Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи прискореної обробки текстової інформації для організації сховищ інформаційних об'єктів

(тема)

Виконав:
студент II курсу, групи КСМм-19-1
Зайцевої С.Г.
(прізвище, ініціали)

Спеціальність
123 – Комп'ютерна інженерія
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп'ютерні системи та мережі
(повна назва освітньої програми)

Керівник: доц. Барковська О.Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 – Комп'ютерна інженерія
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютерні системи та мережі
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Зайцевій Софії Геннадіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Методи прискореної обробки текстової інформації для організації сховищ інформаційних об'єктів

затверджена наказом по університету від " 30 " жовтня 2020 р. № 1487Ст

2. Термін подання студентом роботи до екзаменаційної комісії 14 грудня 2020 р.

3. Вхідні дані до роботи Текстові корпуси, центральний процесор, графічний процесор, алгоритми та методи попередньої обробки тексту

4. Перелік питань, що потрібно опрацювати в роботі
- розробка моделі сховищ інформаційних об'єктів;
- аналіз методів попередньої обробки тексту на виявлення можливості реалізації на системах із масовим паралелізмом;
- дослідження впливу характеристик обчисюлвальної системи на час реалізації модифікованого методу визначення ваги слів у корпусі тексту.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Слайд презентація – 10 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
	залишити пустим, якщо кон-		
	сультантом є керівник роботи		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Проектування організаційної моделі сховища	03.11.20-09.11.20	
	інформаційних об'єктів		
2	Огляд етапів накопичення інформації	10.11.20-17.11.20	
3	Виконання експериментів послідовного	18.11.20-23.11.20	
	упорядкування частотного словнику		
4	Виконання експериментів паралельного	24.11.20-01.12.20	
	упорядкування частотного словнику		
5	Аналіз отриманих результатів	02.12.20-04.12.20	
6	Оформлення матеріалів атестаційної роботи	05.12.20-07.12.20	
7	Подання атестаційної роботи керівникові та	08.12.20-09.12.20	
	її попередній захист		
8	Подання атестаційної роботи на рецензування	10.12.20-11.12.20	

Дата видачі завдання 02 листопада 2020 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Барковська О.Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 67 с., 16 рис., 9 табл., 1 дод., 20 джерел.

ОБРОБКА ТЕКСТУ, ПРИСКОРЕННЯ, МОДЕЛЬ, ІНФОРМАЦІЙНИЙ ПОШУК, ЧАСТОТНИЙ СЛОВНИК, УПОРЯДКУВАННЯ.

Метою атестаційної роботи є створення моделі сховищ інформаційних об’єктів із модифікованими та вдосконаленими методами прискореної обробки текстової інформації.

У ході виконання атестаційної роботи запропоновано впаровадження сортування вагів термів на системах із масовим паралелізмом у побудованму частотному словнику, що скорочує час роботи блоку синтаксичного рівня запропонованої моделі майже на 18%.

ABSTRACT

Master’s thesis: 67 pages, 16 figures, 9 tables, 1 appendices, 20 sources.

TEXT PROCESSING, SPEEDUP, MODEL, INFORMATION SEARCH, FREQUENCY VOCABYLARY, SORTING.

The major goal of this thesis is to create a model of storage of information objects with modified and improved methods of accelerated processing of textual information.

It is proposed to pair the sorting weights of terms on systems with mass parallelism in the constructed frequency dictionary, which reduces the operating time of the block of syntactic level of the proposed model by almost 18%.

ЗМІСТ

ВСТУП

Мозок людини забезпечує короткочасну оперативну пам'ять. Однак зі смертю людини інформація в цій оперативної пам'яті зникає. І для довгострокового запам'ятовування людина використовує інші, штучні методи і носії – рукописні, друковані або електронні варіанти носіїв інформації.

Щоб зрозуміти значущість інформації в сучасному світі потрібно згадати, що її накопичення ведеться ще з древніх часів. З перших років свого існування людина використовувала таку природну інформаційну технологію, як мова. Згодом поряд із промовою для збереження і передачі інформації людина стала використовувати зображення і писемність. У міру розвитку мови і загальної культури народів стали з'являтися різні види писемності як «тверда копія», якийсь замінник мови. Основним призначенням писемності виступає функція збереження інформації. Таким чином, основне завдання писемності – фіксувати інформацію на носіях і передавати її іншим людям.

Більшість мов використовують фонетичне письмо, при якому графічний знак (графема) прив'язаний до певного звучання. Такий письмовий знак може позначати окремий склад або окремий звук мови. Крім організації мови на рівні форнетіческом, виділяють також графематичний, морфологічний, синтаксичний рівень і семантичний рівні.

Про важливість збереження та накопичення інформації свідчить також те, що в одній з найбільш відомих бібліотек, в якій працювали Архімед, Евклід, Ератосфен Кіренський, Герон Олександрійський, Клавдій Птолемей і багато інших, – Александріісказ бібліотек – діяв закон, згідно з яким всі рукописи, виявлені на кораблях, які прибували в александрійську гавань, повинні були обов'язково направлятися в бібліотеку для переписування. В Олександрійській бібліотеці чи не вперше в історії людства виявилися зібрані літературні пам'ятники багатьох народів Близького Сходу. Тут вперше в бібліотечній практиці почали класифікувати сотні тисяч книг за галузями

знань і складати каталоги з позначенням автора і назви кожної книги.

Збільшення цих обсягів досягалося двома шляхами. З одного боку, це було безпосереднє збільшення числа самих носіїв інформації, наприклад глиняних табличок. А з іншого боку, відбувалася зміна технології зберігання інформації, що дозволяє збільшувати концентрацію інформації, що зберігається при зниженні габаритів самих носіїв (папірус – пергамент – береста – папір – перфокарта).

Дана практика активно підтримується і розвивається в сьогоднішні дні, проте в збільшених обсягах, з більш жорсткими вимогами щодо швидкості обробки і доступу до нінформаціїеї, а також її захищеності, що робить роботу над наведеною темою актуальною.

1 ВИМОГИ ТА АКТУАЛЬНІСТЬ ОРГАНІЗАЦІЇ СХОВИЩ ІНФОРМАЦІЙНИХ ОБ’ЄКТІВ

Людина і людське суспільство зобов'язані своєму становленню та розвитку двом основним факторам: праці й мові. Аналіз динаміки коштів і способів зберігання та обміну інформацією дозволяє виявити прискорені темпи розвитку сучасного суспільства. Однак, це призводить до таких проблем, як зниження швидкості обробки інформації і подорожчання сховищ інформаційних об'єктів (ІО) [1].

Інформаційним об'єктом будемо називати сукупність логічно пов'язаної інформації, що зберігається на інформаційному носії (паперовому, магнітному, електронному, лазерному…)

Існують прості і комплексні інформаційні об'єкти. Прості інформаційні об'єкти – звук, зображення, текст, число. Комплексні (структуровані) інформаційні об'єкти – елемент, таблиця, база даних, гіпертекст, гіпермедіа.

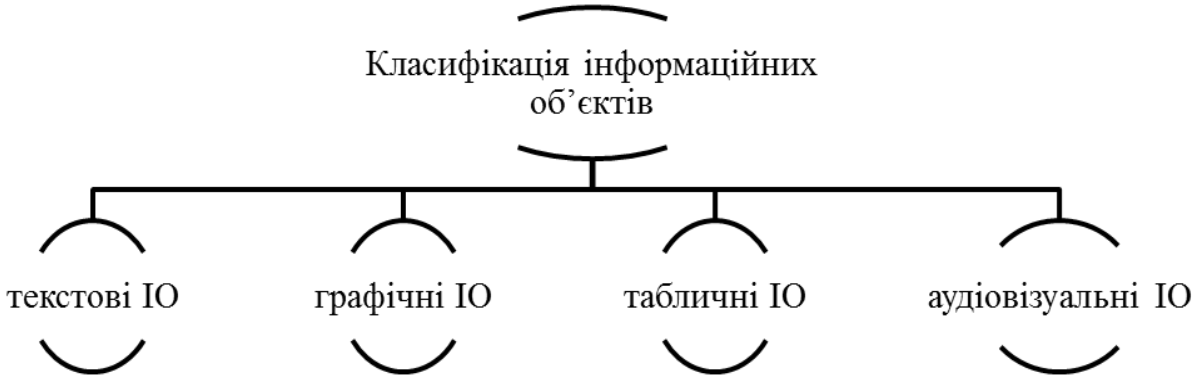

Рисунок 1.1 – Класифікація інформаційних обєктів

ІО діляться на наступні класи (рисунок 1.1):

- текстові інформаційні об'єкти – літературний твір, газетна стаття, наказ;

- графічні інформаційні об'єкти – малюнки, креслення, схеми;
- табличні інформаційні об'єкти – різні документи в табличній формі; аудіовізуальні інформаційні об'єкти – відео та музика.

Більшість ІО є складними, тобто містять інформацію, представлену в різних формах.

