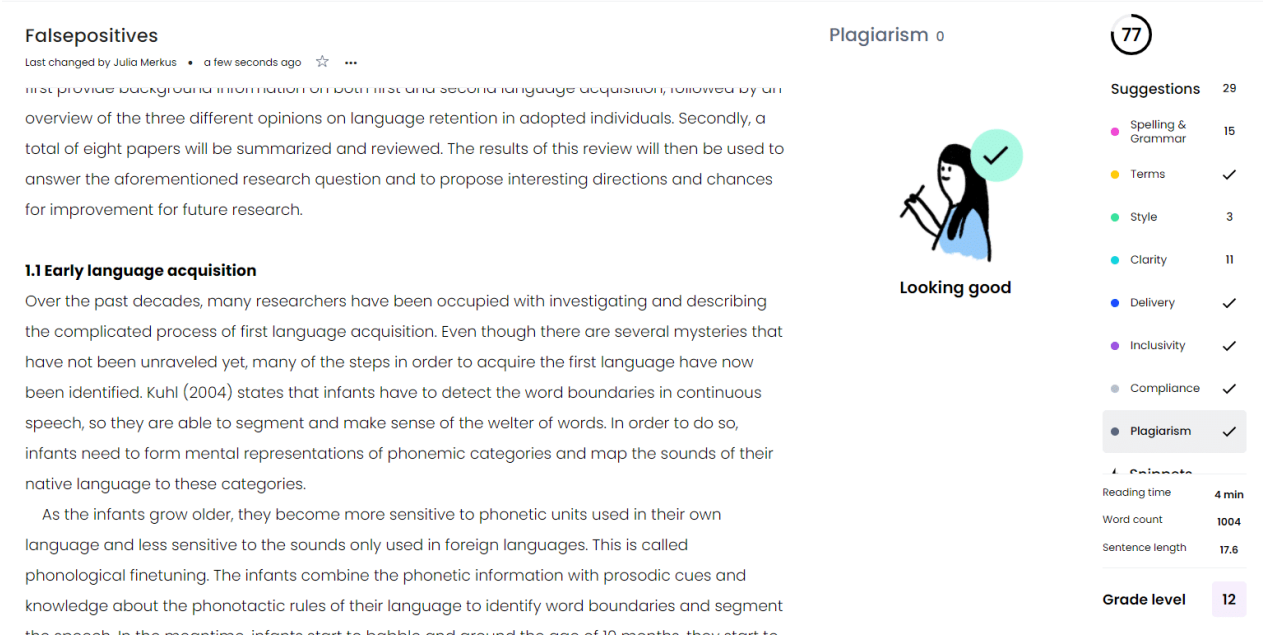


Рисунок 1.2 – Результат роботи Writer [17]

Writer має дві версії: безкоштовний веб-сайт, що дозволяє перевірити до 2000 слів, та платне розширення для браузера на суму 11 доларів США на місяць, з безкоштовною пробною версією доступною на один місяць [17].

З недоліків можна виділити [17]:

– версія розширення браузера не може виявити будь-які форми плагіату.

– додаткові покупки в програмі здійснюються після завершення безкоштовної пробної версії.

– веб-версія надає занадто мало інформації порівняно з версією програми.

– немає помітного прогресу під час тривалого сканування.

– лише вставте або введіть текст, а не завантажуйте документи

Також потрібно відмітити і Unicheck (рисунок 1.3), але безкоштовна пробна послуга, яку надається користувачу, дозволяє оцінити роботу програми з обмеженням на 200 слів [17]. Тому малоймовірно, що роботу можна буде перевірити. Однак завжди можна купити преміум версію, якщо пробна версія виявиться закороткою для потреб користувача.

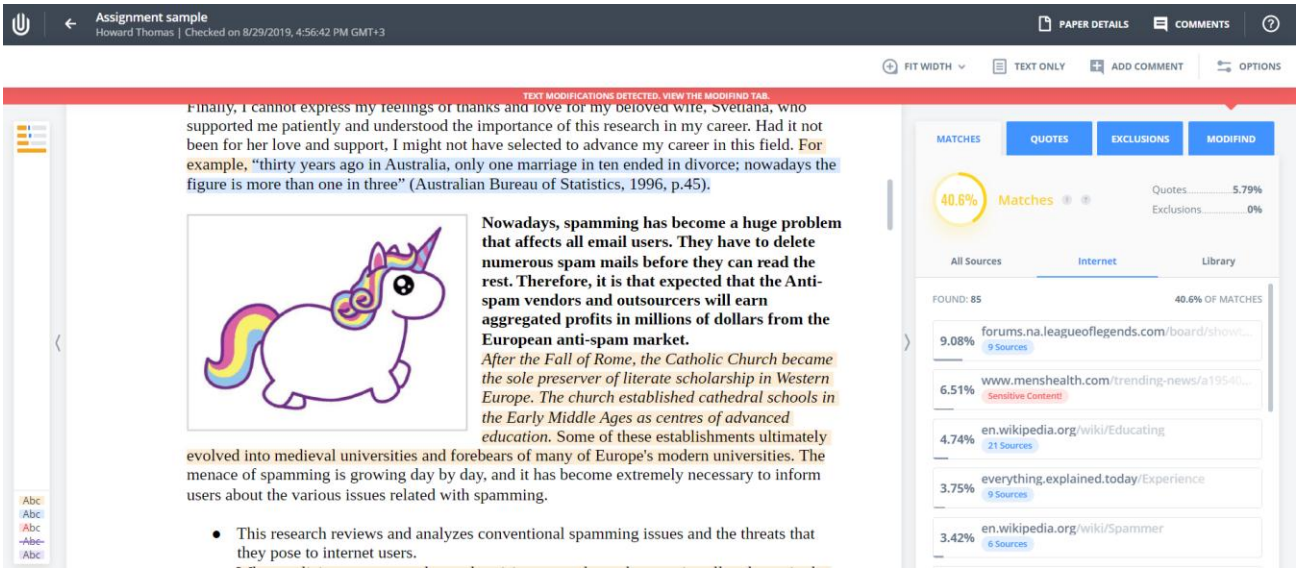


Рисунок 1.3 – Результат пошуку плагіату програми Unicheck [18]

Ціни змінюються в залежності від кількості сторінок: 5 доларів за кожні 20 сторінок, 10 доларів за кожні 50 і 15 доларів за кожні 100 [17]. Unicheck добре впорався зі співставленням деяких уривків та пошуком джерела плагіату.

Отже, було проаналізовано програмні системи для пошуку плагіату за семантичним аналізом текстів – Plagiarisma, Writer та Unicheck. Кожна має свої

переваги та недоліки, але вибір програми залежить від потреб та можливостей користувача. Plagiarisma та Writer можуть бути відмінним вибором для швидкої та доступної перевірки, а Unicheck – для точної, проте він є платною версію. З вищесказаного, даний напрям залишається актуальним напрямом інформаційних технологій, а подальша розробка програмного забезпечення в області пошуку плагіату – актуальним завданням.

1.4 Особливості аналізу україномовного контенту для задач пошуку плагіату

Задачі пошуку плагіату актуальні й для української мови, що вимагає аналізу багатьох аспектів тексту, з урахуванням унікальної структури мови, яка надає додаткових труднощів для розуміння тексту. Деякі з завдань включають аналіз анафор, порядку слів, граматики та морфології, використання офіційних слів і знаків пунктуації та неологізмів [20].

Науковці досліджували цю тему, провівши порівняння різних програмних рішень для виявлення запозичень. Найбільш поширені методи - синтаксичні (метод шинглів, Long Sent, Heavy Sent) та лексичні (TF-IDF та Метод Коудури). Але у кожного з методів є недоліки [21].

Якщо розглядати синтаксичні методи, то головними недоліками у [21]:

– перевіряючи об’ємні тексти, алгоритм Бродера, як правило, стає менш ефективним. Це пов’язано з тим, що збільшення кількості шинглів призводить до більшої кількості відповідних порівнянь, що призводить до збільшення операцій та зниження продуктивності;

– методи Long Sent та Heavy Sent схожі тому недоліками є повільна і низька точність, особливо під час вивчення великих текстів.

Щодо лексичних методів, то недоліками у [21]:

– методі TF-IDF є те що в сигнатуру не включаються рідкісні слова та низький процент плагіату при перевірці великого тексту;

– незначні зміни у вмісті документа можуть зробити метод Koudura нестабільним.

Особливим пунктом є технічна частина, яка є обмежувачем для того, щоб виконувати порівняння, система потребує багато пам’яті та обчислювальної потужності [22]. Перефразувати тексти з кількох джерел проблематично, оскільки важко визначити джерело плагіату. Деякі методи виявлення плагіату можуть аналізувати лише одне джерело за раз, незважаючи на те, що визначити кілька джерел складніше, ніж знайти одне джерело [22]. Це тому, що багато з цих методів визначають джерело підозрілого документа як єдине джерело.

Отже, перевірка на плагіат вимагає аналізу багатьох аспектів твору, унікальна структура української мови навіть створює додаткові труднощі для розуміння текстів. Деякі з цих завдань включають аналіз анафор, порядку слів, граматики та морфології, використання офіційних слів і знаків пунктуації та неологізмів. Крім того, при перевірці на плагіат необхідно враховувати різні мовні відтінки та використання синонімів, які можуть вказувати на копіювання іншого тексту. Також важливо враховувати стиль та тон твору, щоб визначити, чи є вони оригінальними, або ж скопійовані з іншого джерела.

1.5 Постановка задачі

Метою роботи є розробка методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату та відповідної інформаційної системи пошуку плагіату за семантичним аналізом текстів, що дозволяє за текстовим контентом визначати числову оцінку встановленого обсягу текстових запозичень шляхом його семантичного аналізу з використанням n-грам.

У рамках досягнення поставленої мети ставляться наступні задачі:

– розробити метод семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату;

– спроєктувати структуру інформаційної системи пошуку плагіату за семантичним аналізом текстів й структуру відповідної БД;

– розробити інформаційну систему пошуку плагіату за семантичним аналізом текстів;

– провести тестування створеної інформаційної системи пошуку плагіату за семантичним аналізом текстів;

– провести дослідження практичної ефективності розробленого методу за використання створеної інформаційної системи.

Висновки до розділу 1

У першому розділі було проведено характеристику предметної області та проаналізовано семантичний аналіз тексту й задачу виявлення плагіату, де було розглянуто види ідентифікації схожих або копійованих текстів в інтернеті або в базах даних та методи детектування плагіату, що можна використати для виявлення плагіату під час використання семантичного аналізу тексту.

У наступному підрозділі розглядаються методи та засоби, які використовуються для семантичного аналізу тексту та виявлення плагіату. Було досліджено такі методи, як пошук плагіату за ключовими словами та утворення n-грам.

Під час аналізу існуючих рішень для пошуку плагіату за семантичним аналізом текстів було проведено огляд ряду існуючих комерційних систем і систем з відкритим кодом, які використовують різні методології та алгоритми для виявлення плагіату.

Далі було розглянуто унікальні особливості виявлення плагіату в аналізі україномовного контенту, такі як аналіз анафор, порядку слів, граматики та морфології, використання офіційних слів і знаків пунктуації та неологізмів, що є важливими міркуваннями при розробці системи виявлення плагіату для україномовного контенту.

Отже, для ефективного виявлення плагіату необхідно застосовувати комплексні методи і засоби семантичного аналізу текстів, а також ураховувати особливості аналізу мовного контенту для кожної конкретної мови. Дослідження

показали, що існують різні методи та підходи для виявлення плагіату, і їх вибір залежить від конкретних умов та вимог задачі. Україномовний контент має свої особливості, тому необхідно використовувати підходи, що враховують ці особливості, для ефективного виявлення плагіату в україномовних текстах.

Розділ 2 Метод семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату

2.1 Опис і кроки методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату

Схема реалізації методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату зображена на рисунку 2.1. Метод семантичного аналізу текстів з використанням n-грам перетворює вхідні дані у вигляді цифрової текстової інформації у вихідні дані вигляду відсоткового рівня плагіату на знайдений аналог. Запропонований метод призначений для вирішення задач пошуку плагіату.

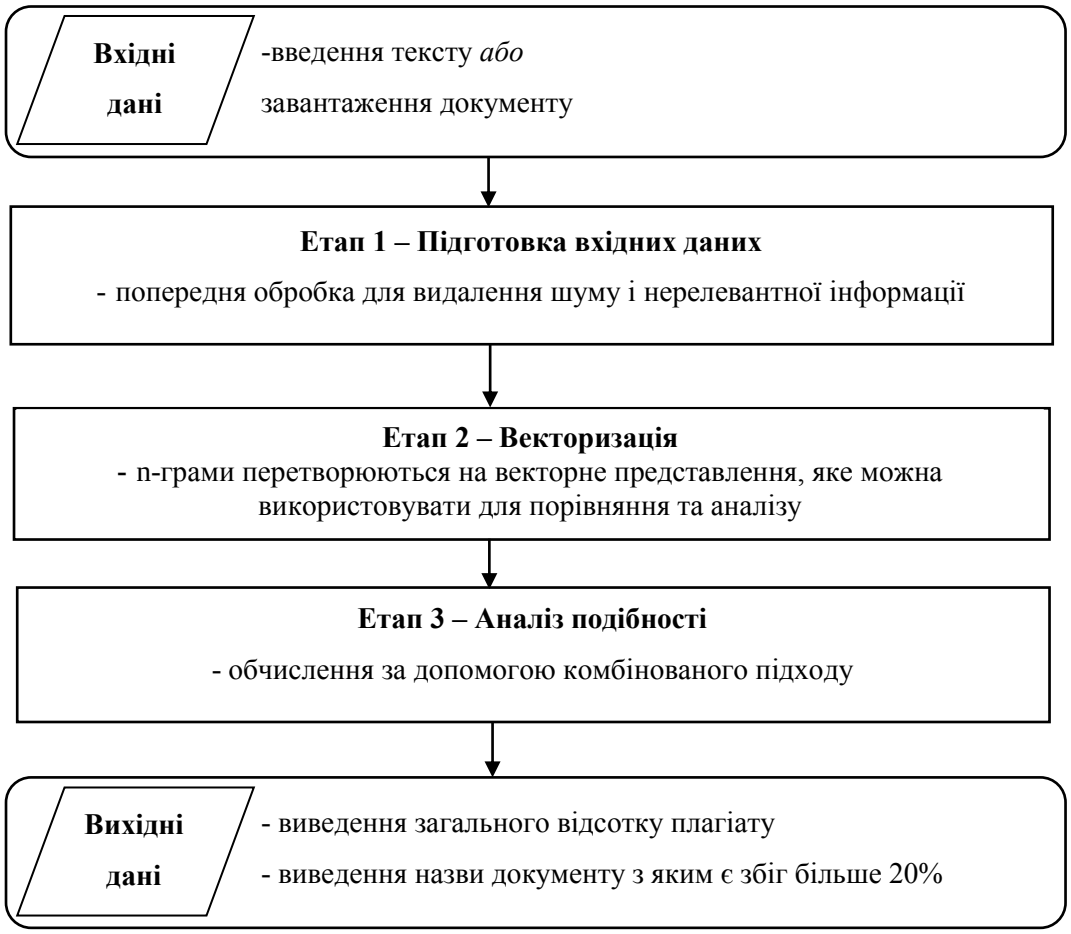


Рисунок 2.1 – Схема методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату

Для перевірки тексту на плагіат необхідні вхідні дані, а саме: текст або документ, який необхідно перевірити та текст або документ, з яким вони будуть порівнюватися.

Вихідними даними буде число збігів у тексті, що визначатиметься у відсотках.

Щоб підготувати текстові документи до аналізу, початковий етап передбачає усунення нерелевантної інформації та шуму. Цей процес передбачає:

– видалення знаків пунктуації. Це включає будь-які символи, що не є літерами або цифрами, такі як крапки, коми, знаки питання та інші. Видалення знаків пунктуації допомагає збільшити ефективність аналізу тексту, оскільки дозволяє ігнорувати незначні елементи, які не несуть корисної інформації для розуміння тексту.

– видалення стоп-слів. Стоп-слова - це слова, які зазвичай зустрічаються в тексті дуже часто, такі як "і", "він", "вона", "тому", "це" тощо. Ці слова не мають значення для розуміння контексту тексту та можуть заважати аналізу. Вилучення стоп-слів допомагає зменшити розмір тексту та збільшити точність аналізу.

– текст ділиться на токени. Токени – це окремі слова або символи, які використовуються для побудови тексту [23]. Вони можуть бути розділені пробілами, знаками пунктуації або іншими роздільниками.

Потім формуються n-грами шляхом вилучення послідовності з n послідовних токенів.

Після генерації n-грам наступний етап включає перетворення їх у векторні представлення, які можуть бути корисними для аналізу та порівняння. Як правило, це досягається за допомогою таких методів, як:

– "bag-of-words" (мішок слів). У цьому методі документ представляється як набір слів, де кожен елемент відповідає одному слову в документі. Для векторизації слова підраховуються в кожному документі та порівнюється їх частота появи. Таким чином, ми отримуємо матрицю векторів, де кожен

стовпець представляє один документ у корпусі текстів, а кожен рядок представляє одне слово;

– Марківська модель: допомогає визначити ймовірність переходу між різними станами тексту [24]. Ці стани включають початок токену, кінець токену та середина токену. Навчаючись на корпусі текстів, модель Маркова застосовується для токенізації нового тексту.

Формула для ланцюгів Маркова другого порядку, або двограм, може бути записана наступним чином [24]:

$$P(w_i|w_i - 1) = \frac{count(w_i-1,w_i)}{count(w_i-1)},$$

де: $P(w_i|w_i - 1)$ – ймовірність того, що слово wi з'явиться після слова wi-1; $count(w_i - 1, w_i)$ – кількість разів, коли двіграма wi-1, wi з'являється в тексті; $count(w_i - 1)$ – кількість разів, коли слово wi-1 з'являється в тексті.

Формула для розрахунку триграм [24]:

$$P(w_i|w_i - 1, w_i - 2) = \frac{count(w_i-2,w_i-1,w_i)}{count(w_i-2,w_i-1)},$$

де: $P(w_i|w_i - 1, w_i - 2)$ – триграма, яка складається з трьох слів;
$count(w_i - 2, w_i - 1, w_i)$ – кількість разів, коли триграма зустрічається у тексті; $count(w_i - 2, w_i - 1)$ – кількість разів, коли перші два слова триграми зустрічаються у тексті.

Після векторизації n-грам третій етап виявлення плагіату передбачає використання формул для обчислення подібності двограм між документами, а саме:

$$percnt_{Bi} = \frac{num_{Bi}}{total_{Bi}} \cdot 100\%,$$

де $percnt_{Bi}$ – відсоток збігів за біграмами; num_{Bi} – кількість співпадінь біграм серед двох текстів; $total_{Bi}$ – загальна кількість біграм серед двох текстів.

Формула для обчислення подібності триграм між документами:

$$percnt_{Tri} = \frac{num_{Tri}}{total_{Tri}} \cdot 100\% ,$$

де $percnt_{Tri}$ – відсоток збігів за триграмами; num_{Tri} – кількість співпадінь триграм серед двох текстів; $total_{Tri}$ – загальна кількість триграм серед двох текстів.

$$percnt_{total} = percnt_{Bi} + percnt_{Tri} ,$$

де $percnt_{total}$ – загальний відсоток збігів.

Після отримання індексу подібності між двома документами можна буде зробити висновок щодо того, чи документ містить плагіат, якщо індекс перевищує певний поріг.

Отже, для боротьби з проблемою плагіату було проаналізовано етапи роботи над методом семантичного аналізу текстів з використанням n-грам. Розроблений метод семантичного аналізу текстів з використанням n-грам перетворює вхідні дані у вигляді цифрової текстової інформації у вихідні дані вигляду відсоткового рівня плагіату на знайдений аналог. Запропонований метод призначений для вирішення задач пошуку плагіату. Концепція передбачає розбиття тексту на більш дрібні компоненти та піддавання їх математичному аналізу. Цей підхід дуже ефективний у виявленні випадків плагіату, таким чином забезпечуючи автентичність і оригінальність написаного вмісту.

2.2 Особливості використання n-грам для задач пошуку плагіату у методі семантичного аналізу текстів

Використання n-грамів у семантичному аналізі тексту передбачає врахування кількох важливих факторів. Одним із них є вибір найкращого значення для n на основі довжини тексту та бажаного рівня деталізації. Менші розміри n-грамів, наприклад 2 грами та 3 грами, можуть бути ідеальними для коротших текстів, тоді як для довших текстів можуть бути корисні більші розміри, наприклад 5 грамів або 6 грамів.

Ще один фактор, який слід враховувати, це вибір, коли використовувати n-грами, а коли їх уникати. Їх використання може призвести до розрідженої матриці у великих наборах даних із багатьма нульовими значеннями. Однак це можна пом’якшити за допомогою таких методів, як кластеризація та зменшення розмірності, щоб мінімізувати кількість функцій. Варто зазначити, що тлумачення n-грам може бути неоднозначним, деякі з них мають кілька можливих значень.

Особливості n-грам наведені нижче [25]:

– вибір n елементів: Вибір ідеального розміру n-грам під час аналізу тексту залежить від довжини тексту та бажаного рівня точності дослідження. Для невеликих текстів із меншою кількістю слів дво- або триграми є більш показовими, оскільки пропонують детальне уявлення про зв’язки між окремими словами та фразами. Коли ви маєте справу з довгими текстами, що складаються з багатьох слів і речень, вибір більших розмірів n-грамів, таких як п’ять або шість, може дати більш бажані результати. Це пов’язано з тим, що ці більші розміри сприяють більш глибокому розумінню вмісту, уможливлюючи відображення вагоміших контекстних зв’язків між словами та фразами. В результаті текст трактується більш комплексно і цілісно. Що стосується аналізу тексту, визначення ідеального розміру в n-грам залежить від кількох факторів, включаючи довжину тексту, його специфіку та передбачувану мету дослідження.Overlapping n-grams: у деяких випадках може знадобитися

використовувати n-грами, що перекриваються, щоб отримати більш повну та значущу інформацію. Це передбачає створення n-грам, які мають спільні токени з сусідніми n-грамами.

– sparse data: При використанні n-грам з великими наборами даних виникає значна проблема у вигляді розрідженого матричного представлення. Незліченна кількість комірок матриці, ймовірно, міститиме нульові значення, що призведе до проблематичних обчислень і зниження ефективності моделі. Існують різноманітні тактики для вирішення цієї проблеми, і одна з них зосереджена на зменшенні розмірності даних. Цей підхід має на меті мінімізувати кількість вимірів, присутніх у матриці, зберігаючи при цьому значний рівень інформації про дані. Деякі методи, які використовуються для досягнення цього, включають головні компоненти або матричну факторизацію. Кластеризація даних представляє альтернативний підхід. Це передбачає групування однакових n-грам, зменшення кількості ознак набору даних і надання більш короткого представлення. Однак розуміння та інтерпретація кластерів є вирішальними для їх використання в подальшому аналізі та моделюванні.

– неоднозначність: Багатовимірні n-грами, хоча й корисні для аналізу набору даних, можуть створювати розріджені матричні представлення з численними клітинками з нульовим значенням. Це викликає труднощі при роботі з великими наборами даних. На противагу цьому, можна використовувати методи зменшення функцій, такі як кластеризація або зменшення розмірності. Ці методи працюють, щоб зменшити кількість ознак, наявних у наборі даних. Використання N-грам може призвести до неточного тлумачення результату через можливість кількох значень. Однак контекст та інші методи обробки природної мови можуть роз’єднати ці неоднозначності. Довжина тексту, а також його складність можуть визначити ідеальний розмір у n-грамах, щоб гарантувати точний аналіз тексту.

Загалом, використання n-грам може значно розширити можливості аналізу тексту. Однак для досягнення оптимальних результатів потрібен ретельний вибір значення n, щоб пом’якшити проблеми неоднозначності та

розрідженості. Глибоке розуміння цих факторів допоможе забезпечити точність та ефективність виявлення плагіату за допомогою аналізу n-грам. Даний інструмент є неймовірно корисним для отримання цінної інформації з текстових даних, що робить його ідеальним для багатьох програм, включаючи виявлення плагіату.

2.3 Проєктування структури інформаційної системи пошуку плагіату за семантичним аналізом текстів

Схема структури інформаційної системи пошуку плагіату за семантичним аналізом текстів зображена на рисунку 2.2.

Рисунок 2.2 – Схема структури інформаційної системи пошуку плагіату за семантичним аналізом текстів

БД – модуль бази даних, призначений для зберігання та організації даних для аналізу. Це забезпечує швидкий доступ до даних, пошук і усуває зайву інформацію. У результаті бази даних можуть вміщувати величезні обсяги інформації, включаючи дані користувача, текстову документацію, метадані, серед інших деталей.

Модуль редагування бази даних незамінний для управління інформацією. Це дозволяє легко змінювати збережену інформацію, додавати нові записи, редагувати існуючі, видаляти надлишки та оновлювати дані. Крім того, він надає ефективні засоби пошуку та фільтрації даних, а також має кілька інших важливих функцій, таких як сортування, експорт та імпорт даних, керування правами доступу до даних, забезпечення безпеки даних і створення звітів.

Модуль обробки тексту – це універсальний інструмент, який дозволяє користувачам аналізувати текстову інформацію та маніпулювати нею. Його основні функції включають розпізнавання та виділення ключових слів, аналіз тонів і ідентифікацію важливих деталей, таких як імена, дати та місця розташування. Завдяки різноманітним додаткам, починаючи від аналізу соціальних медіа й закінчуючи обробкою блогів, цей модуль може допомогти зібрати цінні дані для подальшого використання.

