НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«На правах рукопису»
УДК _____

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 121 Інженерія програмного забезпечення

на тему: «Модифікований метод автоматизованого порівняння текстів на ідентичність з використанням ущільнення даних»

Виконав:
студент VI курсу, групи КП-61м
Шевчук Михайло Михайлович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент,
Заболотня Т.М. _____

Рецензент:
Доцент кафедри ММСА ІПСА, к.т.н., доцент,
Дідковська М.В. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2018 року

ЗМІСТ

ВСТУП

Задача пошуку дублікатів в текстових даних тривалий час залишається актуальною для багатьох галузей діяльності людини. Прикладами застосування рішень цієї задачі є пошук плагіату в студентських роботах, пошук схожих текстів у стрічках новин, перевірка творів на унікальність у видавництвах тощо.

Одними із класичних методів пошуку плагіату в текстових документах є група методів, що базується на використанні шинглів. Шинглом називається фрагмент тексту, довжина якого фіксована і визначається кількістю слів, що входять до нього. Головними недоліками методів цього сімейства є те, що більшість з них має високу обчислювальну складність [1].

Альтернативою до використання шинглів є залучення методів, що використовують практики з інших галузей обробки даних. Одним з таких методів є метод пошуку плагіату, що базується на ущільненні документів і подальшому використанні нормалізованої відстані ущільнення (normalized compression distance, NCD) [2]. Цей метод передбачає оброблення пари текстових документів, які потрібно перевірити на ідентичність, і їх конкатенації певним алгоритмом ущільнення. На основі отриманих довжин ущільнених документів обчислюється нормалізована відстань ущільнення, яка використовується в якості показника ідентичності вихідних документів. Головною перевагою цього методу, є невисока обчислювальна складність, яка визначається лише використовуваним алгоритмом ущільнення. Проте описаний метод не надає конкретних вказівок щодо того, який алгоритм ущільнення слід застосувати, хоча саме від цього залежить точність ідентифікації дублікатів. Задача пошуку на ідентифікації дублікатів – це задача, де головним критерієм є точність ідентифікації (при умові адекватних значень швидкодії по часу та пам’яті). Саме тому підвищення даного критерію є найактуальнішою і важливою задачею. Виходячи з того, що метод, що розглядається у роботі, в основі

своєї роботи використовує ущільнення, перспективним напрямком роботи по покращенню даного методу, є робота над використанням адаптацій тексту для специфічних нюансів алгоритмів ущільнення та базових принципів, що він використовує.

Тому модифікація методу автоматизованого порівняння текстів на ідентичність з використанням ущільнення даних, що базується на описаних вище нюансах дозволить покращити його роботу за критерієм точності.

1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ПОРІВНЯННЯ ТЕКСТІВ НА ІДЕНТИЧНІСТЬ

1.1. Загальний опис задачі порівняння текстів на ідентичність

Введемо деякі визначення, якими надалі будемо оперувати у даній роботі.

Текстом будемо називати сукупність речень, що складені будь-якою природною мовою.

Під текстовим файлом або текстовим документом будемо розуміти програмний файл, що містить дані, які повністю можуть бути розпізнані та трактовані людиною як текст.

Ідентичністю текстових даних будемо називати те наскільки один текст за своїми синтаксичними, лексичними та іншими конструкціями схожий на інший.

Під автоматизованим порівнянням текстів на ідентичність мається на увазі таке порівняння текстів, при якому всі рішення приймаються автоматично ЕОМ, без участі людини.

Ущільненням даних будемо називати алгоритмічне перетворення даних, що зменшує їх об’єм.

Задачею порівняння текстів на ідентичність називається задача, при якій маючи два тексти у вигляді текстових документів необхідно визначити ступінь ідентичності даних текстів за певними критеріями.

Навіть використання даного визначення не зменшує неоднозначність результату порівняння текстових документів на ідентичність, бо не існує однозначних найкращих критеріїв ідентичності текстів. Тому що при усьому різноманітті методів порівняння текстових документів на ідентичність, різні методи будуть давати різні найкращі результати, відповідно до різних критеріїв. Саме тому множина можливих рішень задачі порівняння текстових документів на ідентичність залежить від методу порівняння, що використовується та критеріїв порівняння.

Також варто зазначити, що при достатньо великому об’ємі текстового документу, від вибору методу порівняння суттєво залежить час та ефективність роботи відповідного алгоритму.

Основною перешкодою для успішного рішення даної задачі є гігантський об'єм даних, що зберігаються в базах сучасних пошукових машин. Такий об’єм робить практично неможливим (за розумний період часу) її «пряме» рішення шляхом попарного порівняння текстових документів. Тому в останній час велика увага приділяється розробці методів пониження обчислювальної складності алгоритмів, що створюються за рахунок вибору різних евристик (наприклад, хешування певного фіксованого набору «значущих» слів або речень текстового документу, семплування набору підрядків тексту, використання дактилограм та інші).

При застосуванні наближених підходів спостерігається зменшення (іноді дуже значне) показника повноти знаходження дублів.

Важливою вимогою, що впливає на якість алгоритмів для порівняння текстів на ідентичність є їх стійкість до «невеликих» змін вихідних документів та можливість стабільно обробляти короткі документи.

1.2. Огляд існуючих методів порівняння текстів на ідентичність

Одними з перших досліджень в області порівняння текстів на ідентичність, а саме пошуку нечітких дублікатів, є роботи U. Manber [3] та N. Heintze [4]. В цих роботах для побудови вибірки використовуються послідовності сусідніх букв. Дактилограма файла ([3]) або документа ([4]) містить в собі усі текстові підрядки фіксованої довжини. Чисельне значення дактилограм обчислюється за допомогою алгоритму випадкових поліномів Карпа-Рабіна [5]. У якості міри подібності двох документів використовується відношення кількості загальних підрядків до розміру

файла чи документа. Пропонується ряд методів, що направлені на зниження обчислювальної складності алгоритму.

U. Manber використовував цей метод для знаходження схожих файлів (утиліта sif), а N. Heintze – для виявлення нечітких дублікатів документів (система Koala).

У 1997 році A. Broder et al. [6, 7, 8] запропонували новий, «синтаксичний» метод оцінки ідентичності між документами, що базується на представленні документа у вигляді множини різноманітних послідовностей фіксованої довжини k, що складаються з сусідніх слів. Такі послідовності були названі «шинглами». Два документа вважались схожими, якщо множини їх шинглів суттєво перетинались. Оскільки кількість шинглів приблизно рівна довжині документа в словах, автори запропонували два методи семплування для отримання репрезентативних підмножин.

Перший метод залишав тільки ті шингли, чиї «дактилограми», що вираховувались за алгоритмом Карпа-Рабіна [5], ділились без залишку на деяке число m. Основним недоліком є залежність виборки від довжини документа. Саме тому документи невеликого розміру (за кількістю слів) представлялися або дуже короткими вибірками, або взагалі не мали таких. Другий метод відбирав тільки фіксовану кількість s шинглів з найменшими значеннями дактилограм або залишав усі шингли, якщо їх кількість не перевищувала s.

Подальшим розвитком концепцій A. Broder et al. були дослідження D. Fetterly et al. [9], A. Broder et al. [10].

Для кожного ланцюжка вираховуються 84 дактилограми за алгоритмом Карпа-Рабіна [5] за допомогою взаємно-однозначних та незалежних функцій, що використовують випадкові набори («minwise independent») простих поліномів. В результаті кожен документ був представлений у вигляді 84 шинглів, що мінімізували значення відповідної функції.

Потім 84 шингли розбивались на 6 груп по 14 (незалежних) шинглів в кожній. Ці групи називаються «супершинглами». Якщо два документи мають схожість, наприклад, $p \sim 0.95\,(95\%)$, то 2 відповідних супершингли співпадають з ймовірністю $p^{14} \sim 0.95^{14} \sim 0.49\,(49\%)$.

Оскільки кожен документ представлений у вигляді 6 супершинглів, то ймовірність того, що у двох документів співпадуть не менше 2х супершинглів, дорівнює $1-(1-0.49)^6-6\cdot 0.49\cdot(1-0.49)^5 \sim 0.9\,(90\%)$. Для порівняння: якщо два документа мають схожість тільки $p \sim 0.8\,(80\%)$, то ймовірність співпадіння не менше двох супершинглів є всього $0.026\,(2.6\%)$.

Таким чином для ефективної перевірки співпадіння не менше двох суперншинглів (і, відповідно, підтвердження гіпотези про схожість вмісту) кожен документ представляється усіма попарними комбінаціями з 6 супершинглів, які називають «мегашинглами». Кількість таких мегашинглів дорівнює 15 (кількість комбінацій з 6 по 2). Два документа схожі за вмістом, якщо у них співпадає хоча б один мегашингл.

Головною перевагою даного алгоритму у порівнянні з дослідженнями A. Broder et al. є те, що, по-перше, будь-який документ (у тому числі і дуже маленький) завжди представляється у вигляді вектора фіксованої довжині та, по-друге, схожість визначається простим порівнянням координат вектора і не потребує (як у A. Broder) виконання теоретико-множинних операцій.

Інший сигнатурний підхід, що базується вже не на синтаксичних, а на лексичних принципах, був запропонований A. Chowdhury et al. у 2002 та покращений у 2004 роках. [11, 12, 13].

Основна ідея цього підходу є у обчисленні дактилограми I-Match для представлення вмісту документів. З цією метою спочатку для початкової колекції документів будується словник L, що містить слова з середніми значеннями IDF, оскільки такі слова забезпечують, як правило, більш точні

результати при знаходженні нечітких дублікатів. Слова з великими та маленькими значеннями IDF відкидаються.

Потім для кожного документа формується множина U різних слів, що входять в нього, та визначається перетин U та словника L. Якщо розмір цього перетину більше деякого мінімального порога (що визначається емпірично), то список слів, що входять у перетин, впорядковується і для нього обчислюється I-Match сигнатура (хеш-функція SHA1).

Два документи вважаються схожими, якщо в них співпадають I-Match сигнатури (але може виникнути проблема з колізією хеш-кодів). Алгоритм має велику обчислювальну ефективність, що перевищує показники алгоритму A. Broder. Іншою перевагою алгоритму (також у порівнянні до A. Broder) є його висока ефективність при порівнянні невеликих за розміром документів. Основним недоліком є його нестійкість при невеликих змінах вмісту документа.

Для подолання цього недоліку оригінальний алгоритм був модифікований авторами, і у нього була додана можливість багаторазового випадкового перемішування основного словника. Суть нових модифікацій полягає у наступному.

Додатково до основного словника L створюються K різноманітних словників $L_1 - L_K$, які отримуються шляхом видалення з вихідного словника деякої невеликої фіксованої частини p слів, що складають приблизно 30-35% від початкового об'єму L.

Для кожного документа замість однієї обчислюється K+1 I-Match сигнатур за вищеописаним алгоритмом, тобто документ представляється у вигляді вектора розмірності K+1, і два документи вважаються ідентичними, якщо у них співпаде хоча б одна з координат.

Якщо документ зазнає невеликих змін (близько n слів), то ймовірність того, що хоча б одна з K додаткових сигнатур залишиться незмінною буде дорівнювати $1-(1-p^n)^K$.

Дійсно, ймовірність того, що зміни не торкнуться якого-небудь одного словника, дорівнює p^n - ймовірності того, що усі зміни попадуть в видалену частину вихідного словника. Тоді $(1-p^n)$ – ймовірність того, що сигнатура зміниться, а $(1-p^n)^K$ – ймовірність того, що усі сигнатури зміняться (оскільки додаткові словники формуються незалежно) і, відповідно, $1-(1-p^n)^K$ – і є шукана ймовірність.

Рекомендованими значеннями параметрів, що гарно показали себе на практиці, є $p=0.33$ та $K=10$.

Алгоритм показав високі результати при використанні в різноманітних застосуваннях веб-пошуку та фільтрації спаму.

Схожий підхід описаний у патенті US Patent 6,658,423 W. Pugh [14] з Google. Автор пропонує повний словник документа розбити на фіксовану кількість списків слів за допомогою якоїсь функції хешування (наприклад, за допомогою залишку від ділення хеш-кода на кількість списків). Далі для кожного списку обчислюється дактилограма і для документа вважаються подібними, якщо вони містять хоча б одну спільну дактилограму. Автор приводить оцінки стійкості алгоритму до невеликих змін вихідного документа. В роботі немає ніяких відомостей про ефективність практичного застосування через міркування конфіденційності.

Ще одним сигнатурним підходом, що також базується на лексичних принципах (тобто, використанні словнику), є метод «опорних» слів, що був запропонований С. Ільїнським та ін. [15]. Його суть полягає у наступному.

Спочатку з індексу за деяким правилом (див. нижче) обирається множина з N слів (N визначається емпіричним шляхом), які називаються «опорними». Потім кожен документ представляється N-мірнім двійковим вектором, де і-та координата дорівнює 1, якщо і-те «опорне» слово має в документі відносну частоту більше «певного» порога (що встановлюється окремо для кожного слова), і дорівнює 0 в іншому випадку Цей двійковий

вектор називається сигнатурою документа. Два документа «схожі», якщо їх сигнатури є схожими.

Загальні міркування для побудови множини «опорних» слів наступні:

- множина слів повинна охоплювати максимально можливу кількість документів;
- «якість» слова (див. нижче) має бути якомога більшою;
- кількість слів у наборі має буду мінімальною.

Опишемо алгоритм побудови множини и вибору порогових частот.

Нехай «частота» це нормована внутридокументна частота слова у документа TF, що лежить у діапазоні від 0 до 1, де 1 – частота слова, що зустрічається у документі більше всього разів.

Для кожного слова (однократно) будується розподіл документів за такою внутридокументною «частотою».

Проводимо декілька ітерацій, кожна з яких складається з двох фаз. У першій максимізується покриття документів в індексі при фіксованій (обмеженій знизу) точності; у другій максимізується точність при фіксованому покриті.

Визначимо «точність» слова наступним чином: точність тим вище, ніж менша зустрічаємість слова в дельта-околі даного значення відносної частоти (тобто чим менше документів з TF, рівним $TF_{гран} \pm \delta$). Частоту, що має найкращу точність ми називаємо пороговою та запам’ятовуємо для подальшого використання в алгоритмі.

Після кожної ітерації відкидаємо найбільш «погані» слова. Після останньої ітерації залишимо достатньо слів для гарного покриття.

Цей метод дозволяє, починаючи з вибірки в сотні тисяч слів, залишити набір в 3-5 тисяч, розрахунок сигнатур по якому з застосуванням повнотекстового індексу виконується на міліарному індексі декілька хвилин на пошуковому кластері системи Яндекс.

1.3. Опис методу порівняння текстів на ідентичність на основі ущільнення

Одним з методів порівняння текстів на ідентичність є метод на основі ущільнення, що використовує normalized compression distance (NCD). NCD – це спосіб виміру подібності між двома об’єктами. [16] Цей параметр вираховується за формулою (1).