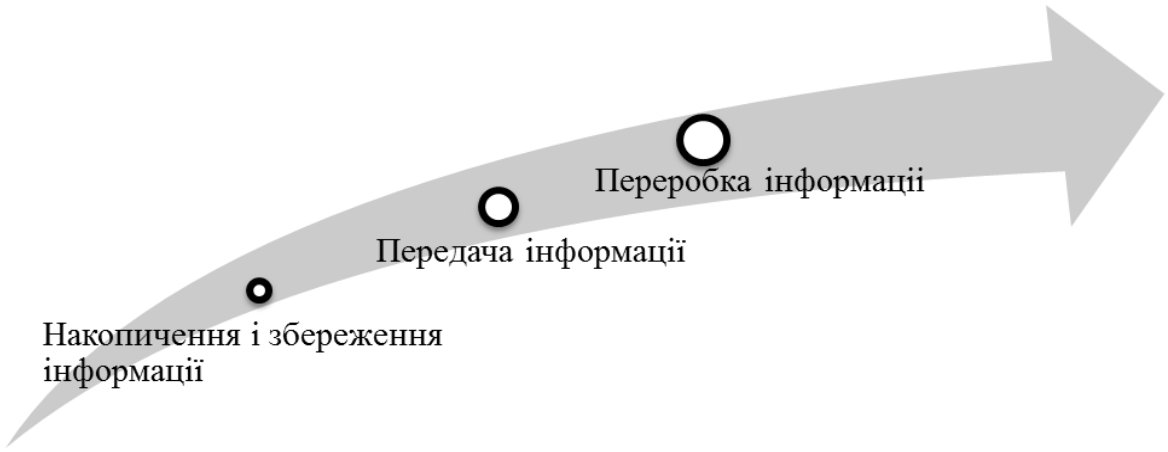

Рисунок 1.2 – Перебіг науково-освітньоі діяльності

Будь-яка науково-освітня діяльність, і університетська в тому числі, передбачає роботу зі сховищами ІО, включає в себе три основних інформаційних процеси (рисунок 1.2):

- накопичення і збереження інформації для забезпечення багаторазового використання інформації. Критеріями, що висуваються до способів зберігання інформації, є надійність і доступність, дотримання яких залежить також і від носія інформації (папір, фото- і кіноплівка, магнітні та оптичні диски і ін), що знаходиться в сховищі інформації. Сховище інформації – це певним чином організована інформація на зовнішніх носіях, призначена для тривалого зберігання і постійного використання (наприклад, архіви документів, бібліотеки, картотеки). Основні властивості сховища інформації: обсяг інформації, що зберігається, надійність зберігання, час доступу (тобто час пошуку потрібних відомостей), наявність захисту інформації;

- передача цієї інформації іншим;
- переробка інформціі, необхідна для збору найбільш повної інформації і підвищення ймовірності прийняття правильного рішення з метою генерації нового знання. Для прискорення процесу отримання якнайповнішої інформації складаються каталоги (алфавітний, предметний і ін.).

Приблизну послідовність способів поширення інформації можна уявити як: передача інформації «з уст в уста», за допомогою читання книг, за допомогою вивчення та обміну електронними інформаційними об'єктами.

Поява технології зберігання інформації у вигляді електронних файлів і її транспорт по мережі Інтернет зробила можливим створення розподілених електронних бібліотек, що, в свою чергу, призвело до створення віртуальних дистанційних університетів, де студенти і викладачі можуть бути розділені тисячами кілометрів і перебувати на різних материках.

В інженерно-лінгвістичній практиці під мовою прийнято розуміти звукову форму тексту, а під текстом - письмову форму мови.

1.1 Класифікація задач в області лінгвістичних інформаційних технологій

Основні класи задач, що зустрічаються при роботі з текстом, як способом подання інформації, такі:

- розпізнавання усної мови;
- синтез мови за текстом;
- підтримка введення тексту на електронні носії;
- машинний переклад;
- інформаційний пошук;
- компресія тексту (реферування і анотування);
- класифікація текстів;
- витяг фактів і знань;

- виявлення текстової близькості;
- питально-відповідні системи;
- діалог з комп'ютерними системами на природній мові.

Розпізнавання усної мови і синтез мови за текстом [2]. Перший пристрій для розпізнавання мови з'явилося в 1952 році (рисунок 1.3), воно могло розпізнавати вимовлені людиною цифри. Існує кілька основних способів розпізнавання мови. Розпізнавання окремих команд з невеликого заздалегідь заданого словника дозволяє досягти найвищої достовірності розпізнавання. Прикладом використання є голосова навігація по сайтам. Розпізнавання фраз, які відповідають певним заданим правилам (граматиці) широко застосовується в системах голосового самообслуговування. Пошук ключових слів в потоці злитної мови, в цьому випадку мова не повністю перетворюється в текст - в ній автоматично знаходяться лише ті ділянки, які містять задані слова чи словосполучення. Використовується в пошукових системах, в системах мо-ніторінга мови. Розпізнавання злитого мовлення на великому словнику - ця технологія найбільш близька до мрії людини про взаємодію людини і машини - все, що сказано, дослівно перетворюється в текст. Тому іноді ця технологія так і називається STT - speech to text. До кінця ця задача не вирішена ніде в світі, проте, достовірність розпізнавання вже досить висока для використання технології на практиці.

Рисунок 1.3 – Прилад для розпізнавання голосу (1952)

Підтримка введення тексту на електронні носії. Одним з перших додатків в цьому напрямку були програми автоматичного переносу слів і програми орфографічної перевірки тексту (правопис). Ці програми забезпечують корекцію на лексико-морфологічному і синтаксичному рівні, т. Е. Виробляють звірення входу зі списком допустимих структур (розпізнавання з дискретним входом) і, в разі невдачі, а також пошук найближчого відповідності. Близькі до цих завдань також розпізнавання друкованого і рукописного тексту і автозавершення.

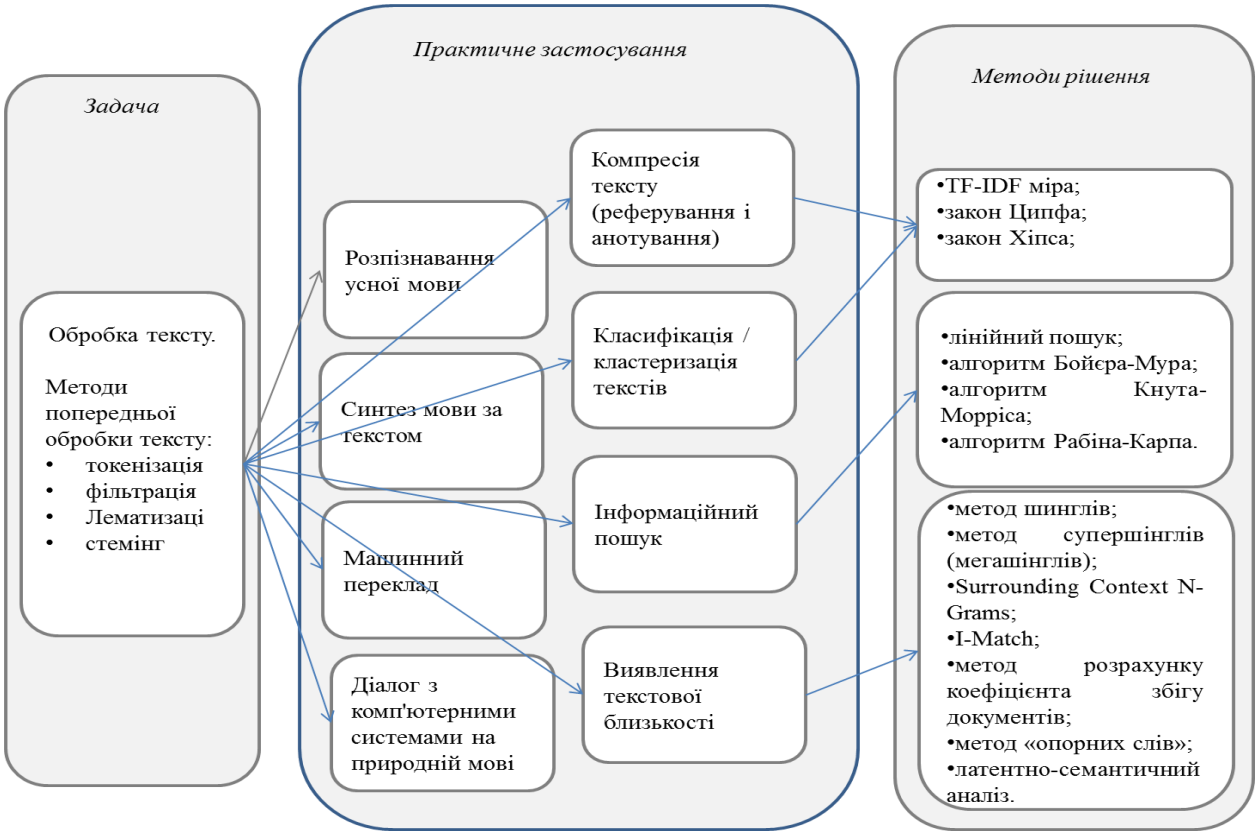

Рисунок 1.4 – Аналіз проблемної області

Машинний переклад. Перші програми перекладу були побудовані більше 50 років тому і були засновані на найпростішої стратегії послівного перекладу. Однак досить швидко було усвідомлено, що машинний переклад вимагає повної лінгвістичної моделі, що враховує всі рівні мови, аж до семантики і прагматики. В даний час існує цілий спектр комп'ютерних

систем переказу різної якості, але, не дивлячись на багато десятиліть розвитку всього цього напрямку, в цілому завдання ма-шинного перекладу ще досить далека до повного вирішення.