Модуль для виявлення плагіату дозволяє користувачам оцінити, наскільки схожі два фрагменти тексту один на одного, використовуючи різні методи аналізу. Це, зокрема, порівняння n-грам, оцінка синтаксичної та семантичної подібності текстів тощо. Це незамінний інструмент для навчальних закладів і видавництв, які мають певні стандарти оригінальності та унікальності написання.

Використовуючи n-грами в аналізі тексту, модуль ретельно перевіряє вживання слів і контекст, виділяє часті слова та послідовності та ефективно визначає шаблони у великих обсягах текстових даних. N-грами стосуються послідовності з N слів, які зустрічаються разом у аналізованому тексті. Цей аналітичний інструмент ідеально підходить для ретельного вивчення текстових даних із соціальних мереж, бібліотек, наукових статей, документів тощо.

Отже, підчас проєктування структури інформаційної системи пошуку плагіату за семантичним аналізом текстів було описано чотири модулі, які дозволяють працювати з різними типами даних та виконувати різні завдання: БД для зберігання та організації даних, модуль роботи з текстами для аналізу та обробки текстової інформації, модуль пошуку плагіату для визначення подібності текстів та виявлення плагіату, та модуль роботи з н-грамами для аналізу тематичного контексту тексту та визначення частот вживання слів.

2.4 Структура бази даних інформаційної системи

Для зберігання даних створено БД, що включає у себе таблиці: «ngrams», «Plagiarism», «Documents». Схема БД зображена на рисунку 2.3.

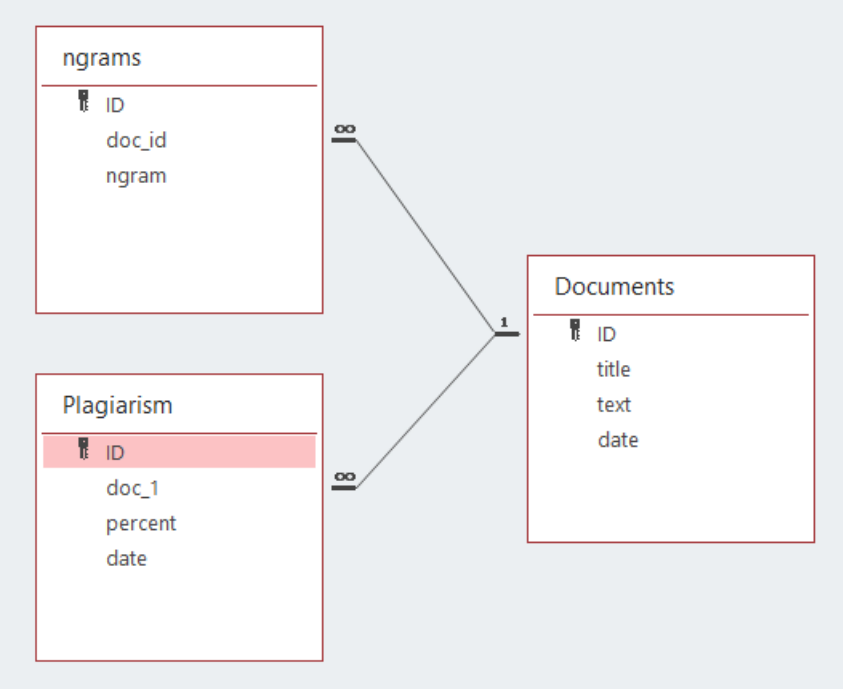


Рисунок 2.3 – Схема БД методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату

У таблиці «Documents» зберігається інформація про кожен документ, який потрібно перевірити на плагіат, включаючи його ідентифікатор, назву та необроблений текст документа.

Таблиця «ngrams» зберігає n-грами для кожного документа. Кожен рядок представляє одну n-граму та містить ідентифікатор n-грами, ідентифікатор документа, до якого вона належить, і сам текст n-грами.

Таблиця «Plagiarism» містить інформацію про перевірені документи, а саме: ідентифікатор документу, який перевірено, відсоток збігу з іншим текстом та дату проведення перевірки.

Під час пошуку плагіату користувач зазвичай виконує ряд кроків, щоб переконатися, що аналізований текст оригінальний чи ні (рисунок 2.4).

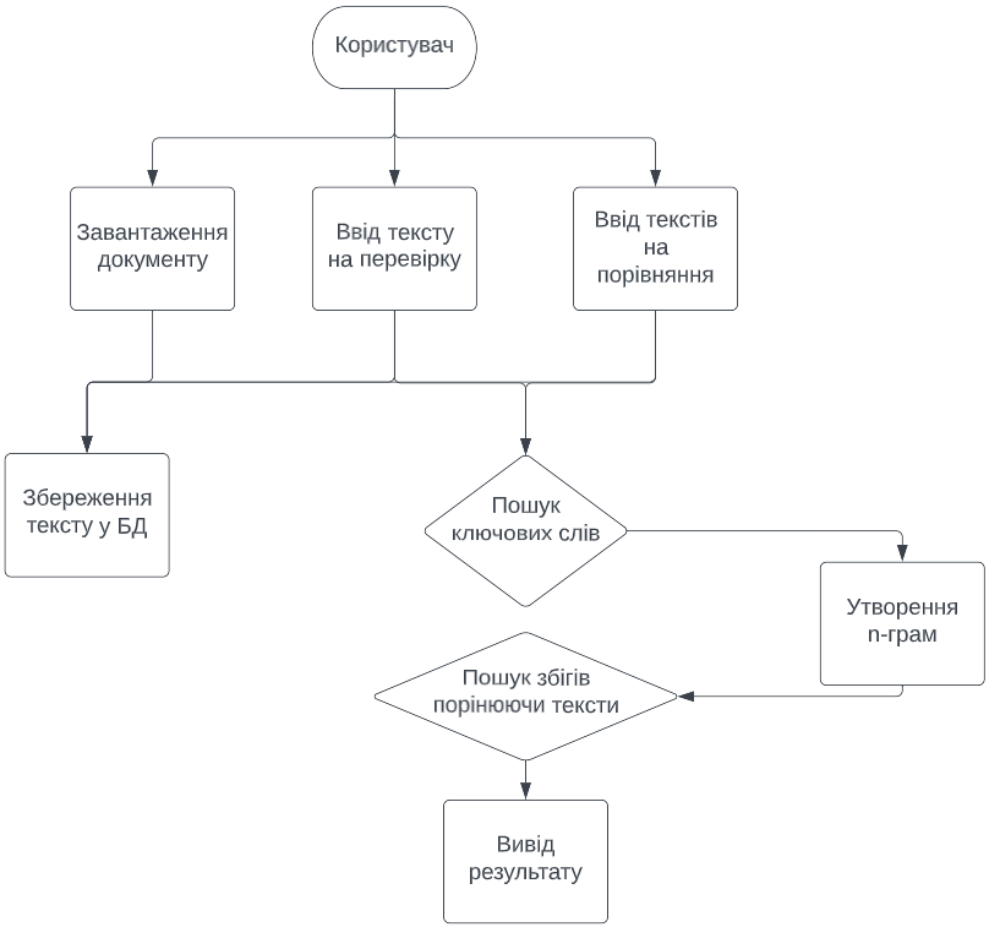


Рисунок 2.4 – Діаграма дій користувача при перевірці тексту на плагіат

Перед початком перевірки на плагіат користувач повинен підготувати текст. Це може бути, як скопійований текст так і текстовий документ, який потрібно перевірити, або порівняти його з іншим.

Щоб почати перевірку на плагіат, користувач завантажує документ або вставляє текст у відповідне поле, потім натиснувши кнопку відбувається порівняння його з іншим текстом, або ж з внутрішню БД.

Після завершення початкової фази визначаються та виділяються важливі ключові слова. Потім текст розбивається на n-грами та порівнюється з іншими текстами на предмет подібності. Після цього тексти можна зберегти в БД для подальшого використання та редагування.

Після завершення перевірки на плагіат користувач перевіряє результати, щоб оцінити ступінь оригінальності відображеного в тексті.

Також користувач може відредагувати або дізнатися, які є в дво- та триграми в тексті, що був збережений в БД (рисунок 2.5).

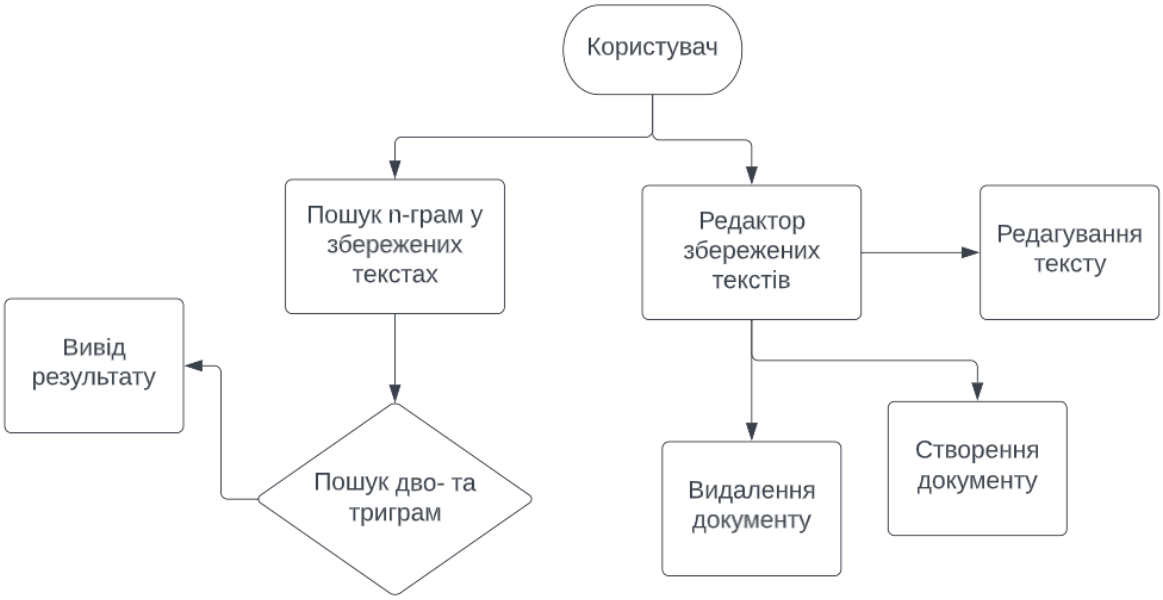


Рисунок 2.5 – Діаграма дій користувача з редагуванням збереженого тексту та пошуку n-грам

Взаємодія користувача з системою включає наступні дії: він може створити новий текст або відредагувати збережений текст, і в разі помилкового збереження текст може бути видалений з системи. Для аналізу n-грам в збереженому тексті користувач повинен вибрати текст зі списку, відкрити його та розпочати пошук n-грам. Після завершення пошуку, система поверне список

дво- та триграм, який може бути використаний користувачем для подальшого аналізу.

Отже, під час перевірки тексту на оригінальність система вибирає ключові слова та генерує n-грами для порівняння з іншими текстами. Отримані дані зберігаються в базі даних і надаються користувачеві для перегляду. За результатами користувач може оцінити рівень оригінальності тексту. Також система дозволяє користувачу зручно зберігати, редагувати та аналізувати текст, забезпечуючи зручну та ефективну роботу з текстовою інформацією.

2.5 Датасети для інформаційної системи пошуку плагіату за семантичним аналізом текстів

Під час розробки інформаційної системи було використано датасети, що складаються з великої кількості текстової інформації. Ці датасети були використані для забезпечення якості та точності роботи програми перевірки на плагіат. Зокрема, вони були використані для порівняння вихідного тексту з іншими текстами з дата сетів та виявлення можливих випадків плагіату.

До використаних дата сетів належать UberText 2.0, Браунський корпус української мови, Microsoft Research Paraphrase Corpus та PAN (Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection) Plagiarism Corpus 2010. Вони містять різні типи текстів, що дає змогу робити більш точні порівняння. Також вони відповідають міжнародним стандартам у галузі виявлення плагіату.

UberText 2.0 – це відкритий датасет українських текстів, що складається з понад 100 мільйонів токенів [26]. Датасет був створений компанією Texty.org.ua з метою покращення рівня обробки природньої мови для української мови [26].

Датасет UberText 2.0 містить тексти різної тематики з українськомовного Інтернету. Зокрема, до датасету входять новини, блоги, форуми, соціальні мережі, художня література та інші типи текстів [26]. Кожен текст в датасеті має відповідний URL, що дозволяє легко знайти оригінальний джерело тексту.

Для створення датасету використовувались методи автоматичної обробки природньої мови, такі як токенізація, нормалізація, лематизація та інші [26]. Крім того, були застосовані методи зменшення шуму та очищення даних.

UberText 2.0 може бути використаний для навчання моделей машинного навчання для різних задач, пов'язаних з обробкою природньої мови [26]. Наприклад, для класифікації текстів за темою, побудови рекомендаційних систем, аналізу емоцій тощо.

Загальнодоступність датасету дозволяє розширити область досліджень з обробки природньої мови для української мови (рисунок 2.6).

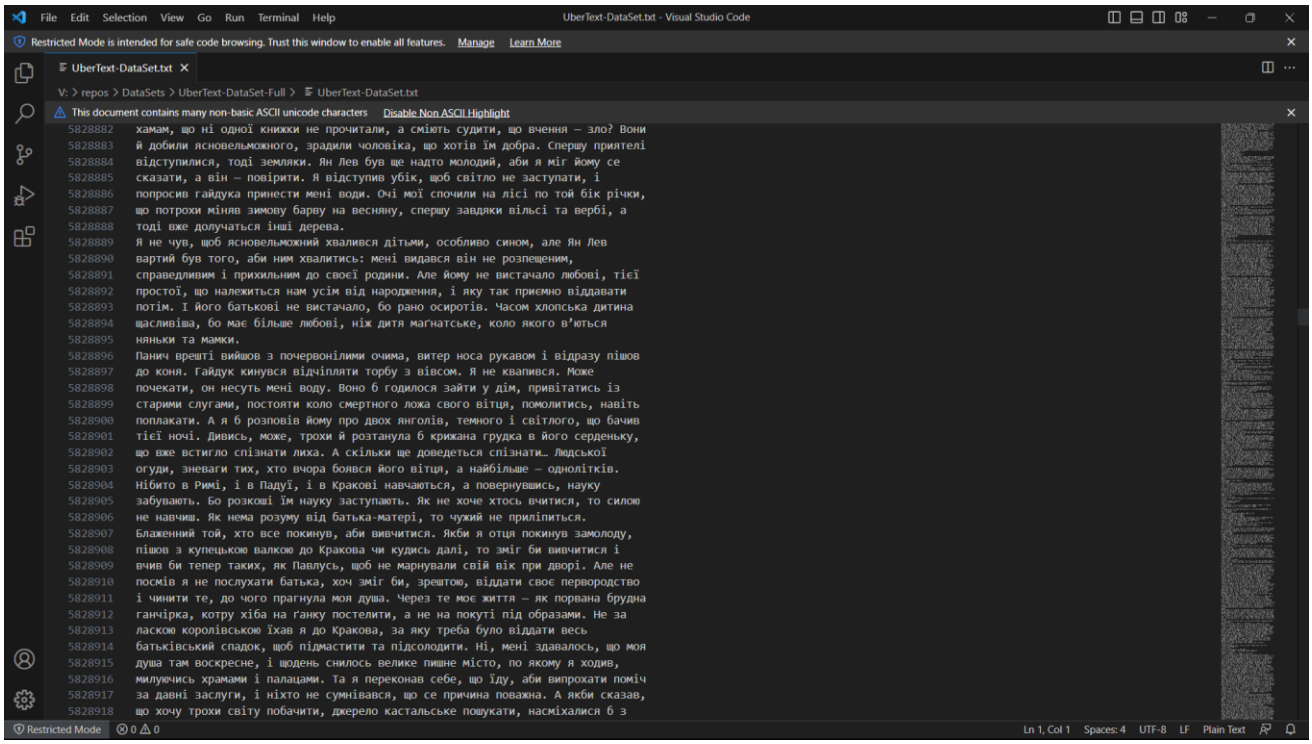


Рисунок 2.6 – Дані датасету UberText [26]

Браунський корпус української мови є одним з найвідоміших датасетів української мови, який був створений на основі Браунського корпусу англійської мови [27].

Датасет містить більше 1 мільйона слів з різних джерел, таких як художня література, періодичні видання, наукові публікації, аудіозаписи та інші [27]. Тексти у датасеті позначені тематичними мітками, що дозволяє використовувати

корпус для досліджень у різних сферах, включаючи лінгвістику, комп'ютерну лінгвістику, машинне навчання та ін.

Крім того, Браунський корпус української мови містить інформацію про граматичну структуру речень, а також теги для кожного слова, що дає змогу проводити дослідження у галузі обробки природньої мови, зокрема для побудови моделей машинного навчання для аналізу текстів, машинного перекладу та інших застосувань [27].

Датасет є відкритим і доступним для завантаження та використання в наукових дослідженнях та комерційних проектах згідно з умовами ліцензії [27]. Браунський корпус української мови є важливим ресурсом для дослідників, які працюють у галузі природної мови та машинного навчання (рисунок 2.7).

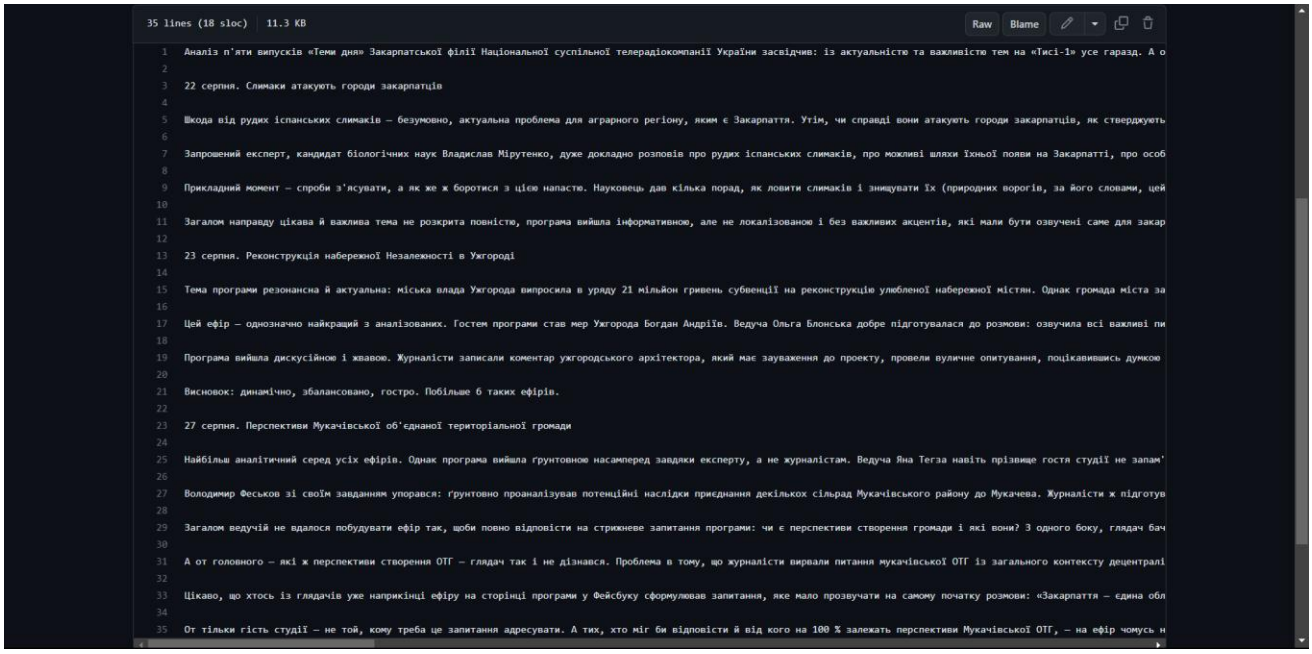


Рисунок 2.7 – Дані датасету браунського корпусу української мови [27]

MSRP (Microsoft Research Paraphrase Corpus) - це англомовний датасет, що містить понад 5800 пар англійських речень, зібраних з різних джерел, таких як статті, форуми, блоги тощо [28].

Кожна пара речень у датасеті містить оригінальне речення та його перефразування, які оцінені за рівнем семантичної близькості. MSRP використовується для завдання перефразування речень та оцінки різних методів

машинного навчання, що дозволяють визначати, наскільки два речення є семантично близькими [28].

MSRP був створений командою Microsoft Research з метою створення датасету для оцінки різних методів перефразування, в тому числі з використанням глибинного навчання [28]. Датасет є відкритим і доступним для безкоштовного використання в наукових дослідженнях та комерційних проектах згідно з умовами ліцензії.

MSRP (рисунок 2.8) є важливим ресурсом для дослідження у галузі обробки природньої мови, зокрема для розробки систем автоматичного перекладу, сумаризації текстів, а також для використання в задачах відповіді на запитання та інших застосуваннях, які вимагають розуміння семантики тексту [28].

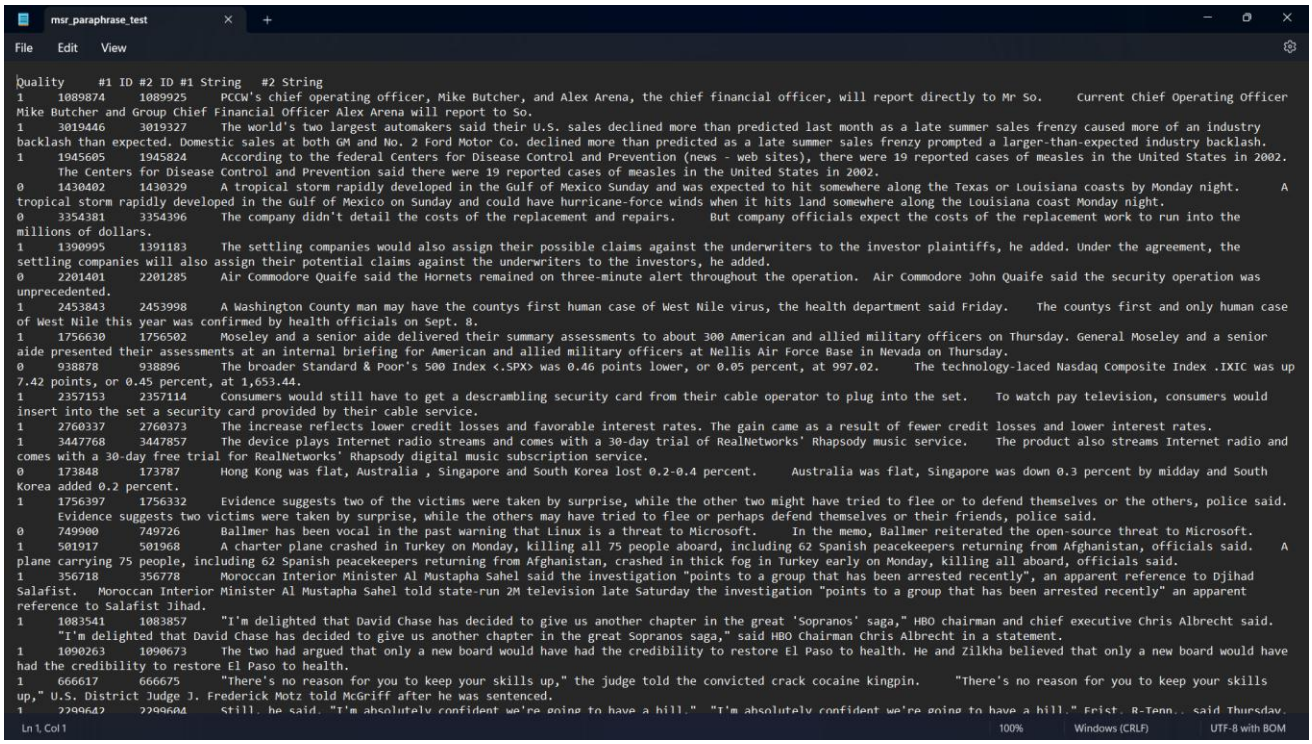


Рисунок 2.8 – Дані датасету MSRP [28]

PAN Plagiarism Corpus 2010 - це датасет, що містить текстові документи, які були використані для тестування систем виявлення плагіату в рамках PAN-10 shared task, що проходив у 2010 році [29].

Датасет містить текстові файли з різних джерел, які були створені з метою тестування різних систем виявлення плагіату, включаючи системи на основі векторної моделі, машинного навчання та статистичного аналізу. Кожен файл містить різні види плагіату, включаючи цитування без джерела, переробку тексту та повністю скопійований матеріал [29].

Цей датасет є важливим ресурсом для дослідження у галузі виявлення плагіату та розробки систем, що здатні виявляти різні типи плагіату. Він може бути використаний для тестування ефективності різних методів виявлення плагіату, а також для порівняння різних систем виявлення плагіату [29].

PAN Plagiarism Corpus 2010 є відкритим датасетом, що доступний для безкоштовного використання у наукових дослідженнях та комерційних проектах згідно з умовами ліцензії [29]. Він може бути використаний у різних застосуваннях, що вимагають виявлення плагіату, таких як системи авторства та контент-аналізу, системи відстеження авторства та багато інших (рисунок 2.9).