Даний метод є доволі новим, він був запропонований С. Ващіліном і базується на засадах механізму ущільнення даних, а саме – усунення надлишку інформації, що міститься у вихідних даних [17]. Його суть полягає у наступному.

Нехай ми маємо текстові документи A, B. Для порівняння їх на вмісту на ідентичність необхідно виконати наступні кроки (кроки 2-3 наведені на рис 1.1).

1. Зчитати текстові документи A, B у двійковому вигляді у відповідні змінні.
2. Виконати конкатенацію A, B у зміну AB .
3. Почергово застосувати обраний алгоритм ущільнення на змінні A, B, AB та записати результат у змінні cA, cB, cAB відповідно.
4. Вирахувати нормалізовану відстань ущільнення за формулою (1.1).

$$NCD = \frac{Length(cAB) - Min(Length(cA), Length(cB)}{Max(Length(cA), Length(cB))} \quad (1.1)$$

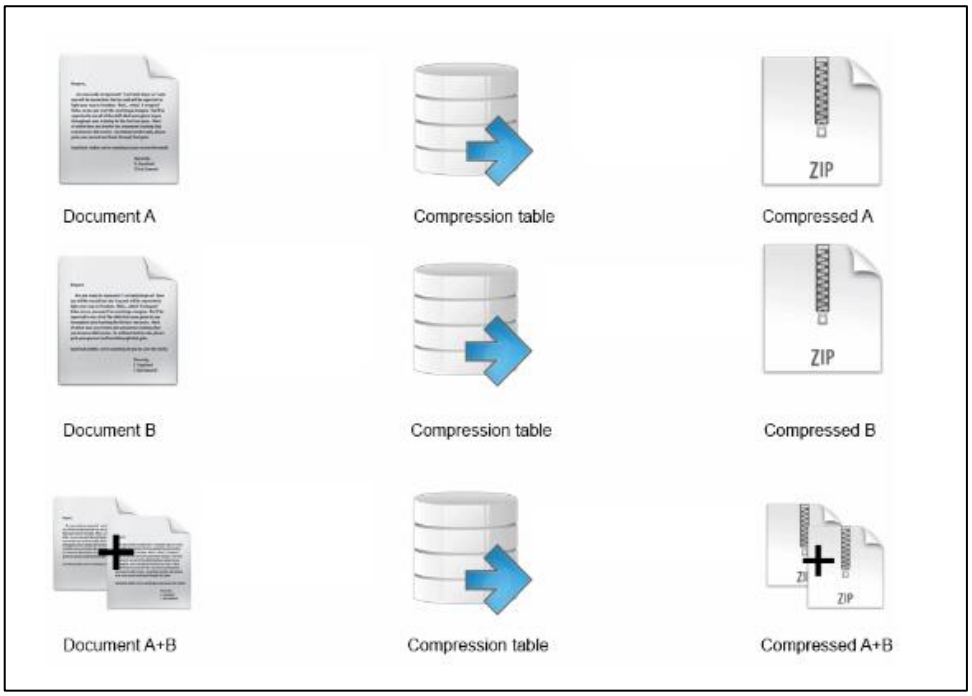

Рис. 1.1. Хід виконання методу

Далі, відповідно до значення NCD та встановлених допустимих меж, що обираються емпірично, визначається ступінь схожості документів. Даний метод не визначає конкретний алгоритм ущільнення, що необхідно застосовувати. Саме тому порівняння самих розповсюджених методів ущільнення розглядається далі у даній дисертації. Також розглядається можливість модифікації наведеного методу з метою покращення результатів за критерієм точності. Задача пошуку плагіату не є задачею реального часу, де важлива швидкість виконання. В цій задачі основним критерієм є точність, але, при цьому, не можна нехтувати критерієм швидкості. Якщо ж точність підвищується при розумному зменшенні швидкості – то це є допустимим.

1.4. Висновки

На основі огляду методів для порівняння текстів на

ідентичність, можливо визначити основний критерій, за яким можна оцінити ефективність даних методів.

Відповідно, при розгляді новітнього методу, а саме методу автоматизованого порівняння текстів на ідентичність з використанням ущільнення даних, було виділено основні нюанси, що лежать у його основі та визначено шляхи модифікації даного методу, що ведуть до підвищення точності.

Для вирішення задачі пошуку дублікатів у текстових документах пропонується модифікований метод автоматизованого порівняння текстів на ідентичність з використанням ущільнення даних, модифікований у частинах:

– попереднього оброблення текстових даних;

– ущільнення даних.

2. МОДИФІКОВАНИЙ МЕТОД АВТОМАТИЗОВАННОГО ПОРІВНЯННЯ ТЕКСТІВ НА ІДЕНТИЧНІСТЬ НА ОСНОВІ УЩІЛЬНЕННЯ ДАНИХ

2.1. Способи попереднього оброблення текстових даних перед виконанням порівняння на ідентичність

Визначимо та опишемо для модифікованого методу автоматизованого порівняння текстів на ідентичність на основі ущільнення даних декілька способів попереднього оброблення текстів перед виконанням основного методу порівняння. Метою такої попереднього оброблення є вичистка тексту від частин, які не впливають позитивно або не використовуються при порівнянні, але займають місце, або, навпаки, дають похибку при роботі методу. На виході процедури оброблення, після послідовного застосування усіх нижчеописаних підходів, ми отримаємо довгий рядок з цифер від одиниці до семірки, що буде показувати вигляд документу після попереднього оброблення і містити лише логічно значущу інформацію.

2.1.1. Нормалізація тексту

Метою даного способу є вилучення несуттєвих для пошуку деталей, які в силу помилок або умисної атаки на алгоритм порівняння можуть бути добавлені у файл. Нормалізація тексту включає в себе різні операції, які можуть застосовуватись або ні, в залежності від різних параметрів, таких як мова тексту, стиль тексту та інші. Далі наведено список деяких операцій, що відносяться до нормалізації тексту.

- вилучення знаків пунктуації;
- вилучення зносок, посилань (наприклад конструкцій типу «[стр. 56]»);
- нормалізація UTF-8 [18];
- приведення тексту до нижнього регістру;

– спрощення складних символів (наприклад, для російської мови, приведення «ё» до «е», «й» до «и» та ін.).

Головна ідея полягає в тому, що після даної процедури схожі тексти залишаться схожими, а несхожі – несхожими. Нюансом цього підходу є те, що швидкодія операцій дуже залежить від розробника, що їх реалізує. При погано реалізованій обробці текстів можуть бути проблеми швидкодії за критеріями часу та пам’яті.

2.1.2. Вилучення коротких слів

В цьому способі попереднього оброблення текстових даних із тексту вилучаються короткі слова. Емпіричним шляхом було виявлено, що для кирилиці короткими словами є слова з 6 літер та менше, а для латиниці – 5 та менше. Як правило, тексти, що відносяться до однієї теми, містять загалом приблизно однакові слова. Причому великі слова будуть знаходитись на різних позиціях. Ми нехтуємо маленькими варіаціями і отримуємо тексти, меншого об’єму, які не втрачають своєї унікальності. При цьому, також, цей спосіб дозволяє підвищити швидкодію за критеріями пам’яті та часу, оскільки основному алгоритму необхідно буде обробляти меншу кількість даних.

2.1.3. Soundex

Для надійної ідентифікації слова нам не потрібні голосні літери. Більш того, для більшості слів нам не є важливими приголосні літери, а важливим є їх класифікація – дзвінка, м’яка, шипляча. Тому метою даного способу є видалення голосних літер та приведення тих, що залишились до групового еквіваленту. Існує аналогічний алгоритм, що називається Soundex і використовується в англійській мові для пошуку слів зі схожим звучанням. Для даної задачі ми використаємо його варіацію (наведені кроки використовують алфавіт англійської мови).

1. Вилучимо з тексту усі голосні літери та літери «h» та «w».
2. Виконаємо заміну букв на цифри, відповідно до таблиці 2.1. У результаті ми отримаємо рядок з цифер 1-6, що розбита пробілами на слова

Таблиця 2.1

Таблиця замін у алгоритмі Soundex

Літери	*Відповідна їм цифра*
b, f, p, v	1
c, g, j, k, q, s, x, z	2
d, t	3
l	4
m, n	5
r	6

3. Для подальшого скорочення, ми в усіх слів, що довше чотирьох знаків (тобто слова з 5 або більше приголосними) видаляємо цифри, починаючи з п’ятої і ставимо їм цифру 7 замість видалених (тобто, цифра 7 у нас буде відмічати особо довгі слова).
4. Видаляємо пробіли, переноси рядків та інше. У результаті отримуємо з тексту довгий рядок, що складається з цифр від 1 до 7.

2.2. Опис та порівняння алгоритмів ущільнення

Так як метод порівняння текстів на ідентичність, що використовує ущільнення та NCD не надає певних вимог або рекомендацій до методу алгоритму ущільнення, логічним було би порівняти декілька

найрозповсюдженіших алгоритмів ущільнення в рамках модифікації обраного методу.

Для порівняння в рамках даного дослідження обрано чотири найрозповсюдженіших алгоритми ущільнення: lzo, deflate, bwt, lzma [19].

Алгоритм lzo створений для швидкого та невимогливого до ресурсів ущільнення, завдяки чому він використовується у високонавантажених системах. Рекомендований рівень ущільнення за замовчуванням – 3.

Алгоритм deflate базується на комбінації алгоритмів lz77 та Хаффмана та лежить в основі роботи утиліти gzip і, де-факто, є основним алгоритмом ущільнення для unix-систем. Рекомендований рівень ущільнення за замовчуванням – 3.

Алгоритм bwt базується на трансформації порядку символів в рядках і, порівнюючи з іншими наведеними у цьому пункті алгоритмами, є досить повільним. Лежить у основі роботи архіватора bzip2. Рекомендований рівень ущільнення за замовчуванням – 9.

В основі алгоритму lzma лежить ущільнення даних за словником, а основною його перевагою є високий коефіцієнт ущільнення, у порівнянні з іншими обраними алгоритмами. Лежить у основі архіватору 7-zip. Рекомендований рівень ущільнення за замовчуванням – 7.

Чисельне значення нормалізованої відстані ущільнення лежить у діапазоні від 0 до 1. Вважається, що чим ближче отримане значення до нуля – тим більш схожими є тексти. Відповідно, основними критеріями порівняння обраних алгоритмів ущільнення буде наближеність значення нормалізованої відстані ущільнення до нуля при схожих текстових документах та наближеність цього ж параметру до одиниці – для різних текстових документів.

Експериментальні дослідження обраних алгоритмів проводились шляхом порівняння даних, отриманих після виконання операцій, передбачених описаним вище методом, за критерієм точності. Для реалізації алгоритмів було використано бібліотеку з відкритими

програмними кодами – SharpCompress [20]. Обраний рівень ущільнення – 5.

Дослідження проводились на декількох наборах документів англійською мовою однакової довжини, побудованих на основі художньої літератури. Дані текстові документи не зазнавали ніякої попереднього оброблення такої як нормалізація, вилучення прийменників і т.п..

Відповідні результати порівняння наведенні у таблиці 2.2.

Таблиця 2.2

Результати проведення тестування

	Алгоритм ущільнення			
Вхідні дані	lzo	deflate	bwt	lzma
AA (25)	0.18	0.17	0.13	0.11
AB (25)	0.40	0.40	0.41	0.44
AA (100)	0.23	0.22	0.15	0.13
AB (100)	0.44	0.47	0.48	0.51
AA (250)	0.26	0.24	0.17	0.16
AB (250)	0.52	0.56	0.57	0.58
AA (500)	0.29	0.27	0.21	0.18
AB (500)	0.57	0.63	0.65	0.64
AA (1000)	0.30	0.31	0.23	0.21
AB (1000)	0.6	0.75	0.84	0.79

Примітки

1. AA значить, що для порівняння були подані однакові документи, відповідно AB – різні (але які відповідали вищеописаним критеріям)
2. Цифра в дужках показує текстову довжину документа в символах (включаючи службові)

Відповідно до отриманих результатів, найкращим за описаними вище критеріями є алгоритм lzma, який і рекомендується використовувати при реалізації методу автоматизованого порівняння текстів на ідентичність з використанням ущільнення даних.

2.3. Підходи до попереднього оброблення текстів сирцевого коду при порівнянні на ідентичність

Вищенаведені способи попереднього оброблення текстів на ідентичність загалом можливо застосовувати до текстів, що відносяться до літературної категорії. Проте, їх використання не є можливим для аналізу сирцевих кодів програм, які, очевидно, що є текстовими даними, на ідентичність. Це пов’язано з їх специфікою, а саме те що сирцеві коди складаються з змінних та операторів, що легко обробляються комп’ютером. Тобто, вони не представляють собою літературний текст, в класичному розумінні. Саме тому, використання вищевказаних методів попереднього оброблення не є можливим. Саме тому було проведено додаткове дослідження, під час якого була помічена проблема з ускладненням порівняння сирцевих кодів на ідентичність через використання «синтаксичного цукру».

Під «синтаксичним цукром» розуміється будь-який елемент синтаксису мови програмування, що дублює існуючий аналог, але є більш зручним у використанні, або більш коротким при записуванні, або виглядає більш природнім чи більш зрозумілим при читанні програмного коду. Таким чином, «синтаксичний цукор» призначений лише для того, щоб зробити мову програмування більш зручною для програміста.

В загалому, існуючі методи пошуку плагіату в програмному коді прийнято поділяти на три великі групи:

- текстові методи;
- структурні методи;

– семантичні методи.

Текстові методи пошуку плагіату в програмному коді базуються на розгляданні коду як послідовності токенів – символів, що представляють собою оператор або групу операторів мови програмування [21]. При цьому параметри цих операторів повністю ігноруються.

Токенізація (процес перетворення програмного коду на послідовність токенів) методами цієї групи виконується наступним чином:

1. Кожному оператору мови програмування, який не є операндом, приписується цифровий код, що раніше був обраний для відповідного класу операторів. Також коди можна приписувати блоковим операторам (в мові програмування C#, наприклад, це фігурні дужки) та підключенням бібліотек (для C# – просторів імен).
2. Будується рядок з отриманих кодів, зберігаючи порядок їх слідування відповідно до порядку в програмному коді.

На основі результатів аналізу цієї процедури можна зробити висновок, що дані методи пошуку плагіату дозволяють ефективно працювати з фрагментами коду, що належать до перших двох типів розглянутої класифікації. Разом з тим слід зазначити, що текстові методи були першими методами пошуку плагіату в програмному коді і на сьогодні до них належать найбільш сучасні та ефективні методи [21].

Методи, що базуються на дослідженні структури програми (яка часто представляється ними у вигляді графу потоку керування або абстрактного синтаксичного дерева), відповідно отримали назву структурних методів [22]. Методи даної групи також є ефективними для оброблення фрагментів коду перших двох типів розглянутої класифікації і непогано працюють з кодом, що відноситься до третього типу. Головним недоліком методів пошуку плагіату в програмному коді цієї групи є їхня обчислювальна складність і потреба в досить великій кількості додаткової

пам’яті для зберігання відповідного подання програмного коду (графу потоку керування або абстрактного синтаксичного дерева).