Інформаційний пошук. Створення пошукового образу документа передбачає індексування його тексту, тобто виділення в ньому ключових слів. Оскільки дуже часто набагато точніше тему і зміст документа відображають не окремі слова, а словосполучення, в якості ключових слів стали розглядатися словосполучення. Це істотно ускладнило процедуру індексування документів, так як для відбору значущих словосполучень тексту потрібно використовувати різні комбінації статистичних і

Компресія тексту (реферування і анотування) [3]. Вирішення цього завдання складається з двох етапів: 1) сегментація на висловлювання (частини висловлювань) і потім 2) вибір найбільш значущих (синтез). Реферування тексту – скорочення його обсягу і отримання його короткого викладу – реферату. Загальний реферат може складатися також для кількох близьких по темі документів. Основним методом автоматичного реферування є відбір найбільш значущих пропозицій реферованих тексту, для чого зазвичай спочатку обчислюються ключові слова тексту, і розраховується коефіцієнт значущості речень тексту. Близьке до реферування завдання – анотування тексту документа, тобто складання його анотації. У простій формі анотація являє собою перелік основних тем тексту, для виділення яких можуть використовуватися процедури індексування.

Класифікація текстів [4]. При створенні колекцій документів актуальні завдання класифікації і кластеризації текстів з метою створення класів близьких по темі документів. Класифікація означає віднесення кожного документа до певного класу із заздалегідь відомими параметрами, а кластеризація – розбиття безлічі документів на кластери, тобто підмножини тематично близьких документів. Дуже близька до класифікації завдання рубріціювання тексту – його віднесення до однієї з заздалегідь відомих тематичних рубрик (зазвичай рубрики утворюють ієрархічне дерево

тематик). Завдання класифікації набуває все більшого поширення, вона вирішується, наприклад при розпізнаванні спаму, а порівняно новий додаток - класифікація SMS-повідомлень в мобільних пристроях.

Використання рубрикаторів-класифікаторів дозволяє скоротити трудомісткість на пошук потрібної інформації, представленої електронними текстами.

Витяг фактів і знань (Information Extraction). Витяг інформації з текстів часто потрібно робити при вирішенні завдань економічної та виробничої аналітики. Для цього здійснюється виділення в тексті певних об'єктів - іменованих сутностей (імен, персоналій, географічних назв), їх стосунків і пов'язаних з ними подій. Як правило, це реалізується на основі часткового синтаксичного аналізу тексту, що дозволяє виконувати, наприклад, обробку потоків новин від інформаційних агентств.

Виявлення текстової близькості полягає в необхідності пошуку відстані між документами. Виникає в різних завданнях, таких як виявлення плагіату, визначення авторства документа, пошук інформації, машинний переклад, формування тестів і завдань, автоматична побудова рефератів.

Питально-відповідні системи (Question Answering). Це завдання вирішується шляхом визначення типу питання, пошуком текстів, потенційно містять відповідь на це питання, і витяганням відповіді з цих текстів.

Завдання реферування, виділення феноменів і понять, класифікації та кластеризації, відповідей на запити, тематичного індексування і пошуку за ключовими словами прийнято відносити до технологій Text Mining, тобто інтелектуального аналізу текстів.

Діалог з комп'ютерними системами на природній мові. Найбільш часто це завдання вирішувалося для спеціалізованих баз даних - в цьому випадку мова запитів досить обмежений (лексично і граматично), що дозволяє використовувати спрощені моделі мови. Запити до бази, сформульовані на природній мові, переводяться на формальну мову, після чого виконується пошук потрібної інформації і будується відповідна фраза відповіді.

1.1.1 Методи виявлення текстової близькості

При аналізі тексту для виявлення текстової близькості розглянутого ІО із заздалегідь класифікованими ІО, можуть бути використані наступні методи:

- метод шинглів;
- метод супершінглів (мегашінглів);
- Surrounding Context N-Grams;
- I-Match;
- метод розрахунку коефіцієнта збігу документів;
- метод «опорних слів»;
- латентно-семантичний аналіз, заснований на розкладанні матриці по методу сингулярного розкладання (SVD – singular value decomposition).

Основними критеріями для аналізу перерахованих методів є стійкість до модифікацій тексту, точність і повнота алгоритму.

Розглянемо деякі з них.

В основі методу шинглів лежить розбиття текстів на групи слів однакової довжини і наступне порівняння їх хешів.

Був запропонований в 1997 році A. Broder [5]. Заснований на представленні документа у вигляді послідовностей фіксованої довжини N, що складаються з сусідніх слів. При цьому на послідовності можуть накладатися обмеження, наприклад, слова повинні знаходитися в одному реченні. Такі послідовності в одних джерелах називають "шинглі", в інших "N-грамами" [4]. Два документа вважаються схожими, якщо безлічі їх N-грам істотно перетинаються. Аналогічно можна вважати схожість двох пропозицій, або ж пропозиції і тексту. Число N-грам для кожного документа є досить великим, використовуються різні способи усічення їх безлічі, наприклад автором був запропонований наступний метод для зменшення числа шинглів: залишати тільки ті шингли, для яких статистичні функції приймають фіксовані значення. Алгоритм не може використовуватися для

визначення конкретних запозичень, так як будь-яка модифікація тексту при запозиченні призведе до того, що шингл зміниться, і випадок запозичення виявлений не буде, що призведе до зменшення повноти. Алгоритм може використовуватися для відбору документів на основі того, чи вважаються перевіряється документ і можливий документ-джерело схожими, але в разі використання цього алгоритму повнота буде низькою.

Модифікація алгоритму шинглів від D. Fetterly [6] полягає в тому, що кожен документ представлявся 84 шинглі. Вибір зі всієї безлічі шинглів відбувається за такою схемою: для всіх шинглів документа розраховується значення 84 хеш функцій. Для кожної хеш функції вибирається шингл з максимальним значенням хеш функції. Потім ці 84 шингли розбиваються на 6 груп по 14 шинглів. Такі групи називаються 9 "супершінглів". Далі документ видається всілякими попарними поєднаннями з 6 супершінглів, які називаються "мегашінглів". Число таких мегашінглів дорівнює 15 (число сполучень з 6 по 2). Два документа подібні за змістом, якщо у них збігається хоча б один мегашінглів. Алгоритм є нестійким при модифікації тексту при запозиченнях, так як при модифікації тексту, відповідного шинглів зміниться також і мегашінглів. Також алгоритм не може виявити запозичення в разі малого збіги документів: в разі збігу документів на 96 відсотків 2 супершінглів збігаються з ймовірністю 49 відсотків. Відповідно, запозичення одного абзацу в двох відносно великих статтях буде недостатнім для виявлення цим алгоритмом. Описаний алгоритм застосуємо для відбору документів для пошуку, якщо критерієм відбору документів є масштабні запозичення, для пошуку конкретних запозичених фрагментів алгоритм застосуємо, так як мегашінглів містить лише невелику підмножину шинглів документа, і, відповідно, не може містити інформацію про всіх можливих запозиченнях. Також алгоритм не є придатним через нестійкість до модифікацій:

Алгоритм Surrounding Context N-Grams є модифікацією алгоритму шинглів, запропонованої Diego A. Rodríguez Torrejón і José Manuel Martín

Ramos [7]. В основу алгоритму береться припущення про те, що текст може модифікуватися шляхом видалення одиничних слів, граматичних помилок, перестановкою слів, і стандартний алгоритм шинглів не виявить збіги. Для цього збільшується число генеруються шинглів. Для кожної послідовності слів N береться певна кількість M, менше N, і з послідовності слів N забирається M слів всілякими способами. Решта слова упорядковано в лексикографічному порядку і являють собою нові N-грами. Недоліком цього алгоритму є істотне зростання числа шинглів, що призводить до істотного уповільнення роботи програми. Також це призводить до збільшення числа помилкових спрацьовувань. Описаний алгоритм є стійким до невеликих модифікацій тексту (якщо модифікуються не більше M слів поспіль), і застосуємо для пошуку конкретних запозичень, проте не може бути застосованим при модифікації більш М слів поспіль. Задати константу M чималий не є можливим, так як це призведе до серйозного зменшення точності алгоритму. Також може застосовуватися для попереднього відбору документів за ознаками, аналогічним стандартним алгоритмом шинглів.

Жоден з описаних алгоритмів не задовольняє повністю одночасно всім поставленим вимогам, тобто не є стійким до довільних модифікацій тексту, і дозволяє проводити попередню фільтрацію документів з подальшим визначенням конкретних запозичених фрагментів. Найбільш повно задовольняє поставленим вимогам алгоритм Surrounding Context N-Grams, який стійкий до деяких видів модифікацій.

1.1.2 Методи пошуку тексту в документі

Зі збільшенням різноманіття текстових редакторів, розширенням її функціональних можливостей, зростанням обсягів інформації, яка зберігається та передається по мережі, роль задачі швидкого пошуку слова-образу у тексті стає більш актуальною. Основним застосуванням алгоритмів пошуку може бути контекстний пошук у базах та банках даних,

бібліографічний пошук, пошук фрагменту тексту та його заміна при редагуванні тексту, у задачах стиску даних, алгоритмах прогнозування тощо (рисунок 1.5). Саме внаслідок описаної важливості рішення задачі пошуку слів-образів у тексті, цією задачею займається велика кількість дослідників у різних країнах світу [1,14].