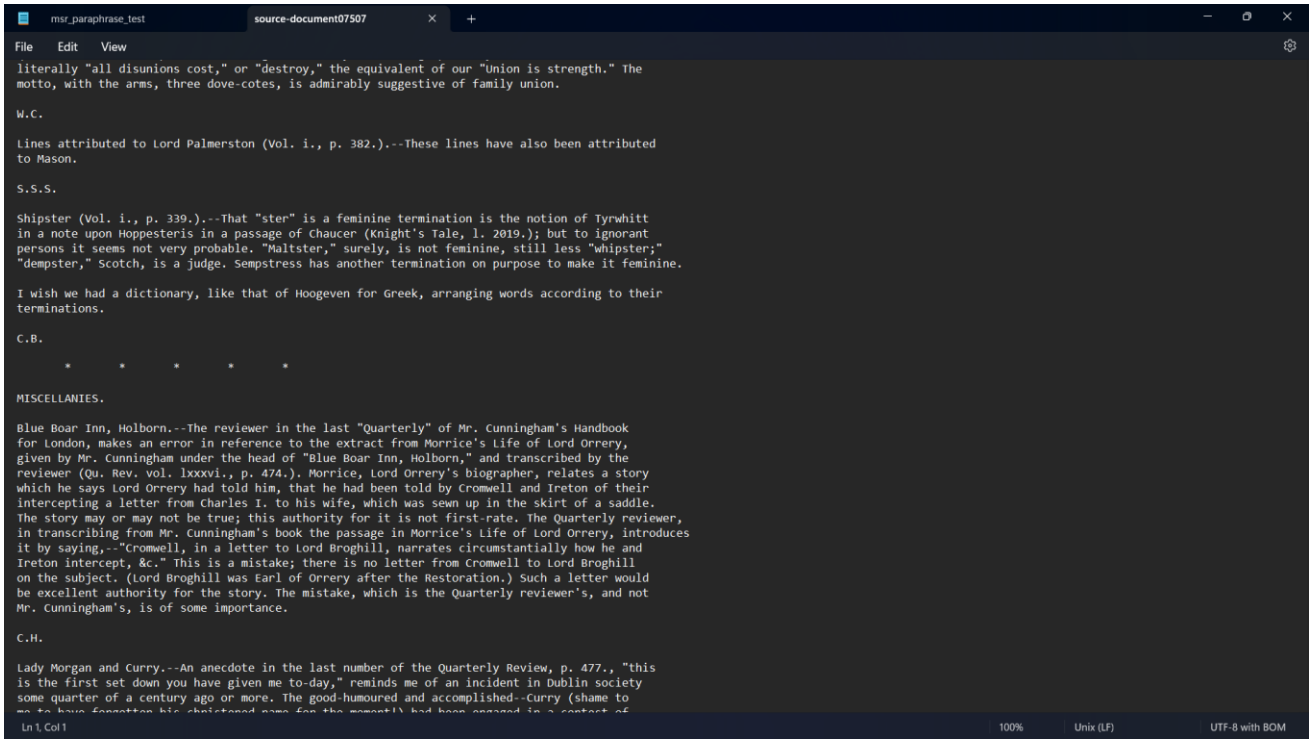


Рисунок 2.9 – Дані датасету PAN Plagiarism Corpus 2010 [29]

Отже, додання датасетів до інформаційної системи пошуку плагіату за семантичним аналізом текстів допоможе підвищити ефективність та точність

виявлення плагіату. Це дозволить виявляти потенційні випадки плагіату з більшою точністю та надійністю.

2.6 Вибір засобів розробки інформаційної системи пошуку плагіату за семантичним аналізом текстів

OpenNLP – це програмне забезпечення з відкритим вихідним кодом, створене Apache Software Foundation, пропонує широкий спектр інструментів обробки тексту [30]. Воно допомагає розпізнавати іменовані об’єкти, знаходити речення, аналізувати синтаксис, розпізнавати частини мови, класифікувати текст тощо. Програмне забезпечення підходить для створення різноманітних програм обробки природної мови, таких як пошукові системи, системи автоматичного перекладу та автоматичного аналізу тексту [30].

Для перевірки тексту на плагіат за допомогою n-грам OpenNLP використовують для виконання таких завдань, як [30]:

– виявлення слів у текстах, що дає змогу ідентифікувати спільні слова в різних текстових порівняннях;

– порівняння двох або більше текстів;

– категоризація текстів у кластери, організовані за темою чи стилем написання;

– автоматичний переклад, його можна використовувати для аналізу вхідної мови, що у свою чергу допоможе ідентифікувати слова та фрази іншої мови.

Одним з ключових інструментів бібліотеки OpenNLP є TokenizerME. TokenizerME – це універсальний метод, який пропонує гнучкість налаштування параметрів на основі конкретних вимог [31]. Його можна налаштувати для визначення різних мовних структур, діалектів та інших мовних нюансів. Цей метод також надзвичайно швидкий і ефективний для обробки великих обсягів текстових даних, що робить його незамінним інструментом для завдань, які включають обробку величезних обсягів інформації [30].

У бібліотеці OpenNLP TokenizerME є часто використовуваним методом токенізації тексту. Він розділяє текст на окремі лексеми або слова, забезпечуючи основу для різноманітних інших алгоритмів обробки тексту, зокрема у: обробці природної мови, машинному перекладі, розпізнаванні мовлення та інших сферах [30].

Щоб розділити текст на окремі токени, метод TokenizerME використовує ряд правил: розділення за пробілами, знаки пунктуації, символи нового рядка та інше [31]. Інші методи обробки природної мови, такі як ланцюги Маркова, часто інтегруються з TokenizerME для підвищення точності операцій обробки тексту.

Отже, використання OpenNLP у інформаційній системі пошуку плагіату за семантичним аналізом для задач пошуку плагіату може значно підвищити ефективність та точність виявлення плагіату. OpenNLP має ряд унікальних здатностей, таких як розпізнавання речень, слів, порівняння текстів та їх кластеризація, що дозволяє зменшити кількість помилкових виявлень плагіату та збільшити точність визначення оригінальності тексту.

Висновки до розділу 2

У другому розділі було описано метод семантичного аналізу текстів з використанням n-грам для пошуку плагіату. Розглянуто процес розбиття текстів на n-грами, обробку отриманої інформації та формування семантичної моделі тексту. Описано кроки реалізації методу та алгоритм пошуку плагіату.

Описуючи особливості використання n-грам для задач пошуку плагіату у методі семантичного аналізу текстів було проаналізовано особливості використання n-грам для пошуку плагіату. Розглянуто вплив розміру n-грамів на точність та швидкість пошуку плагіату.

В результаті розроблено метод пошуку плагіату, який дозволяє виявляти запозичення текстів з використанням семантичного аналізу із використанням n-грам. Розроблений метод семантичного аналізу текстів з використанням n-грам

перетворює вхідні дані у вигляді цифрової текстової інформації у вихідні дані вигляду відсоткового рівня плагіату на знайдений аналог.

Також в розділі було описано структуру інформаційної системи для пошуку плагіату за семантичним аналізом текстів. Розглянуто складові системи, їх взаємодію та функціональні можливості. Далі було описано структуру бази даних інформаційної системи для пошуку плагіату за семантичним аналізом текстів. Розглянуто сутності, атрибути та зв'язки між ними. Описано процес зберігання та оновлення даних.

До інформаційної системи було додано чотири датасети для забезпечення більш точної та ефективної перевірки оригінальності тексту та більш широкого покриття мов та тематик.

Під час опису засобів розробки інформаційної системи для пошуку плагіату за семантичним аналізом текстів було розглянуто технологію та інструменти, що були використані для реалізації проекту.

Розділ 3 Програмна реалізація інформаційної системи пошуку плагіату за семантичним аналізом текстів

3.1 Структура модулів інформаційної системи пошуку плагіату за семантичним аналізом текстів

Інформаційна система містить реалізацію методу семантичного аналізу текстів з використанням n-грам, яка перетворює вхідні дані у вигляді цифрової текстової інформації у вихідні дані у вигляді відсоткового рівня плагіату на знайдений аналог. Діаграма класів є основним інструментом у дослідженні та розумінні модулів інформаційної системи виявлення плагіату, що використовують семантичний аналіз тексту. Діаграма класів надає візуальне відображення взаємозв'язків між різними класами програми і показує, як вони взаємодіють між собою для досягнення поставленої мети – виявлення плагіату та оцінки оригінальності текстових матеріалів (рисунок 3.1).

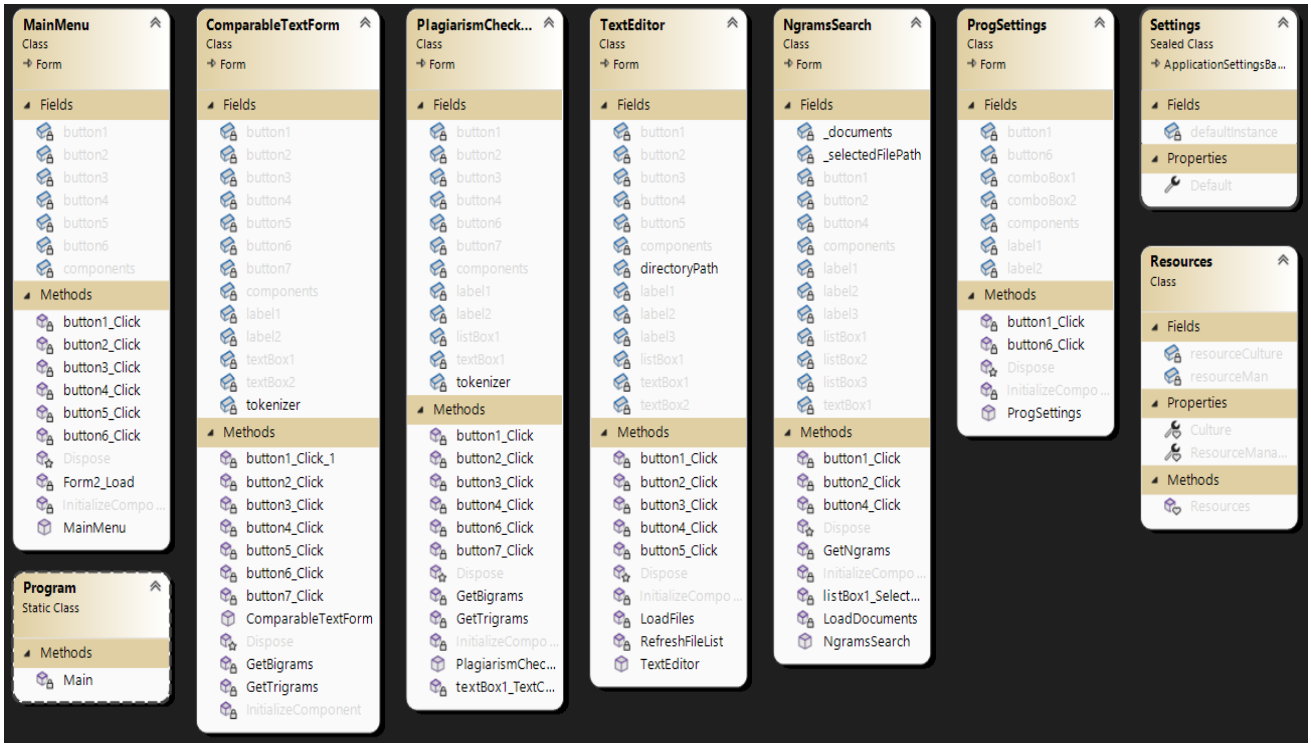


Рисунок 3.1 – Діаграма класів інформаційної системи пошуку плагіату за семантичним аналізом текстів

Клас MainMenu є формою головного меню програми, яка відображається користувачу і забезпечує зручний спосіб взаємодії з програмою (рисунок 3.2).

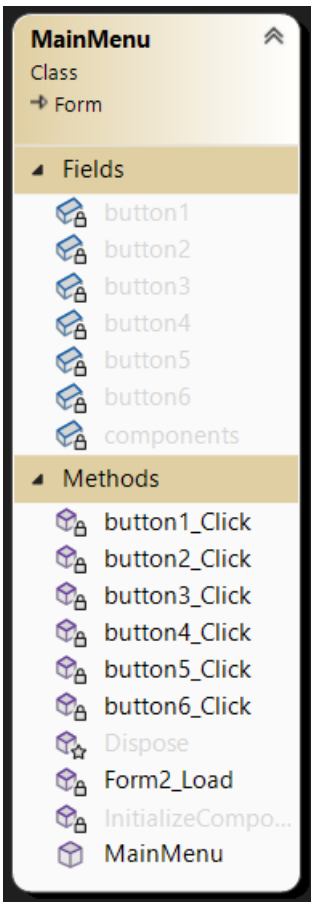


Рисунок 3.2 – Клас MainMenu

Даний клас містить різні кнопки, які відповідають різним функціям або модулям програми. Основна функція цього класу полягає в перехопленні подій натискання цих кнопок і запуску відповідних дій або відкриття відповідних форм. Наприклад, при натисканні кнопки "Comparable Text" створюється об'єкт класу ComparableTextForm і відображається відповідна форма. Аналогічно, при натисканні інших кнопок створюються та відображаються відповідні форми або виконуються певні дії у програмі. Це дозволяє користувачеві зручно вибирати потрібні функціональності та взаємодіяти з програмою шляхом натискання кнопок у головному меню.

Клас ComparableTextForm виконує порівняння двох введених текстів, використовуючи методи токенізації tokenizer та функції генерації дво-

(GetBigrams) та три-грам (GetTrigrams) (рисунок 3.3). Результат порівняння відображається у вікні повідомлення за допомогою MessageBox.Show.

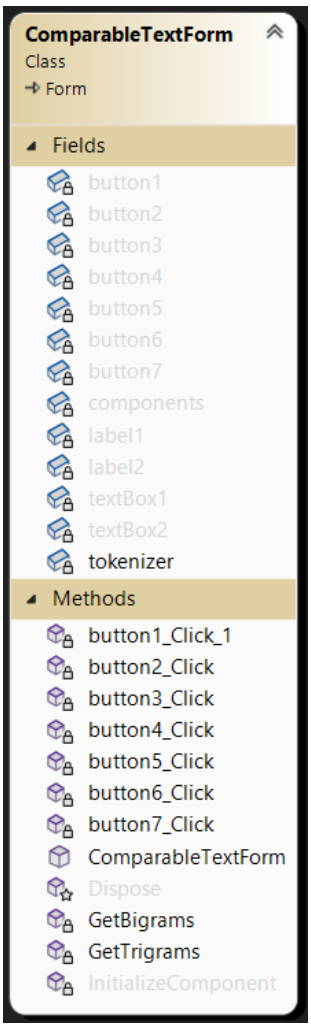


Рисунок 3.3 – Клас ComparableTextForm

Клас ComparableTextForm також має інші методи, які відповідають за інші функції програми, такі як видалення стоп-слів та розділових знаків (button2_Click, button3_Click), збереження документу(button7_Click), завантаження текстових файлів у текстове поле (button4_Click, button5_Click) та перехід до іншої форми (button6_Click). Аналогічні кнопки присутні в класі PlagiarismChecker.

Клас PlagiarismChecker є формою програми, яка використовується для перевірки плагіату в текстових документах (рисунок 3.4). Він містить різні функції та логіку, яка дозволяє користувачеві ввести текст, виконати аналіз

тексту, поділивши його на токени, та порівняти з вже існуючими текстами у БД. Результати перевірки відображаються у вікні повідомлень, що містить та відсотки плагіату, а також в список, який містить імена файлів.

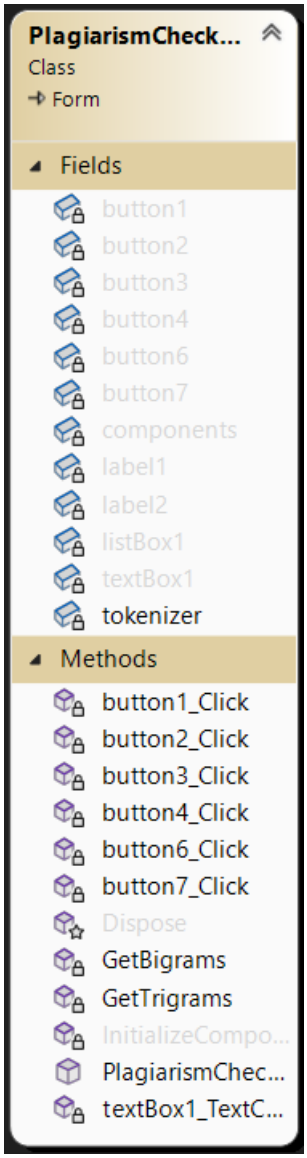


Рисунок 3.4 – Клас PlagiarismChecker

Клас TextEditor є формою програми, яка дозволяє користувачу редагувати та керувати текстовими файлами (рисунок 3.5). Він містить функціонал для завантаження списку файлів у директорії, за допомогою методу LoadFiles, та відображення їх назв у listBox1 (елемент управління типу ListBox), створення та видалення файлів, а також редагування їх вмісту.

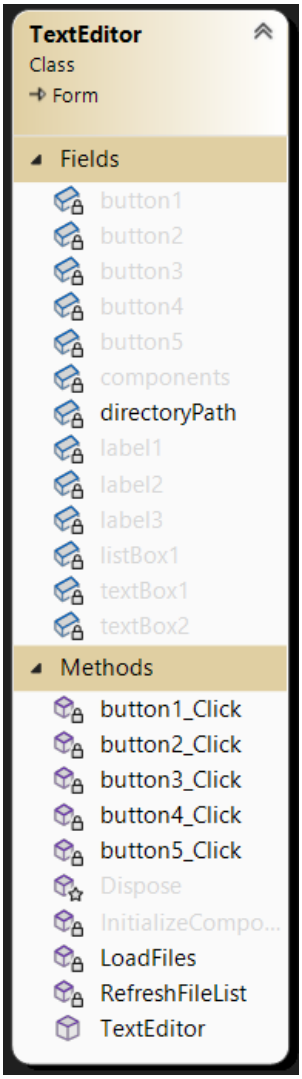


Рисунок 3.5 – Клас TextEditor

Метод RefreshFileList використовується для оновлення списку файлів після виконання деяких операцій, таких як видалення або створення файлу.

Клас NgramsSearch є формою програми, яка виконує пошук n-грам у вибраному текстовому файлі (рисунок 3.6). Він має функціонал для завантаження доступних документів, вибору файлу, токенізації вмісту файлу та пошуку біграм та триграм.

На основі токенів обчислюються біграми та триграми, які зберігаються у відповідні словники bigrams та trigrams. Після чого, список біграм та триграм виводяться у listBox2 та listBox3.

NgramsSearch використовує бібліотеку OpenNLP для токенізації тексту. Він дозволяє виявляти співпадіння у текстах для виявлення плагіату.

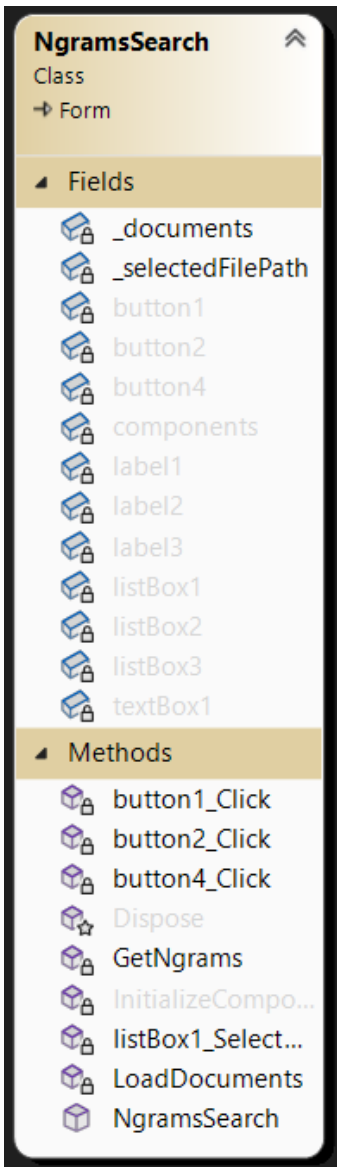


Рисунок 3.6 – Клас NgramsSearch

Клас ProgSettings є формою, яка дозволяє змінювати спосіб перевірки тексту шляхом вибору між використанням лише двограм, лише триграм або комбінованого методу, який поєднує дво- та триграми (рисунок 3.7). За допомогою цього класу можна налаштовувати параметри аналізу тексту для виявлення плагіату.

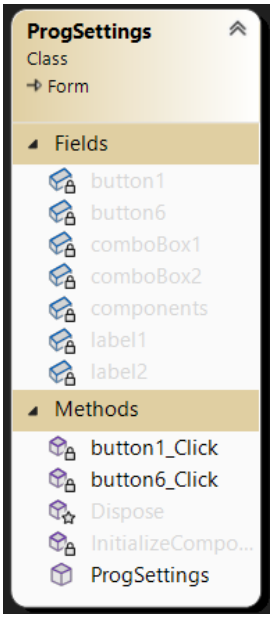


Рисунок 3.7 – Клас ProgSettings

Отже, розроблені модулі інформаційної системи пошуку плагіату за семантичним аналізом текстів мають класи, які взаємодіють між собою для забезпечення функціональності системи. Клас TextEditor відповідає за редагування та збереження текстових файлів, які підлягатимуть подальшому аналізу. Клас NgramsSearch здійснює пошук дво- та триграм у текстових документах, а також відображає результати аналізу. Клас ProgSettings дозволяє налаштувати параметри перевірки, зокрема вибрати метод аналізу – двограми, триграми або комбінований підхід. Клас ComparableTextForm є формою для порівняння введених текстів з метою виявлення плагіату. Клас PlagiarismChecker виконує процес порівняння тексту з БД та виявлення плагіату на основі семантичного аналізу. Разом ці класи створюють повноцінну структуру модулів інформаційної системи, яка дозволяє ефективно виявляти плагіат у текстових документах і спрощує користування системою для користувачів.

3.2 Особливості розробки складових інформаційної системи

Особливостями розробки складових інформаційної системи є поглиблений аналіз основних аспектів створення програми перевірки тексту на плагіат методом n-грам. Інформаційна система, яка розглядається в цьому

дослідженні, заснована на аналізі текстових даних і здатна ідентифікувати плагіат. Метод n-грам є ключовим аспектом цієї системи, оскільки він дозволяє створювати дво- та триграми з текстових сегментів. Код, відповідальний за генерування цих граматичних одиниць, є невід'ємною складовою загальної ефективності та майстерності системи у виявленні плагіату.

Функція для генерації двограм, представлена за допомогою схеми, та наведена на рисунку 3.8.

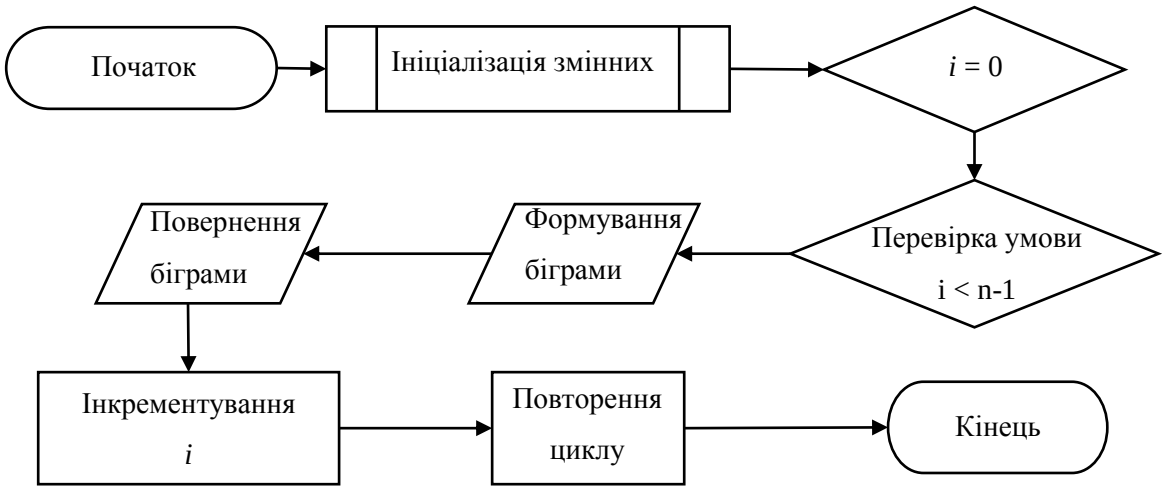


Рисунок 3.8 – Схема функції генерації двограм

Функція для генерації триграм наведена на рисунку 3.9 і призначена для автоматизації процесу переведення тексту в триграми.

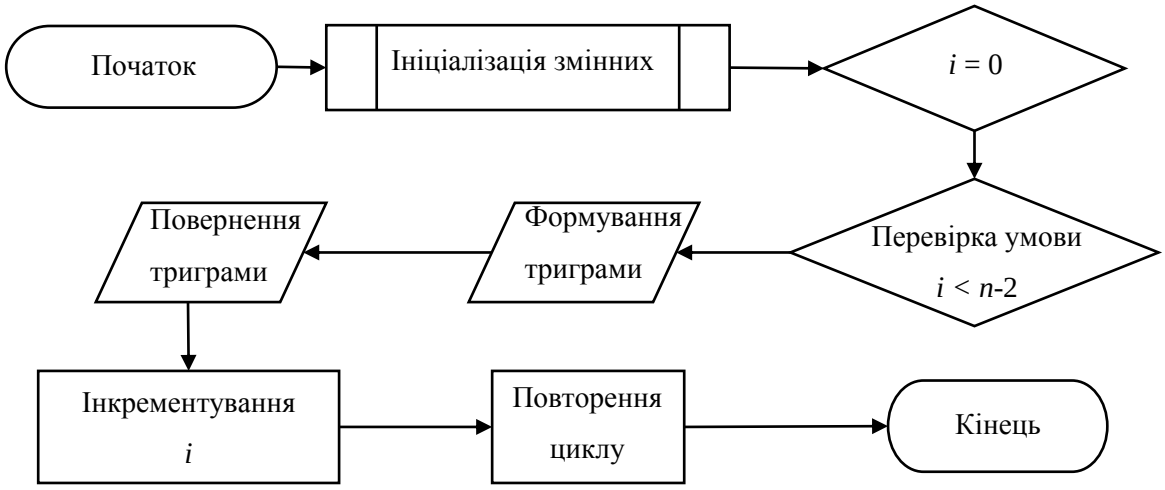


Рисунок 3.9 – Схема функції генерації триграм

Для формування n-грам, на вхід функції передається масив «tokens», який містить токени. Функція починає з ініціалізації змінної і зі значенням 0.