Семантичні методи є подібними до структурних методів, проте в їх основі лежить знання семантики операторів [23]. Наприклад, методами цієї групи може використовуватись подання програмного коду у вигляді особливого семантичного графу, що має вершини двох типів. Вершини одного типу будуються на основі послідовностей операторів, що характеризуються певною семантикою (наприклад, це можуть бути арифметичні операції, умовні оператори, оператори циклу тощо). Вершини іншого типу задають відношення, в якому перебувають сусідні з ними вершини – таким чином з’єднуючи пари вершин першого типу.

За своїм визначенням найбільш стійкими до заміни операторів на «синтаксичний цукор» для маскування плагіату є структурні та семантичні методи. Проте ці методи, як зазначено, є обчислювально складними і часто потребують залучення великої кількості додаткової пам’яті. Текстові методи є більш привабливими з точки зору обчислювальної складності та необхідної пам’яті, проте є зовсім нестійкими до маскування плагіату за допомогою «синтаксичного цукру».

Саме тому, вирішенням проблеми неможливості виявлення плагіату в програмному коді, що містить «синтаксичний цукор», текстовими методами може стати розроблення та практичне впровадження підходів до попереднього аналізу програмного коду на наявність в ньому «синтаксичного цукру» та подальшої нейтралізації впливу останнього на результати процедури пошуку плагіату.

Підходи до оброблення «синтаксичного цукру» при пошуку плагіату в програмному коді. Принциповими ознаками «синтаксичного цукру» є те, що:

– все, що може бути написаним за допомогою використання «синтаксичного цукру», може бути написаним без нього цією ж мовою програмування;

– заміна частини програмного коду на «синтаксичний цукор» не змінює хід виконання програми.

Також слід зазначити, що конструкції «синтаксичного цукру» є попередньо відомими і вони чітко визначаються у коді.

На цих ознаках базуються два наступні запропоновані підходи до оброблення «синтаксичного цукру». Обидва підходи передбачають певні дії на етапі попереднього оброблення програмного коду перед виконанням будь-якого методу пошуку плагіату у програмному коді і розглядають останній як звичайний текст.

Reduce. Основна ідея цього підходу полягає в заміні базових конструкцій мови програмування, що мають еквіваленти у вигляді «синтаксичного цукру», на «синтаксичний цукор».

Map. Основна ідея цього підходу є протилежною до Reduce-підходу, і полягає в заміні операторів «синтаксичного цукру» на їх еквіваленти, що використовують базові конструкції мови програмування.

Свої назви дані підходи отримали відповідно до результату їх застосування:

– у зв’язку з тим, що в більшості випадків код з використанням «синтаксичного цукру» є більш коротким, підхід, що передбачає заміну базових конструкцій мови програмування на «синтаксичний цукор», начебто «стискає» програмний код – звідси походить назва Reduce;
– протилежний підхід в більшості випадків збільшує обсяг програмного коду, «відображаючи» «синтаксичний цукор» на базові синтаксичні конструкції мови програмування – звідси походить назва Map.

Заміна базових конструкцій мови програмування на «синтаксичний цукор» та навпаки може бути реалізована по-різному. Наприклад, можливе використання регулярних виразів для пошуку та заміни конструкцій. Також очевидним є те, що дані підходи потребують попередньо створеного

списку «синтаксичного цукру» конкретної мови програмування та його еквівалентів, записаних базовими конструкціями цієї ж мови.

Результатами реалізації даних підходів є те, що всі оператори програми приводяться до одного виду – або з використанням «синтаксичного цукру», або без. Отже, у випадку, якщо плагіат програмного коду було замасковано за допомогою «синтаксичного цукру», після оброблення програмного коду буде отримано його початкову форму і плагіат буде знайдено при подальшому виконанні будь-якого методу пошуку плагіату.

Загальний алгоритм використання запропонованих підходів оброблення «синтаксичного цукру» при пошуку плагіату програмного коду буде мати такий вигляд:

1. Читання сирцевого коду, який потрібно перевірити на наявність в ньому плагіату
2. Читання коду, який є потенційним джерелом плагіату
3. Попереднє перетворення обох кодів відповідно до одного з запропонованих підходів (Map або Reduce) до оброблення «синтаксичного цукру» в коді програмного забезпечення.
4. Виконання основного класичного методу пошуку плагіату в програмному коді

2.4. Модифікований метод порівняння текстів на ідентичність

Описаний нижче метод порівняння текстів на ідентичність базується на використанні попереднього оброблення текстів та ущільненні алгоритмом, що дає найкращу ефективність для даної задачі. Суть методу полягає у тому, щоб почистити текст від «сміттєвих» даних, що не допомагають при порівнянні текстів на ідентичність та формуванні більш явного паттерна документу.

Модифікований метод порівняння текстів на ідентичність складається з таких етапів:

1. Попередня обробка текстових документів, що включає в себе наступні способи: нормалізація тексту, вилучення коротких слів, soundex;
2. Зчитування оброблених в попередньому текстів у бінарному вигляді та їх конкатенація;
3. Застосування обраного алгоритму ущільнення – lzma на дані з попереднього пункту (тексти у бінарному вигляді та їх конкатенацію);
4. Вирахування нормалізованої відстані ущільнення за формулою з оригінального алгоритму (1.1).
5. Визначення степені схожості текстових документів відповідно до емпірично встановлених границь та значення нормалізованої відстані ущільнення.

2.5. Висновки

У даному розділі розглянуто способи попереднього оброблення текстових документів та порівняння алгоритмів ущільнення для подальшого застосування у методі автоматизованого порівняння текстів на ідентичність на основі ущільнення даних.

Запропонований метод автоматизованого порівняння текстів на ідентичність на основі ущільнення даних складається з таких етапів: попереднє оброблення текстових даних, до якого відносяться наступні способи: нормалізація тексту, вилучення коротких слів, soundex; зчитування у бінарному вигляді оброблених на попередньому кроці текстів та їх конкатенація; ущільнення цих бінарних даних алгоритмом lzma; вирахування нормалізованої відстані ущільнення та оцінка степені

схожості текстових документів емпірично встановлених границь та значення нормалізованої відстані ущільнення.

Розроблений метод автоматизованого порівняння текстів на ідентичність може використовуватись в різних галузях, де необхідно порівнювати текстові документи на ідентичність.

3. ОПИС РОЗРОБЛЕННОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО РЕАЛІЗУЄ МОДИФІКОВАНИЙ МЕТОД

3.1. Основні вимоги до програмного забезпечення

Сформулюємо та опишемо основні вимоги до програмного забезпечення для автоматизованого порівняння текстів на ідентичність, що базується на ущільненні. Система, що розроблюється, повинна забезпечувати такі функціональні можливості, як:

- підключення модулів, що забезпечують попередню обробку тексту;
- підключення модулів, що реалізують ущільнення тексту певними алгоритмами;
- автоматизоване порівняння текстів на ідентичність за допомогою методу, що базується на ущільненні та NCD;
- можливість роботи з текстовими документами на англійській мові.

Також, крім функціональних вимог до системи, наявні такі нефункціональні вимоги:

- компоненти з бізнес-логікою методу мають працювати на будь-якій ОС, на якій встановлений фреймворк, сумісний до стандарту .NET Standart 2.0;
- система має реалізовувати паралельну обробку та порівняння текстових документів;
- система має бути протестована за допомогою unit та integration tests.

Відповідно, програмне забезпечення для автоматизованого порівняння текстів на ідентичність, що базується на ущільненні буде реалізовано у вигляді dynamic link library, написане під .NET Standart 2.0. Також, для демонстрації буде розроблена host-система у вигляді

консольного застосування, що буде використовуватись для взаємодії з користувачем.

Діаграма варіантів використання розробленого програмного забезпечення наведена на рис 3.1.

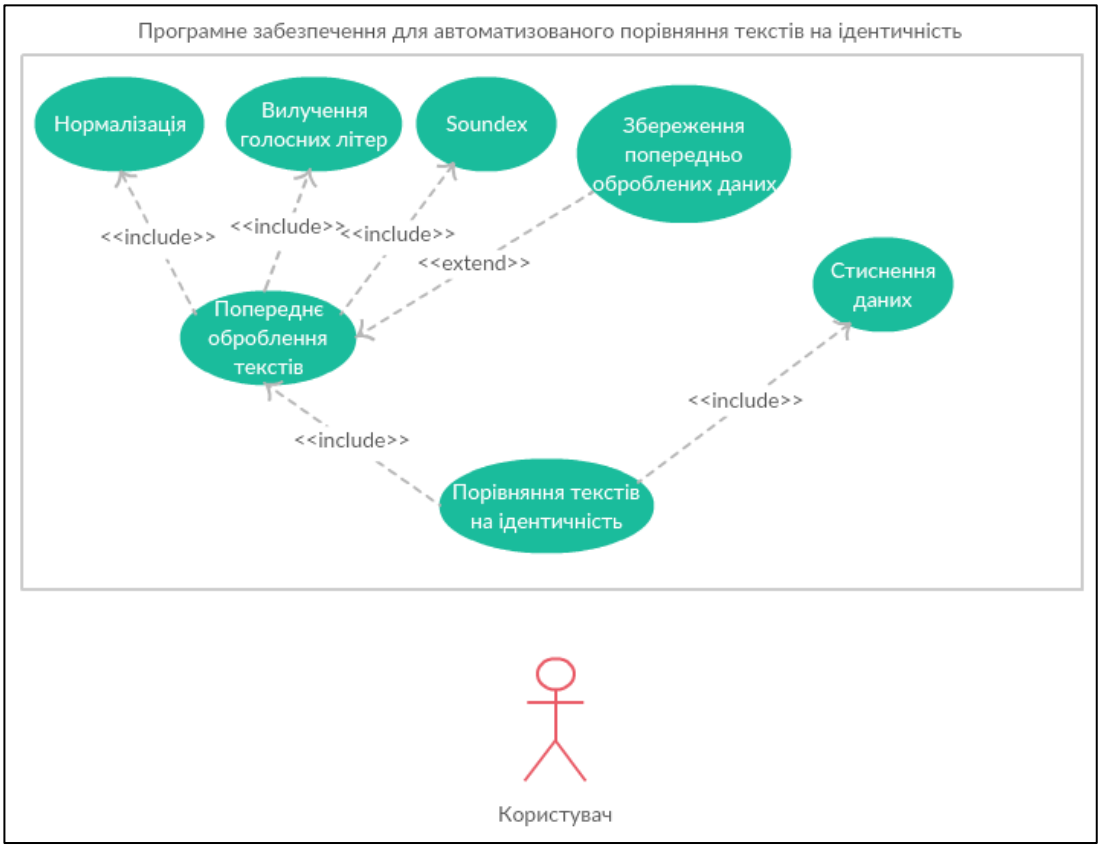

Рис. 3.1. Діаграма варіантів використання програмного забезпечення для автоматизованого порівняння текстів на ідентичність, що базується на ущільненні

3.2. Опис обраних засобів розроблення програмного забезпечення

Програмне забезпечення для автоматизованого порівняння текстів на ідентичність розроблено на для платформ, сумісних з .NET Standart з використанням мови програмування C#. В якості середовища розроблення використання Visual Studio 2017, а в якості системи керування версіями – git.

Коротко розглянемо обрані засоби.

3.2.1. .NET Standart та платформа .NET

.NET Standart являє собою формальну специфікацію .NET-інтерфейсів, які мають бути доступними для всіх реалізацій .NET. Ідея .NET Standart - це встановлення більшої однорідності в екосистемі .NET. [24]

.NET – це програмна платформа, випущена компанією Microsoft. Головними складовими даної платформи є загальномовне середовище виконання (CLR) та бібліотека класів .NET Framework (FCL).

Як зрозуміло з назви, головною задачею CLR є виконання та управління кодом під час виконання (такий код називається «керованим»), проте вона також виконує інші функції, такі як керування пам’яттю, завантаження коду, оброблення виключень, керування типізацією та безпекою [25]. Програма для .NET Framework, написана на будь-якій підтримуваній мові програмування, спочатку перекладається компілятором в єдиний для .NET проміжний байт-код Common Intermediate Language (CIL) (раніше називався Microsoft Intermediate Language, MSIL). У термінах .NET результат називається збіркою. Потім код або виповнюється віртуальною машиною CLR, або транслюється утилітою NGen.exe в виконуваний код для конкретного цільового процесора. Проте використання віртуальної машини є більш бажаним, а можливість компіляції для конкретного процесору застосовується не так часто. У разі використання віртуальної машини CLR вбудований в неї JIT-компілятор «на льоту» перетворює проміжний байт-код в машинні коди для потрібного процесора [25].

Бібліотека Framework Class Library містить об’єктні класи .NET, які є доступними для всіх підтримуваних мов програмування. Базові типи знаходяться в ядрі FCL та носять назву Base Class Library (BCL). Крім них, FCL містить класи для створення застосунків з використанням технологій

Windows Forms, WPF, ASP.NET, WCF, Language Integrated Query та інші [26].

Платформа .NET має наступні переваги:

– використання компонентно-орієнтованого підходу, що значно підвищує роздільність коду та його незалежне використання;
– незалежність від мови програмування – різні частини програмного забезпечення можуть бути написаними на різних підтримуваних мовах (C#, F#, Managed C++ та інші);
– велика кількість створених спільнотою пакетів, що надають різні функціональні можливості;
– простий процес інсталяції розробленого програмного забезпечення для кінцевого користувача;
– наявність Language Integrated Query (LINQ) – мови запитів, подібної до SQL, що значно спрощує маніпуляції з даними;
– наявність бібліотеки паралельних задач (Task Parallel Library, TPL), що значно спрощує процес додавання паралелізму в програмні застосунки.

3.2.2. Мова програмування C#

C# – об’єктно-орієнтована мова програмування високого рівня, що була розроблена для використання разом з платформою .NET. На сьогодні C# є однією з найбільш поширених мов для розроблення прикладних застосунків [27].

Синтаксис даної мови є Сі-подібним і найбільш близьким до таких мов, як C++ та Java. Як будь-яка об’єктно-орієнтована мова, C# підтримує поняття поліморфізму, інкапсуляції та наслідування, а також має строгу статичну типізацію. Перейнявши багато що від своїх попередників – мов C++, Delphi, Modula і Smalltalk – C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем [27].

Дану мову обрано для розроблення програмного забезпечення в рамках даної роботи завдяки наступним перевагам:

- строга статична типізація значно зменшує кількість помилок та спрощує їх виявлення ще на етапі написання коду;
- підтримка LINQ;
- наявність таких конструкцій мови, як властивості, перевантаження операторів, атрибути;
- наявність делегатів – строго типізованих та безпечних «посилань» на функції, використання яких значно спрощу реалізацію реакцій на події;
- можливість ведення документації разом з кодом у вигляді XML коментарів, яка буде доступна всім при використанні документованого коду.

При розробленні використано останню версію мови – C# 7.0 – котра надає такі додаткові можливості, як null-умовні оператори, ініціалізатори автоматичних властивостей, інтерполяція рядків, імпорт статичних функцій та інші.