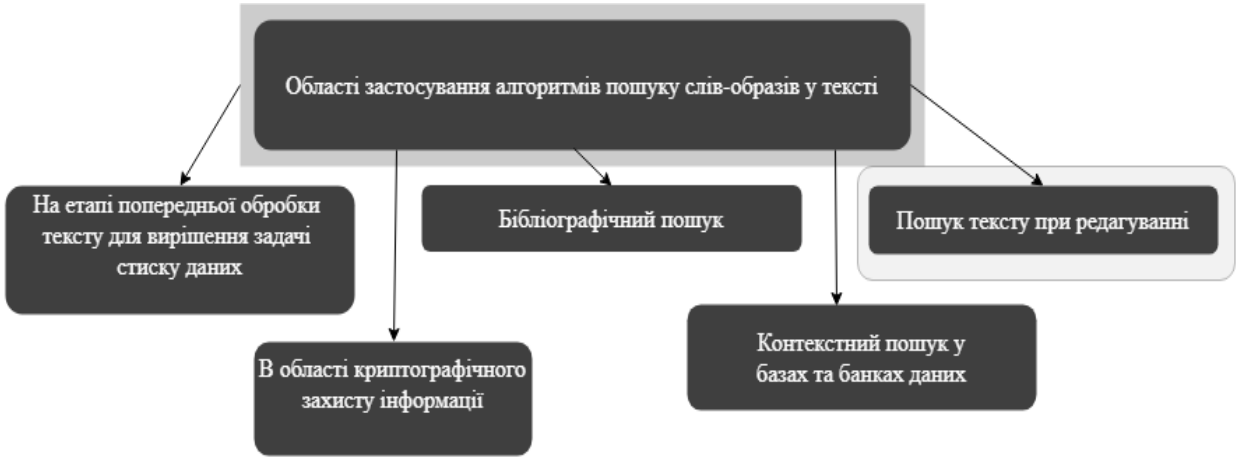

Рисунок 1.5 – Області застосування алгоритмів пошуку слів-образів у тексті

Серед існуючих алгоритмів пошуку слів-образів у текст можна виділити наступні алгоритми (рисунок 1.6):

- лінійний пошук;
- алгоритм Бойєра-Мура (БМ);
- алгоритм Кнута-Морріса;
- алгоритм Рабіна-Карпа (РК).

У роботі розглядаються такі області застосування алгоритмів пошуку, як швидкий пошук слів-образів у тексті для використання у текстових редакторах та процесорах на етапі редагування тексту (пошук тексту або пошук та заміна тексту).

Словом-образом будемо називати послідовність символів або букв деякого алфавіту, яке необхідно знайти в тексті. Для рядку важливий склад і кількість символів в ньому, а також порядок проходження символів в рядку.

Довжина рядка – це кількість символів в ньому. Наприклад, слово "ababcaba" має довжину 8. Порожній рядок – це рядок, що не містить символів, тобто має нульову довжину.

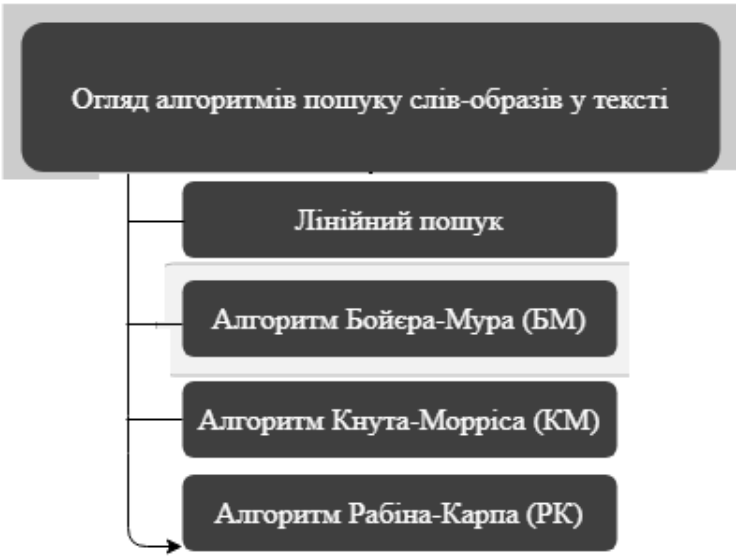

Рисунок 1.6 – Огляд алгоритмів пошуку слів-образів у тексті

Під операцією компарації розуміється порівняння символів шуканого слова і тексту для виявлення їх збігу або розходження в процесі ідентифікації рядка. Два рядки є співпадаючими, якщо вони мають один і той же склад символів, рівну довжину і однаковий порядок проходження символів. Рядки "aba" і "aab" не збігаються, тому що, хоча їх склад і довжина однакові, але порядок символів різний. Аналогічно, рядки "aaa" і "aaaaa" теж різні, оскільки мають різну довжину.

Будь-яка послідовно взята частина рядка є її підсловом. Будемо позначати символи слова буквами W з індексами, відповідними номерами цих символів в слові. При цьому символи з різними номерами можуть збігатися, тобто літери в рядку не обов'язково всі повинні бути різними. Наприклад, слово «ababcaba» $= W_1 W_2 \dots W_7 W_8$, де довжина слова визначається як $j = (\overline{1,l})$. Символи тексту будемо позначати буквами T з

відповідними індексами: $T_1T_2 \dots T_y$, де довжина тексту визначається як $i = (\overline{1,t})$.. Входження заданого образу в певний текст вважається знайденим, якщо він є підсловом цього тексту і визначено його початок в тексті.

Трудомісткість алгоритмів пошуку слів-образів у тексті визначається кількістю компарації символів у шуканому слові-образі довжиною l із вихідним текстом довжиною t, яка має бути виконана у процесі роботи алгоритму. Середня трудомісткість розглянутих існуючих алгоритмів наведена у таблиці 1.1.

Таблиця 1.1 – Середня трудомісткість існуючих алгоритмівпошуку слів-образів у тексті

Алгоритм пошуку слів-образів у тексті	Середня трудомісткість алгоритму
Лінійний пошук	*O((t-l)×l)*
Алгоритм Бойєра-Мура (БМ)	*O(t/l))*
Алгоритм Кнута-Морріса	*O(t+l)*
Алгоритм Рабіна-Карпа (РК)	*O((t-l)×l)*

1.1.3 Метод «мішку слів»

Стандартними способами подання документів є векторне подання, або, так званий, мішок слів [7]. Модель мішка слів – це спосіб представлення текстових даних при моделюванні тексту за допомогою алгоритмів машинного навчання.

Модель «мішок слів» проста для розуміння і реалізації і має великий успіх в таких проблемах, як моделювання мови та класифікація документів.

Модель мішка слів, або скорочено BoW (Bag of Words) – це спосіб вилучення особливостей з тексту для використання в моделюванні, наприклад, в алгоритмах машинного навчання.

Підхід дуже простий і гнучкий, і його можна використовувати безліччю способів для вилучення найчастіше вживаних слів в документах, а також ключових.

Мішок слів – це уявлення тексту, який описує входження слів в документ. Включає в себе дві речі:

- словник відомих слів;
- міра наявності відомих слів.

Метод називається BoW тому, що будь-яка інформація про порядок або структуру слів в документі відкидається. Модель стосується тільки того, чи зустрічаються в документі відомі слова, а не де в документі.

Формальна постановка задачі виглядає наступним чином:

Нехай $f_1, \dots, f_m$ множина, яка складається із m ознак (атрибутів), які можуть бути присутні в документі; нехай $n_i(d)$ – кількість потраплянь ознаки f_i в документ d. Далі кожен документ d представляється у вигляді вектору наступним чином:

$$d = (n_1(d), n_2(d), \dots, n_m(d))$$

Виділяють два основних типи атрибути:

- частотні кожне значення в векторі d відповідає кількості потраплянь ознак в докумнет d; тотді $ni(d) \in (0; +\infty)$;
- бінарні (наявності/відсутності), кожне значення у векторі d бінарне (true/false або 0/1) і відображає факт присутності ознак fi в документі d. Тоді $ni(d) = \{0,1\}$.

Далі документи, представлені у вигляді векторів своїх ознак (атрибутів), використовуються для навчання класифікатора, реалізованого за допомогою одного з методів машинного навчання.

Роль метода в задачах класифікації велика і важлива, оскільки документи схожі, якщо вони мають схожий зміст.

1.2 Рівнева організація комп’ютерного аналізу природно-мовного тексту

При виконанні аналізу ІС, виділяють такі рівні:

- фонетичний рівень;
- графематичний рівень;
- морфологічний рівень;
- синтаксичний рівень;
- семантичний рівень.

На фонетичному рівні виконується аналіз мовних звуків і звукового будови мови (склади, звукосполучення, закономірності з'єднання звуків в мовний ланцюг).

Етап графематичного аналізу призначений для виділення елементів структури тексту: параграфів, абзаців, пропозицій, окремих слів і так далі. У завдання графематіческого аналізу входять:

- виділення абзаців, заголовків, приміток;
- виділення пропозицій з вхідного тексту;
- поділ вхідного тексту на слова, цифрові комплекси, формули, тобто токенізація;
- збірка слів, написаних в розрядку;
- виділення стійких оборотів, які не мають словозмінних варіантів;
- виділення ПІБ (прізвище, ім'я, по батькові), коли ім'я та по батькові написані ініціалами;
- виділення іноземних лексем, записаних латиницею;
- виділення електронних адрес і імен файлів.