Потім вона перевіряє, чи «i» менше «tokens.Length - 1» (для двограм) та «tokens.Length - 2» (для триграм). Це умова циклу, яка перевіряє, чи існує достатня кількість токенів, щоб сформувати дво- або триграму.

Усередині циклу формується n-грама з поточного, наступного та чергового токенів за допомогою рядка форматування «tokens[i] + " " + tokens[i + 1]» (для двограм) та «tokens[i] + " " + tokens[i + 1] + " " + tokens[i + 2]» (для триграм). Отриману n-граму функція повертає за допомогою «yield return».

Після повернення n-грами збільшується значення «i» на 1, і цикл повторюється для наступних токенів.

Коли умова циклу стає хибною (коли «i» стає більше або дорівнює «tokens.Length - 1» (для двограм) та «tokens.Length - 2» (для триграм)), цикл завершується, і функція повертає кінцевий результат.

При збереженні тексту виконується перевірка наявності попередньо збереженого тексту. Цей процес має на меті виявлення дублікатів або перевірку, чи був такий самий текст збережений раніше. Перевірка допомагає уникнути зайвого збереження дублікатів і забезпечити економію пам’яті.

На рисунку 3.10 зображено фрагмент інформаційної системи, що відповідає за вивід повідомлення про збереження файлу:

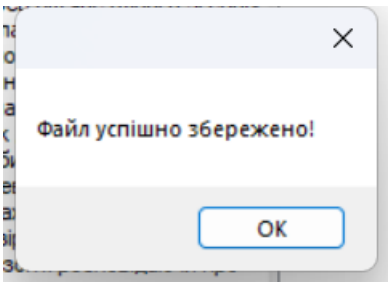


Рисунок 3.10 – Результат збереження тексту при перевірці

Спочатку, для збереження тесту, отримується список усіх файлів у вказаній теці за допомогою функції Directory.GetFiles(folderPath). Далі, для

кожного файлу, зчитується його вміст за допомогою System.IO.File.ReadAllText(file) і порівнюється з текстом, введеним у textBox1.Text та textBox2.Text.

Якщо вміст файлу збігається з textBox1.Text та відмінний від fileName1, тоді файл, що був збережений попередньо у fileName1, видаляється за допомогою System.IO.File.Delete(fileName1), а змінна fileName1 оновлюється, зберігаючи шлях до нового файлу.

Аналогічно, якщо вміст файлу збігається з textBox2.Text та відмінний від fileName2, відповідний файл, збережений у fileName2, видаляється, і fileName2 оновлюється.

Таким чином, код перевіряє наявність попередньо збережених текстів та, якщо знаходить дублікати, виконує їх видалення, зберігаючи оновлені шляхи до файлів. Це допомагає уникнути зайвого збереження дублікатів та економії пам'яті шляхом збереження лише унікальних текстів.

На рисунку 3.11 зображено процес обробки введеного тексту, що включає видалення знаків пунктуації та оновлення текстових полів для подальшого використання у контексті даної дослідної роботи.

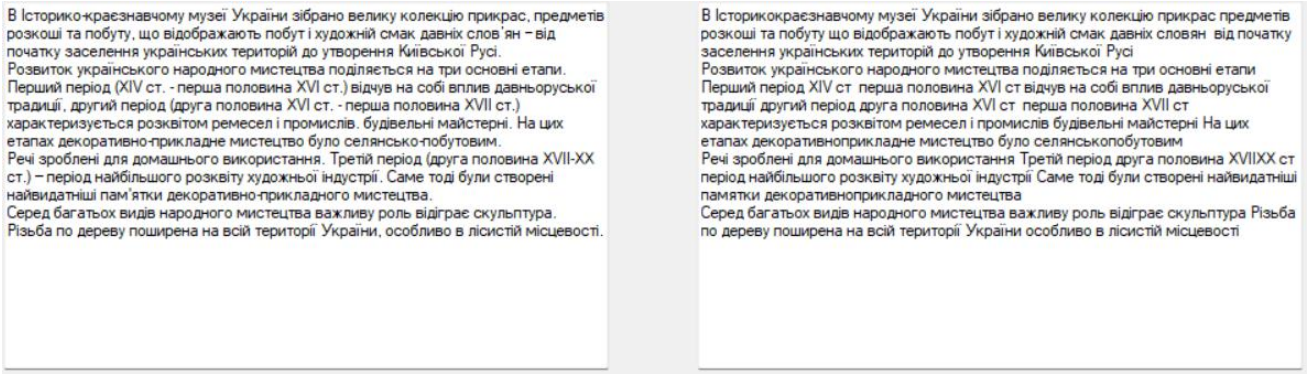


Рисунок 3.11 – Результат роботи функції видалення розділових знаків

У першому рядку коду, виконується зчитування тексту, введеного у поле textBox1, та збереження його у змінні text1.

Далі, застосовуються методи обробки тексту, які дозволяють видалити знаки пунктуації з введених рядків. За допомогою методу Where та умовного

оператора, відбираються лише символи, які не є знаками пунктуації, з рядка text1. Після фільтрації, отримані символи перетворюються у масив, а потім об'єднуються знову у рядок noPunctuationText1.

Останні два рядки коду відповідають за оновлення текстових полів. Значення поля textBox1.Text оновлюється з обробленим текстом noPunctuationText1, тим самим відображаючи текст без знаків пунктуації.

Наведений рисунок 3.12 відображає процес обробки введеного тексту, який включає видалення стоп-слів та оновлення текстового поля.

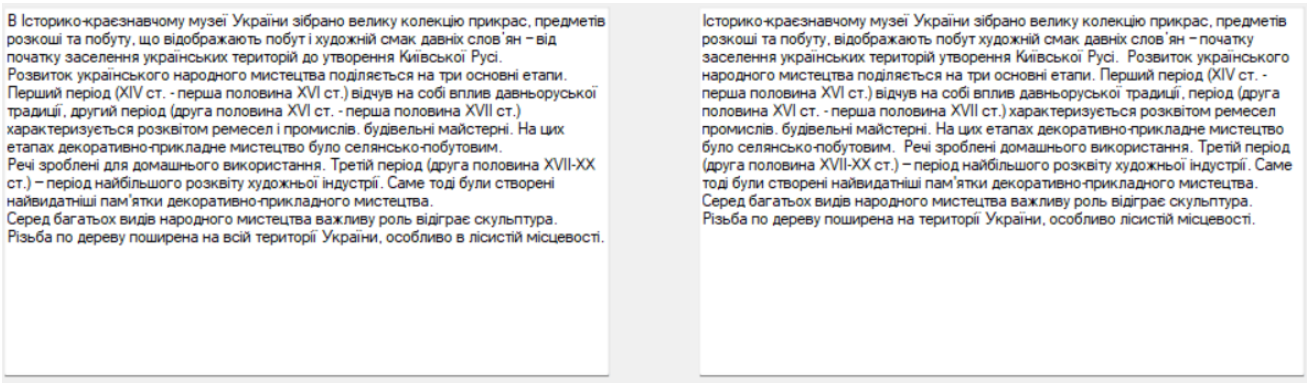


Рисунок 3.12 – Метод видалення стоп слів

У першому рядку коду, здійснюється зчитування тексту, введеного у поле textBox, та збереження його у змінну text1.

Далі, використовується файл "stopwords.txt", який містить список стоп-слів. За допомогою методу ReadAllLines, вміст цього файлу зчитується із файлової системи та зберігається у змінну stopWords.

Потім, застосовується обробка тексту з використанням методів розбиття рядка на окремі слова, фільтрації стоп-слів та знову об'єднанням в рядок. У третьому рядку коду, рядок text1 розбивається на окремі слова за допомогою методу Split(). Потім, за допомогою методу Where та умовного оператора, з використанням методу Contains, фільтруються лише ті слова, які не містяться у списку stopWords (при цьому, порівнювання відбувається у нижньому регістрі). Наступно, відфільтровані слова знову об'єднуються в рядок, використовуючи метод string.Join, та зберігаються у змінну noStopWordsText.

Останній рядок коду відповідає за оновлення текстового поля. Значення поля textBox1.Text оновлюється з обробленим текстом noStopWordsText1, тим самим відображаючи текст без стоп-слів.

На рисунку 3.13 розглядається процес відкриття та читання текстового файлу за допомогою діалогового вікна відкриття файлів. Це дозволяє користувачеві вибрати текстовий файл із файлової системи та відображає його зміст у текстовому полі textBox.

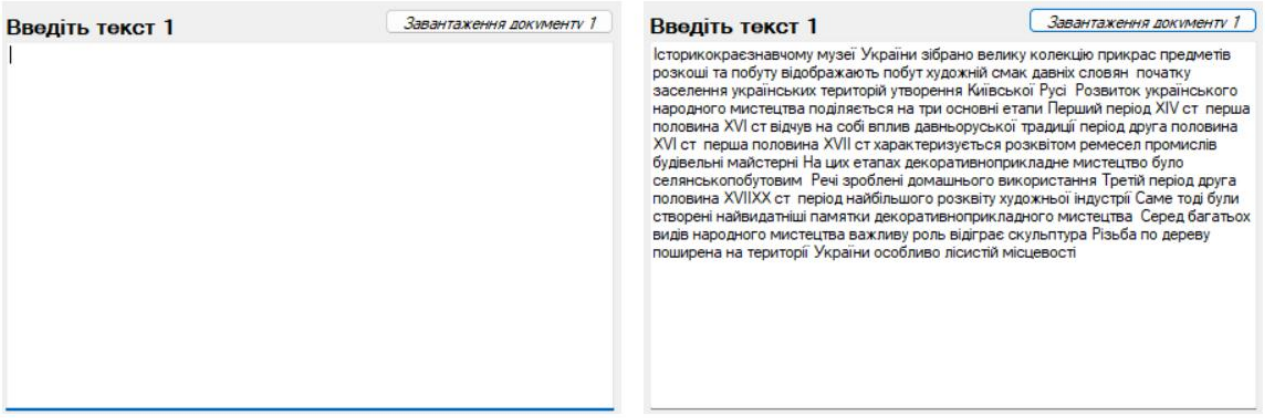


Рисунок 3.13 – Результат роботи алгоритму завантаження документу

Даний код відкриває діалогове вікно вибору файлу та дозволяє користувачеві вибрати текстовий файл.

Після чого фільтруються документи для відображення лише текстових файлів у діалоговому вікні вибору файлу. Це реалізовано за допомогою властивості Filter, де вказані шаблони файлів для відображення.

Вміст файлу зчитується за допомогою System.IO.File.ReadAllText(fileName) і встановлюється як текстове значення textBox1.Text. Таким чином, вибраний файл відображається в текстовому полі для подальшого використання або обробки.

Як висновок, особливості розробки складових інформаційної системи, зокрема аналізу текстових даних та перевірки наявності дублікатів серед збережених файлів, грають важливу роль у покращенні ефективності та точності методу семантичного аналізу текстів з використанням n-грам для задач пошуку

плагіату. В коді реалізовані процеси обробки введеного тексту, такі як видалення знаків пунктуації та стоп-слів, для підготовки тексту до аналізу. Діалогове вікно вибору файлу дозволяє користувачеві відкрити та прочитати вміст текстового файлу для подальшого використання. Крім того, ці особливості сприяють оптимізації використання ресурсів пам'яті, що є значущим аспектом розробки інформаційних систем для обробки текстових даних.

3.3 Тестування створеної інформаційної системи пошуку плагіату за семантичним аналізом текстів

Для перевірки ефективності та коректності роботи інформаційної системи для реалізації методу семантичного аналізу текстів з використанням n-грам, який призначений для перетворення вхідних даних у вигляді цифрової текстової інформації у вихідні дані вигляду відсоткового рівня плагіату на знайдений аналог, було проведено ряд юніт-тестів. В цих тестах було перевірено функціональність окремих компонентів системи, зокрема функції генерації біграм і триграм. Тести включали в себе ручне введення текстових даних, їх обробку та порівняння результатів з очікуваними значеннями.

У тест-кейсах, наведених нижче, представлена перевірка коректної роботи функції генерації двограм та триграм. За допомогою даних функцій формуються послідовності дво- та трьохелементних груп слів з вхідного масиву "tokens". Далі, за допомогою оператора "Assert.IsTrue", порівнюються отримані n-грами з очікуваними значеннями. Цей тестовий метод виконує перевірку правильності роботи функції та допомагає забезпечити коректну роботу системи у виявленні плагіату за допомогою методу n-грам.

Кроки тест-кейсу перевірки коректної роботи функції генерації дворам наведені у таблиці 3.1.

Таблиця 3.1 – Тест-кейс MCAT001

Тест-кейс ID: MCAT001	**Пріоритет: 1**	**Створено:** 14.05.2023, М. Шиманський
Назва: Коректна робота функції генерації двограм **Вхідні дані:** Речення для подальшого утворення двограм		
Кроки	**Очікуваний результат**	
1. Введення текстових даних 2. Введення очікуваного результату 3. Запуск перевірки 4. Порівняння фактичного результату з очікуваним	Результатом є збіг фактичного та очікуваного результату	
Результат виконання тест-кейсу: пройдено успішно		

Кроки тест-кейсу перевірки коректної роботи функції генерації триграм наведені у таблиці 3.2.

Таблиця 3.2 – Тест-кейс MCAT002

Тест-кейс ID: MCAT002	**Пріоритет: 1**	**Створено:** 14.05.2023, М. Шиманський
Назва: Коректна робота функції генерації триграм **Вхідні дані:** Речення для подальшого утворення триграм		
Кроки	**Очікуваний результат**	
1. Введення текстових даних 2. Введення очікуваного результату 3. Запуск перевірки 4. Порівняння фактичного результату з очікуваним	Результатом є збіг фактичного та очікуваного результату	
Результат виконання тест-кейсу: пройдено успішно		

Результати перевірки методу «GetBigrams» (рисунок 3.14).

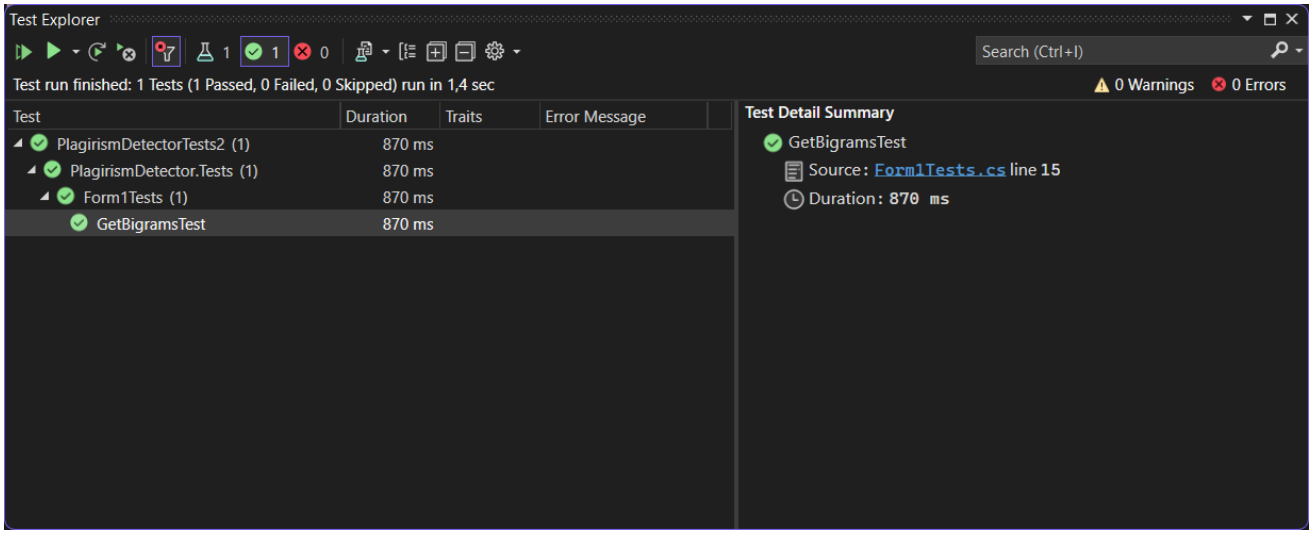


Рисунок 3.14 – Результат Unit-тесту двограм

Результати перевірки методу «GetTrigrams» (рисунок 3.15).

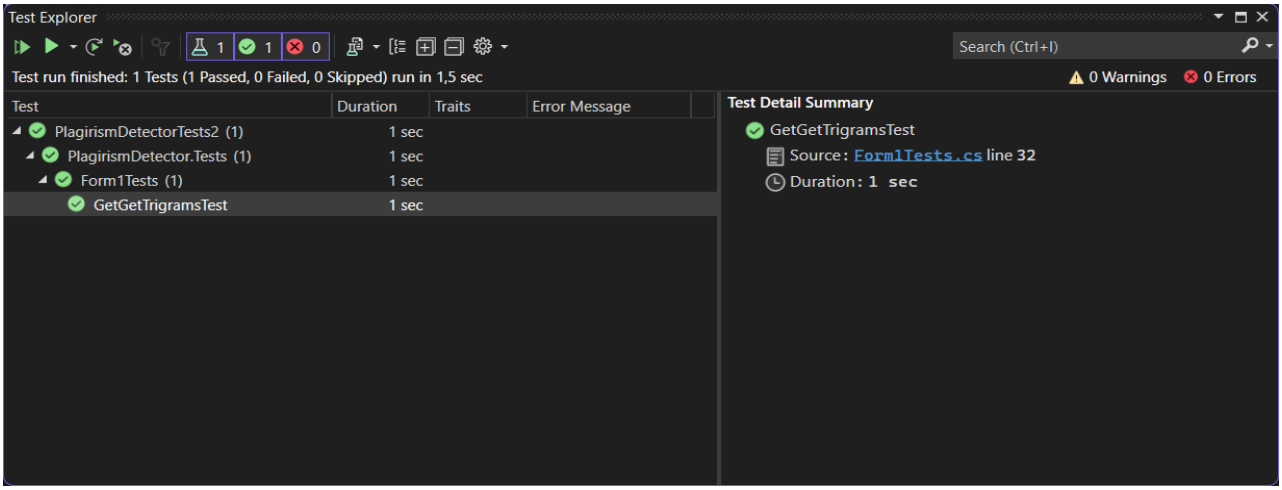


Рисунок 3.15 – Результат Unit-тесту триграм

Етапи процесу тестування функції генерування дво- та триграм.

1. Для початку створюємо екземпляр класу Form1 та ініціалізуємо масив токенів, які будуть використані для формування n-грам. Крім того, встановлюємо очікувані n-грами та зберігаємо їх у змінній під назвою “expectedBigrams”.

2. Виконуємо функцію “GetBigrams” та “GetTrigrams” категорії Form1 із раніше створеними маркерами. Отримані результати зберігаються у змінній під назвою “result”.

3. Порівнюємо отримані n-грами “result” з очікуваними n-грамами “expectedBigrams” за допомогою методу “SequenceEqual”. Якщо всі n-грами збігаються, тест вважається успішним.

У результаті проведених юніт-тестів було підтверджено правильну роботу функцій генерації біграм і триграм. Отримані результати відповідали очікуваним значенням, що свідчить про коректну інтеграцію цих функцій у загальну систему. Таке тестування дозволило впевнитись в надійності та стабільності роботи інформаційної системи, що дозволить їй ефективно використовуватись для виявлення плагіату та забезпечення оригінальності текстових документів.

3.4 Інструкція користувача до реалізованої інформаційної системи

Підчас запуску інформаційної системи відкриється головне меню, де є шість кнопок, а саме: «Порівняння текстів», «Перевірка текстів», «Редактор текстів», «Пошук N-грам», «Налаштування» та «Вихід». При натисканні на будь-яку копку поточна форма закривається (рисунок 3.16).

Користувач може зразу перевірити тексти, натиснувши кнопку «Перевірити», але для точнішого результату варто видалити стоп слова, а також розділові знаки, за допомогою кнопок «Забрати розділові знаки» та «Видалити стоп-слова» (рисунок 3.17).

Натиснувши на кнопку «Порівняння текстів» відкриється форма де користувач може скопіювати або завантажити два тексти та перевірити наявність співпадінь між ними (рисунок 3.18).