3.2.3. Visual Studio 2017

Microsoft Visual Studio – серія продуктів компанії Microsoft, яка де-факто є стандартом для розроблення програмних застосунків на мові C# і платформі .NET. Visual Studio – це інтегроване середовище розробки, що також включає в себе ряд інших інструментальних засобів, з можливістю їх розширення за допомогою доповнень (Add-Ins). Дана IDE дозволяє розроблювати консольні застосунки, застосунки з графічним інтерфейсом (за допомогою технологій Windows Forms та WPF), веб-сайти, веб-застосунки, веб-сервіси і інші.

Visual Studio включає в себе редактор програмного коду, що підтримує мови платформи .NET та інші, з підтримкою технології

IntelliSense і можливістю рефакторингу коду. Крім цього, доступні такі інструменти, як:

- вбудований відладчик Visual Studio Debugger, що може працювати як на рівні програмного коду, так і на рівні машинних команд;
- редактор форм для розроблення застосунків з графічним інтерфейсом за допомогою технологій Windows Forms та WPF;
- веб-редактор для розроблення веб-сайтів з застосуванням технологій ASP.NET, ASP.NET MVC;
- дизайнер UML, що підтримує генерацію коду (наприклад класів), з розроблених моделей;
- дизайнер баз даних, з підтримкою підходів Model First (генерація БД з моделі) та Database First (генерація моделі з БД).

Visual Studio 2017 – це остання на сьогодні версія Visual Studio, яка була представлена 7 березня 2017 року [28]. Ця версія містила наступні нові можливості та суттєві зміни [28]:

- підтримка .NET Framework 4.7.1;
- підтримка багатьох цільових платформ – крім можливості розроблення застосунків для ОС Windows була додана можливість розроблення мобільних застосунків для ОС iOS та Android (за допомогою технологій Xamarin або Apache Cordova);
- підтримка фреймворку Unity для розроблення багатоплатформних ігор;
- підтримка Universal Windows Platform – універсальної платформи Windows, що дозволяє розроблювати застосунки для будь-яких пристроїв під керуванням ОС Windows 10.

3.2.4. git

git – це розподілена система керування версіями, перша версія якої була розроблена Лінусом Торвальдсом і випущена 7 квітня 2005 року [29].

Ядро git є набором утиліт командного рядка з параметрами. Всі налаштування зберігаються в текстових файлах конфігурації. Така реалізація дає можливість легко інтегрувати git в інші системи (зокрема, створювати графічні git-клієнти з будь-яким бажаним інтерфейсом) [29].

git, на відміну від Subversion і подібних до неї систем, не зберігає інформацію як список змін (патчів) для файлів. Замість цього git зберігає дані набором зліпків. Кожного разу при фіксації поточної версії проекту git зберігає зліпок того, як виглядають всі файли проекту. Але якщо файл не змінювався, то дається посилання на раніше збережений файл. Вся база даних git зберігається в теці з назвою .git. В git файли можуть знаходитися в одному із 3-х станів: зафіксованому (файл вже збережено в локальній базі даних), зміненому (файл було змінено, але зміни не зафіксовано) і підготовленому (файли було змінено і відмічено для фіксації) [29].

Більшість дій можна виконувати на локальній файловій системі без використання інтернет підключення. Вся історія змін зберігається локально і при необхідності вивантажується у віддалений репозиторій – такий підхід характерний для будь-яких розподілених систем керування версіями [29].

3.3. Опис розробленого програмного забезпечення

В рамках даної роботи розроблене наступне програмне забезпечення:

- система, що реалізує модифікований метод для автоматизованого порівняння текстів на ідентичність, що базується на ущільненні;
- консольний застосунок-хост для автоматизованого порівняння текстів на ідентичність.

Розглянемо кожне розроблене програмне забезпечення окремо.

3.3.1. Система, що реалізує модифікований метод для автоматизованого порівняння текстів на ідентичність, що базується на ущільненні

Відповідно до описаного у розділі 2 модифікованого методу для автоматизованого порівняння текстів на ідентичність, було спроектовано архітектуру системи для реалізації даного методу. Відповідно до шаблона bounded context були виділені основні логічні частини, що були реалізовані у вигляді окремих бібліотек, які можуть бути підключені за необхідністю. Bounded context – це реалізація принципу слабої зв'язності на більш високому рівні – на рівні підсистем (модулів). Така сегментація дозволяє більш гнучко управляти розробкою, аж до виключення або перепису з нуля цільних модулів, можливо, що навіть із повною заміною команди та/або технологічного модуля стека [30].

Архітектура розробленої системи зображена на рис. 3.2. Її можна розділити на логічні модулі, опис яких наведений нижче.

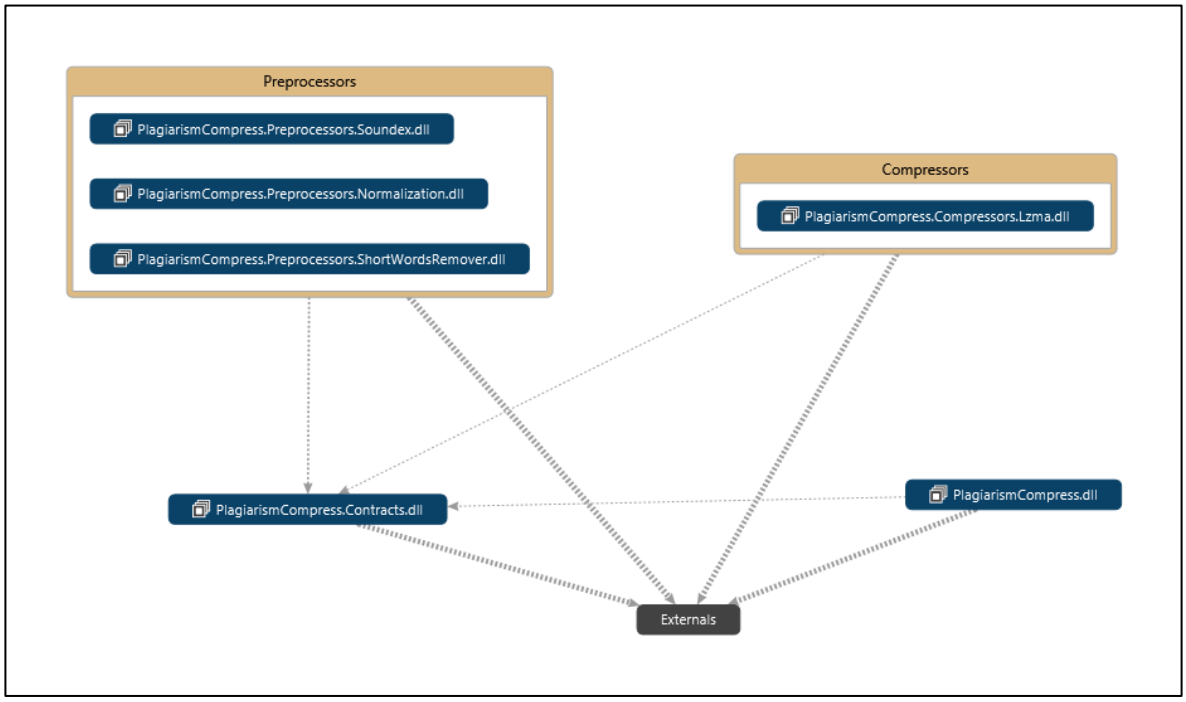

Рис. 3.2. Високорівнева архітектура системи PlagiarismCompress

Відповідно система складається з наступних модулів-бібліотек:

– PlagiarismCompress.dll – бібліотека, що реалізує бізнес-логіку методу для автоматизованого порівняння текстів на ідентичність, що базується на ущільненні;
– PlagiarismCompress.Contracts.dll – бібліотека, що містить контракти (програмні інтерфейси) для бібліотек, які будуть реалізовувати конкретні методи попереднього оброблення текстів та ущільнення;
– PlagiarismCompress.Preprocessors.Soundex – бібліотека, що містить реалізацію способу попереднього оброблення текстового документу шляхом застосування варіації алгоритму soundex;
– PlagiarismCompress.Preprocessors.Normalization – бібліотека, що містить реалізацію способу попереднього оброблення текстового документу шляхом нормалізації;
– PlagiarismCompress.Preprocessors.ShortWordsRemover – бібліотека, що містить реалізацію способу попереднього оброблення текстового документу шляхом вилучення коротких слів;
– PlagiarismCompress.Compressors.Lzma – бібліотека, що містить реалізацію ущільнення шляхом lzma.

Відповідно, бібліотека з бізнес-логікою працює з відповідними програмними інтерфейсами – контрактами, що дозволяє забезпечити низьку зв’язність. Також подібна архітектура дозволяє просто розширювати попередні обробники та методи ущільнення – створюючи нові модулі та реєструючи їх в основній бібліотеці в черзі обробки.

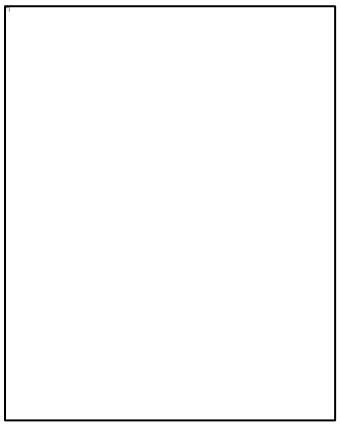

Рис. 3.3. Публічні методи PlagiarismCompress

Точкою входу для використання бібліотеки є клас PlagiarismCompress, що міститься в однойменній бібліотеці. Він містить методи для додавання та вилучення модулів попередньої обробки тексту та метод для реєстрації модуля, що реалізує ущільнення. Ну і головним робочим методом є метод Compare, що приймає на вхід шляхи до файлів текстових документів і повертає значення нормалізованої відстані ущільнення.

Відповідно, модулі, що організують попередню обробку текстів та їх ущільнення організовані у ієрархію зі спільним предком у вигляді інтерфейсів, що містяться у бібліотеці PlagiarismCompress.Contracts.dll. Така їх організація дозволяє використовувати шаблон проектування «Стратегія».

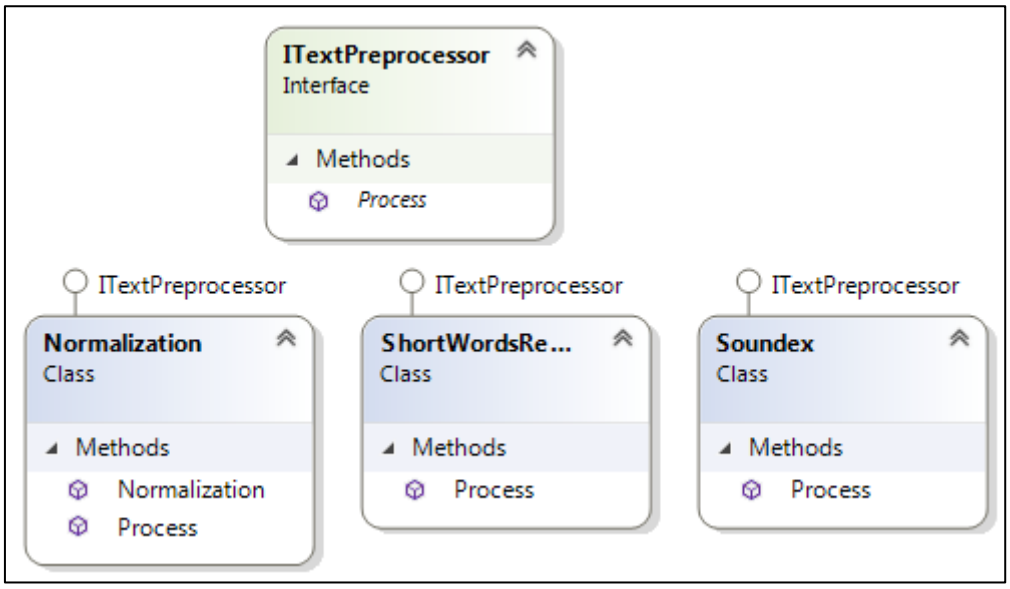

Рис. 3.4. Організація модулів попереднього оброблення текстових документів.

Аналогічним чином реалізована робота з модулем ущільнення текстових документів.

Важливими передумовами для роботи системи є наявність одного зареєстрованого модуля ущільнення (цей модуль є важливою частиною системи, тому при його відсутності буде отримана помилка) та нуля чи більше модулів попереднього оброблення текстових документів. Без зареєстрованих модулів система буде реалізовувати базовий метод для автоматизованого порівняння текстів на ідентичність, що базується на ущільненні. Модулі попереднього оброблення текстових документів відпрацьовують у порядку реєстрації в системі.

Розглянемо приклад організації модуля нормалізації, що складається з різних кроків. Загальна структура організації кроків та класу-фабрики, що надає доступ до них наведена на рисунку 3.5.

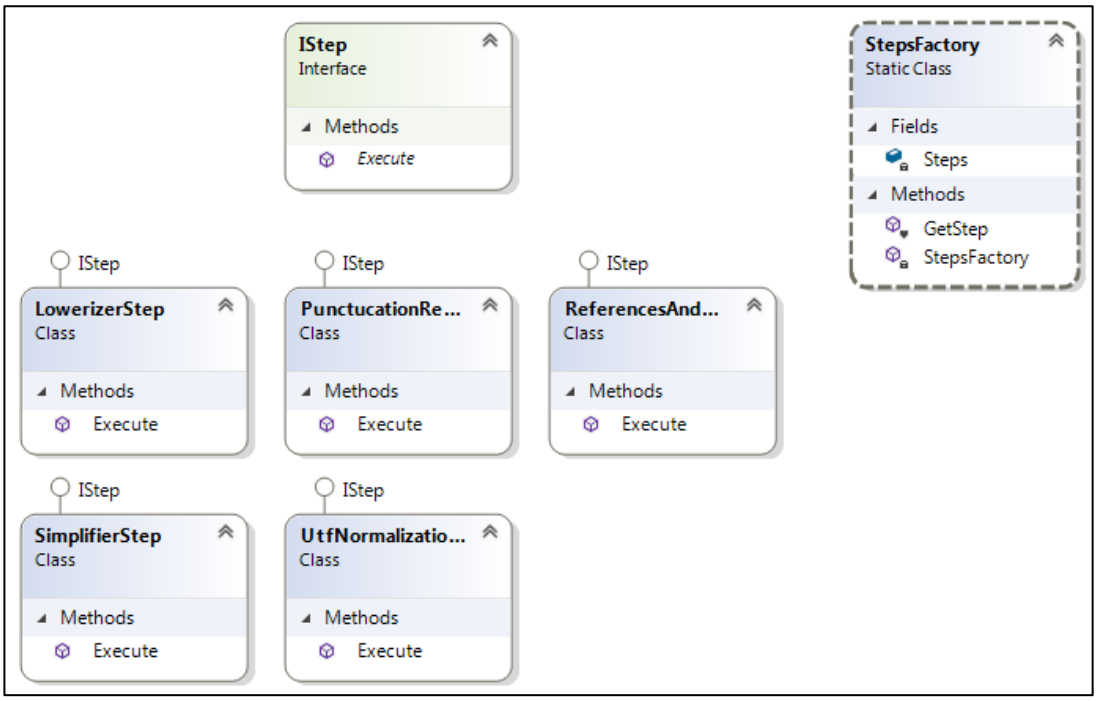

Рис. 3.5. Загальна структура організації кроків

Так як концептуально зазначено, що модулі є закритими логічними частинами, то кроки та фабрика, що надає до них доступ, реалізовані у вигляді internal-класів (в мові c# цей модифікатор вказує, що доступ до об'єктів, що помічені даним словом, є можливим тільки зі цієї збірки). Відповідно, усі кроки, які містяться в модулі нормалізації, реалізують інтерфейс IStep, що містить метод Execute. Доступ до них надається за допомогою фабрики, що містить метод GetStep, який приймає строкове значення та видає конкретний клас, що прив'язаний до даного значення і реалізує крок. Концептуально розуміється, що дані значення є описаними та будуть передані в конфігурації.