Мінімальні лінійні компоненти тексту, кото-які в подальшому розглядаються як неподільні одиниці, називаються токенами.

На етапі морфологічного аналізу обробляються окремі слова, в них виділяються основи і флексії (змінювані частини слів) – приставки, суфікси,

закінчення. Надалі флексії використовуються для встановлення граматичних відносин між словами в межах одного речення.

Морфологічний аналіз забезпечує визначення нормальної форми, від якої була утворена дана словоформа, і набору параметрів, приписаних даній словоформі. Нормальна форма (називний відмінок для іменників, інфінітив для дієслів і так далі) називається леммою, а сам процес визначення лем - лематизації. Лематизації проводиться для того, щоб орієнтуватися в подальшому тільки на нормальну форму, а не на всі словоформи, використовуючи параметри, наприклад, для перевірки узгодження слів.

При синтаксичному аналізі по-перше, колекцію треба розділити на складові частини меншої довжини. В результаті синтаксичного аналізу лінійна послідовність токенів (слів) перетворюється в набір синтаксичних відносин. Граматично побудовані пропозиції є зв'язковими, тобто позбавленими розривів в ланцюжку синтаксичних відносин. Відносини є бінарними. Важлива особливість синтаксичних залежностей полягає в тому, що вони далеко не завжди пов'язують слова, що знаходяться поруч в ланцюжку.

Семантичний аналіз забезпечує можливість визначення значення слова в залежності від контексту і конкретної ситуації, розуміння сенсу фрази.

1.3 Обгрунтування значущості попередньої обробки тексту

Передобробка тексту – один з ключових етапів більшості алгоритмів інтелектуального аналізу текстів. Класична методологія категоризації текстів включає в себе етапи предобработки, вилучення ознак і класифікації. Незважаючи на те, що, як було показано в роботах [8], [9], [10], витяг ознак, їх відбір і метод класифікації вносять значний вклад в процес класифікації, попередня обробка може серйозно вплинути на її результат. Етап попередньої обробки складається з токенізації, фільтрації, лематизації і стемінг.

Токенізація: розбиття послідовності символів на частини (слова / фрази), звані токенами. Також може включати в себе видалення певних символів, наприклад, знаків пунктуації.

Фільтрація полягає у видаленні деяких слів з тексту. Найпоширеніший вид фільтрації - видалення стоп-слів. Під стоп-словами розуміються такі слова, які часто зустрічаються в тексті і не несуть змістовної інформації (прийменники, сполучники і тому подібне).

Лематизація включає в себе морфологічний аналіз слів, при якому різні форми слова групуються для того, щоб їх можна було обробляти як один об'єкт. При лематизації документів для кожного слова необхідно визначити частину мови. Так як визначення частини мови дуже складний процес, схильний до помилок, на практиці частіше користуються методами стемінг.

Стемінг – процес знаходження основи слова, яка не обов'язково збігається з його морфологічним коренем. Алгоритми стемінг залежать від мови. Перший алгоритм для англійської мови був запропонований в 1968 році [11]. Найбільш поширеним на сьогоднішній день є Стеммер Портера [12]. Опублікований в 1980 році, оригінальний алгоритм був призначений для англійської мови, але згодом автором були запропоновані Стеммер для поширених індоєвропейських мов, в тому числі для російської мови.

Подання текстів у вигляді векторів з деякого загального для всіх текстів векторного простору є основним етапом попередньої бработкі тексту, оскільки дозволяє розглянути текст не як набір токенів або символів, а в більш зручному для розуміння комп'ютера вигляді - вигляді вектора [13]. Цей підхід є одним з основних інструментів в області інтелектуального аналізу текстів, інформаційного пошуку, класифікації та кластеризації текстових документів.

Кожна координата вектора в рамках моделі відповідає окремому терму. Визначення терма залежить від сфери застосування і в його ролі можуть виступати окремі слова, групи слів, комбінації цифр і букв. Якщо терм присутній в документі, то відповідне значення в векторі відмінно від нуля.

Існує кілька стандартних способів підрахунку цих значень, відомих також як ваги термів. Це може бути булева вага, рівна 1, якщо терм зустрівся в документі і 0 в іншому випадку. Інший варіант - кількість входжень терма в документ. У класичній векторної моделі, запропонованої Селтоном і ін. [14], ваги термів є множина локальних і глобальних параметрів. Така модель відома як tf-idf (англ. Term frequency – inverse document frequency, частота терма – зворотна частота документа).

Деякі слова можуть зустрічатися майже у всіх документах деякої колекції і, відповідно, надавати малий вплив на належ-ність документа до тієї чи іншої категорії, а значить не бути ключовими для цього документа. Для зниження значущості слів, які зустрічаються майже у всіх документах, вводять инверсную частоту терміна *IDF* (inverse document frequency) – це логарифм відношення числа всіх документів *D* до числа документів *d*, що містять деяке слово.

$$IDF = \lg D \times d$$

Значення цього параметра тим менше, чим частіше слово зустрічається в документах колекції. Таким чином, для слів, які зустрічаються у великій кількості документів, *IDF* буде близький до нуля (якщо слово зустрічається у всіх документах *IDF* дорівнює нулю), що допомагає виділити важливі слова.

Параметр *TF* (term frequency) – це відношення числа раз *k*, яке деяке слово зустрілося в документі, до загальної кількості слів у документі *n*. Нормалізація довжини документа потрібна для того, щоб зрівняти в правах короткі і довгі документи.

$$TF = k \times n_i$$

Коефіцієнт TF * IDF дорівнює добутку TF і IDF, при цьому TF грає роль підвищує множника, IDF - понижуючого. Тоді ваговими параметрами векторної моделі деякого документа можна прийняти коефіцієнти TF * IDF

входять до нього слів. Для того щоб ваги знаходилися в інтервалі (0, 1), а вектори документів мали рівну довжину, зна-чення TF * IDF зазвичай нормалізуються.

Відзначимо, що ця формула оцінює значимість терміна тільки з точки зору частоти входження в документ, тим самим не враховуючи порядок проходження термінів в документі і їх синтаксичну роль; іншими словами, семантика документа зводиться до лексичної семантиці назв термінів, а композиційна семантика не розглядається.

Ключовими в даному випадку будуть слова набрав найбільшу вагу. Слова з малою вагою, взагалі можна не враховувати при класифікації.

Таким чином, терм матиме велику вагу, якщо в деякому тексті він зустрічається часто, а в інших – рідко. З іншого боку, для поширених термів ваги будуть невеликими.

Таблиця 1.2 – Категорії текстів корпусу Брук

Іденти-фікатор	Категорія	Відсот-кове відно-шення	Опис текстів
1	2	3	4
1 – A	Преса	25%	Репортажі, огляди, статті, листи в редакцію; національні і регіональні видання; тематичні: політика, спорт, суспільство; економіка і фінанси, короткі новини; культура: театр, література, музика, танці
2 – B	Релігійна література	3%	Книги, ппріодика, брошури

Продовження таблиці 1.2

1	2	3	4
3 – C	Професійно-популярна література	7%	Книги і періодика : домоводство, ремесла, "сад і город", хобі, ремонт і будівництво, конструювання, музика і танці, домашні тварини, спорт, їжа і вино, подорожі, фермерство, робочі професії і тому подібне.
4 – D	Естетичні інформативні тексти	7%	Інформативні тексти, що не потрапляють в інші категорії, зокрема, біографії, мемуари, есе, передмови, особисті листи, художня, критика, рекламні тексти
5 – E	Адміністративні документи	3%	Закони, урядові акти, звіти організаций/фондов/ компаній, офіційні листи
6 – F	Науково-популярна література	5%	Наукові журнали, періодика, курсові роботи, дипломні роботи
7 – G	Наукова література	10%	Книги і періодика; природні і гуманітарні науки, техніка і інженерна справа
8 – H	Учбова література	15%	Підручники, посібники, гуманітарні і природні науки і т.д
9 – I	Художні тексти	25%	Романи, повісті, оповідання, новели, з тематики: детективи, фантастика, пригодницька, любовна, гумористична і так далі

Для моделювання колекції документів вектора, відповідні документам, групують в матрицю так, що рядок визначає терм, а кожен стовпець відповідають деякому документу.

Послідовність N елементів (символів, термів, звуків, складів) називається N-грамою. N-програмних моделі використовуються для широкого кола досліджень і розробок в області обробки природної мови, як, наприклад, розпізнавання мови, машинний переклад, вилучення інформації.

Попередньо оброблений текст може бути класифікований, використовуючи навчений класифікатор. Оскільки мова йде про українську мову, то навчання класифікаторів має певні обмеження. А саме – відсутність великих корпусів у різних тематиках. Серед відомих корпусів можна виділити:

- Брауновский корпус української мови [2];
- великий електронний словник украинкого мови [3];
- російсько-український словник [4];
- LanguageTool API NLP UK [5].

Брауновський корпус української мови (Брук) знаходиться у вільному доступі. Це дає можливість безперешкодній роботі з накопиченою базою. Корпус налічує 730 документів - файлів формату .txt з текстом і описом інформації про нього. Також кожен файл розпочинається з певної букви, що означає клас, до якого належить текст. Кількість класів Брук фіксована і рівна дев'яти. Категорії текстів у Брук і їх процентне наповнення надані в таблиці 1.2.