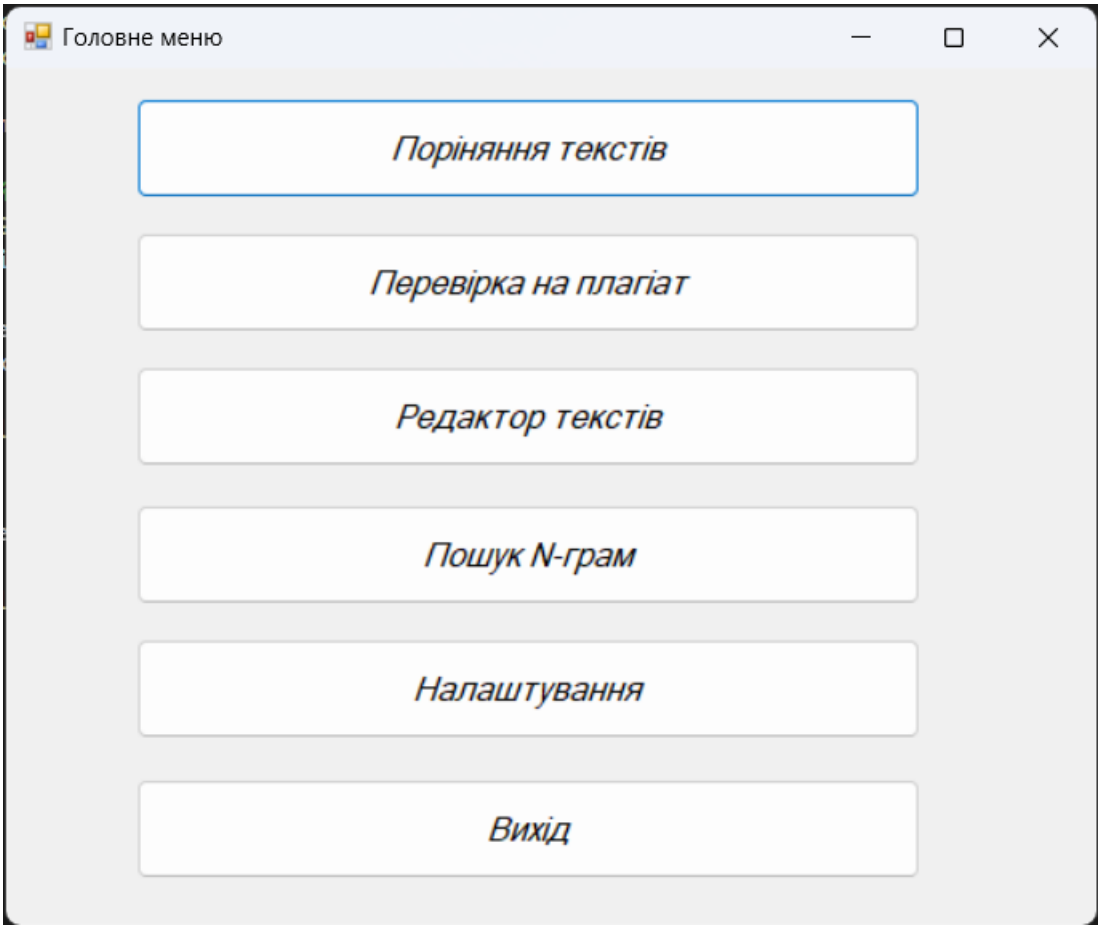


Рисунок 3.16 – Головне меню інформаційної системи

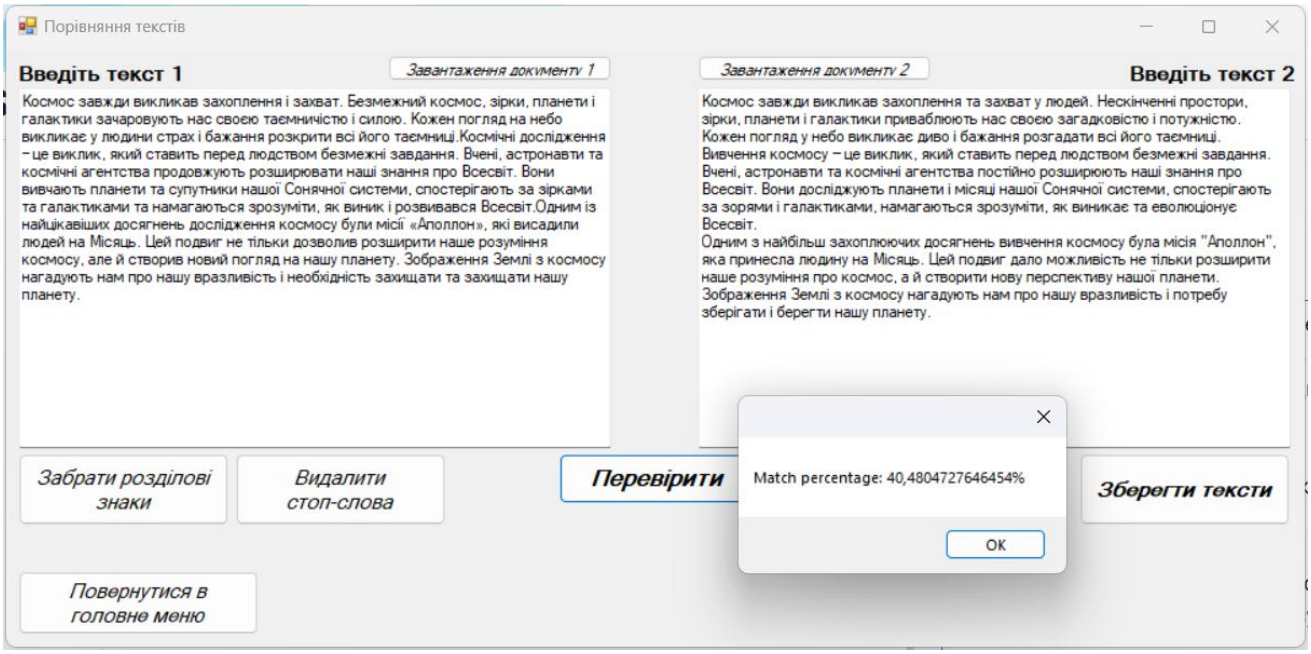


Рисунок 3.17 – Порівняння двох текстів на сумісність збігів

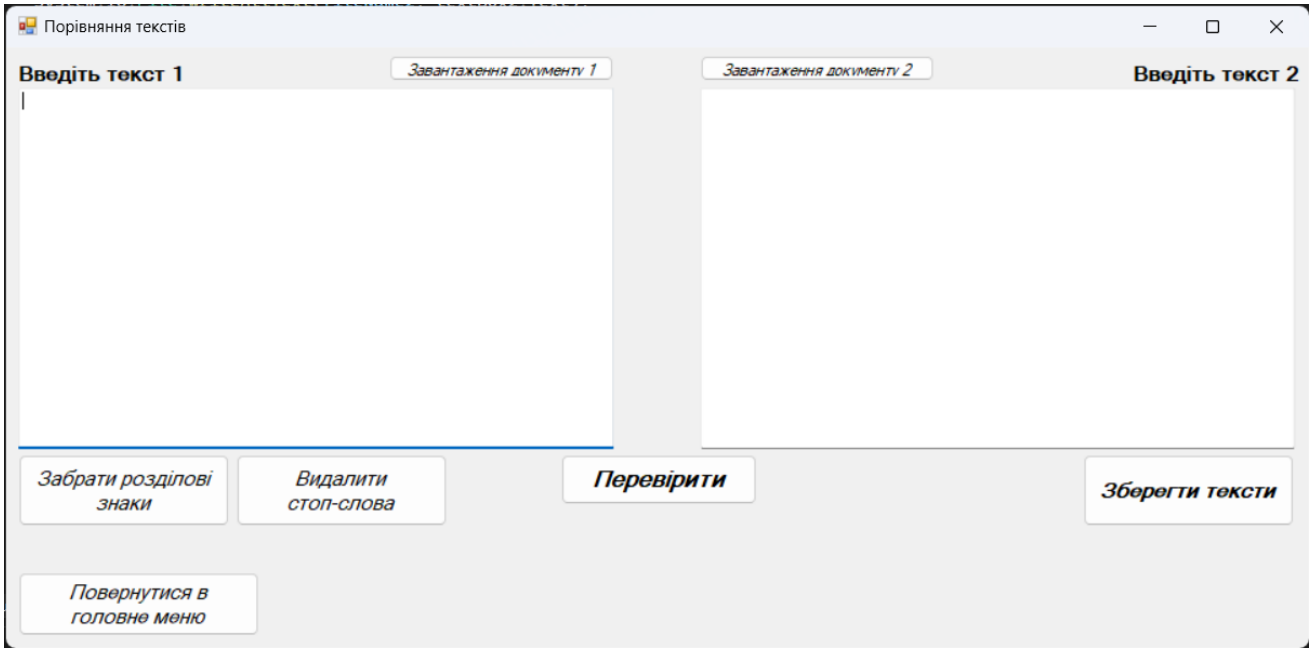


Рисунок 3.18 – Форма «Порівняння текстів»

За бажанням користувач має можливість зберегти обидва тексти у БД натиснувши на кнопку «Зберегти тексти».

Якщо користувач хоче скористатися іншими функціями програми, він може перейти у головне меню натиснувши відповідну кнопку.

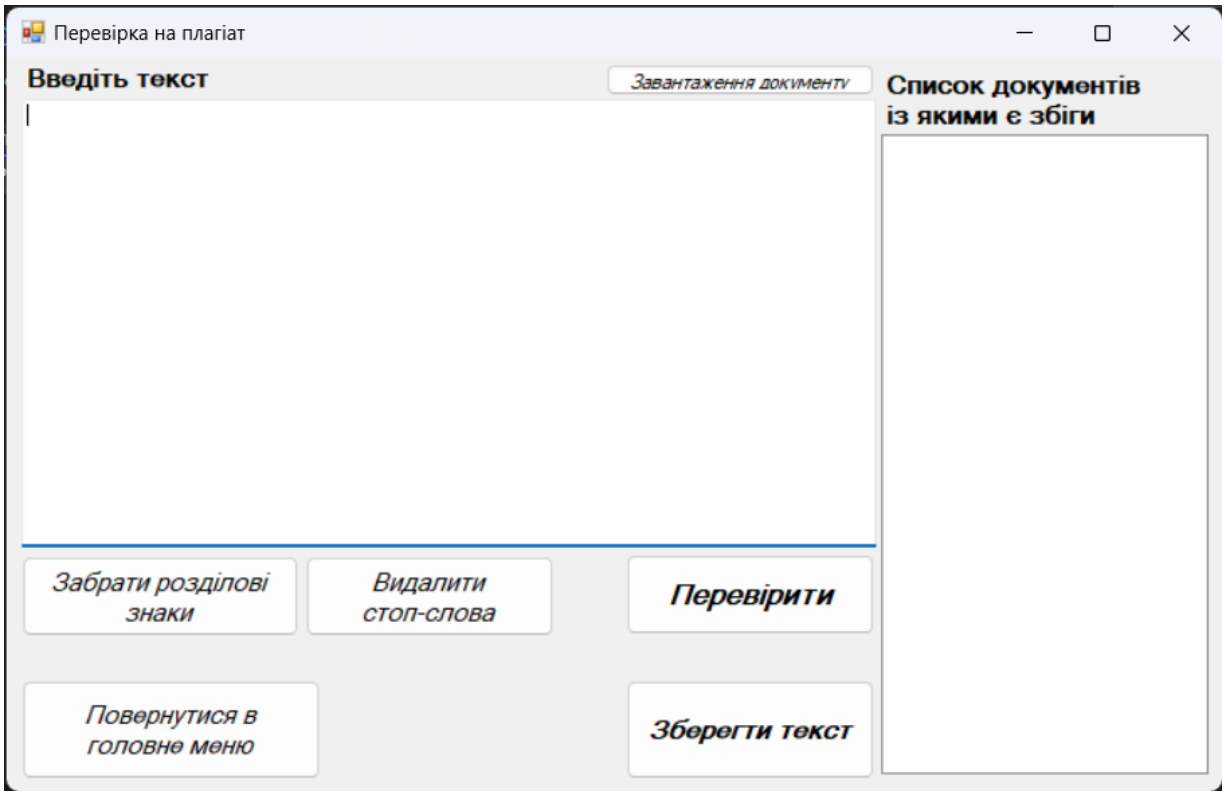


Рисунок 3.19 – Форма «Перевірка на плагіат»

Якщо користувач хоче порівняти текст з існуючою БД текстів, він може перейти на форму «Перевірка на плагіат» (рисунок 3.19).

Аналогічно як і формі «Порівняння текстів», у даній формі можна забирати розділові знаки та стоп-слова, для кращої ефективності перевірки тексту, та зберігати за потреби текст у БД (рисунок 3.20). При великій кількості збігів із конкретним документом, що збережений у БД, його ім’я буде виведено в окремий список (рисунок 3.21).

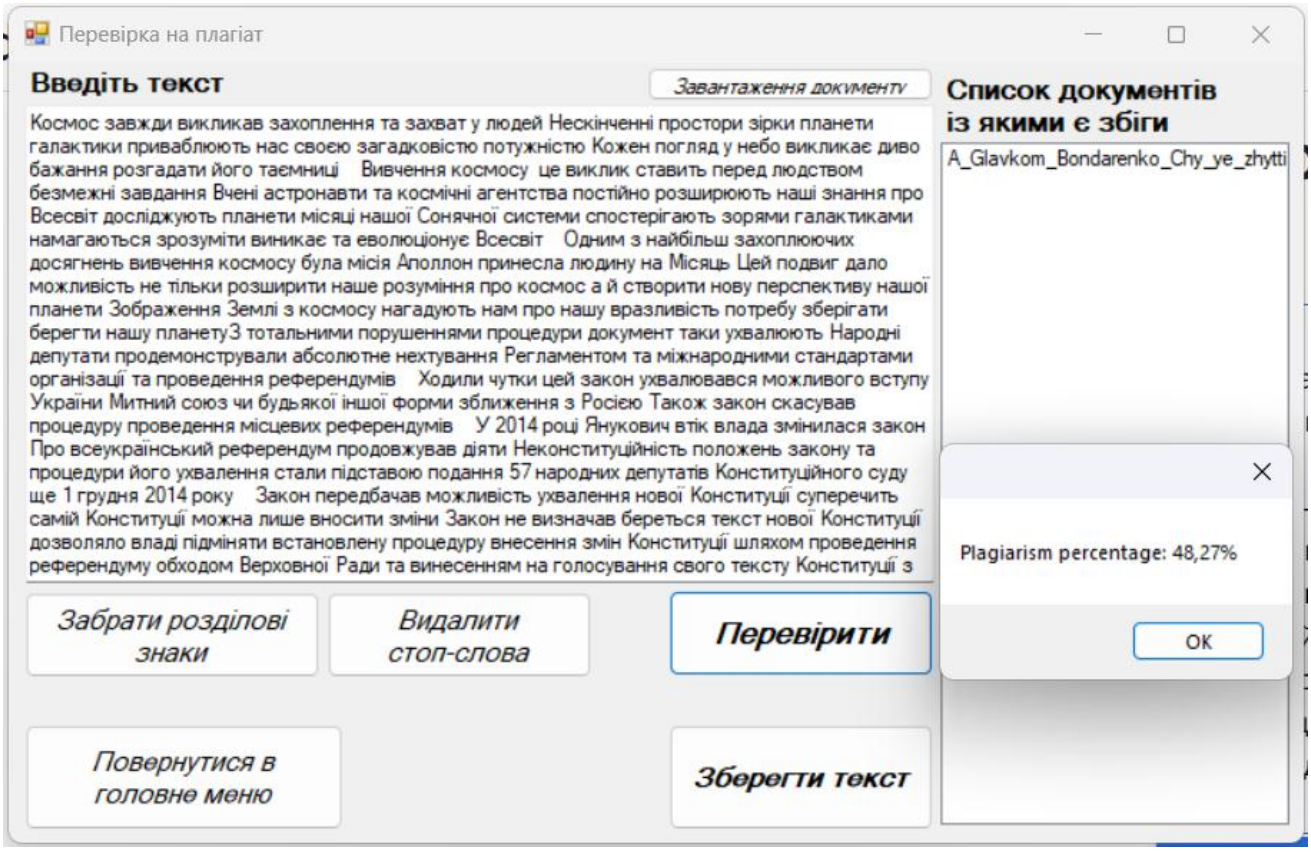


Рисунок 3.20 – Перевірка тексту на плагіат

Якщо користувач, зберіг текст, але допустив помилку або має бажання відредагувати збережений текст, то він може скористатися формою «Редактор текстів» (рисунок 3.21), вибрати потрібний документ та відредагувати його.

Серед запропонованих можливостей, користувачеві дозволяється зберігати, видаляти та редагувати тексти.

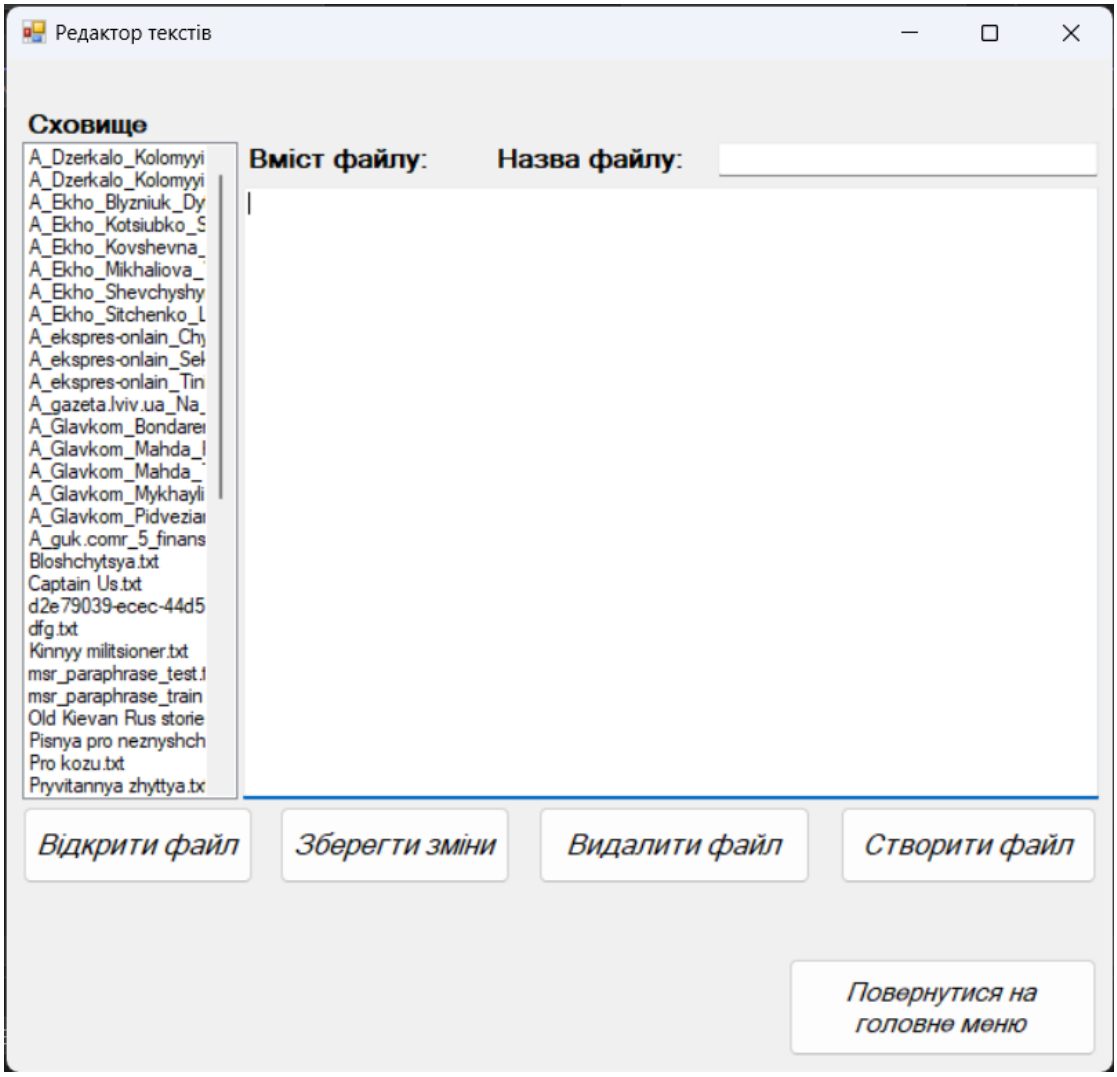


Рисунок 3.21 – Форма «Редактор текстів»

Користувач може зберігати тексти, введені в текстове поле, за допомогою кнопки "Зберегти зміни", а також створювати нові файли за допомогою кнопки "Створити файл", файли зберігаються в «Сховище». Кнопка "Видалити файл" видаляє файл з БД. Якщо ж користувач помилково видалив файл, то його можна відновити, так як текстове поле після цього не оновлюється.

Кнопка "Відкрити файл" дозволяє користувачеві завантажувати тексти з уже існуючих файлів.

Також користувач може дізнатися кількість дво- та триграм у тексті, для цього йому потрібно перейти на форму «Пошук N-грам» (рисунок 3.22).

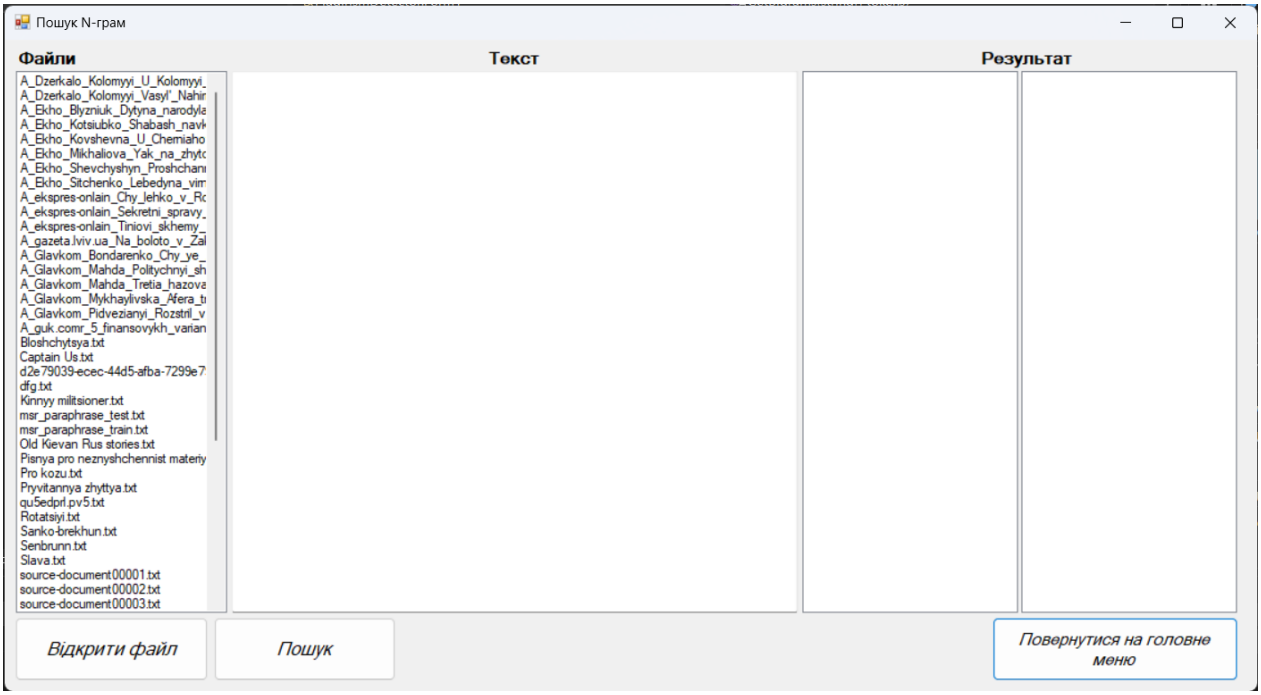


Рисунок 3.22 – Форма «Пошук N-грам»

У даній формі файли завантажуються з БД, вибір файлу відбувається за допомогою списку. Кнопка "Відкрити файл" виводить файл в текстовому полі та зчитує список токенів, отриманих за допомогою бібліотеки OpenNLP. Кнопка "Пошук" розраховує кількість дво- та триграм виводячи їх у відповідні списки (рисунок 3.23).

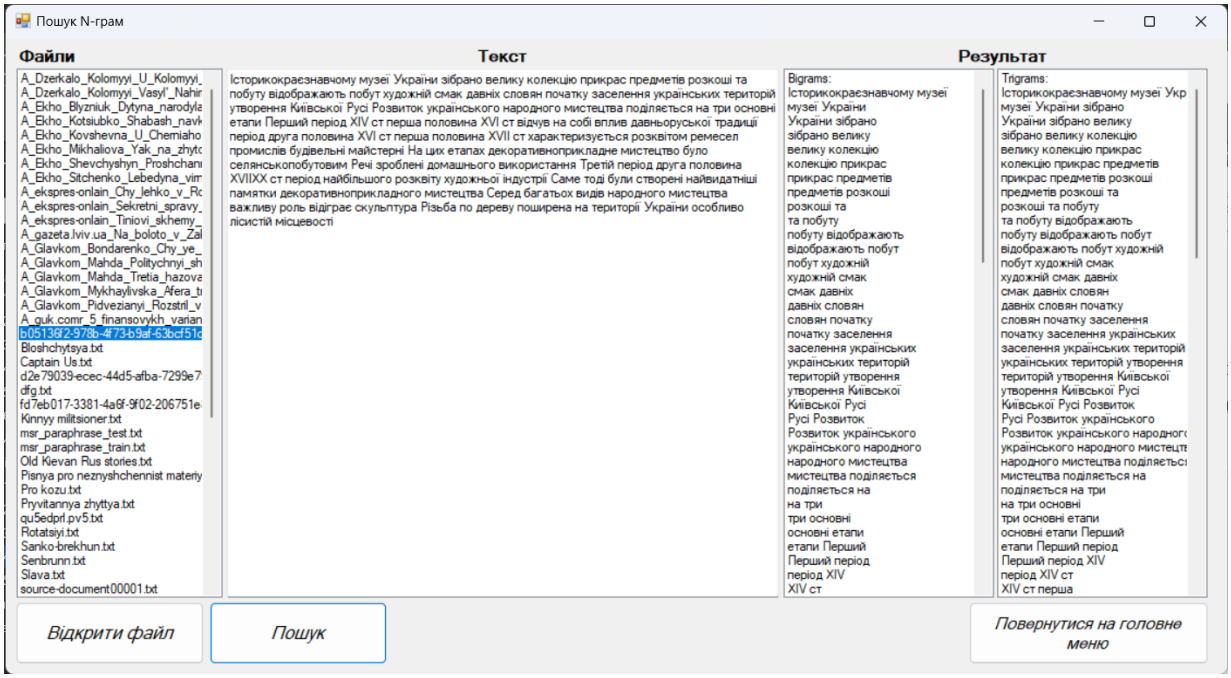


Рисунок 3.23 – Вивід списків дво- та триграм залежно від вибраного тексту

Якщо ж користувач хоче перевірити текст конкретною мовою та способом він може скористатися формою «Налаштування» (рисунок 3.24).

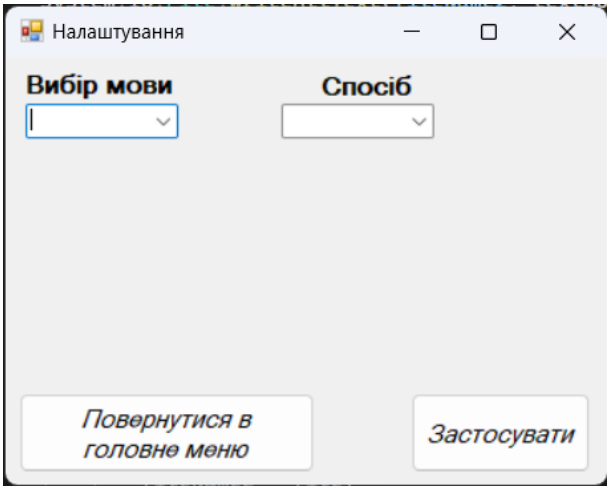


Рисунок 3.24 – Форма «Налаштування»

У даній формі можна вибрати якою мовою буде виконана перевірка, а саме: англійська та українська. Також користувач може обрати спосіб за допомогою якого буде здійснена перевірка, а саме: двограми, триграми та комбінований варіант.

Отже, було розроблено інформаційну систему, що використовує методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату та реалізовано: перевірку текстів між собою, перевірка тексту з БД, редагування збережених текстів, пошук n-грам та можливість обирати конкретні засоби перевірки.

3.5 Дослідження ефективності методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату

Для дослідження ефективності методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату було проаналізовано два тексти трьома способами.

Перший спосіб включає пошук збігів між текстами за допомогою двограм, другий спосіб включає пошук збігів за допомогою триграм, останній спосіб є комбінацією двох методів.