Отже, наведемо приклад використання розробленої бібліотеки для автоматизованого порівняння текстів на ідентичність, що базується на ущільненні PlagiarismCompress – див. лістинг 3.1.

Лістинг 3.1. Приклад використання розробленої бібліотеки

```
using PlagiarismCompress;
using PlagiarismCompress.Compressors.Lzma;
using PlagiarismCompress.Preprocessors.Normalization;
using PlagiarismCompress.Preprocessors.ShortWordsRemover;
using PlagiarismCompress.Preprocessors.Soundex;
```

```
var lzmaConfiguration = new LzmaConfiguration
{
// Зазначаємо деякі коректні налаштування для роботи модулю ущільнення
};

var compressor = new Lzma();

var normalizationConfiguration = new INormalizationConfiguration
{
      // Зазначаємо деякі коректні налаштування для роботи модулю Нормалізації
};

var normalization new Normalization(normalizationConfiguration);
var shortWordsRemover = new ShortWordsRemover();
var soundex = new Soundex();

var pl = new PlagiarismCompress();
pl. UseCompressor(compressor);
pl. AddPreprocessor(normalization);
pl. AddPreprocessor(shortWordsRemover);
pl. AddPreprocessor(soundex);

var result = pl.Compare("PATH_1","PATH_2");
```

Unit та integration тести бібліотеки створені за допомогою NUnit (фреймворк для тестування, що є більш продуктивним, ніж MSTest – фреймворк для тестування, що встановлюється за замовчуванням разом з Visual Studio та має більш широку функціональність).

Відповідно кожен з модулей має власні unit тести, що тестують функціональність конкретного модуля. І основна бібліотека містить integration тести, що тестують взаємодію модулей.

3.3.2. Консольний застосунок для автоматизованого порівняння текстів на ідентичність

В якості кінцевого застосунку для автоматизованого порівняння текстів на ідентичність розроблено застосунок з інтерфейсом командного

рядка. Такий варіант реалізації користувацького інтерфейсу дозволяє легко підтримувати пакетну обробку текстових документів, а також повну автоматизацію виконання порівняння текстів на ідентичність. Наприклад, консольний застосунок легко викликати з різних автоматичних скриптів, передаючи різні налаштування, що значно спростило подальше тестування розроблених методів та підходів.

Сама структура консольного застосунку є досить простою, оскільки вся функціональність, що реалізує бізнес-логіку порівняння текстів на ідентичність, виділена в окремі бібліотеки. Таким чином, відповідальністю консольного застосунку є лише створення екземплярів підключаємих модулів, їх реєстрація у основній бібліотеці, зчитування даних та вивід результатів.

Застосунок приймає налаштування у вигляді параметрів командого рядка при своєму запуску. На поточний момент параметрами є шляхи до файлів, які необхідно порівняти на ідентичність.

3.3.2. Службові скрипти для збірки проекту

Так як проект написаний під .NET Standart 2.0, то можливість його зборки під різні платформи закладалась з самого початку. Тому, для спрощення процесу зборки проекту були написані службові скрипти з використанням мови PowerShell.

PowerShell – це інструмент з відкритим сирцевим кодом від Microsoft, що складається з консольного застосунку та мови автоматизації. Цей інструмент за своїми функціональними можливостями не поступається bash у Unix-системах та є основним при використанні у Windows-системах.

Розроблені скрипти можна використовувати при процесі CI/CD та запускати для запуску проекту автоматично чи за розкладом. Відповідно, використовуючи аналогічні команди, можна написати аналогічні скрипти для unix-платформ.

Процес побудови складається з трьох кроків, кожен з яких реалізований відповідним скриптом

– dotnet restore – цей крок і відповідна команда дозволяють відновити пакети залежностей проекту з системи керування пакетами NuGet. Результат виконання даної команди наведений на рис. 3.6.

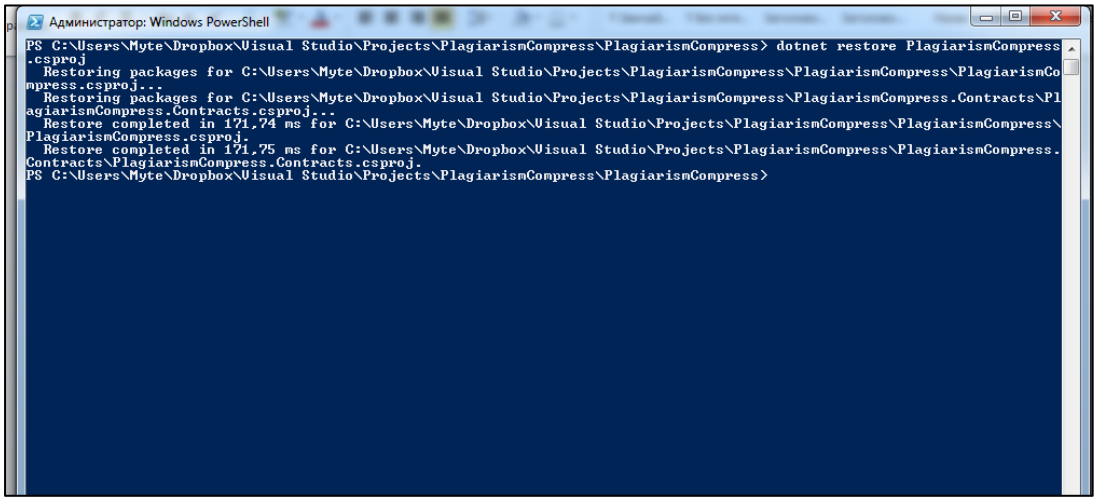

Рис. 3.6. Результат виконання команди dotnet restore

– dotnet build – цей крок і відповідна команда створені для будівлі проекту. Вона будує проект та його залежності в набір бінарних файлів. На виходу будуть файли з розширенням dll, що містять IL-код проекту, файли pdb, що необхідні для відладки проекту та файл .deps.json, у якому указані залежності проекту. Результат виконання даної команди наведений на рис 3.7.

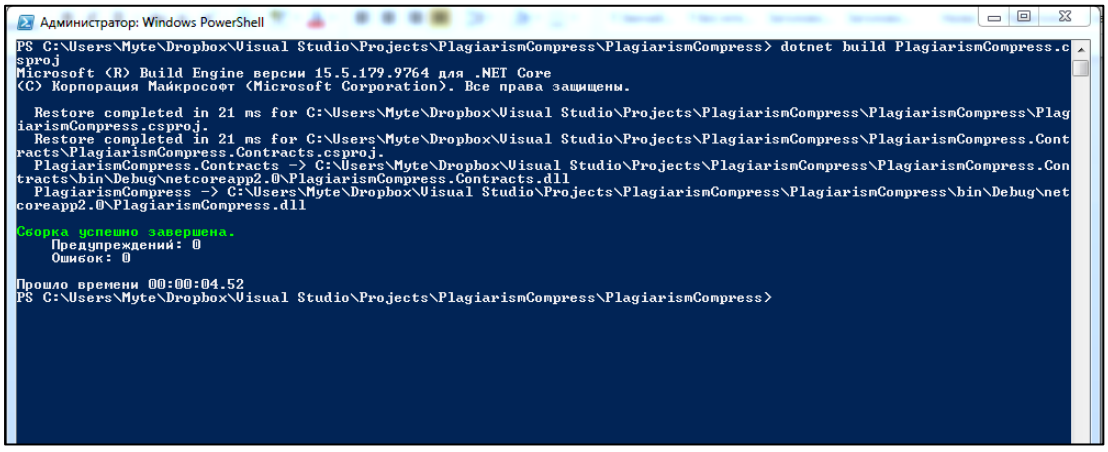

Рис. 3.7. Результат виконання команди dotnet restore

– dotnet publish – цей крок і відповідна команда використовуються для пакування застосування та усіх його залежностей в папку для розгортання на кінцеве оточення. Результат виконання цієї команди наведений на рис 3.8.

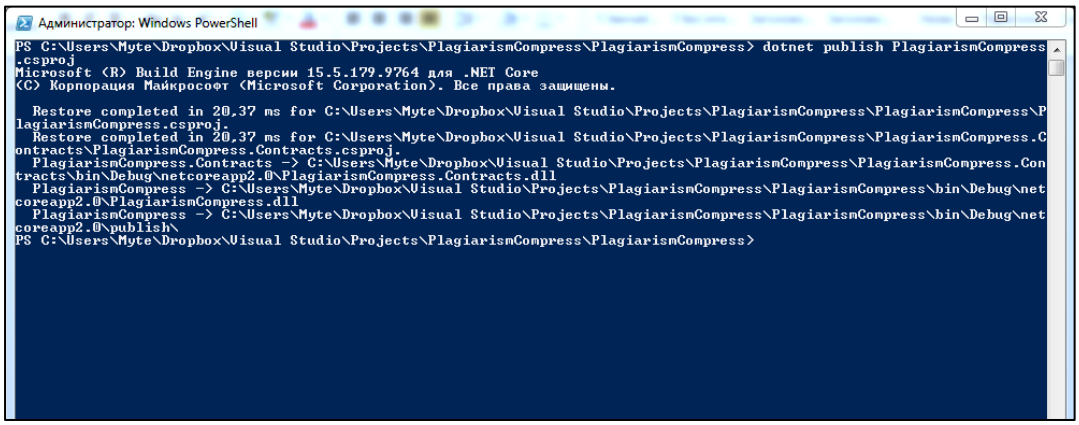

Рис. 3.8. Результат виконання команди dotnet publish

Оскільки на вищенаведених прикладах явно не вказувалася цільова платформа, то процес білду відбувався під платформу windows. Список файлів, що знаходяться в папці publish після виконання усіх команд наведений на рис. 3.9.

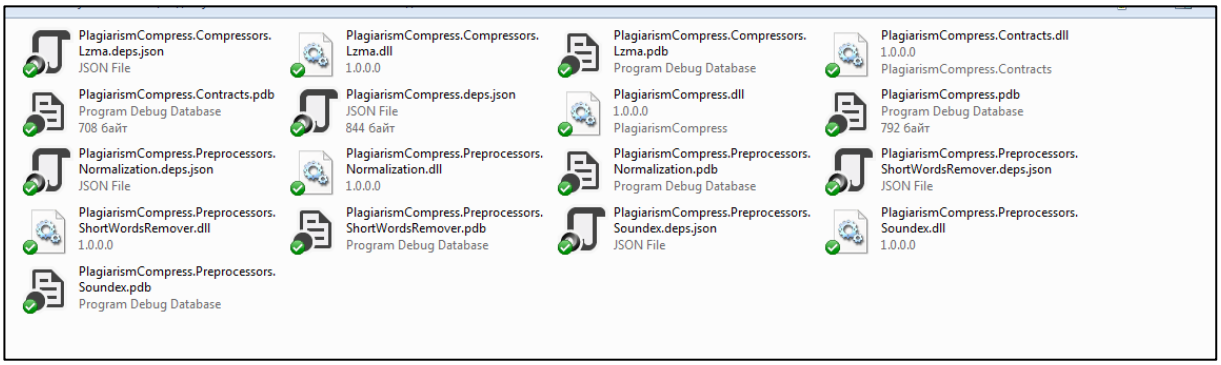

Рис. 3.9. Список файлів, що лежать в кінцевій папці після виконання процесу побудови.

Відповідно, використання скриптів є доволі гнучким підходом. Наведені скрипти можна більш гнучко сконфігурувати чи розширити, використовуючи можливості мови PowerShell та функціональність, що надається засобами dotnet. Дані скрипти зроблені для використання на build-серверах з ОС Windows, але легко можуть бути портовані для Unix-систем.

3.4. Висновки

В рамках даної роботи розроблено програмне забезпечення, яке призначене для порівняння текстів на ідентичність та реалізує модифікований метод автоматизованого порівняння текстів на ідентичність з використанням ущільнення даних.

Можливо зробити наступні висновки про розроблене програмне забезпечення в цілому:

- розроблене програмне забезпечення можливо використовувати для автоматизованого порівняння текстів на ідентичність;
- розроблений консольний за стосунок надає можливості роботи з розробленими бібліотеками, що реалізовують бізнес-логіку;
- архітектура розробленого програмного забезпечення дозволяє легко додавати нові модулі, що реалізують попереднє оброблення текстових документів та ущільнення за певними алгоритмами завдяки використанню таких шаблонів проектування як «стратегія» та «boundary context» з domain driven development;
- завдяки реалізації бізнес-логіки методу у вигляді окремих бібліотек, що сумісні з стандартом .NET Standart 2.0 розроблене програмне забезпечення не залежить від інтерфейсу користувача і може використовуватись з будь-яким інтерфейсом (консольним, графічним, веб і т.д.) або бути інтегрованим до існуючих програмних комплексів та систем та бути запущеній на будь-якій платформі, що містить компоненти .NET, що сумісні зі стандартом .NET Standart 2.0 (наприклад, на Windows з .NET Framework 4.7+, на Unix системах, що містять .NET Core 2.0+);

– завдяки розробленим білд-скриптам, процес побудови проекту є досить простим та тривіальним. Розроблені скрипти можна використовувати як і при ручній побудові проекту, так і у процесі CI/CD.

4. АНАЛІЗ ЕФЕКТИВНОСТІ МОДИФІКОВАНОГО МЕТОДУ

4.1. Критерії оцінки ефективності

Загалом, основними критеріями при оцінці методів автоматизованого порівняння текстів на ідентичність є критерії точності, часу та пам’яті.

Якщо з критеріями часу та пам’яті все є досить прозорим, то критерій точності для цієї задачі є спеціалізованим. В методі автоматизованого порівняння текстових документів на ідентичність, що базується на ущільненні, результуючим показником є значення нормалізованої відстані ущільнення. Чисельне значення нормалізованої відстані ущільнення лежить у діапазоні від 0 до 1. Вважається, що чим ближче отримане значення до нуля – тим більш схожими є тексти. Відповідно, критерієм точності для оцінки та порівняння обраних методів автоматизованого порівняння текстів на ідентичність буде наближеність значення нормалізованої відстані ущільнення до нуля при схожих текстових документах та наближеність цього ж параметру до одиниці – для різних текстових документів.