Українська інтерпретація оригінального проекту Брауновского корпусу зазнала значні зміни у зв'язку з необхідністю адаптувати категоризацію текстів під язик слов'янської групи, який в лінгвістичному аспекті має значні відмінності від груп германських мов.

2 РОЛЬ ВИСОКОПРОДУКТИВНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ У ПРИСКОРЕННІ МЕТОДІВ ОБРОБКИ ТЕКСТУ

2.1 Апаратний паралелізм

Підвищення продуктивності обчислювальної системи за рахунок збільшення числа процесорів є не новим підходом. Перші промислові зразки мультипроцесорних систем з'явилися на базі векторно-конвеєрних комп'ютерів в середині 80-х років. В даний час існує безліч різних варіантів організації паралельних обчислень [15], які можна розбити на наступні класи.

Масивно-паралельні системи (MPP) – складаються з однорідних обчислювальних вузлів, що включають один або кілька процесорів і локальну пам'ять (прямий доступ до пам'яті інших вузлів неможливий). Вузли системи пов'язані через деяку комутаційну середу. Програмування здійснюється в рамках моделі передачі повідомлень (наприклад, MPI).

Кластерні системи – це дешевий варіант MPP. Такі системи складаються з набору персональних комп'ютерів (робочих станцій) загального призначення. Для зв'язку обчислювальних вузлів використовуються стандартні мережеві технології. Програмування, як і в попередньому класі, здійснюється в рамках моделі передачі повідомлень, зазвичай MPI. Реалізації систем даного класу дешевше, але мають вроджену затримку через використання стандартних мережевих інтерфейсів.

Grid (обчислювальна мережа) – це об'єднання різнорідних обчислювальних ресурсів в мережах (наприклад, Інтернет). Цей клас реалізує розподілені обчислення. Обчислення проводяться на віддалених один від одного пристроях, що породжує великі затримки при передачі даних між пристроями.

Паралельні векторні системи (PVP). Складаються з обчислювальних вузлів, кожен з яких містить кілька векторно-конвеєрних процесорів. Вузли можуть бути пов'язані комутаційної середовищем.

Симетричні мультипроцесорні системи (SMP) – це система з декількох однорідних процесорів і масиву загальної пам'яті, до якої мають доступ усі процесори. Хоча загальна пам'ять і спрощує взаємодію процесорів системи, але, з іншого боку, обмежує їх кількість в реальних системах. Програмування здійснюється в рамках моделі загальної пам'яті, як правило, це технологія OpenMP.

Системи з неоднорідним доступом до пам'яті (NUMA). Така система складається з декількох однорідних базових модулів, кожен з яких має кілька процесорів і блок пам'яті. Модулі об'єднані між собою. Не дивлячись на фізичний поділ пам'яті, логічно вона є відкритою. Програмування аналогічно SMP-систем.

«Хмарні» обчислення – обчислення, що проводяться в готовій інфраструктурі, до якої є доступ через мережу. Інфраструктура може складатися з тисяч, сотень тисяч обчислювальних вузлів, дискових масивів. Все це пов'язано в єдину мережу і функціонує як одна обчислювальна машина. «Хмарні» обчислення представляють собою динамічно масштабується спосіб доступу до зовнішніх обчислювальних ресурсів у вигляді сервісу, що надається за допомогою Інтернету, при цьому користувачеві не потрібно ніяких особливих знань про інфраструктуру «хмари» або навичок управління «хмарної» технологією.

Поряд з розбивкою на класи паралельних систем, можна класифікувати і паралельні архітектури. Спроби такої класифікації приймалися неодноразово, найбільш відомою є класифікація М. Флінна, запропонована в 1966 році [16]. Дана класифікація базується на понятті потоку, під яким розуміється послідовність команд або даних, які обробляються процесором.

2.2 Програмний паралелізм

Умовно способи розпаралелювання можна розділити на наступні:

- директиви оптимизирующего компілятора;
- спеціальні директиви, що розширюють можливості мови до паралелізації;
- паралельні мови програмування;
- комунікаційні засоби або засоби межпроцессорного інтерфейсу.

Здатність алгоритму до розпаралелювання потенційно пов'язана з одним з двох (або одночасно з обома) внутрішніх властивостей, які характеризуються як паралелізм завдань (message passing) і паралелізм даних (data parallel).

Якщо алгоритм заснований на паралелізм завдань, обчислювальна задача розбивається на кілька відносно самостійних підзадач, кожна з яких завантажується в окремий процесор. Кожна подзадача реалізується незалежно, але використовує спільні дані і (або) обмінюється результатами своєї роботи з іншими подзадачами. Для реалізації такого алгоритму на багатопроцесорної системі необхідно виявляти незалежні підзадачі, які можуть виконуватися паралельно. Часто це виявляється далеко не очевидною і досить важким завданням.

При наявності в алгоритмі властивості паралелізму даних одна операція може виконуватися відразу над всіма елементами масиву даних. У цьому випадку різні фрагменти масиву можуть оброблятися незалежно на різних процесорах. Для алгоритмів цього типу розподіл даних між процесорами зазвичай здійснюється до виконання завдання на ЕОМ.

2.3 Огляд GPGPU технології

Довгий час основним і єдиним пристроєм, що виконує операції (інструкції), був центральний процесор. Пізніше з'явилися багатоядерні

процесори і багатопроцесорні системи, в яких обчислювальних компонентів було кілька [17]. Це дозволило машинам виконувати кілька завдань одночасно, а загальна (теоретична) продуктивність системи піднялася рівно в стільки разів, скільки ядер було встановлено в машині, чого не можна сказати про фактичну продуктивності багатопроцесорних комп'ютерів.

Більш того, виявилося, що виробляти і конструювати багатоядерні процесори занадто складно і дорого. У кожному ядрі необхідно розміщувати повноцінний процесор x86 або x64-архітектури зі своїм кешем, конвеєром команд, блоками SSE і іншими елементами.

В силу цих та інших проблем, напрямок збільшення обчислювальної потужності за рахунок зростання кількості ядер центрального процесора можна вважати безперспективним.

Графічні процесори позбавлені багатьох недоліків центрального процесора. Популярна на сьогоднішній день технологія використання графічного процесора для загальних обчислень (GPGPU) дозволяє використовувати потокові процесори для неграфічних обчислень. Хоча дана технологія і має обмежене коло застосування, GPU пристрої зробили крок далеко вперед центрального процесора в плані нарощування сумарною обчислювальною потужності і загальної кількості ядер.

По суті, основне завдання GPU зводиться до математичних розрахунків за допомогою простих алгоритмів, які отримують на вхід передбачувані дані, і близька по ідеології RISC архітектурі CPU.

Технологія GPGPU реалізована декількома виробниками [18]:

- Khronos Group: OpenCL - мова програмування завдань загального призначення, пов'язаних з обчисленнями на різних графічних і центральних процесорах;
- Microsoft: DirectCompute - інтерфейс програмування додатків, який входить до складу DirectX;

- Advanced Micro Devices: AMD FireStream - технологія GPGPU, що дозволяє реалізовувати обчислювальні алгоритми, що виконуються на графічних процесорах прискорювачів ATI;
- nVidia: CUDA - технологія GPGPU, що дозволяє реалізовувати на мові Сі (або іншими мовами програмування) обчислювальні алгоритми, що виконуються на графічних процесорах прискорювачів GeForce восьмого покоління і старше.

GPU CUDA і ATI Stream виконані з урахуванням прямого доступу до апаратних можливостей відеокарт [19]. Реалізація технології GPGPU CUDA від компанії nVidia на сьогоднішній день є фаворитом в даній області. Інші технології, в тому числі і ATI Stream, поки не мають такої розвинутої апаратної і програмної реалізації, як у nVidia. Постійний технічний прогрес в розвитку графічних чіпів загального призначення, а також стрімкий розвиток nVidia CUDA SDK дозволяють вважати компанію nVidia лідером в області високопродуктивних обчислень з використанням GPGPU.

CUDA - це архітектура паралельних обчислень на GPU компанії nVidia, що підтримують технологію GPGPU обчислень загального призначення [20].

Згідно вище наведеної класифікації, обчислювальну архітектуру CUDA можна віднести до класу SIMD, але компанія nVidia використовує термін SIMT - Single Instruction / Multiple Thread. На відміну від SIMD систем, де вхідний потік даних, що складається з однорідних елементів, обробляється однією інструкцією, SIMT-системи ділять вхідні дані на безліч дрібних підзадач, кожна з яких обробляється своєї ниткою. Нитки виконуються паралельно на обчислювальних модулях, в якості яких використовується набір потокових мультипроцессоров.

Мультипроцессор – це багатоядерний SIMD-процесор, що дозволяє в кожний певний момент часу виконувати на всіх ядрах тільки одну інструкцію, при цьому кожен потоковий процесор виконує цю інструкцію над власними даними.

Для виконання на GPU завдання розділяється програмістом на безліч потоків (thread), які об'єднуються в блоки (block), а блоки, в свою чергу, об'єднуються в сітку (grid). Процедура, яка виконується кожним з потоків, називається ядром (kernel). Обчислювальні ресурси між потоками розподіляються драйвером CUDA. Логічно пристрій можна уявити як набір мультипроцессоров плюс драйвер CUDA.

Терміном хост (host) позначається частина програми, що виконується на CPU, а пристрій (device) – це відеокарта або інше обчислювальний пристрій, що підтримує CUDA.