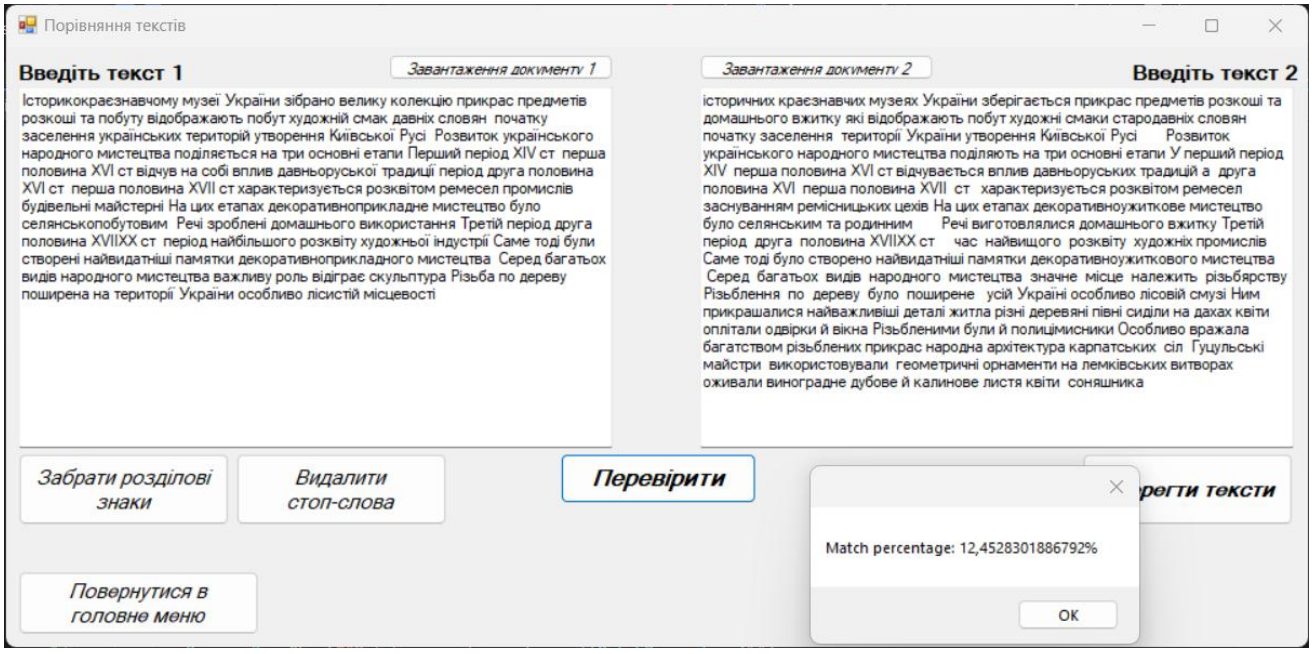


Рисунок 3.25 – Результат пошуку збігів за допомогою методу утворення двограм

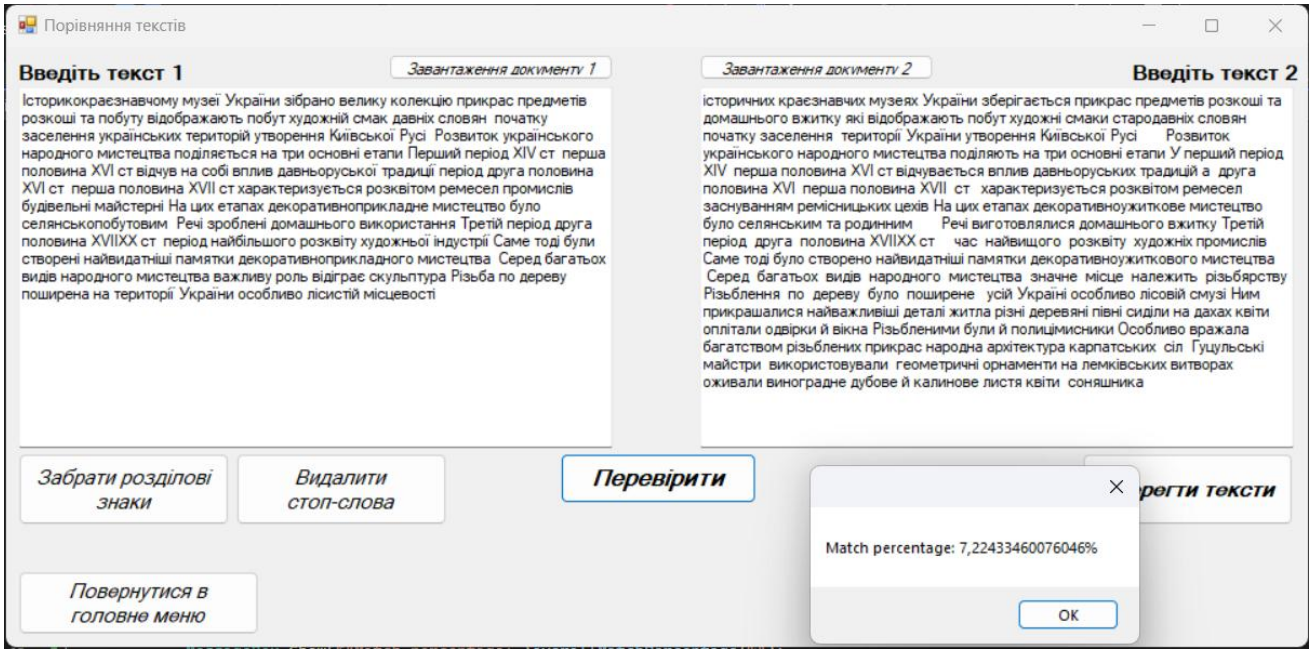


Рисунок 3.26 – Результат пошуку збігів за допомогою методу утворення триграм

Результати аналізу показали, що метод двограм мав результат 12.45% знаходження збігів між текстами (рисунок 3.25). Метод триграм показав менший

результат, становлячи 7.22% (рисунок 3.26). Натомість, комбінований метод продемонстрував найкращі результати зі значенням 19.67% знаходження збігів (рисунок 3.27).

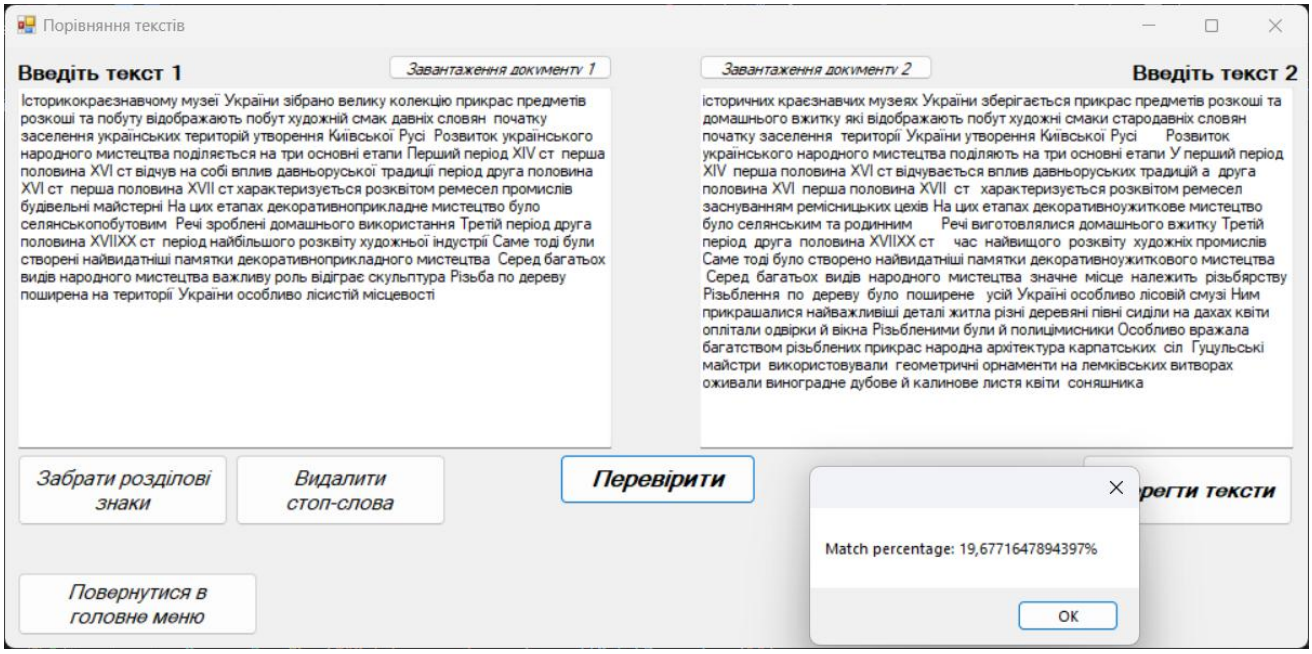


Рисунок 3.27 – Результат пошуку збігів за допомогою комбінованого методу

Для кращої візуалізації результатів порівняння ефективності методів семантичного аналізу текстів з використанням n-грам для пошуку плагіату була створена діаграма (рисунок 3.28). Ця діаграма дозволяє швидко та зрозуміло зробити висновок про те, який метод є найбільш ефективним для виявлення збігів між текстами у контексті пошуку плагіату.

Таким чином, порівнюючи ефективність трьох методів, можна зробити висновок, що комбінований метод є найкращим для виявлення збігів між текстами у контексті пошуку плагіату. Він показав найвищий результат у порівнянні з методами двограм та триграм. Його поєднання двох методів сприяє більш точному та комплексному аналізу текстів, що покращує можливості виявлення плагіату.

Для додаткового порівняння ефективності був використаний ще один метод – існуючий, який використовує крім дво- та триграм, чотири- та п'ятиграми. Результати цього методу показали значення 25%. Для кращої

візуалізації цього порівняння була створена діаграма, яка дозволяє зрозуміти відмінності між існуючим методом та комбінованим методом.

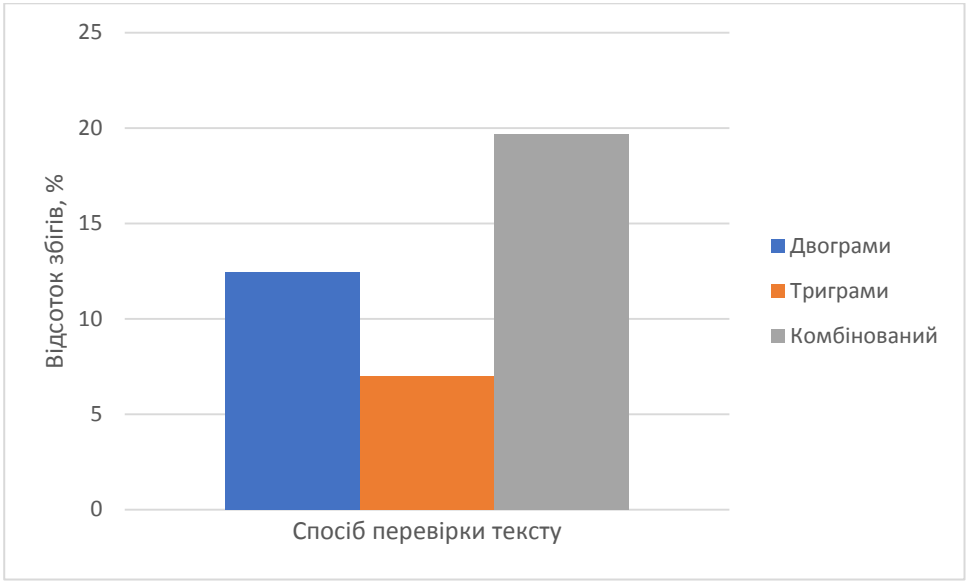


Рисунок 3.28 – Діаграма, що порівнює результати перевірки трьох методів

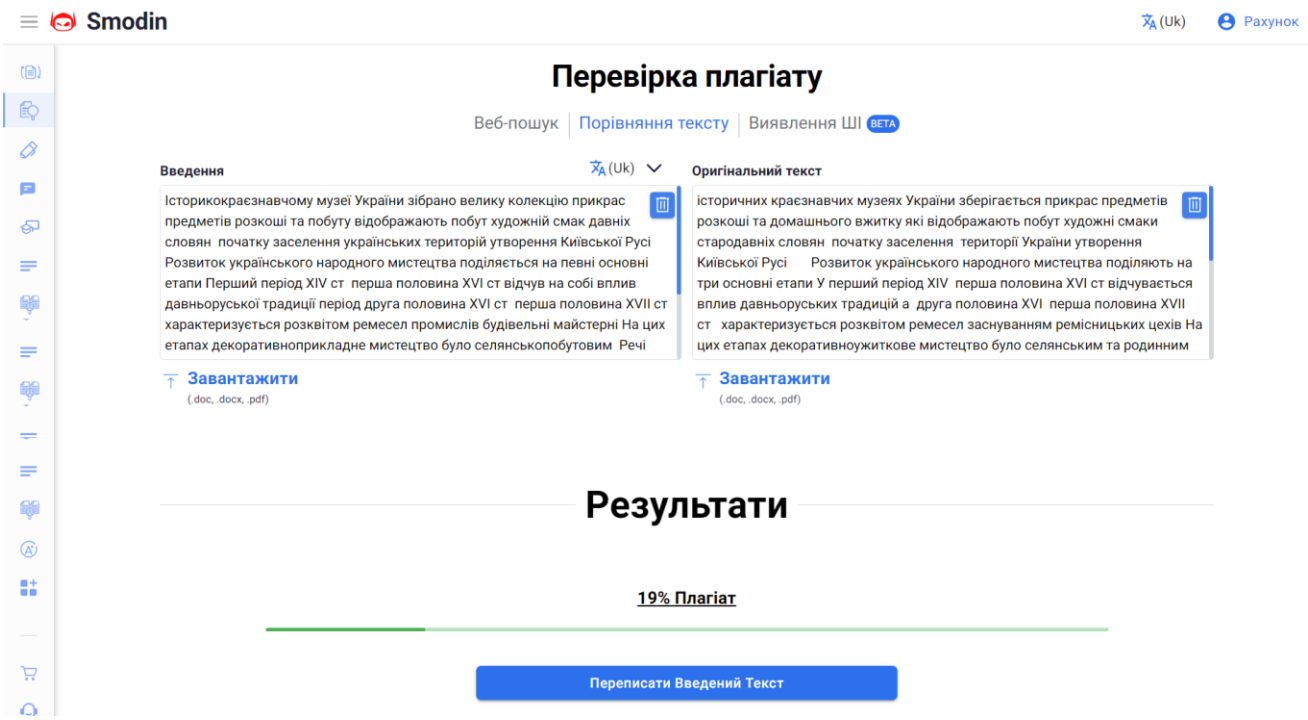


Рисунок 3.29 – Результат пошуку збігів за допомогою існуючого методу

Порівнюючи результати існуючого методу з використанням чотири- та п'ятиграм та комбінованого методу з дво- та триграмами, виявляється, що між ними існує невелика різниця. Існуючий метод, який використовує п'ятиграми,

продемонстрував кращий результат з показником 25% (рисунок 3.29), тоді як комбінований метод досяг результату 19.67% (рисунок 3.27).

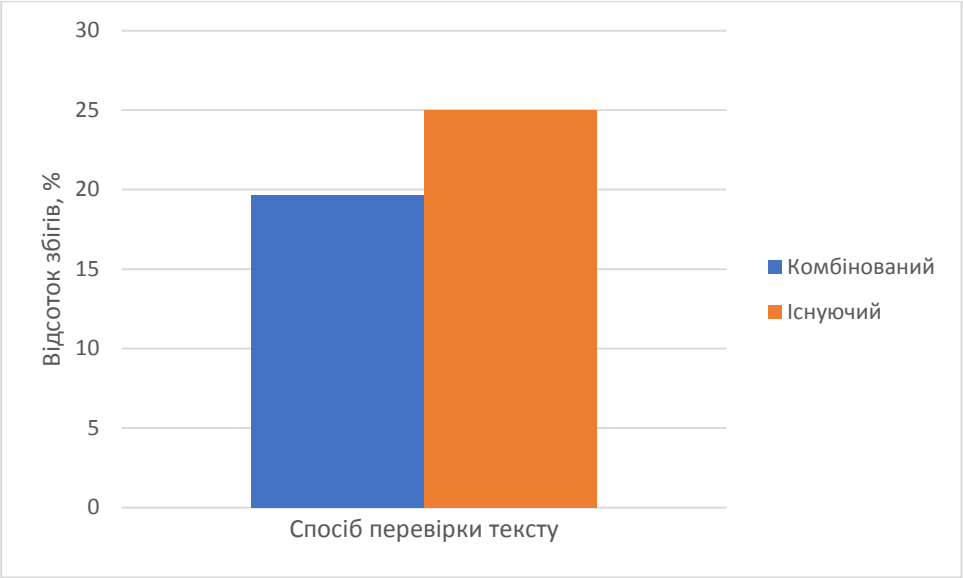


Рисунок 3.30 – Діаграма, що порівнює існуючий та комбіновані методи

Це свідчить про те, що існуючий метод може вважатися дещо довершеним з точки зору виявлення збігів між текстами. Однак, різниця у показниках є незначною, і комбінований метод з дво- та триграмами також демонструє добрі результати.

При виборі між цими двома методами важливо враховувати особливості конкретної задачі, а також ресурси та обмеження, з якими можуть зіштовхнутися інформаційні системи. Детальний аналіз і порівняння різних аспектів обох методів можуть допомогти зробити обґрунтований вибір залежно від потреб і вимог конкретної ситуації.

На підставі проведеного порівняльного аналізу між методами, такими як двограм, триграм, комбінований та існуючий, можна зробити наступні висновки.

Комбінований метод, що поєднує дво- та триграми, показав найкращий результат з показником 19.67%. Цей підхід поєднує переваги обох методів, забезпечуючи ширший спектр аналізу текстів та більшу точність у виявленні збігів та плагіату.

Щодо існуючого методу, який використовує п'ятиграми та показав результат 25%, він демонструє дещо вищу ефективність порівняно з комбінованим методом. Цей метод може бути вважаний більш довершеним у виявленні збігів між текстами.

Загалом, вибір оптимального методу залежить від конкретної задачі, обмежень ресурсів та вимог щодо точності. Комбінований метод з дво- та триграмами може бути привабливим компромісом, забезпечуючи хорошу ефективність і прийнятну точність при обробці текстових даних та виявленні плагіату.

Висновки до розділу 3

У третьому розділі було реалізовано інформаційну систему пошуку плагіату за семантичним аналізом текстів. Розділ включає розгляд різних підрозділів, які детально описують структуру модулів системи, особливості розробки складових, тестування функцій генерації дво- та триграм, інструкцію користувача та дослідження ефективності методу семантичного аналізу текстів з використанням n-грам для пошуку плагіату.

Структура модулів інформаційної системи була описана з використанням діаграми класів, що дозволяє зрозуміти взаємозв'язки між різними компонентами системи. Особливості розробки складових включали розробку функцій генерації дво- та триграм, збереження та редагування текстів, видалення стоп-слів та знаків пунктуації. Ці складові відіграють ключову роль у процесі пошуку плагіату та забезпечують високу точність і швидкість обробки текстів.

Протестована інформаційна система показала задовільні результати. Функції генерації дво- та триграм були протестовані, і їх коректність та ефективність були підтверджені.

Інструкція користувача, яка була розроблена, надає чіткі та докладні кроки для користування інформаційною системою. Це дозволяє користувачам з

легкістю використовувати систему та проводити пошук плагіату в їхніх текстових документах.

Дослідження ефективності методу семантичного аналізу текстів з використанням n-грам також було проведено відповідно до завдання. Порівнювалися різні методи, включаючи метод двограм, триграм, комбінований метод і комбінований метод з існуючим. Результати дослідження показали, що комбінований метод з використанням n-грам має достатню ефективність у виявленні плагіату.

Отже, на основі проведених досліджень та успішної реалізації інформаційної системи, можна зробити висновок, що розроблений метод семантичного аналізу текстів з використанням n-грам є ефективним і потенційно корисним інструментом для пошуку плагіату. Розроблена ж система може бути використана в освітніх, наукових та інших сферах, де є важливим виявлення оригінальності текстів та боротьба з плагіатом.

Висновки

Відповідно до поставленої мети кваліфікаційної роботи бакалавра, було розроблено метод семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату та відповідну інформаційну систему пошуку плагіату за семантичним аналізом текстів, що дозволив за текстовим контентом визначати числову оцінку встановленого обсягу текстових запозичень шляхом його семантичного аналізу з використанням n-грам. Метод семантичного аналізу текстів з використанням n-грам перетворює вхідні дані у вигляді цифрової текстової інформації у вихідні дані вигляду відсоткового рівня плагіату на знайдений аналог.

Процес розробки включав детальний аналіз існуючих методів виявлення плагіату, розробку методу та його практичну реалізацію.

В результаті проведеного дослідження було встановлено, що запропонований метод семантичного аналізу текстів з використанням n-грам є ефективним засобом виявлення плагіату. Його основною перевагою є здатність виявляти плагіат навіть у випадках, коли текстові фрагменти були значно змінені або перекладені.

Розроблена інформаційна система демонструє надійну та ефективну роботу в процесі виявлення плагіату. Вона забезпечує імпорт та індексування текстових даних, а також виконання пошуку та порівняння текстів. Впровадження даної системи дозволить виявляти плагіат та забезпечувати інтелектуальну безпеку в різних сферах, включаючи наукові дослідження, літературну творчість та академічну сферу.

Було проведено порівняння розробленого методу з існуючими методами виявлення плагіату. Отримані результати підтвердили високу ефективність та точність запропонованого методу. Виявлено, що він здатний ефективно працювати з різними типами текстів та забезпечувати надійне виявлення плагіату.

Загалом, розроблений метод семантичного аналізу текстів з використанням n-грам є ефективним підходом у напрямку виявлення плагіату. Отримані результати дослідження підтверджують високу потенційну корисність і практичну застосовність цього методу в різних сферах. Запропонований метод семантичного аналізу текстів з використанням n-грам може знайти своє застосування у різних задачах, де виявлення плагіату є важливим завданням.

Незважаючи на досягнуті успіхи, є деякі аспекти, які можуть бути вдосконалені. Подальші дослідження можуть бути спрямовані на покращення точності методу, розширення функціональності системи та врахування особливостей конкретних типів текстів. Також можна провести додаткові експерименти з використанням великого обсягу даних для підтвердження стабільності та ефективності методу у реальних умовах.

Перелік посилань

1. Hindawi. Survey of Natural Language Processing Techniques in Bioinformatics. URL: https://www.hindawi.com/journals/cmmm/2015/674296/

2. SHRDLU. URL: http://hci.stanford.edu/winograd/shrdlu/

3. Monkeylearn Blog. Semantic Analysis, Explained. URL: https://monkeylearn.com/blog/semantic-analysis/

4. Semantic Role Labeling. URL: https://web.stanford.edu/~jurafsky/slp3/24.pdf

5. Tutorialspoint. Compiler Design – Semantic Analysis. URL: https://www.tutorialspoint.com/compiler_design/compiler_design_semantic_analysis.htm

6. Grammarly. 7 Common Types of Plagiarism, With Examples. URL: https://www.grammarly.com/blog/types-of-plagiarism/

7. Plagiarism: Taxonomy, Tools and Detection Techniques. URL: https://arxiv.org/ftp/arxiv/papers/1801/1801.06323.pdf

8. Hubspot. How to Do Keyword Research for SEO: A Beginner's Guide. URL: https://blog.hubspot.com/marketing/how-to-do-keyword-research-ht

9. Textrank for summarizing text. URL: https://cran.r-project.org/web/packages/textrank/vignettes/textrank.html

10. Emerald insight. A STATISTICAL INTERPRETATION OF TERM SPECIFICITY AND ITS APPLICATION IN RETRIEVAL. URL: https://www.emerald.com/insight/content/doi/10.1108/eb026526/full/html

11. Yet Another Keyword Extractor. URL: https://github.com/LIAAD/yake

12. Goldsmiths. Keywords and Advanced Search Techniques. URL: https://libguides.gold.ac.uk/aso/library/advancedsearch

13. Analytics Vidhya. «What are n-grams and How to Implement Them in Python?». URL: https://www.analyticsvidhya.com/blog/2021/09/what-are-n-grams-and-how-to-implement-them-in-python/

14. Devopedia. N-Gram Model. URL: https://devopedia.org/n-gram-model

15. Dupli Checker. URL: https://www.duplichecker.com/

16. Plagiarisma. URL: http://plagiarisma.net/

17. Scribbr. 12 Best Free Plagiarism Checkers in 2022 | Tested & Reviewed. URL: https://www.scribbr.com/plagiarism/best-free-plagiarism-checker/

18. Unicheck. URL: https://support.unicheck.com/hc/en-us/articles/360015995794-New-Unicheck-Report-Design-and-Functionality

19. eLearning industry. Top 10 Free Plagiarism Detection Tools For eLearning Professionals. URL: https://elearningindustry.com/top-10-free-plagiarism-detection-tools-for-teachers

20. Можливості та труднощі використання обробки природної мови. URL: http://lviv-forum.inf.ua/save/2020/20-21.12.2020/%D1%87%D0%B0%D1%81%D1%82%D0%B8%D0%BD%D0%B0%201.pdf#page=74

21. 2022 International Conference on Innovative Solutions in Software Engineering. Програмні аспекти виявлення академічного плагіату. URL: https://www.researchgate.net/profile/Mykola-Kozlenko/publication/366841043_2022_International_Conference_on_Innovative_Solutions_in_Software_Engineering_ICISSE/links/63b4bd36c3c99660ebc8a6a7/2022-International-Conference-on-Innovative-Solutions-in-Software-Engineering-ICISSE.pdf#page=278

22. Розробка сервера пошукового ядра Антиплагіат. URL: https://elartu.tntu.edu.ua/handle/lib/33596

23. Wikipedia. N-gram. URL: https://en.wikipedia.org/wiki/N-gram

24. Wikipedia. Ланцюг Маркова. URL: https://uk.wikipedia.org/wiki/Ланцюг_Маркова

25. Sciencedirect. Algorithmically generated malicious domain names detection based on n-grams features. URL: https://www.sciencedirect.com/science/article/abs/pii/S0957417420311957

26. UberText 2.0. URL: https://lang.org.ua/en/ubertext/

27. Браунський корпус української мови. URL: https://github.com/brown-uk/corpus

28. Microsoft Research Paraphrase Corpus. URL: https://www.microsoft.com/en-us/download/details.aspx?id=52398

29. PAN Plagiarism Corpus 2010 (PAN-PC-10). URL: https://zenodo.org/record/3250123#.ZFgEx3

30. Apache OpenNLP Developer Documentation. URL: https://opennlp.apache.org/docs/1.9.3/manual/opennlp.html#intro.description