Метою дослідження було підвищення точності методу автоматизованого порівняння текстів на ідентичність, тому критерій точності буде основним, за яким буде проводитись оцінка та порівняння методів. Вторинним критерієм буде критерій часу, бо завжди треба співвідносити збільшення точності та потенційне зменшення швидкодії.

4.2. Дані, що використовувались при аналізі ефективності

В силу того, що в основі методу лежить ущільнення, метод не висуває окремих вимог до стилю, мови і т.д. текстів. Тому було вирішено обрати мовою текстів, що використовувались при перевірці, англійську, оскільки в сучасному світі вона є основною мовою та для неї можна використовувати усі методи попередньої обробки та усі підходи, що були

наведені у розділі 2, у пункті про нормалізацію тексту. Дані були взяти з відкритого доступу, а точніше з сайту бібліотеки Гарвардського університету [31].

Приклад даних, що використовувались для порівняння: “On the basis of recurrent word-combinations in the form of 3-grams extracted from a corpus of English original fiction texts and a corpus of English fiction texts translated from Norwegian, we investigate the validity of the second point [1] Preliminary observations suggest that the original and translated texts seem to cluster in similar ways; e.g. the top 20 3-grams in English originals vs. English translations have an overlap of close to 100 per cent, although some differences can be noted in the ranking of the word-combinations. Our starting point is thus that the two varieties of English (EO / ET) exhibit the same type and number of 3-grams [2] The aim of this study is to outline and analyse the nature of such combinations in English originals vs. English translations in terms of their function and frequency. More specifically, the study explores to what extent EO and ET 3-grams differ functionally. By taking a statistical approach, we will be able to establish whether variety has a significant effect on the function of 3-grams or not. Another aim of the study is to outline a functional taxonomy and method by which such combinations in the two varieties of English can be adequately compared and analysed. The study can be said to be exploratory and experimental in the sense that it tests the potential of using quantitative methods to steer the qualitative research in interesting and meaningful directions. The following quotation from Firth (1957) underlines the relevance of studying translations and of comparing translations with non-translated texts to uncover how ‘fresh ideas’ are clothed:”.

4.3. Приклад попереднього оброблення текстових даних

Застосуємо описані у розділі 2 способи попереднього оброблення текстових даних до вищенаведеного тексту.

4.3.1. Дані після застосування нормалізації тексту

Після виконання операцій, що відносяться до нормалізації тексту, а саме вилучення знаків пунктуації; вилучення зносок, посилань; нормалізації UTF-8; приведення тексту до нижнього регістру та спрощення складних символів, текст, що оброблюється, набуде наступного вигляду:

“on the basis of recurrent word combinations in the form of grams extracted from a corpus of english original fiction texts and a corpus of english fiction texts translated from norwegian we investigate the validity of the second point preliminary observations suggest that the original and translated texts seem to cluster in similar ways e g the top grams in english originals vs english translations have an overlap of close to per cent although some differences can be noted in the ranking of the word combinations our starting point is thus that the two varieties of english eo et exhibit the same type and number of grams the aim of this study is to outline and analyse the nature of such combinations in english originals vs english translations in terms of their function and frequency more specifically the study explores to what extent eo and et grams differ functionally by taking a statistical approach we will be able to establish whether variety has a significant effect on the function of grams or not another aim of the study is to outline a functional taxonomy and method by which such combinations in the two varieties of english can be adequately compared and analysed the study can be said to be exploratory and experimental in the sense that it tests the potential of using quantitative methods to steer the qualitative research in interesting and meaningful directions the following quotation from firth underlines the relevance of studying translations and of comparing translations with non translated texts to uncover how fresh ideas are clothed”.

4.3.2. Дані після вилучення коротких слів

Після вилучення коротких слів, до яких відносяться слова довжиною в 5 та менше символів для латиниці, текст, що оброблюється, набуде наступного вигляду:

“recurrent combinations extracted corpus english original fiction corpus english fiction translated norwegian investigate validity second preliminary observations suggest original translated cluster similar english originals english translations overlap although differences ranking combinations starting varieties english exhibit number outline analyse nature combinations english originals english translations function frequency specifically explores extent differ functionally taking statistical approach establish whether variety significant effect function another outline functional taxonomy method combinations varieties english adequately compared analysed exploratory experimental potential quantitative methods qualitative research interesting meaningful directions following quotation underlines relevance studying translations comparing translations translated uncover clothed”

4.3.2. Дані після застосування soundex

Відповідно, після застосування останнього способу попереднього оброблення, наші дані набудуть наступного вигляду:

“62667251572362726125242625412352612524212353652735625512371433225316457126172222762543652724237254652426254752423652716414323116765257251572363716325242213551634554253625157524262547524236527152371625721217214672353311615237325223327116223147361632251711231523753634515237325553325157163252423234251675423214672165713537253375332243376262536275552736237144572335536476415723357365272516736527365275216243 3”

4.4. Результати аналізу ефективності методу

При аналізі ефективності модифікованого методу автоматизованого порівняння текстових документів на ідентичність, що базується на ущільненні у порівнянні з оригіналом використовувався метод ущільнення lzma, оскільки, як описано у розділі 2, він дає найкращі результати за критерієм точності для даної задачі.

Для реалізації базового методу методу автоматизованого порівняння текстових документів на ідентичність, що базується на ущільненні використовувалось розроблене та описане у розділі 3 програмне забезпечення. Різниця була у кроках попереднього оброблення текстових документів у модифікованому методі.

Для моніторингу швидкодії використовувались стандартні компоненти платформи .NET – stopwatchs, оскільки цей критерій не був обраний як основний. При більш серйозному акценті на цих критеріях, можливо, доречним є використання бібліотеки BenchmarkDotNet [32], що є дуже потужним інструментарієм для аналізу швидкодії програмних засобів.

Був взятий текст для порівняння та на його основі, шляхом вилучення частини тексту та заміни її на рівноцінну по довжині, але унікальну за вмістом, отриманий інший текст, з яким і велось порівняння. Відповідно, у першій колонці таблиці 4.1 вказана довжина тексту (в символах) та відсоткове значення схожості текстів. Дані аналізу ефективності модифікованого методу наведені у таблиці 4.1.

Таблиця 4.1

Результати проведення тестування

Довжина тексту та відсоток схожості	*NCD (orig)*	*NCD (modified)*	*Time (orig) (seconds)*	*Time (modified) (seconds)*

Продовження таблиці 4.1

500 (100)	0.12	0.07	2.31s	2.49s
500 (75)	0.31	0.36	2.34s	2.48s
500 (50)	0.63	0.52	2.33s	2.48s
500 (0)	0.77	0.89	2.41s	2.63s
1000 (100)	0.14	0.08	5.62s	6.1s
1000 (75)	0.3	0.34	5.7s	6.13s
1000 (50)	0.67	0.56	5.69s	6.24s
1000 (0)	0.83	0.91	6.1s	7.3s
5000 (100)	0.09	0.03	15.3s	17.24s
5000 (75)	0.29	0.25	15.6s	17.28s
5000 (50)	0.7	0.56	15.6s	17.96s
5000 (0)	0.9	0.95	16.1s	18.67s

Відповідно до даних, що наведені у таблиці 4.1, можна побачити, що модифікований метод автоматизованого порівняння текстів на ідентичність з використанням ущільнення показує найбільш кращі результати за критерієм точності, ніж оригінальний метод при прийнятному зменшенні швидкості.

Це пов’язане з тим, що способи попереднього оброблення, що були запропоновані у роботі, дозволили зменшити кількість «сміття», що містилось у текстах та виділити доволі чіткі структурні шаблони текстового документу (строка цифр від одиниці до сіми у нашому випадку). При цьому завдяки зменшенню об’єму даних, що відбувався на етапі попереднього оброблення, основний крок методу, а саме стиснення

відбувався швидше. Саме тому зменшення швидкодії за критерієм швидкості не було значним.

При цьому стоїть відмітити, що схожі результати були отримані на різних об’ємах тексту та ступенях схожості текстових документів, що дозволяє сказати про те, що модифікований метод порівняння текстових документів на ідентичність, що використовує ущільнення, є більш ефективним за метод, що брався за основу для вдосконалення.

На жаль, модифікований метод, так як і основний, не є стійким до змін. Це пов’язано з основним концептом, що використовується у методах – те, що у текстах повинен бути якийсь спільний шаблон, а зміни у одному з текстів можуть залишити лише локальний шаблон (на прикладі кількох слів) і «зломати» більш глобальним. На мою думку, в даному випадку, є актуальним використання підходів класичних методів для пошуку нечітких дублікатів, а саме підходів, що лежать у основі концепції шинглів. Це дозволить використовувати переваги модифікованого методу, що пропонується у даній роботі і усунути недолік, що описаний вище.

Також під час дослідження, був виявлений ще один недолік. Через те, що в способі нормалізації тексту є етап, що вилучає посилання і т.д., метод не враховує дані конструкції. Варіантом вирішення цього недоліку є додаткова логіка оброблення посилань та цитат, причому допустимими є варіанти як і вилучення тексту цитат та посилань разом зі зносками, так і додаткова логіка по їх обробленню та аналізу.

Відповідно, вищеописані недоліки потребують додаткових досліджень і робота над їх усуненням може бути темою подальших досліджень.

4.5. Перелік конструкцій «синтаксичного цукру» мови C#

Під час додаткового дослідження, що описане в пункті 2.3, була обрана мова C# та досліджені її конструкції на предмет «синтаксичного цукру».

Поняття «синтаксичного цукру» в деякій мірі є умовним – різні конструкції можуть бути як віднесені до «синтаксичного цукру» мови програмування, так і не бути віднесеними до нього. Наприклад, більшість сучасних мов програмування як високого, так і низького рівня, мають декілька конструкцій для запису циклів (з умовою, без умови, з лічильником, з післяумовою) або декілька умовних конструкцій. Віднесення однієї з них до базових конструкцій, а інших варіантів до «синтаксичного цукру» є помилковим і часто не доцільним, особливо при пошуку плагіату. До таких же конструкцій можна віднести різноманіття арифметичних операторів: наприклад, оператор += та подібні до нього оператори формально підпадають під визначення «синтаксичного цукру», проте вони є досить низькорівневими. В той час, як програмний код стає все більш складним та високорівневим, доцільним є віднесення до «синтаксичного цукру» саме конструкцій високого рівня абстракції для пошуку плагіату. Розгляд «синтаксичного цукру» лише високого рівня також дозволяє зменшити кількість хибно-позитивних помилок, таким чином підвищуючи якість отримуваних результатів за різними оцінками (наприклад, за такими, як точність та F міра).

До «синтаксичного цукру» мови програмування C# версії 7.0 та вище, за допомогою якого можливе маскування плагіату програмного коду, пропонується відносити:

- оператор null-об’єднання;
- неявну типізацію локальних змінних;
- ініціалізатори колекцій;

– ініціалізатори об’єктів;

– лямбда-вирази;

– методи розширення;

– автоматичні властивості;

– іменовані параметри;

– null-умовні оператори;

– лямбда-визначення функцій;

– інтерполяція рядків;

– out змінні;

– опціональні параметри.

Перелік конструкцій «синтаксичного цукру» мови C# та відповідні йому базові конструкції наведено в табл. 4.2.

Таблиця 4.2

«Синтаксичний цукор» мови C#

№	«Синтаксичний цукор»	Еквівалент
1	return obj1 **??** obj2;	return obj1 != null ? obj1 : obj2;
2	**var** obj = 1	int obj = 1
3	string[] arr = **{“A”, “B”}**	string[] arr = new string[2]; arr[0] = “A”; arr[1] = “B”;
4	Customer c = new Customer **{** **Name = "James",** **Age = 30** **}**;	Customer c = new Customer(); c.Name = "James"; c.Age=30;
5	obj.Act(**x => x**)	obj.Act(delegate(obj x) { return x; })
6	x**.Do()**	StaticClass.Do(x)

Продовження таблиці 4.2

7	public string A **{ get; set; }**	private string a;
		public string A
		{
		  get { return a; }
		  set { a = value; }
		}

8	Method(**a:** 10, **b:** 25)	Method(10, 25)
9	var obj1 = a**?.**Do(); var obj2 = arr**?[**0];	var obj1 = a != null ? a.Do() : null; var obj2 = arr != null ? arr[0] : null;

10	public object Do() => **new object();**	public object Do()
		{
		  return new object();
		}

11	var s = **$**"A**{obj.ToString()}**"	var s = string.Format("A{0}", obj)
12	int.TryParse("1", **out int** value);	int value; int.TryParse("1", out value);
13	public void Method(int a **= 5**); Method();	public void Method(int a); Method(5);

До окремого переліку слід включити «цукор», що відноситься до LINQ (Language Integrated Query). Цей механізм платформи .NET додає до мов програмування, що її підтримують (в тому числі і до мови програмування C#), синтаксис мови запитів (подібну до SQL), яка дозволяє оперувати даними з різних джерел – бази даних, пам'яті, XML і т.д.

LINQ в мові програмування C# можна використовувати двома шляхами:

- за допомогою операторів мови;
- за допомогою методів розширення.

Методи розширення виступають в ролі «синтаксичного цукру» для операторів LINQ, таким чином вони також відносяться до області застосування запропонованих підходів пошуку плагіату в програмному коді. Причиною їх виділення в окремий перелік (див. табл. 4.3) авторами цієї роботи є те, що перетворення методів розширення LINQ на оператори та навпаки потребує розроблення та застосування більш складних та комплексних правил для того, щоб зберегти правильний порядок оброблення даних та викликів.

Таблиця 4.3

Оператори та методи розширення LINQ

№	Метод розширення	Відповідний оператор
1	GroupBy	group … by
2	Join	join … in … on … equals …
3	OrderBy	orderby …
4	OrderByDescending	orderby … descending
5	Select	select
6	SelectMany	декілька операторів from
7	ThenBy	orderby …, …
8	ThenByDescending	orderby …, … descending
9	Where	where

4.6. Висновки

У даному розділі було наведено критерій оцінки ефективності модифікованого методу автоматизованого порівняння текстових документів на ідентичність, що базується на ущільненні. Обґрунтовано вибір даних, що використовувались для аналізу ефективності. Та, власне, проведено порівняльний аналіз ефективності оригінального та модифікованого методів автоматизованого порівняння текстових документів на ідентичність, що базується на ущільненні.

Також, були наведені конструкції «синтаксичного цукру» мови C# та еквівалентні їм базові конструкції мови.

За результатами порівняння можна зробити наступні висновки:

- модифікація методу дозволяє отримати більш кращій результат за критерієм точності;
- при цьому алгоритм значно не програє за критерієм швидкодії по часу;

У подальших дослідженнях можна проаналізувати стійкість методу до незначних змін.