Для ідентифікації потоків і блоків використовуються індекси, за допомогою яких програміст визначає, з якою саме частиною даних працює конкретний потік. Для зручності індексація може бути одно-дво-або тривимірної. Максимальні індекси, які можуть використовуватися для блоку потоків – (512, 512, 64), при цьому кількість потоків в блоці не повинно перевищувати 512. Максимальні розміри сітки становлять (65535, 65535, 1), тобто фактично мережа потоків двумірна.

Виконуваної одиницею CUDA-програми є warp. Розмір warp становить 32 потоку. Пов'язано це з тим, що затримка (latency) при виконанні однієї інструкції на мультипроцесорі становить 4 такту. Тільки відносно warp можна говорити про паралельному виконанні потоків, ніяких інших припущень робити не можна.

Однак це не означає, що warp-и виконуються на мультипроцесорі послідовно. Виконання warp-ів може бути паралельним, наприклад, в тому випадку, коли один warp очікує дані з глобальної пам'яті, інші warp-и можуть в цей час виконуватися. Взаємодія між потоками може здійснюватися тільки всередині блоку. Обмін даними здійснюється за допомогою розділяється (shared) пам'яті, загальної для всіх потоків в блоці.

Ще один ключовий момент архітектури CUDA – легка масштабованість. Одного разу написаний код буде запускатися на всіх пристроях, що підтримують CUDA. Для розробки і налагодження коду для

запуску на GPU можна використовувати звичайні відеокарти, які за вартістю доступні кожному. А коли продукт готовий – запускати його вже на потужних Tesla або кластерах, побудованих на архітектурі GPU.

2.3.1 Додаткові бібліотеки комп’ютерної лінгвістики

Існує досить багато бібліотек, за допомогою інтерфейсу яких можливо легко та швидко будувати моделі машинного навчання. Найбільш популярними бібліотеками для машинного навчання саме в області обробки тексту є Scikit-learn, XGBoost, LightGBM, Tensorflow, PyTorch, spacy, nltk, word2vec.

Scikit-learn[37] надає доступ до найбільш широкого спектру класичних моделей та алгоритмів обробки тексту. Інші бібліотеки машинного навчання мають досить складний інтерфейс для розробки алгоритмів та містять невеликий спектр моделей які належать тільки до одного сімейства моделей. Також ці бібліотеки мають досить сильну підтримку тільки деяких сферах використання, наприклад як обробка тексту або сфера комп’ютерного зору.

Scikit-learn (раніше Scikits.learn і також відомий як sklearn) - це безкоштовна бібліотека машинного навчання для мови програмування «Python». Вона містить різні алгоритми класифікації, регресії та кластеризації, включаючи «SVM», «Random Forest», «Gradient Boosting Trees», «k-means» та «DBSCAN», та призначений для взаємодії з числовими та науковими бібліотеками «Python» «NumPy» та «SciPy».

Scikit-learn написаний на «Python», і широко використовує бібліотеку «Numpy» для високоефективних операцій лінійної алгебри та масивів. Крім того, деякі основні алгоритми написані використовуючи технологію «Cython» для підвищення швидкості алгоритмів. «SVM» реалізовані обгорткою «Cython» навколо «LIBSVM»; логістична регресія та лінійні підтримуючі векторні машини аналогічною обгорткою навколо

«LIBLINEAR». У таких випадках розширення цих методів за допомогою «Python» може бути неможливим.

Scikit-learn інтегрується буде з багатьма іншими бібліотеками «Python», такими як «matplotlib» та «plotly» для побудови графіків, «numpy» для матриці векторизації.

PyNLPl – це бібліотека Python для обробки природних мов. Містить різні модулі, корисні для загальних і менш поширених завдань NLP. PyNLPl може бути використаний для основних завдань, таких як вилучення n-грамів та списків частот, а також для побудови простої мовної моделі. Існують також більш складні типи даних та алгоритми. Більше того, існують аналізатори форматів файлів, поширені в NLP (наприклад, FoLiA / Giza / Moses / ARPA / Timbl / CQL). Є також клієнти для взаємодії з різними NLP-серверами. PyNLPl, зокрема, має дуже велику бібліотеку для роботи з FoLiA XML (Format for Linguistic Annotatation). Бібліотека розділена на кілька пакетів та модулів. Зокрема:

- pynlpl.statistics – частотні словники, відстань Левенштейну, загальні функції статистики та теорії інформації;
- pynlpl.textprocessors – простий токенізатор, n-грам розподілення.

Для лематизації текстів БрУК використовувалася бібліотека pymorphy2, яка створена для обробки російської мови, але містить у собі також українських словників для морфологічного аналізу:

- pymorphy2-dicts-ru – для російської мови;
- pymorphy2-dicts-uk – для української моови (экспериментальний).

Словники поширюються окремими пакетами.

Також бібліотека pymorphy2 пропонує можливість видалити стоп-слова з тексту документа. Стопслова - це слова, які не містять смислового навантаження тексту і зустрічаються в кожному текстовому документі (передлоги, союзи та ін.). Список стоп-слів доповнюється списком власних стоп-слів.

Для видалення особистих тегів, посилань, номерів телефонів, електронних адрес і подібного використана бібліотека pynlple.

2.4 Огляд хмарних сховищ даних

Хмарне сховище даних – модель онлайн-сховища, в якому дані можуть зберігатися на численних, розподілених в мережі серверах, що надаються в користування клієнтам. У протидію моделі зберігання даних на власних, виділених серверах, орендованих або придбаних спеціально для подібних цілей, внутрішня структура серверів клієнтові не видна.

Таблиця 2.1 – Порівняльний аналіз хмарних сервісів

Хмарні сховища	Переваги	Недоліки
1	2	3
Google Drive	Зручний інтерфейс. Вбудовані редактори. Висока швидкість роботи.	Для перегляду PDF або прослуховування FLAC знадобляться плагіни. Не можна вивантажити відео довше 15 хв. Не можна викладати фото здатністю вище 2048x2048 пікселів.
OneDrive	Редагування Word, Excel. Відео програється у вбудованому плеєрі.	MP3-файли не програються.
DropBox	Перегляд файлів всіх відомих форматів. Фільтр завантажених файлів.	Синхронізація розповсюджується тільки на папку DropBox. Спочатку маленький обсяг сховища.

Продовження таблиці 2.1

1	2	3
Box	Проста реєстрація. Просунутий пошук. Додавання в Вибране.	Передперегляд доступний тільки для файлів розміром до 500 МБ.
Яндекс. Диск	Авто розподіл за категоріями.	З редакторів доступна тільки примітивна обробка зображень.
Хмара Mail.ru	Перевірка на віруси. Створення резервних копій.	Присутня реклама.
MEGA	Відсутність ліміту на вагу файлу, який завантажується. Високий рівень конфіденційності.	Затримки в синхронізації з мобільними пристроями. Немає можливості переглядати файли онлайн.
Copy.com	Зрозумілий інтер- фейс. Можливість отримати додаткові ГБ за рахунок розміщення реферальних посилань.	Не передбачено програвання відео форматів.
iCloud	Прив'язані до того ж аккаунту. Синхронізація з усіма iOS-пристроями.Автозавантаження музики з iTunes.	Обмеження на кількість збережених фотографій: не більше 1000.

Дані зберігаються і обробляються в так званій хмарі, яка представляє собою, з точки зору клієнта, один великий, віртуальний сервер.

Хмарне зберігання даних - це найбільш практичний спосіб забезпечити практично повсюдний доступ до своїх файлів. Кожен хмарний сервіс має свій власний функціонал та багато сервісіві.

Проаналізувавши 9 найпопулярніших хмарних сховищ за останні роки, була складена порівняльна таблиця (таблиця 2.1). Також під час аналізу хмарних сховищ була складена порівняльна таблиця за критеріями оцінки (таблиця 2.2)

Таблиця 2.2 – Порівняння хмарних сервісів за критеріями

Хмарні сховища даних	Безкоштовний простір, що надається (Гб)	Платне розширення (до Тб)	Редагування файлів	Сумісний доступ	Мобільний доступ
Google Drive	15	30	Так	Так	Так
OneDrive	15	1	Так	Так	Так
DropBox	2	1	немає	Так	Так
Box	10	-	Так	Так	Так
Яндекс.Діск	10	1	Так	Так	Так
хмара Mail.ru	25	1	Так	Так	Так
MEGA	50	4	немає	Так	Так
Copy.com	15	1	немає	Так	Так
iCloud	10	2	немає	Так	Так

2.5 Мета роботи. Постановка задачі

Метою роботи є створення моделі сховищ інформаційних об’єктів із модифікованими та вдосконаленими методами прискореної обробки текстової інформації.

Для досягнення поставеної мети мають бути вирішені наступні задачі:

- розробка моделі сховищ інформаційних об’єктів;
- аналіз методів попередньої обробки тексту на виявлення можливості реалізації на системах із масовим паралелізмом;
- дослідження впливу характеристик обчисюлвальної системи на час реалізації модифікованого методу визначення ваги слів у корпусі тексту;
- аналіз отриманих результатів.

3 РІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ. ТЕОРЕТИЧНІ НАПРАЦЮВАННЯ

3.1 Організація сховищ ІО

Перехід від традиційної форми зберігання до електронної набуває масового характеру і знаходить відображення на функціонуванні бібліотек. У зв'язку з цим відбувається трансформація бібліотек в електронні бібліотеки. Відповідно до цього міняються функції і процеси управління. Таким чином, представлення інформації в електронній формі: створення електронних документів, організація її у вигляді електронних видань, різноманітних електронних колекцій і електронних бібліотек –це завдання актуальне.