31. Class TokenizerME. URL: https://opennlp.apache.org/docs/1.9.3/apidocs/opennlp-tools/opennlp/tools/tokenize/TokenizerME.html

ДОДАТКИ

Додаток А

Програмні коди

Form1.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Windows.Forms;
using java.io;
using opennlp.tools.tokenize;

namespace PlagirismDetector
{
    public partial class ComparableTextForm : Form
    {

        private readonly TokenizerME tokenizer;

        public ComparableTextForm()
        {
            InitializeComponent();
            var text = textBox1.Text;
            using (var modelIn = new FileStream("en-tokens.bin", FileMode.Open))
            {
                var modelStream = new MemoryStream();
                modelIn.CopyTo(modelStream);
                var modelBytes = modelStream.ToArray();
                var inputStream = new ByteArrayInputStream(modelBytes);
                var model = new TokenizerModel(inputStream);
                tokenizer = new TokenizerME(model);
            }

        }

        private void button1_Click_1(object sender, EventArgs e)
        {

            var text1 = textBox1.Text;
            var text2 = textBox2.Text;

            var tokens1 = tokenizer.tokenize(text1);
            var tokens2 = tokenizer.tokenize(text2);

            var bigrams1 = GetBigrams(tokens1);
            var bigrams2 = GetBigrams(tokens2);
            var trigrams1 = GetTrigrams(tokens1);
            var trigrams2 = GetTrigrams(tokens2);

            var bigramMatchCount = bigrams1.Intersect(bigrams2).Count();
            var totalBigrams = bigrams1.Count() + bigrams2.Count();
            var bigramMatchPercentage = totalBigrams > 0 ? bigramMatchCount * 100.0 /
totalBigrams : 0;

            var trigramMatchCount = trigrams1.Intersect(trigrams2).Count();
            var totalTrigrams = trigrams1.Count() + trigrams2.Count();
            var trigramMatchPercentage = totalTrigrams > 0 ? trigramMatchCount * 100.0 /
totalTrigrams : 0;
```

```
            var overallMatchPercentage = bigramMatchPercentage + trigramMatchPercentage;
            MessageBox.Show($"Match percentage: {overallMatchPercentage}%");
        }

        // Функція для генерації дво-грам
        private IEnumerable<string> GetBigrams(string[] tokens)
        {
            for (var i = 0; i < tokens.Length - 1; i++)
            {
                yield return $"{tokens[i]} {tokens[i + 1]}";
            }
        }
        // Функція для генерації три-грам
        private IEnumerable<string> GetTrigrams(string[] tokens)
        {
            for (var i = 0; i < tokens.Length - 2; i++)
            {
                yield return $"{tokens[i]} {tokens[i + 1]} {tokens[i + 2]}";
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            var text1 = textBox1.Text;
            var text2 = textBox2.Text;

            var noPunctuationText1 = new string(text1.Where(c =>
!char.IsPunctuation(c)).ToArray());
            var noPunctuationText2 = new string(text2.Where(c =>
!char.IsPunctuation(c)).ToArray());

            textBox1.Text = noPunctuationText1;
            textBox2.Text = noPunctuationText2;
        }

        private void button3_Click(object sender, EventArgs e)
        {

            var text1 = textBox1.Text;
            var text2 = textBox2.Text;

            var stopWords = System.IO.File.ReadAllLines("stopwords.txt");
            var noStopWordsText1 = string.Join(" ", text1.Split().Where(w =>
!stopWords.Contains(w.ToLower())));
            var noStopWordsText2 = string.Join(" ", text2.Split().Where(w =>
!stopWords.Contains(w.ToLower())));

            textBox1.Text = noStopWordsText1;
            textBox2.Text = noStopWordsText2;
        }

        private void button4_Click(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*";

            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
                string fileName = openFileDialog.FileName;
                textBox1.Text = System.IO.File.ReadAllText(fileName);
            }
        }
```

```
        private void button5_Click(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*";

            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
                string fileName = openFileDialog.FileName;
                textBox2.Text = System.IO.File.ReadAllText(fileName);
            }
        }

        private void button6_Click(object sender, EventArgs e)
        {
            MainMenu form2 = new MainMenu();
            form2.FormClosed += (s, args) => this.Close();
            form2.Show();
            this.Hide();

        }

        private void button7_Click(object sender, EventArgs e)
        {
            string folderPath = @"V:\repos\TextsForPlag";
            string fileName1 = Path.Combine(folderPath, Guid.NewGuid().ToString() +
".txt");
            string fileName2 = Path.Combine(folderPath, Guid.NewGuid().ToString() +
".txt");

            System.IO.File.WriteAllText(fileName1, textBox1.Text);
            System.IO.File.WriteAllText(fileName2, textBox2.Text);

            MessageBox.Show("Файли успішно збережено!");

            // Перевірка, чи був такий текст збережений раніше
            var allFiles = Directory.GetFiles(folderPath);
            foreach (var file in allFiles)
            {
                string content = System.IO.File.ReadAllText(file);
                if (content == textBox1.Text && file != fileName1)
                {
                    System.IO.File.Delete(fileName1);
                    fileName1 = file;
                }
                else if (content == textBox2.Text && file != fileName2)
                {
                    System.IO.File.Delete(fileName2);
                    fileName2 = file;
                }
            }
        }
    }
}
```

Form2.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
using System.Windows.Forms;

namespace PlagirismDetector
{
    public partial class MainMenu : Form
    {
        public MainMenu()
        {
            InitializeComponent();
        }

        private void Form2_Load(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            ComparableTextForm form1 = new ComparableTextForm();
            form1.FormClosed += (s, args) => this.Close(); // закриття поточної форми
при закритті Form2
            form1.Show();
            this.Hide(); // приховування поточної форми

        }

        private void button2_Click(object sender, EventArgs e)
        {
            PlagiarismChecker form3 = new PlagiarismChecker();
            form3.FormClosed += (s, args) => this.Close();
            form3.Show();
            this.Hide();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            TextEditor form4 = new TextEditor();
            form4.FormClosed += (s, args) => this.Close();
            form4.Show();
            this.Hide();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            ProgSettings form5 = new ProgSettings();
            form5.FormClosed += (s, args) => this.Close();
            form5.Show();
            this.Hide();

        }

        private void button5_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            NgramsSearch form6 = new NgramsSearch();
            form6.FormClosed += (s, args) => this.Close();
            form6.Show();
            this.Hide();
        }
```

```
    }
}
```

Form3.cs

```
using java.io;
using opennlp.tools.tokenize;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using opennlp.tools.tokenize;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;
using com.sun.org.apache.xpath.@internal.compiler;

namespace PlagirismDetector
{
    public partial class PlagiarismChecker : Form
    {
        private readonly TokenizerME tokenizer;

        public PlagiarismChecker()
        {
            InitializeComponent();

            var text = textBox1.Text;

            using (var modelIn = new FileStream("en-tokens.bin", FileMode.Open))
            {
                var modelStream = new MemoryStream();
                modelIn.CopyTo(modelStream);
                var modelBytes = modelStream.ToArray();
                var inputStream = new ByteArrayInputStream(modelBytes);
                var model = new TokenizerModel(inputStream);
                tokenizer = new TokenizerME(model);
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            var inputText = textBox1.Text;

            var inputKeywords = tokenizer.tokenize(inputText);
            var inputBigrams = GetBigrams(inputKeywords).ToList();
            var inputTrigrams = GetTrigrams(inputKeywords).ToList();

            var matchCounts = new List<int>();
            var totalCounts = new List<int>();

            var files = Directory.GetFiles(@"V:\repos\TextsForPlag");
            foreach (var file in files)
            {
                var fileText = System.IO.File.ReadAllText(file);
                var fileKeywords = tokenizer.tokenize(fileText);
                var fileBigrams = GetBigrams(fileKeywords).ToList();
                var fileTrigrams = GetTrigrams(fileKeywords).ToList();

                var bigramMatchCount = inputBigrams.Intersect(fileBigrams).Count();
```

```
                var totalBigrams = inputBigrams.Count() + fileBigrams.Count();

                var trigramMatchCount = inputTrigrams.Intersect(fileTrigrams).Count();
                var totalTrigrams = inputTrigrams.Count() + fileTrigrams.Count();

                matchCounts.Add(bigramMatchCount + trigramMatchCount);
                totalCounts.Add(totalBigrams + totalTrigrams);
            }

            var totalMatchCount = matchCounts.Sum();
            var totalItemCount = inputBigrams.Count() + inputTrigrams.Count();

            var plagiarismPercentage = totalItemCount > 0 ? totalMatchCount * 100.0 /
totalItemCount : 0;
            var percentageToShow = Math.Min(plagiarismPercentage, 100.0);
            MessageBox.Show($"Plagiarism percentage: {percentageToShow:F2}%");

            // Очистити список перед виведенням нових елементів
            listBox1.Items.Clear();
            for (int i = 0; i < matchCounts.Count; i++)
            {
                double matchPercentage = totalCounts[i] > 0 ? matchCounts[i] * 100.0 /
totalCounts[i] : 0;
                if (matchPercentage >= 20)
                {
                    double inputMatchPercentage = totalItemCount > 0 ? matchCounts[i] *
100.0 / totalItemCount : 0;
                    string fileName = Path.GetFileName(files[i]);
                    listBox1.Items.Add($"{fileName} ({inputMatchPercentage:F2}%)");
                }
            }


        }

        // Функція для генерації дво-грам
        private IEnumerable<string> GetBigrams(string[] tokens)
        {
            for (var i = 0; i < tokens.Length - 1; i++)
            {
                yield return $"{tokens[i]} {tokens[i + 1]}";
            }
        }
        // Функція для генерації три-грам
        private IEnumerable<string> GetTrigrams(string[] tokens)
        {
            for (var i = 0; i < tokens.Length - 2; i++)
            {
                yield return $"{tokens[i]} {tokens[i + 1]} {tokens[i + 2]}";
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            var text1 = textBox1.Text;
            var stopWords = System.IO.File.ReadAllLines("stopwords.txt");
            var noStopWordsText1 = string.Join(" ", text1.Split().Where(w =>
!stopWords.Contains(w.ToLower())));
            textBox1.Text = noStopWordsText1;
        }

        private void button2_Click(object sender, EventArgs e)
        {
```

```
            var text1 = textBox1.Text;
            var noPunctuationText1 = new string(text1.Where(c =>
!char.IsPunctuation(c)).ToArray());
            textBox1.Text = noPunctuationText1;
        }

        private void button6_Click(object sender, EventArgs e)
        {
            MainMenu form2 = new MainMenu();
            form2.FormClosed += (s, args) => this.Close();
            form2.Show();
            this.Hide();
        }

        private void button7_Click(object sender, EventArgs e)
        {
            string folderPath = @"V:\repos\TextsForPlag";
            string fileName1 = Path.Combine(folderPath, Guid.NewGuid().ToString() +
".txt");
            System.IO.File.WriteAllText(fileName1, textBox1.Text);

            MessageBox.Show("Файли успішно збережено!");

            // Перевірка, чи був такий текст збережений раніше
            var allFiles = Directory.GetFiles(folderPath);
            foreach (var file in allFiles)
            {
                string content = System.IO.File.ReadAllText(file);
                if (content == textBox1.Text && file != fileName1)
                {
                    System.IO.File.Delete(fileName1);
                    fileName1 = file;
                }
            }
        }

        private void button4_Click(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*";

            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
                string fileName = openFileDialog.FileName;
                textBox1.Text = System.IO.File.ReadAllText(fileName);
            }
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }
    }
}
```

Form4.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace PlagirismDetector
{
    public partial class TextEditor : Form
    {
        private readonly string directoryPath = @"V:\repos\TextsForPlag\";

        public TextEditor()
        {
            InitializeComponent();
            LoadFiles();
        }

        private void LoadFiles()
        {
            listBox1.Items.Clear();

            // Отримуємо список файлів у директорії
            string[] files = Directory.GetFiles(directoryPath);

            // Виводимо назви файлів у ListBox
            foreach (string file in files)
            {
                listBox1.Items.Add(Path.GetFileName(file));
            }
        }

        private void button4_Click(object sender, EventArgs e)
        {
            MainMenu form2 = new MainMenu();
            form2.FormClosed += (s, args) => this.Close();
            form2.Show();
            this.Hide();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            if (listBox1.SelectedItem != null)
            {
                string fileName = listBox1.SelectedItem.ToString();
                string filePath = Path.Combine("V:\\repos\\TextsForPlag", fileName);
                File.Delete(filePath);
                MessageBox.Show($"File '{fileName}' has been deleted.");
                RefreshFileList();
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            string fileName = textBox2.Text;
            if (fileName == "")
            {
                MessageBox.Show("Введіть назву файлу!");
                return;
            }
            string filePath = @"V:\repos\TextsForPlag\" + fileName + ".txt";

            if (File.Exists(filePath))
            {
```

```
                DialogResult result = MessageBox.Show("Файл з такою назвою вже існує.
Замінити його?", "Попередження", MessageBoxButtons.YesNo);
                if (result == DialogResult.No)
                {
                    return;
                }
            }

            string fileContent = textBox1.Text;
            File.WriteAllText(filePath, fileContent);

            MessageBox.Show("Файл збережено!");
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (listBox1.SelectedItem != null)
            {
                string fileName = listBox1.SelectedItem.ToString();
                string filePath = Path.Combine("V:\\repos\\TextsForPlag", fileName);
                textBox1.Text = File.ReadAllText(filePath);
                textBox2.Text = fileName;
            }
        }

        private void button5_Click(object sender, EventArgs e)
        {
            string fileName = textBox2.Text.Trim();

            if (string.IsNullOrEmpty(fileName))
            {
                MessageBox.Show("Please enter a file name.");
                return;
            }

            string filePath = Path.Combine("V:\\repos\\TextsForPlag", fileName);

            if (File.Exists(filePath))
            {
                MessageBox.Show($"File '{fileName}' already exists.");
                return;
            }

            if (string.IsNullOrEmpty(textBox1.Text.Trim()))
            {
                MessageBox.Show("Please enter some content.");
                return;
            }

            File.WriteAllText(filePath, textBox1.Text);
            MessageBox.Show($"File '{fileName}' has been created.");
        }

        private void RefreshFileList()
        {
            listBox1.Items.Clear();
            string[] fileEntries = Directory.GetFiles("V:\\repos\\TextsForPlag");

            foreach (string fileName in fileEntries)
            {
                listBox1.Items.Add(Path.GetFileName(fileName));
            }
        }
```

```
    }
}
```

Form5.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Windows.Forms;
using java.io;
using Microsoft.ML;
using Microsoft.ML.Data;
using Microsoft.ML.Transforms.Text;

using opennlp.tools.tokenize;

namespace PlagirismDetector
{
    public partial class ProgSettings : Form
    {
        public ProgSettings()
        {
            InitializeComponent();


            // Додати варіанти мови до comboBox1
            comboBox1.Items.Add("en");
            comboBox1.Items.Add("uk");

            // Додати варіанти методу перевірки до comboBox2
            comboBox2.Items.Add("Bigram");
            comboBox2.Items.Add("Trigram");
            comboBox2.Items.Add("Mixed");
        }

        private void button1_Click(object sender, EventArgs e)
        {
            // Отримати вибрану мову та метод перевірки
            string language = comboBox1.SelectedItem.ToString();
            string checkMethod = comboBox2.SelectedItem.ToString();

            // Передати параметри до Form1
            Form1 form1 = new Form1(language, checkMethod);
            form1.Show();

            // Передати параметри до Form3
            Form3 form3 = new Form3(language, checkMethod);
            form3.Show();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            MainMenu form2 = new MainMenu();
            form2.FormClosed += (s, args) => this.Close();
            form2.Show();
            this.Hide();
        }
    }
}
```

Form6.cs

```
using System;
using System.Collections.Generic;
```

```
using System.Data;
using System.IO;
using System.Linq;
using System.Windows.Forms;
using opennlp.tools.tokenize;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;
using java.io;

namespace PlagirismDetector
{
    public partial class NgramsSearch : Form
    {
        private string _selectedFilePath;
        private readonly List<string> _documents = new List<string>();

        public NgramsSearch()
        {
            InitializeComponent();
            LoadDocuments();
        }

        private void LoadDocuments()
        {
            var dirPath = @"V:\repos\TextsForPlag";
            if (Directory.Exists(dirPath))
            {
                var files = Directory.GetFiles(dirPath);
                foreach (var file in files)
                {
                    _documents.Add(file);
                    listBox1.Items.Add(Path.GetFileName(file));
                }
            }
        }

        private void button4_Click(object sender, EventArgs e)
        {
            MainMenu form2 = new MainMenu();
            form2.FormClosed += (s, args) => this.Close();
            form2.Show();
            this.Hide();
        }

        private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            _selectedFilePath = _documents[listBox1.SelectedIndex];
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (_selectedFilePath == null)
            {
                MessageBox.Show("Please select a file.");
                return;
            }

            var modelIn = new FileInputStream("en-tokens.bin");
            var model = new TokenizerModel(modelIn);
            var tokenizer = new TokenizerME(model);

            var fileContent = System.IO.File.ReadAllText(_selectedFilePath);
            var tokens = tokenizer.tokenize(fileContent);
            textBox1.Text = string.Join(" ", tokens);
```

```
            modelIn.close();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            string folderPath = @"V:\repos\TextsForPlag\";
            string filePath = folderPath + listBox1.SelectedItem.ToString();
            if (string.IsNullOrEmpty(filePath))
            {
                MessageBox.Show("Please select a file from the list.");
                return;
            }

            var modelIn = new FileInputStream("en-tokens.bin");
            var model = new TokenizerModel(modelIn);
            var tokenizer = new TokenizerME(model);

            string fileContent = System.IO.File.ReadAllText(filePath);


            string[] tokens = tokenizer.tokenize(fileContent);
            var bigrams = new Dictionary<string, int>();
            var trigrams = new Dictionary<string, int>();

            for (int i = 0; i < tokens.Length - 1; i++)
            {
                string bigram = tokens[i] + " " + tokens[i + 1];
                if (!bigrams.ContainsKey(bigram))
                {
                    bigrams[bigram] = 0;
                }
                bigrams[bigram]++;

                if (i < tokens.Length - 2)
                {
                    string trigram = tokens[i] + " " + tokens[i + 1] + " " + tokens[i +
2];
                    if (!trigrams.ContainsKey(trigram))
                    {
                        trigrams[trigram] = 0;
                    }
                    trigrams[trigram]++;
                }
            }

            string bigramString = string.Join(Environment.NewLine, bigrams.Select(x =>
x.Key + ": " + x.Value));
            string trigramString = string.Join(Environment.NewLine, trigrams.Select(x =>
x.Key + ": " + x.Value));

            listBox2.Items.Clear();
            listBox2.Items.Add("Bigrams:");
            listBox2.Items.AddRange(bigrams.Select(x => x.Key).ToArray()); //+": " +
x.Value
            listBox3.Items.Clear();
            listBox3.Items.Add("Trigrams:");
            listBox3.Items.AddRange(trigrams.Select(x => x.Key).ToArray()); //+": " +
x.Value

        }

        private Dictionary<List<string>, int> GetNgrams(string[] tokens, int n)
        {
```

```
            var ngrams = new Dictionary<List<string>, int>();

            for (int i = 0; i < tokens.Length - n + 1; i++)
            {
                var ngramList = new List<string>();
                for (int j = i; j < i + n; j++)
                {
                    ngramList.Add(tokens[j]);
                }

                if (ngrams.ContainsKey(ngramList))
                {
                    ngrams[ngramList]++;
                }
                else
                {
                    ngrams.Add(ngramList, 1);
                }
            }

            return ngrams;
        }
    }
}
```

Form1Test.cs

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using PlagirismDetector;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PlagirismDetector.Tests
{
    [TestClass()]
    public class Form1Tests
    {
        [TestMethod()]
        public void GetBigramsTest()
        {
            // Arrange
            var form1 = new Form1();
            var tokens = new string[] { "Швидкий", "рудий", "лис", "стрибає", "через",
"ледачого", "собаку"
                //"The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"
            };


            var expectedBigrams = new List<string> { "Швидкий рудий","рудий лис","лис
стрибає","стрибає через","через ледачого","ледачого собаку"
                //"The quick", "quick brown","brown fox","fox jumps","jumps over","over
the","the lazy","lazy dog"
            };
            var result = form1.GetBigrams(tokens);

            Assert.IsTrue(expectedBigrams.SequenceEqual(result));
        }
        [TestMethod()]
        public void GetGetTrigramsTest()
        {
            // Arrange
```

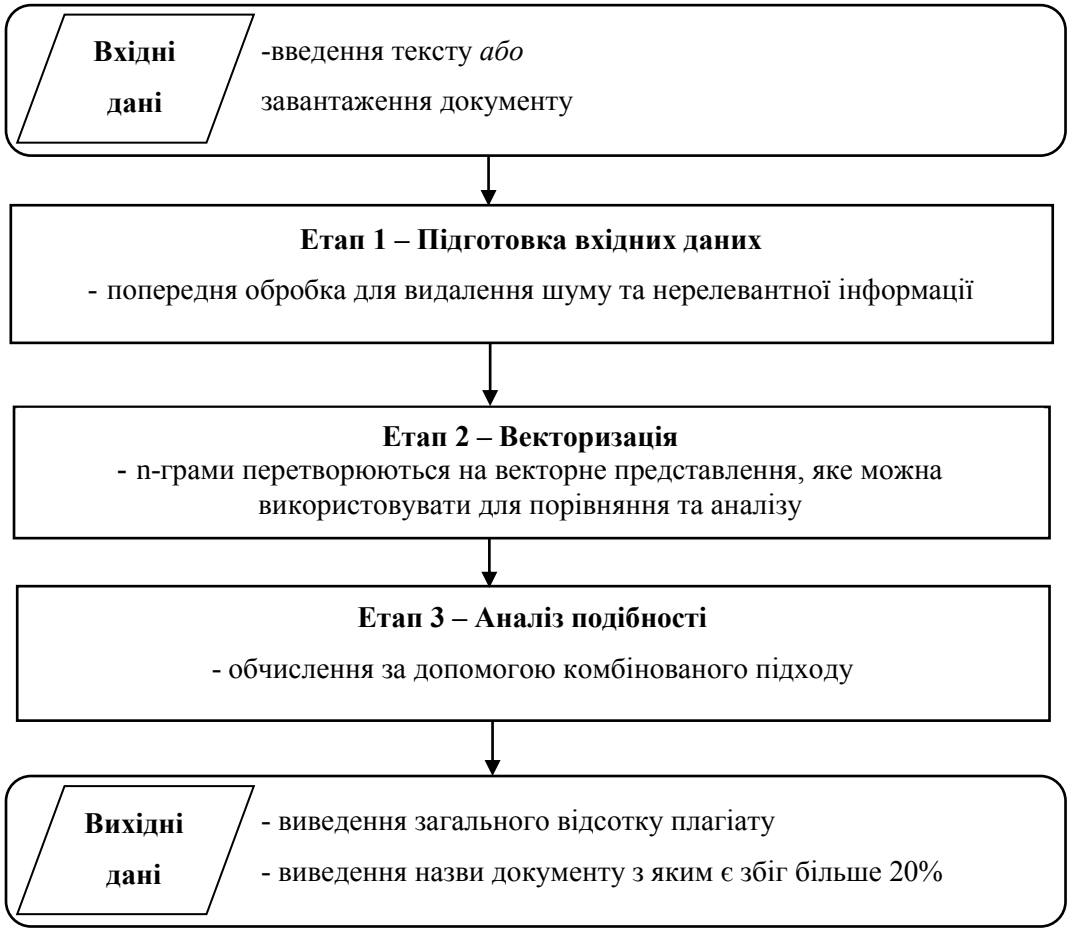
```
            var form1 = new Form1();
            var tokens = new string[] { "Швидкий", "рудий", "лис", "стрибає", "через",
"ледачого", "собаку"
                //"The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"
            };


            var expectedBigrams = new List<string> {
                //"The quick brown","quick brown fox","brown fox jumps","fox jumps
over","jumps over the","over the lazy","the lazy dog",
                "Швидкий рудий лис","рудий лис стрибає","лис стрибає через","стрибає
через ледачого","через ледачого собаку"
            };
            var result = form1.GetTrigrams(tokens);