5. ПОБУДОВА БІЗНЕС-МОДЕЛІ

5.1. Опис проблеми

Розроблений в даній роботі метод автоматизованого порівняння текстових документів на ідентичність, що базується на ущільненні може використовуватись в будь-якій області, де необхідно здійснювати пошук схожих за змістом документів.

Однією з таких на сьогодні областей є перевірка робіт на плагіат у вищих навчальних закладах [33]. В цій області досі є проблема з точністю та швидкістю такої перевірки, що пов’язано з досить великою кількістю факторів.

По-перше, це велика обчислювальна складність алгоритмів, що лежать в основі існуючих рішень. Ці алгоритми часто потребують грубу попередню обробку текстових документів для своєї роботи, яка сама по собі займає доволі велику кількість часу [34]. При цьому існує експоненційна залежність часу попередньої обробки текстів від об’єму.

По-друге, це немалий час пошуку по текстовій базі. Навіть за умови, що усі тексти попередньо оброблені і збережені в базі, пошук по базі є затратним.

По-третє, для роботи існуючих рішень необхідно доволі велика кількість оперативної пам’яті, що не дозволяє паралельну обробку великої кількості документів або потребує великих коштів для закупівлі великої кількості комп’ютерів. Альтернативним рішенням є створення черги документів на обробку, що знов таки збільшує час на обробку документу і отримання результату.

У сукупності ці фактори призводять до довгого очікування від моменту загрузки текстового документу на аналіз до отримання результату. Це є недопустимим для таких організацій як університети, стрічки новин або видавництва, у яких потоки документів є досить

значними і довгий час перевірки на плагіат затримує внутрішні бізнес-процеси.

Також важливим фактором є точність алгоритмів, що лежать в основі використовуваних рішень. Рішення не давати не тільки прийнятну точність, а й не мати багато хибних спрацьовувань. Точність рішень впливає на кількість людських годинно-затрат, що необхідні для пост опрацьовування результатів.

Отже, видно що завдання порівняння текстових документів на ідентичність не є тривіальним і має багато проблем у існуючих рішеннях, що вповільнюють бізнес-процеси у тих підприємствах, що використовують пошук плагіату у своїй роботі.

Описані проблеми узагальнено в дерево проблем, що зображено на рис. 5.1.

Рис. 5.1. Дерево проблем

5.2. Зацікавлені сторони

У вирішенні описаної вище проблеми прямо чи опосередковано зацікавлено досить багато різних сторін.

Є очевидним, що найбільш зацікавленими в вирішені даної проблеми є власники або керуючі особи організацій, де використовується пошук плагіату у текстових документах, оскільки за рахунок її вирішення вони отримують підвищення ефективності роботи підприємства.

Робітники підприємств заінтересовані в збільшенні швидкості опрацювання документів, що дозволить їм бути заблокованим цим процесом менший час та збільшити ефективність праці.

Також в вирішенні проблеми заінтересовані сторони, що надають текстові документи підприємствам. Висока швидкість дозволить їм оцінити документи на відповідність прийнятним підприємствами критеріям. У разі невідповідності, вони зможуть переробити текстовий документ та зробити повторну оцінку, не витрачаючи час на очікування результату.

Опосередковано в вирішенні описаної проблеми зацікавлені такі сторони, як держава, замовники і інші. Держава зацікавлена в отриманні більшої кількості податків, що сплачують підприємства, підвищення швидкості роботи дозволить отримувати підприємствам більший дохід і, відповідно, платити більше податків. Замовники зацікавлені у більш швидкому виконанні їх замовлень. Проте ці зацікавлені сторони мають невеликий вплив.

Описані зацікавлені сторони узагальнено в матрицю зацікавлених осіб, що наведена в табл. 5.1.

Таблиця 5.1

Матриця зацікавлених осіб

Група зацікавлених осіб	Інтереси зацікавленої особи	Вплив	Стратегії приваблення
Власники або керуючі органи підприємств	Підвищення ефективності роботи підприємств	Великий	Рекламні матеріали. Презентації продукту, демонстрації. Участь в тематичних заходах. Надання безкоштовних пробних версій. Забезпечення технічної підтримки.
Робітники підприємств	Підвищення швидкості роботи	Середній	
Сторони, що надають матеріали підприємствам	Зменшення часу отримання результатів при попередній оцінці документів на відповідність критеріям	Середній	

Продовження таблиці 5.1

Держава	Збільшення кількості налогів від підприємств	Малий	
Замовники	Збільшення швидкості виконання замовлень	Малий	

5.3. Рішення. Основні характеристики

Поставлену проблему може вирішити спеціальний програмний продукт, що реалізує описаний метод автоматизованого порівняння текстових документів на ідентичність, що базується на ущільненні. Завдяки описаному в даній роботі алгоритму, текстові дані оброблюються та зберігаються ефективно, що дозволяє збільшити швидкість пошуку, забезпечуючи при цьому прийнятну точність.

Відповідно, ці параметри підвищать швидкість пошуку плагіату в текстових документах, що є важливим для кінцевих користувачів.

Основними клієнтами є університети, видавництва, стрічки новин та paper mill – організації, що спеціалізуються на написанні дипломних, курсових та інших робіт для студентів. Саме тому програмне рішення повинно бути представлене у вигляді веб-сервісу, що буде мати зовнішні API. На їх базі буде побудований веб-сайт, що буде задовольняти потребам більшої кількості користувачів. Завдяки API забезпечується гнучкість рішення, наприклад, великі організації зможуть легко інтегрувати сервіс у свої існуючі програмні комплекси. Запропонований алгоритм гарно кладеться в основу рішення, що легко масштабується, що дозволяє будувати велику і розподілену систему.

5.4. Конкурентні переваги рішення

Конкурентів даного програмного рішення на ринку досить багато. Умовно їх можливо розділити на дві великі групи:

– веб-сервіси у вигляді сайтів;

– спеціалізовані програмні комплекси.

Веб-сервіси у вигляді сайтів, частіше всього, розраховані на індивідуальне використання та мають обмеження у вигляді загальної черги на обробку та прив’язки вартості використання до розміру текстів, що аналізуються. Очевидно, що такі сервіси не можна використовувати корпоративним клієнтам та клієнтам, що часто змінюють текстові документи великого об’єму та роблять періодичні перевірки. Також, зазвичай, такі сервіси не мають великої бази даних документів для звірки.

Спеціалізовані програмні комплекси є дуже залежними від обчислювальних потужностей та використовуються, зазвичай, корпоративними клієнтами. Недоліками є пряма залежність вид наявності потужностей та кількості вхідних даних. Часто програмні комплекси мають велику базу даних текстових документів, що, безперечно, є перевагою, але швидкість роботи з нею обернено пропорційна її розміру, що, очевидно, є недоліком.

Отже, конкурентними перевагами програмного продукту, що пропонується є:

– висока швидкість опрацювання текстових документів за рахунок запропонованого алгоритму;
– ефективне збереження текстових документів та можливість створення глобальної бази та пошуку в ній;
– гарна точність пошуку;
– можливість легкого інтегрування сервісу у існуючі програмні комплекси;
– декілька варіантів роботи з сервісом, що є приязними для різних категорій клієнтів – корпоративних і індивідуальних;
– відсутність залежності вартості від об’єму тексту, що аналізується;
– можливість вдосконалення сервісу під конкретного замовника.

5.5. Клієнти. Сегменти ринку споживання

Як вже зазначалось вище, потенційних клієнтів продукту можна поділити на два сегменти – корпоративні та індивідуальні клієнти.

Відповідно, до кожного з цих сегментів пропонується індивідуальний підхід, що базується на потребах сегменту.

Сегмент корпоративних клієнтів характеризується великою кількістю документів протягом тривалих часових проміжків, наявністю власних програмних комплексів та баз документів. Отже, для таких клієнтів доцільно пропонувати підписки, що характеризуються часовим інтервалом (наприклад, місячну, квартальну, річну т.д.) та кількістю документів, що одночасно перебувають на обробленні. Також, таким клієнтам можна пропонувати послуги інтеграції сервісу до існуючих комплексів, персонального вдосконалення сервісу «під ключ» та розширену технічну підтримку.

Індивідуальні клієнти характеризуються точковими аналізами, тому їм доцільно пропонувати варіанти підписки з обмеженням на кількість аналізів без терміну дії (наприклад, 1, 3, 7 аналізів т.д.) та короткострокові підписки з обмеженням, що на обробленні може бути одночасно тільки один документ (наприклад, на 1, 3, 7, 14 днів). Відповідно, перший тип підписок призначено для тих, хто робить рідко аналізи, а другий – для тих, хто робить аналіз і коректує документ чи документи протягом короткотривалого обсягу часу. Платформою для цього сегменту клієнтів, в основному, є веб-сайт.

5.6. Унікальна ціннісна пропозиція

Ціннісна пропозиція описує ті переваги, які надають споживачу наші товари та послуги і які вирішують проблеми споживачів [35]. Відповідно, унікальною буде така пропозиція, яка чітко відділяє товар від конкурентів і дає зрозуміти, чому необхідно придбати саме наш товар.

Для кожного сегменту споживачів зазвичай пропонується виділяти окрему ціннісну пропозицію [36].

Ціннісною пропозицією для сегменту корпоративних клієнтів є швидка, точна система, що легко, за необхідності, інтегрується в існуючі рішення та дозволяє обробляти одночасно певну кількість документів, верхню межу якої можна динамічно змінювати, в залежності від потреб компанії. Також додатковими є послуги індивідуального налаштування системи та розширеної технічної підтримки.

Для сегменту індивідуальних клієнтів ціннісною пропозицією є наявність декількох варіантів підписок для використання системи, що дозволяє більш гнучко підлаштовувати використання сервісу під конкретні потреби клієнта, також, використовуючи сервіс, клієнт отримує можливість аналізу на великій базі, що динамічно буде наповнюватись.

5.7. Доходи та витрати

Сумарний дохід складається як сума доходів від продаж товарів та супутніх послуг для кожного сегменту споживачів.

Для сегменту корпоративних клієнтів пропонується продавати наступні товари та надавати наступні послуги:

- довготривалі підписки з динамічним налаштуванням максимальної кількості документів, що одночасно знаходяться в черзі;
- індивідуальне налаштування системи під конкретного клієнта;
- розширена технічна підтримка.

Під довготривалими підписками з динамічним налаштуванням максимальної кількості документів, що одночасно знаходяться в черзі мається на увазі, що клієнт може обрати один з запропонованих строків підписки та одне з запропонованих значення максимальної кількості

документів, що одночасно знаходяться у черзі. Відповідно до цих параметрів і буде розраховуватись вартість підписки.

Індивідуальне налаштування системи під конкретного клієнта є окремою послугою, що оплачується за кожен прецедент надання її та включає в себе корекцію параметрів алгоритму, використання або виключення специфічних баз та розширення базової функціональності під конкретного клієнта (що буде ідентифікуватись ключем підписки).

Розширення технічна підтримка пропонується як послуга на термін дії підписки і включає в себе гарантоване оброблення запитів в швидкі терміни в робочі та вихідні дні.

Для сегменту індивідуальних клієнтів пропонується продавати наступні товари:

- підписку з обмеженням на кількість документів для аналізу;
- підписку з короткотривалим часовим обмеженням та обмеження на один документ, що може оброблятись одночасно в будь-який момент часу.

Різниця між короткотривалими та довготривалими часовими обмеженнями полягає в тому, що короткотривалі включать в себе терміни до місяця такі як день, 3 дні, тиждень, 2 тижні, місяць, а довготривалі, відповідно, починаються від місяця – місяць, 3 місяці, півроку, рік т.д.

Розрахований дохід по місяцям протягом першого року наведено в табл. 5.2. Всі суми наведено в доларах США.

Таблиця 5.2

Доходи

	Довготривалі підписки	Індивідуальне налаштування	Розширена технічна підтримка	Підписка з обмеженням на кількість документів	Короткотривалі підписки	Всього
1	180	0	0	50	90	320

Продовження таблиці 5.2

2	420	200	100	200	300	1220
3	800	400	400	400	450	2450
4	1100	700	500	450	600	3350
5	1500	1000	600	500	1000	4600
6	2400	1600	1000	750	1500	7250
7	2000	1400	800	700	1400	5300
8	1900	1500	700	600	1200	5900
9	3000	2000	1100	1100	1500	8700
10	1900	1600	700	700	1100	6000
11	2000	1800	900	900	1000	6600
12	2600	2000	1000	1300	1100	7000
					Сумарно за рік:	58690

Сукупність загальних витрат складають наступні витрати:

- витрати на розроблення, підтримку та вдосконалення програмних продуктів (утримання робочих місць, придбання необхідних інструментів та програмних засобів);
- витрати на надання послуг технічної підтримки (утримання робочих місць, придбання необхідних інструментів та програмних засобів);
- витрати на оплату праці (заробітна плата та інші виплати працівникам);
- витрати на опалення, освітлення, водопостачання, водовідвід;
- адміністративні витрати (витрати на зв’язок, податки та збори, плата за послуги банків тощо);
- витрати на рекламу та дослідження ринку.

На початкове розроблення необхідно 3000$, також доробка буде відбуватись протягом 11 місяців і потребує суму у розмірі 1650$. Отже, сумарно на розробку буде витрачено 4650$.

Витрати по місяцям протягом першого року наведено в табл. 5.3. Всі суми наведено в доларах США.

Таблиця 5.3

Витрати

	Технічна підтримка	Оплата праці	Комунальні та адміністративні	Реклама	Всього
1	500	2000	500	350	3350
2	50	2000	500	400	2950
3	50	2000	500	450	3000
4	50	2000	500	500	3050
5	50	2000	500	600	3150
6	50	2000	500	600	3150
7	50	2000	500	600	3150
8	50	2000	500	600	3150
9	50	2000	500	600	3150
10	50	2000	500	600	3150
11	50	2000	500	600	3150
12	50	2000	500	600	3150
				Сумарно за рік:	37550

Розрахуємо маржинальний прибуток за перший рік за формулою (5.1) [37].

$$\text{Маржинальний прибуток} = \text{Дохід} - \text{Витрати} \quad (5.1)$$

За формулою (5.1) маржинальний прибуток за перший рік складе 58690-(4650+37550) = 16490 доларів США.

5.8. Бізнес-модель

Узагальнимо вище наведену інформацію в виглядів блоків, що складають так звану канву бізнес-моделі або lean canvas [38].

Споживачі:

- ранні клієнти: малі підприємства та окремі індивідуальні клієнти;
- середні і великі підприємства, індивідуальні клієнти.

Проблема:

- низька швидкість аналізу текстових документів на плагіат;
- велика обчислювальна складність алгоритмів;
- значне використання ресурсів;
- не завжди точний пошук.

Рішення:

- програмний продукт, що має високу швидкість аналізу та зберігання текстових документів при гарній точності.

Унікальна ціннісна пропозиція:

- гнучкі підписки для корпоративних і індивідуальних клієнтів;
- можливість легкої інтеграції в існуючі системи і індивідуального налаштування.