            Assert.IsTrue(expectedBigrams.SequenceEqual(result));
        }
    }
}
```

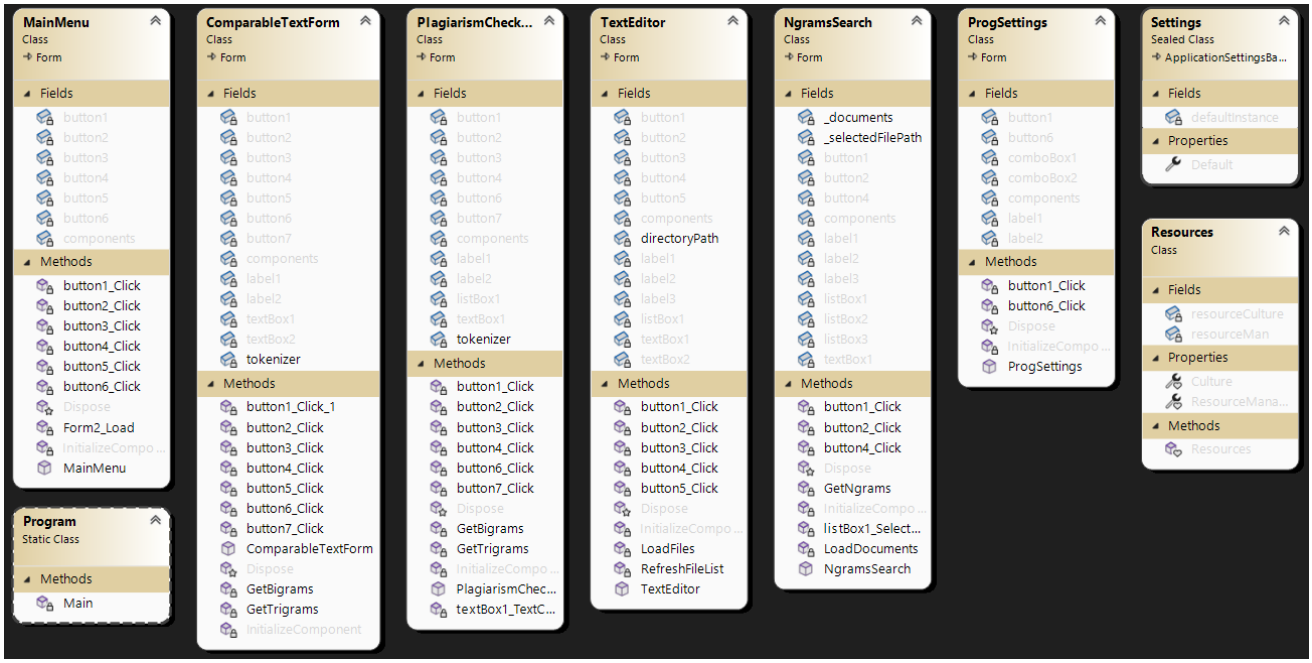
Додаток Б

Схема методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату



Додаток В

Діаграма класів інформаційної системи пошуку плагіату за семантичним аналізом текстів



Додаток Г

Презентаційний матеріал

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

МЕТОД СЕМАНТИЧНОГО АНАЛІЗУ ТЕКСТІВ З ВИКОРИСТАННЯМ N-ГРАМ ДЛЯ ЗАДАЧ ПОШУКУ ПЛАГІАТУ

Виконав:
студент 4 курсу, групи КН-19-2
Шиманський Максим Борисович

Керівник:
викладач кафедри КН
Молчанова Марина Олексіївна

Актуальність

Інтелектуальний аналіз тексту є значним напрямом сучасних досліджень у галузі штучного інтелекту. Перевірка тексту на плагіат широко використовується в освітніх закладах, у paper mill компаніях, видавництвах та журналістиці.

Застосування n-грам та семантичного аналізу дозволяє краще порівнювати тексти, знаходячи аналогічність у використовуваних словах. Використання обробки природної мови (NLP) спрощує взаємодію з текстом, що полегшує завдання і економить час.

Об'єктом дослідження є семантичний аналіз текстів для виявлення плагіату, а предметом дослідження є моделі, методи, алгоритми та засоби для визначення обсягу запозичень шляхом семантичного аналізу.

Робота над методом семантичного аналізу тексту з використанням n-грам для пошуку плагіату є актуальною, оскільки вона допомагає ефективно виявляти плагіат шляхом аналізу семантики та порівняння n-грам.

Мета і задачі роботи

Метою кваліфікаційної роботи бакалавра є розробка й апробація методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату, для чого слід вирішити задачі:

1. провести аналіз предметної області;
2. розробити метод семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату;
3. спроєктувати структуру інформаційної системи пошуку плагіату за семантичним аналізом текстів й структуру відповідної БД;
4. розробити інформаційну систему пошуку плагіату за семантичним аналізом текстів;
5. провести тестування створеної інформаційної системи пошуку плагіату за семантичним аналізом текстів;
6. провести дослідження практичної ефективності розробленого методу за використання створеної інформаційної системи.

Розроблена програмна реалізація методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату в вигляді інформаційної системи на платформі .NET має виконувати наступні основні групи функцій:

- пошук n-грам;
- порівняння тексту з існуючою БД текстів;
- порівняння текстів між собою;
- редагування збережених текстів.

Схема методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату

Математична формула для утворення двограм

Формула для ланцюгів Маркова другого порядку, або двограм, може бути записана наступним чином:

$$P(w_i|w_i-1) = \frac{count(w_i-1, w_i)}{count(w_i-1)}$$

де: $P(w_i|w_i-1)$ – ймовірність того, що слово wi з'явиться після слова wi-1; $count(w_i-1, w_i)$ – кількість разів, коли двіграма wi-1, wi з'являється в тексті; $count(w_i-1)$ – кількість разів, коли слово wi-1 з'являється в тексті.

Математична формула для утворення триграм

$$P(w_i|w_i-1, w_i-2) = \frac{count(w_i-2, w_i-1, w_i)}{count(w_i-2, w_i-1)}$$

де: $P(w_i|w_i-1, w_i-2)$ – триграма, яка складається з трьох слів; $count(w_i-2, w_i-1, w_i)$ – кількість разів, коли триграма зустрічаєтьсяу тексті; $count(w_i-2, w_i-1)$ – кількість разів, коли перші два слова триграми зустрічаютьсяу тексті.

Формула для обчислення подібності n-грам між документами

$$percnt_{Ng} = \frac{num_{Ng}}{total_{Ng}} \cdot 100\%$$

де $percnt_{Ng}$ – відсоток збігів за n-рамами; num_{Ng} – кількість співпадінь n-грам серед двох текстів; $total_{Ng}$ – загальна кількість n-грам серед двох текстів.

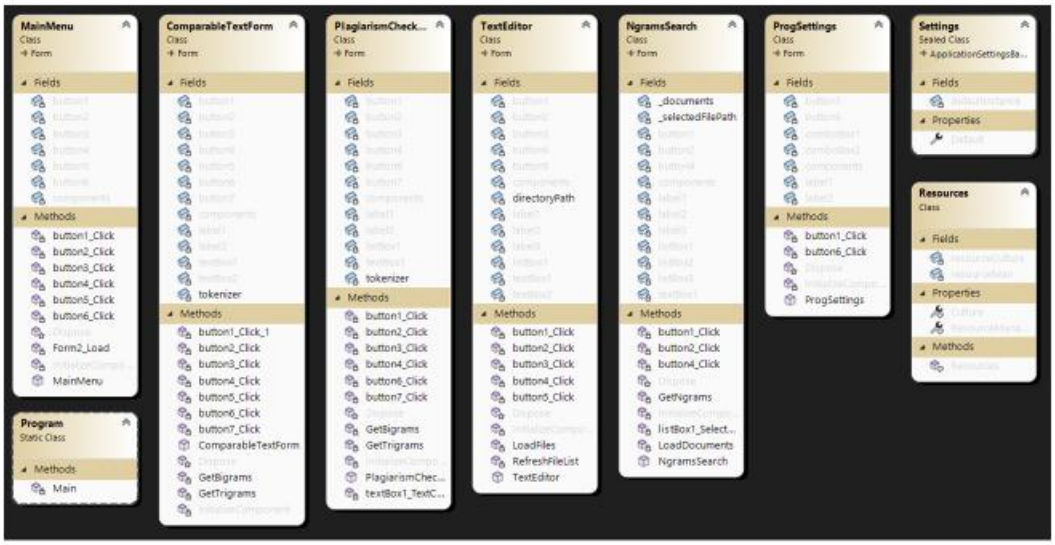
Схема структури інформаційної системи пошуку плагіату за семантичним аналізом текстів

Засоби розробки

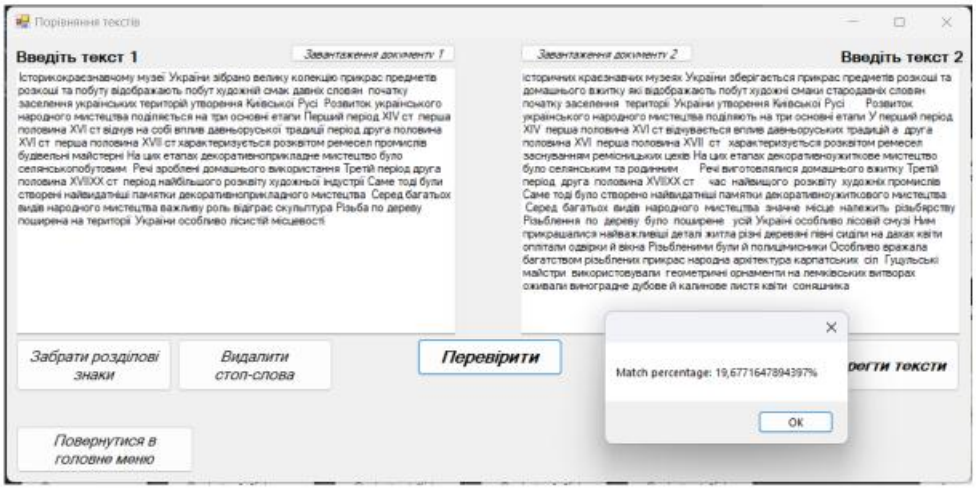
У якості засобів розробки було обрано платформу .NET Framework, мову програмування C#, редактор програмного коду Visual Studio 2022 та СКБД SQLite.

- Для реалізації інтерфейсу користувача було обрано Windows Form, що є засобом, широко використовуваним у розробці графічних інтерфейсів для Windows.
- Для обробки та аналізу тексту використовується бібліотека OpenNLP.
- Для реалізації бази даних в даному проєкті було обрано SQLite. SQLite надає простий у використанні SQL-синтаксис та забезпечує високу продуктивність та надійність при роботі з базою даних.

Діаграма класів програмної реалізації методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату

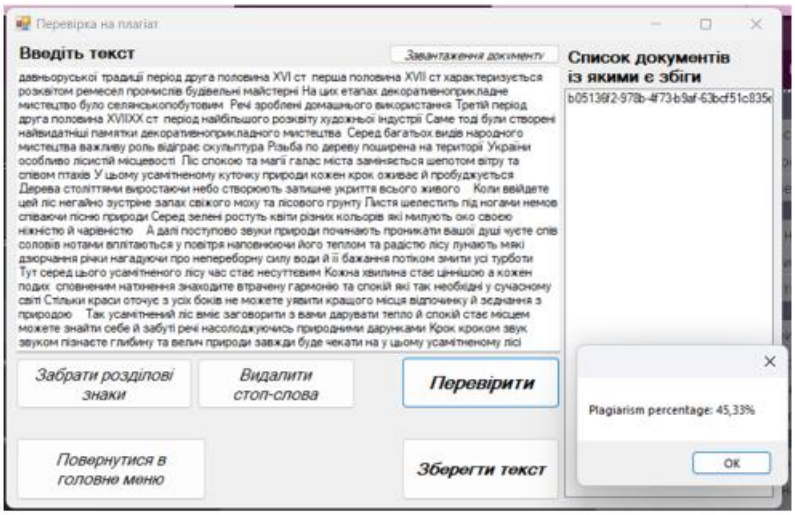


Програмна реалізація методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату



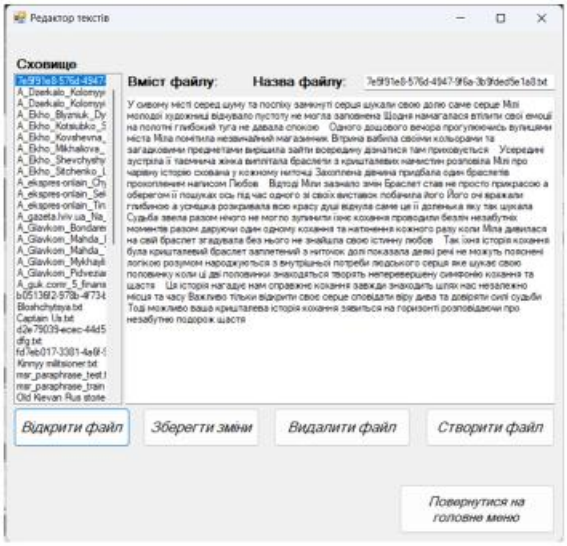
Порівняння двох текстів між собою

Програмна реалізація методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату



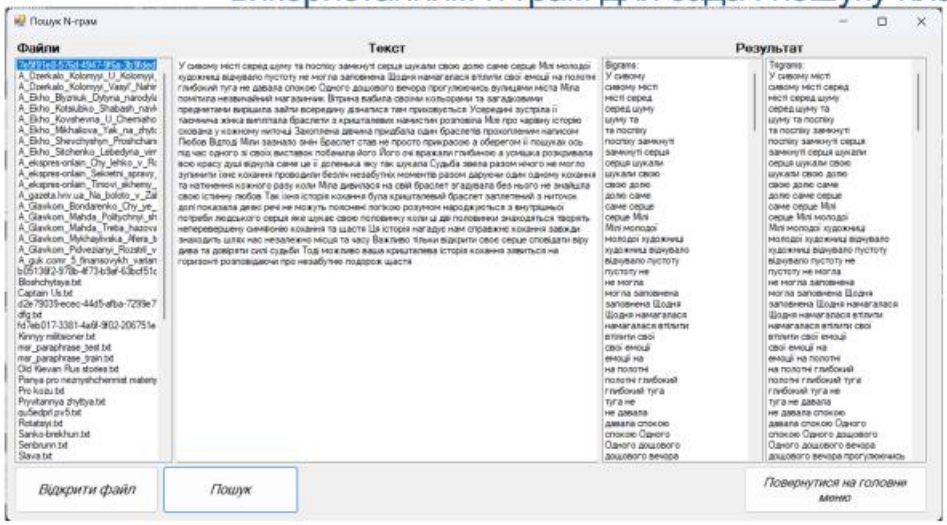
Форма перевірки тексту на плагіат

Програмна реалізація методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату



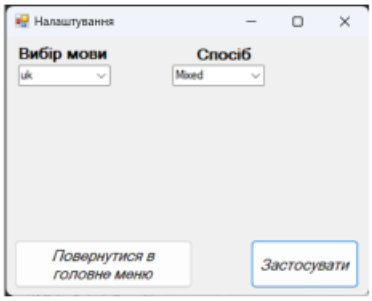
Форма редагування збережених текстів

Програмна реалізація методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату

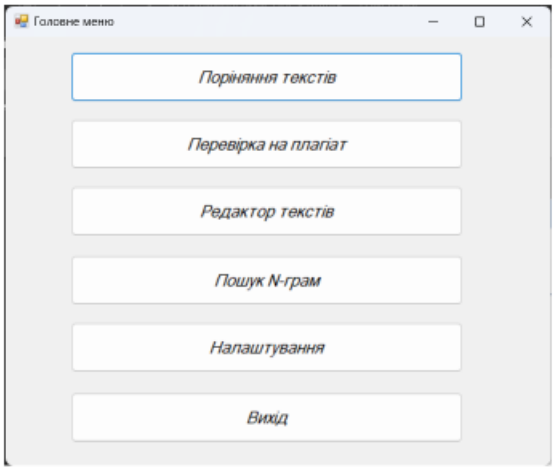


Форма пошуку n-грам

Програмна реалізація методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату

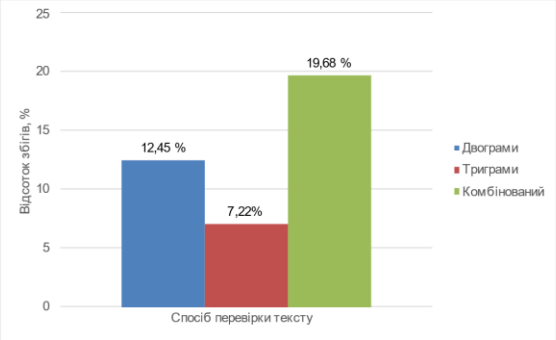


Форма налаштувань



Форма головного меню

Дослідженняефективностіметодусемантичного аналізутекстівз використаннямn-грам для задач пошуку плагіату

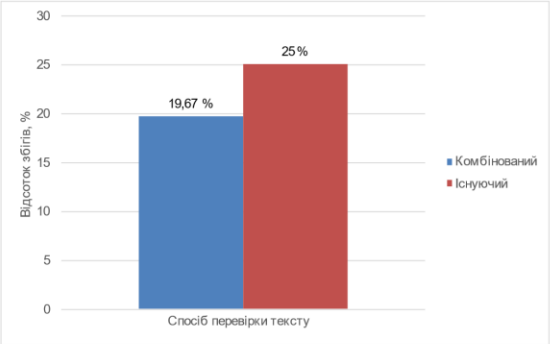


Діаграма, що демонструє відсотки знайденого плагіату за різними параметрами розробленого методу (на основі біграм, триграм, та комбінованого)

В результатіпроведеного досліження було встановлено, що знаходження збігів між текстами становить:

- Метод двограм – 12,45%;
- Метод триграм – 7,22%;
- Комбінований метод – 19,67%.

Дослідженняефективностіметодусемантичного аналізутекстівз використаннямn-грам для задач пошуку плагіату



Діаграма, що показує відсоток знайденого плагіату за різними методами (існуючим та розробленим)

Було проведено порівняння комбінованого методу з існуючими методами виявлення плагіату, де комбінований метод показав результат – 19,67%, а існуючий метод (Smodin) - 25%.

Отримані результатипродемонстрували відмінність між комбінованим та існуючим методами. Виявлено, що комбінований метод здатний ефективно працювати з текстами та забезпечувати надійне виявлення плагіату.

Висновки

У рамках виконання кваліфікаційної роботи бакалавра було виконано **розробку й апробацію методу** семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату. Зокрема, було проведено аналіз предметної області й досліджено сучасні підходи до токенізації тексту та утворення n-грам, розглянуто існуючі програмні реалізації за цим напрямком.

Розроблена **програмна реалізація** методу семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату на платформі .NET виконує наступні основні функції:

- Пошук n-грам.
- Порівняння тексту з існуючою БД текстів.
- Порівняння текстів між собою.
- Редагування збережених текстів.

У якості засобів розробки було обрано фреймворк .NET Framework, мову програмування C#, редактор програмного коду Visual Studio 2022 та СКБД SQLite.

UNICHECK

Ім'я користувача:
Кафедра КН

ID перевірки:
1015400051

Дата перевірки:
02.06.2023 18:08:49 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
02.06.2023 18:12:36 EEST

ID користувача:
100005671

Назва документа: КН-19-2 Шиманський

Кількість сторінок: 69 Кількість слів: 11367 Кількість символів: 84788 Розмір файлу: 4.87 MB ID файлу: 1015063946

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

2.98%
Схожість

Найбільша схожість: 0.94% з джерелом з Бібліотеки (ID файлу: 1011421045)

2.74% Джерела з Інтернету | 296 ..... Сторінка 71

1.96% Джерела з Бібліотеки | 143 ..... Сторінка 73

0% Цитат

Не знайдено жодних цитат

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування | 19 сторінок

Fri Jun 02 17:15:20 EEST 2023, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 2.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 8%**

ID: 114616 Назва: КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА Додано в БД: 2023-06-02 Автора: М.Б. Шиманський Керівники: М.О. Молчанова Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	71549	1105	2110 (3%)	33 (3%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ КАФЕДРИ КОМП'ЮТЕРНИХ НАУК
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: *Метод семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату*

Автор: *студент групи КН-19-2 Шиманський Максим Борисович*

Спеціальність: *122 – Комп'ютерні науки*

Освітня програма: *освітньо-професійна*

Науковий керівник: *викладач Молчанова М.О.*

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	***відповідає***
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі Шиманського Максима Борисовича, не є плагіатом, оскільки: запозичення розміщені в розділі огляду існуючих підходів, не описують безпосередньо авторську роботу і не стосуються її результатів; усі запозичення фрагментарні; серед запозичень знаходяться загальновідомі терміни, скорочення та матеріали статей.

Обсяг запозичень, визначений системами виявлення збігів/ ідентичності/схожості, складає:

- за системою Anti-Plagiarism: 2%;

- за системою Unicheck: 2.98 %.

Керівник роботи _____ *Марина МОЛЧАНОВА*

Гарант ОП _____ *Олександр МАЗУРЕЦЬ*

Завідувач кафедри КН _____ *Олександр БАРМАК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу бакалавра

студента *гр. КН-19-2 Шиманського Максима Борисовича*

за темою *Метод семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату*

1. Актуальність теми

Тема "Метод семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату" залишається актуальною в сучасному інформаційному світі, оскільки проблема плагіату є поширеною і вимагає ефективних інструментів для виявлення та запобігання цьому явищу.

2. Відповідність роботи предметній області Стандарту спеціальності 122 Комп'ютерні науки

Дослідження, проведене у кваліфікаційній роботі, напрямлене на розробку методу семантичного аналізу текстів з використанням n-грам для пошуку плагіату, відповідає предметній області стандарту спеціальності 122 Комп'ютерні науки, оскільки це пов'язано з обробкою текстової інформації та розробкою інформаційних систем.

3. Професійні та особистісні якості бакалавра

Виконавець кваліфікаційної роботи демонструє високий рівень професійної компетентності, зокрема у галузі обробки текстових даних та програмування. Крім того, бакалавр проявив належну самодисципліну, організованість та систематичність під час виконання роботи.

4. Ступінь самостійності під час виконання кваліфікаційної роботи

У ході виконання кваліфікаційної роботи бакалавр виявив значний рівень самостійності. Він власноруч аналізував наукову літературу, використовував методи дослідження та програмні інструменти для розробки інформаційної системи, а також самостійно здійснював аналіз результатів і формулював висновки.

5. Ступінь оволодіння методами дослідження

У процесі виконання кваліфікаційної роботи бакалавр успішно оволодів методами семантичного аналізу текстів з використанням n-грам, а також методами виявлення плагіату. Він використав наукову літературу і практичні знання для впровадження цих методів у розроблену інформаційну систему.

6. Повнота та якість розкриття теми роботи

Розкриття теми роботи є повним і детальним. Бакалавр докладно описав метод, розробку інформаційної системи пошуку плагіату, а також провів експерименти для перевірки ефективності системи. Він також звернув увагу на потенційні обмеження та можливості подальшого розвитку розробленого методу.

7. Логічність, послідовність, аргументованість, літературна грамотність викладення матеріалу

Викладення матеріалу в кваліфікаційній роботі є логічним, послідовним та аргументованим. Бакалавр чітко структурував роботу, використовував адекватні аргументи та підтримував їх науковими джерелами. Також він виявляє високий рівень літературної грамотності.

8. Можливість практичного застосування кваліфікаційної роботи бакалавра, окремих її частин

Кваліфікаційна робота бакалавра має значну практичну цінність. Розроблена інформаційна система з методом семантичного аналізу текстів з використанням n-грам може бути застосована для виявлення плагіату в різних областях, включаючи академічні дослідження, письмові роботи та інші текстові матеріали. Також окремі частини роботи, наприклад, розроблені методи та алгоритми, можуть бути використані як основа для подальших досліджень та розробок.

9. Висновок про можливість допуску кваліфікаційної роботи бакалавра до захисту, на яку оцінку заслуговує робота

*Враховуючи високий рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка «**відмінно**».*

Керівник _____ *викладач кафедри КН Марина МОЛЧАНОВА*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

студента *гр. КН-19-2 Шиманський М.Б.*

за темою: *Метод семантичного аналізу текстів з використанням n-грам для задач пошуку плагіату*

1. Актуальність обраної теми

Тема, що стосується методу семантичного аналізу текстів для пошуку плагіату з використанням n-грам, є вкрай актуальною в сучасному інформаційному суспільстві. Проблема плагіату стає все більш поширеною, особливо в академічному та науковому середовищі. Розробка ефективного методу аналізу текстових даних з метою виявлення плагіату має велике значення для забезпечення чесності та інтелектуальної власності.

2. Повнота розкриття мети та завдань роботи

Кваліфікаційна робота бакалавра повно і зрозуміло розкриває мету дослідження, якою є розробка методу семантичного аналізу текстів з використанням n-грам для пошуку плагіату. Описані завдання роботи відповідають поставленій меті і спрямовані на розробку інформаційної системи, яка забезпечує точне визначення збігів та оцінку обсягу текстових запозичень.

3. Зміст кожного розділу роботи

Кожен розділ кваліфікаційної роботи бакалавра містить відповідну та логічну інформацію. Літературний огляд глибоко аналізує попередні дослідження в області семантичного аналізу текстів та пошуку плагіату. Методологічний розділ ретельно описує методи та підходи, використані для розробки інформаційної системи. Розділ з результатами дослідження детально представляє отримані результати та їх аналіз. Висновки та подальші напрямки дослідження є логічним завершенням роботи.

4. Оцінка розробленої інформаційної системи, її практична цінність

Розроблена інформаційна система пошуку плагіату за семантичним аналізом текстів з використанням n-грам є значним досягненням. Система дозволяє точно виявляти збіги та оцінювати обсяг текстових запозичень. Практична цінність системи полягає в її потенціалі для виявлення плагіату в академічних та наукових роботах, що сприяє забезпеченню інтелектуальної чесності та підвищенню якості наукових досліджень.

5. Якість оформлення кваліфікаційної роботи бакалавра

Кваліфікаційна робота бакалавра має добре організовану структуру, чіткі та логічні розділи, а також зрозумілий стиль викладу. Оформлення відповідає науковим стандартам та вимогам і містить належну кількість посилань на використану літературу.

6. Недоліки кваліфікаційної роботи бакалавра

У кваліфікаційній роботі бакалавра бажаним є більш детальне обґрунтування та аналіз результатів експериментів, а також можливість використання більш широкого спектру методів семантичного аналізу для досягнення більш точних результатів.

7. Загальний висновок (допускається чи не допускається до захисту), та оцінка на яку заслуговує кваліфікаційна робота.

Враховуючи рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка « відмінно *».*

Рецензент Бездольська І.П.