Потоки доходів:

- продаж довготривалих підписок для корпоративних клієнтів з динамічним налаштуванням максимальної кількості документів, що одночасно знаходяться в черзі;
- продаж послуг індивідуального налаштування системи під конкретного клієнта;
- продаж розширеної технічної підтримки;
- продаж підписки з обмеженням на кількість документів для аналізу для індивідуальних клієнтів;

– продаж підписки з короткотривалим часовим обмеженням та обмеження на один документ, що може оброблятись одночасно в будь-який момент часу для індивідуальних клієнтів.

Структура витрат:

– витрати на розроблення, підтримку та вдосконалення програмних продуктів;
– витрати на надання послуг технічної підтримки;
– витрати на оплату праці;
– витрати на опалення, освітлення, водопостачання, водовідвід;
– адміністративні витрати;
– витрати на рекламу та дослідження ринку.

Також в канву бізнес-моделі включаються структурні блоки, які ще не були розглянуті. Це прихована перевага (перевага, яку не можливо скопіювати або купити), ключові метрики (основні показники, що вимірюються) та канали (шляхи до користувачів).

В якості каналів контакту з клієнтами пропонується використовувати наступні шляхи: тематичні ресурси та спільноти, профільні видання, прямі контакти для продажів.

Прихованою перевагою нашого продукту виступає використання технології, що базується на новому алгоритмі для пошуку плагіату серед текстових документів.

Ключовими метриками є наступні: кількість проданих підписок різних типів, кількість продажів індивідуальних налаштувань системи, кількість запитів в службу технічної підтримки.

Описані структурні блоки об’єднано в єдину канву бізнес-моделі, котра наведена в табл. 5.4.

Таблиця 5.4

Канва бізнес-моделі

<table>
<tr>
<td rowspan="2">Проблема
Низька швидкість аналізу текстових документів на плагіат.

Велика обчислювальна складність алгоритмів. Значне використання ресурсів.

Не завжди точний пошук.</td>
<td>Рішення
Програмний продукт, що має високу швидкість аналізу та зберігання текстових документів при гарній точності</td>
<td rowspan="2">Унікальна ціннісна пропозиція
Гнучкі підписки для корпоративних і індивідуальних клієнтів.

Можливість легкої інтеграції в існуючі системи і індивідуального налаштування</td>
<td>Прихована перевага
Використання технології, що базується на новому алгоритмі для пошуку плагіату серед текстових документів</td>
<td rowspan="2">Споживачі
Ранні клієнти: малі підприємства та окремі індивідуальні клієнти.

Середні і великі підприємства, індивідуальні клієнти.</td>
</tr>
<tr>
<td>Ключові метрики
Кількість проданих підписок різних типів.

Кількість продажів індивідуальних налаштувань системи

Кількість запитів в службу технічної підтримки..</td>
<td>Канали
Тематичні ресурси та спільноти.

Профільні видання.

Прямі контакти для продажів</td>
</tr>
</table>

Продовження таблиці 5.4

Структура витрат	Потоки доходів
Витрати на розроблення, підтримку та вдосконалення продуктів. Витрати на надання послуг технічної підтримки. Оплата праці. Комунальні послуги. Адміністративні витрати. Витрати на рекламу та дослідження ринку.	Продаж довготривалих підписок для корпоративних клієнтів з динамічним налаштуванням максимальної кількості документів, що одночасно знаходяться в черзі. Продаж послуг індивідуального налаштування системи під конкретного клієнта. Продаж розширеної технічної підтримки. Продаж підписки з обмеженням на кількість документів для аналізу для індивідуальних клієнтів. Продаж підписки з короткотривалим часовим обмеженням та обмеження на один документ, що може оброблятись одночасно в будь-який момент часу для індивідуальних клієнтів

5.9. Висновки

В даному розділі розглянуто питання практичного застосування розробленого методу пошуку плагіату серед текстових документів.

Незважаючи на те, що побудована бізнес-модель потребує доопрацювань, вона підтверджує життєздатність стартапу, заснованого на застосуванні методу автоматизованого порівняння текстових документів на ідентичність, що базується на ущільненні Описані сегменти споживачів, які готові платити за вирішення даної проблеми, наявні унікальні ціннісні пропозиції для кожного сегменту. Запропонований програмний продукт має конкурентні переваги.

Розраховані доходи та витрати протягом першого року роботи. Знайдений маржинальний прибуток за цей період – 16.5 тисяч доларів США – також підтверджує життєздатність описаної бізнес-моделі.

ВИСНОВКИ

На основі огляду методів для порівняння текстів на ідентичність, був визначний основний критерій, за яким можна оцінити ефективність даних методів.

Відповідно, при розгляді новітнього методу, а саме методу автоматизованого порівняння текстів на ідентичність з використанням ущільнення даних, було виділено основні нюанси, що лежать у його основі та визначено шляхи модифікації даного методу, що ведуть до підвищення точності.

Для вирішення задачі пошуку дублікатів у текстових документах пропонується модифікований метод автоматизованого порівняння текстів на ідентичність з використанням ущільнення даних, модифікований у частинах попереднього оброблення текстових даних.

Було розглянуто способи попереднього оброблення текстових документів та порівняння алгоритмів ущільнення для подальшого застосування у методі автоматизованого порівняння текстів на ідентичність на основі ущільнення даних.

Запропонований метод автоматизованого порівняння текстів на ідентичність на основі ущільнення даних складається з таких етапів: попереднє оброблення текстових даних, до якого відносяться наступні способи: нормалізація тексту, вилучення коротких слів, soundex; зчитування у бінарному вигляді оброблених на попередньому кроці текстів та їх конкатенація; ущільнення цих бінарних даних алгоритмом lzma; вирахування нормалізованої відстані ущільнення та оцінка степені схожості текстових документів емпірично встановлених границь та значення нормалізованої відстані ущільнення.

Розроблений метод автоматизованого порівняння текстів на ідентичність може використовуватись в різних галузях, де необхідно порівнювати текстові документи на ідентичність.

Також в рамках даної роботи розроблено програмне забезпечення, яке призначене для порівняння текстів на ідентичність та реалізує модифікований метод автоматизованого порівняння текстів на ідентичність з використанням ущільнення даних.

Можливо зробити наступні висновки про розроблене програмне забезпечення в цілому:

- розроблене програмне забезпечення можливо використовувати для автоматизованого порівняння текстів на ідентичність;
- розроблений консольний за стосунок надає можливості роботи з розробленими бібліотеками, що реалізовують бізнес-логіку;
- архітектура розробленого програмного забезпечення дозволяє легко додавати нові модулі, що реалізують попереднє оброблення текстових документів та ущільнення за певними алгоритмами завдяки використанню таких шаблонів проектування як «стратегія» та «boundary context» з domain driven development;
- завдяки реалізації бізнес-логіки методу у вигляді окремих бібліотек, що сумісні з стандартом .NET Standart 2.0 розроблене програмне забезпечення не залежить від інтерфейсу користувача і може використовуватись з будь-яким інтерфейсом (консольним, графічним, веб і т.д.) або бути інтегрованим до існуючих програмних комплексів та систем та бути запущеній на будь-якій платформі, що містить компоненти .NET, що сумісні зі стандартом .NET Standart 2.0 (наприклад, на Windows з .NET Framework 4.7+, на Unix системах, що містять .NET Core 2.0+).

Було наведено критерій оцінки ефективності модифікованого методу автоматизованого порівняння текстових документів на ідентичність, що базується на ущільненні. Обґрунтовано вибір даних, що використовувались для аналізу ефективності. Та, власне, проведено порівняльний аналіз ефективності оригінального та модифікованого методів автоматизованого порівняння текстових документів на ідентичність, що базується на ущільненні.

За результатами порівняння зроблені наступні висновки:

- модифікація методу дозволяє отримати більш кращій результат за критерієм точності;
- при цьому алгоритм значно не програє за критерієм швидкодії по часу;

Запропонований можливий вектор подальших досліджень.

Розглянуто питання практичного застосування розробленого методу пошуку плагіату серед текстових документів.

Незважаючи на те, що побудована бізнес-модель потребує доопрацювань, вона підтверджує життєздатність стартапу, заснованого на застосуванні методу автоматизованого порівняння текстових документів на ідентичність, що базується на ущільненні Описані сегменти споживачів, які готові платити за вирішення даної проблеми, наявні унікальні ціннісні пропозиції для кожного сегменту. Запропонований програмний продукт має конкурентні переваги.

Розраховані доходи та витрати протягом першого року роботи. Знайдений маржинальний прибуток за цей період – 16.5 тисяч доларів США – також підтверджує життєздатність описаної бізнес-моделі.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Как работает метод шинглов при проверке текста на плагиат [Електронний ресурс] – дата візиту 28.01.2017. – Режим доступу до ресурсу: https://goo.gl/Cm3Sgg
2. Семантичний аналіз тексту та пошук плагіату. Сергій Ващілін [Електронний ресурс] – дата візиту 28.11.2017. – Режим доступу до ресурсу: https://goo.gl/gJDW2C
3. U. Manber. Finding Similar Files in a Large File System. Winter USENIX Technical Conference, 1994.
4. N. Heintze. Scalable document fingerprinting. In Proc. of the 2nd USENIX Workshop on Electronic Commerce, Nov. 1996.
5. Д. Гасфилд. Строки, деревья и последовательности в алгоритмах. СПб.: Невский диалект, 2003.
6. A. Broder, S. Glassman, M. Manasse and G. Zweig. Syntactic clustering of the Web. Proc. of the 6th International World Wide Web Conference, April 1997.
7. A. Broder. On the resemblance and containment of documents. Compression and Complexity of Sequences (SEQUENCES'97), pages 21-29. IEEE Computer Society, 1998
8. A. Broder. Algorithms for duplicate documents.
9. D. Fetterly, M. Manasse, M. Najork. A Large-Scale Study of the Evolution of Web Pages, WWW2003, May 20-24, 2003, Budapest, Hungary.
10. A. Broder, M. Charikar et al. Min-wise independent permutations, Proceedings of the thirtieth annual ACM symposium on Theory of computing, 1998
11. A. Chowdhury, O. Frieder, D. Grossman, M. McCabe. Collection statistics for fast duplicate document detection. ACM Transactions on Information Systems (TOIS), Vol. 20, Issue 2 (April 2002).

12. A. Chowdhury. Duplicate Data Detection. [Електронний ресурс] – дата візиту 05.04.2018. – Режим доступу до ресурсу: http://ir.iit.edu/~abdur/Research/Duplicate.html
13. A. Kolcz, A. Chowdhury, J. Alspector. Improved Robustness of Signature-Based Near-Replica Detection via Lexicon Randomization. KDD 2004. [Електронний ресурс] – дата візиту 05.04.2018. – Режим доступу до ресурсу: http://ir.iit.edu/~abdur/publications/470-kolcz.pdfi
14. W. Pugh. Detecting duplicate and near — duplicate files. [Електронний ресурс] – дата візиту 06.04.2018. – Режим доступу до ресурсу: http://www.cs.umd.edu/~pugh/google/Duplicates.pdf
15. S. Ilyinsky, M. Kuzmin, A. Melkov, I. Segalovich. An efficient method to detect duplicates of Web documents with the use of inverted index. WWW Conference 2002. [Електронний ресурс] – дата візиту 12.04.2018. – Режим доступу до ресурсу: http://www2002.org/CDROM/poster/187/
16. Normalized compression distance [Електронний ресурс] – дата візиту 12.04.2018. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Normalized_compression_distance
17. Пошук плагіату в текстах та семантичний аналіз тексту. Сергій Ващілін [Електронний ресурс] – дата візиту 12.04.2018. – Режим доступу до ресурсу: https://fwdays.com/en/event/highload-fwdays-17/review/text-semantic-analyze
18. Нормализация Unicode [Електронний ресурс] – дата візиту 14.04.2018. – Режим доступу до ресурсу: https://habr.com/post/45489/
19. Сравнение программ сжатия в применении к передаче больших объёмов данніх [Електронний ресурс] – дата візиту 14.04.2018. – Режим доступу до ресурсу: https://goo.gl/p9Swde
20. GitHub – adamhathcock/sharpcompress [Електронний ресурс] – дата візиту 14.04.2018. – Режим доступу до ресурсу: https://goo.gl/Y2xUA0

21. Huang X., Hardison R.C., Miller W.A space-efficient algorithm for local similarities. // Computer Applications in the Biosciences 6. 1990. P. 373–381.
22. Baxter I., Yahin A., Moura L., Anna M.S., Bier L. Clone Detection Using Abstract Syntax Trees. // Proceedings of ICSM. IEEE. 1998.
23. Moussiades L.M., Vakali A.P Detect: A Clustering Approach for Detecting Plagiarism in Source Code Datasets. // The Computer Journal Advance Access. June 24, 2005
24. .NET Standard [Електронний ресурс]. — дата візиту 14.04.2018. — Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/standard/net-standard
25. .NET Framework [Електронний ресурс]. — дата візиту 14.04.2018. — Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/.NET_Framework
26. .NET Framework [Електронний ресурс]. — дата візиту 14.04.2018. — Режим доступу до ресурсу: https://en.wikipedia.org/wiki/.NET_Framework
27. C Sharp [Електронний ресурс]. — дата візиту 15.04.2018. — Режим доступу до ресурсу: https://goo.gl/m3GBmZ
28. Microsoft Visual Studio [Електронний ресурс]. — дата візиту 15.04.2018. — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio
29. Git [Електронний ресурс]. — дата візиту 15.04.2018. — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Git
30. Ubiquitous Language и Bounded Context в DDD [Електронний ресурс]. — дата візиту 16.04.2018. — Режим доступу до ресурсу: https://habr.com/post/232881/
31. Harvard Library Open Metadata [Електронний ресурс]. — дата візиту 16.04.2018. — Режим доступу до ресурсу: http://library.harvard.edu/open-metadata

32. BenchmarkDotNet [Електронний ресурс]. — дата візиту 17.04.2018. — Режим доступу до ресурсу: https://github.com/dotnet/BenchmarkDotNet
33. Семантичний аналіз тексту та пошук плагіату [Електронний ресурс] – дата візиту 26.11.2017. – Режим доступу до ресурсу: https://goo.gl/gJDW2C
34. Метод шинглов при поиске плагиата [Електронний ресурс] – дата візиту 28.11.2017. – Режим доступу до ресурсу: https://goo.gl/Cm3Sgg
35. Value Proposition Canvas Template [Електронний ресурс]. – дата візиту 04.12.2017. – Режим доступу до ресурсу: https://goo.gl/MbhgG4
36. Шаблон — GN1403: Моделирование бизнес-процессов — Бизнес-информатика [Електронний ресурс]. – дата візиту 04.12.2017. – Режим доступу до ресурсу: https://goo.gl/43VDek
37. Маржинальная прибыль. Пример анализа. Формула расчета [Електронний ресурс]. – дата візиту 04.12.2017. – Режим доступу до ресурсу: https://goo.gl/kyFxfH
38. Как правильно заполнять Lean Canvas | Wiki.Rademade.com [Електронний ресурс]. – дата візиту 04.12.2017. – Режим доступу до ресурсу: https://goo.gl/Gkb4Uq

ДОДАТКИ