Електронні тексти, в порівнянні з друкарськими, характеризуються принципово новими властивостями. Це обумовлено сучасним підходом до їх зберігання і поширення. Електронні тексти відкривають перед лінгвістами широкі перспективи. Це стосується обробки об'ємних масивів інформації з урахуванням додаткової класифікації і нових підходів до рішення традиційних проблем. Дана дослідницька область збирально позначається як "гуманітарна інформатика" (Humanities Computing).

Електронна (цифрова) бібліотечна система (Digital Library System, DLS) – це набір електронних ресурсів і супутніх технічних можливостей для створення, пошуку і використання інформації. Приватним прикладом електронної бібліотеки є електронна бібліотека ВНЗ (ЕБВ), основна мета якої полягає в забезпеченні учбового процесу необхідною літературою (інформацією), а також сприяння підвищенню рівня розвитку студентів і викладачів ВНЗ. Користувачі взаємодіють з електронною бібліотекою через автоматизовану бібліотечну інформаційну систему (АБІС).

АБІС - це складний організаційно-функціональний, технологічний і програмно-технічний комплекс (що вимагає різноманітних засобів забезпечення), призначений для здійснення в автоматизованому режимі

бібліотечно-інформаційних процесів, обслуговування користувачів бібліотеки і забезпечення їх доступу до зовнішніх електронних інформаційних ресурсів, а також для забезпечення життєдіяльності системи. [6]

Проблемою сьогодення є безперервне наростання кількості інформації, що призводить до таких вимог, як збільшення швидкості роботи пошукових систем та систем для категоризації інформації.

Серед типів бібліотек можна виділити тематичні бібліотеки (юридичні, медичні, військові, музичні, транспортні, бібліотеки мистецтв, філософськими) та спеціалізовано-корпоративні, тобто ті, які є актуальними та затребуваними групі читачів із певним статусом, наприклад – студент, аспірант, дослідник, молодий вчений (рисунок 3.1).

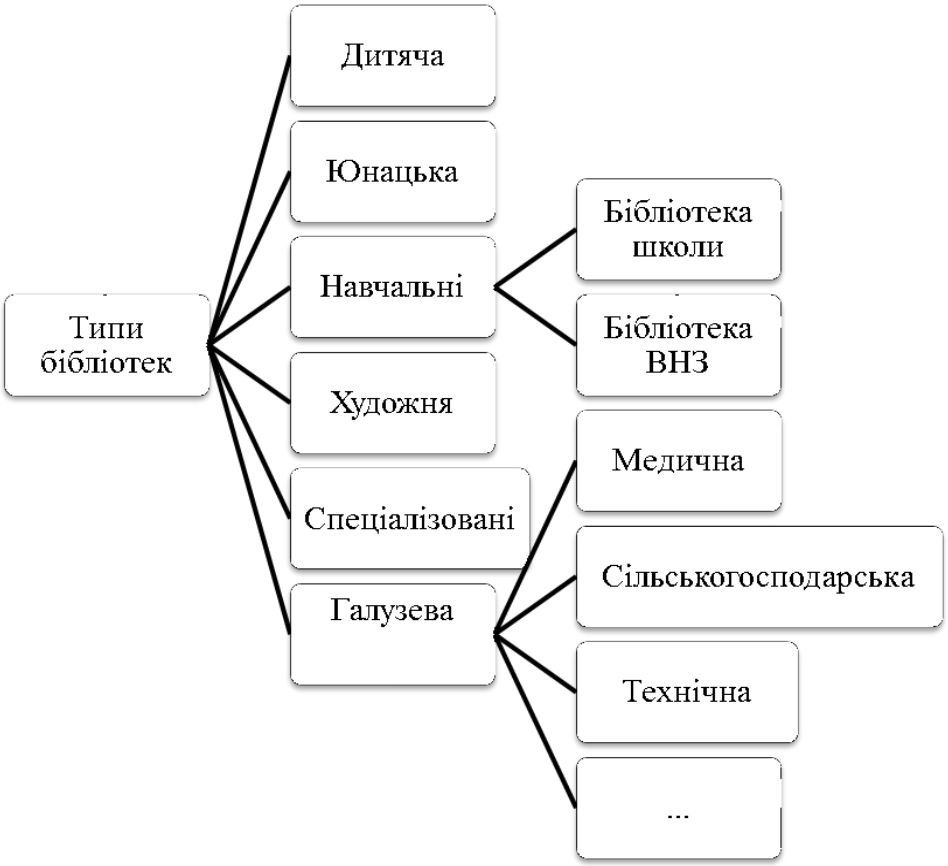

Рисунок 3.1 – Типізація бібліотек

Наведена типізація є актуальною і для електронних сховищ, оскільки забезпечує легкість доступу до цільових даних, а саме – наукових та дослідницьких робіт молодих вчених.

У роботі запропонована організаційна модель електронного сховища ІО для зберігання та доступу до наукових робіт дослідників, викладачів та студентів ВНЗ (рисунок 3.2).

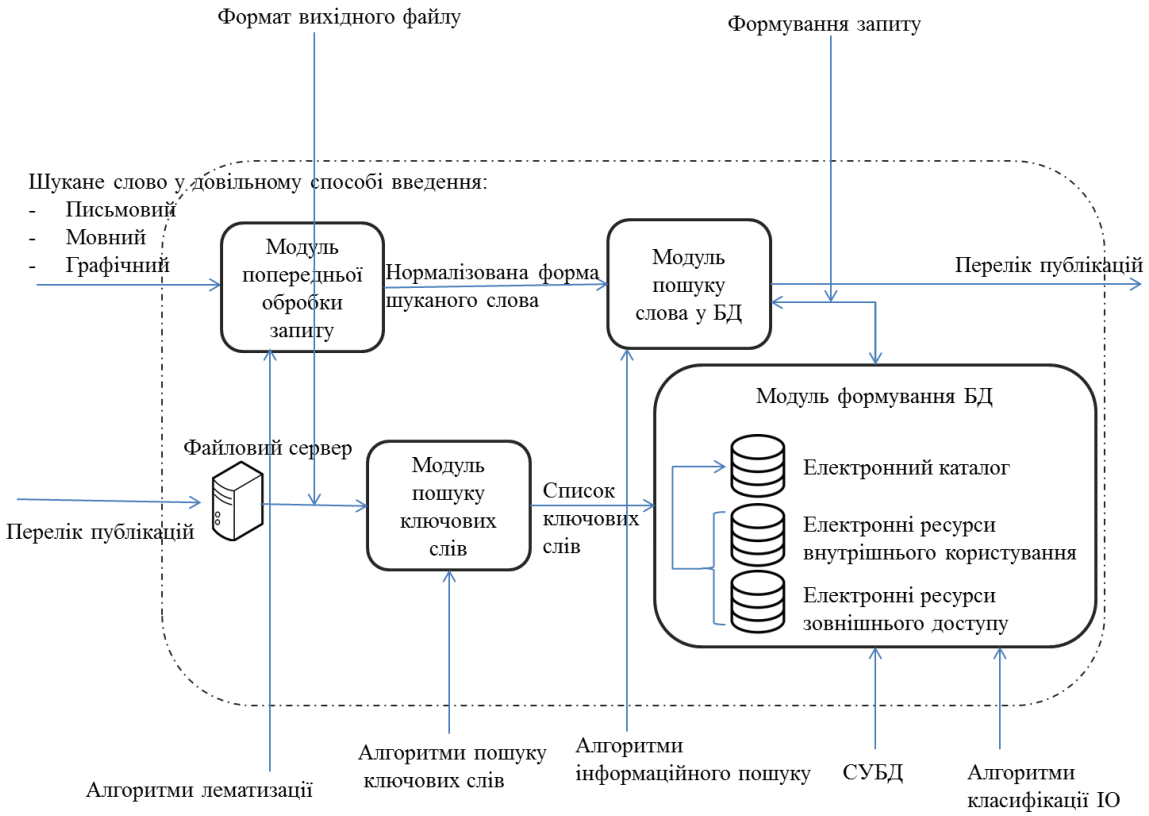

Рисунок 3.2 – Запропонована організаційна модель електронного сховища ІО для зберігання та доступу до наукових робіт дослідників, викладачів та студентів ВНЗ

Запропонована модель складається з наступних модулів:

- модуль попередньої обробки пошукового запиту;
- модуль інформаційного пошуку образу;
- модуль пошуку ключових слів у корпусі;
- модуль формування бази даних.

Організацію сховища у відповідності до рисунку 3.2 можна розділити на два етапи – накопичення інформації та доступ до інформації. Робота кожного з етапів складається з роботи окремих алгоритмів, які описані нижче.

На вхід моделі надходить безліч документів різних форматів (txt, doc, pdf і т. д.) Обирається бібліотека програмного коду в залежності від формату вихідного документа і здійснюється вилучення даних, а саме - ключових слів, з документа у вигляді оновленого тексту. Виділені ключові слова використовуються в якості вхідних значень на етапі класифікації ІС і побудови каталогізатора. Важливим етапом роботи алгоритму є те, що попереднім етапом класифікації є фільтрація тексту по стоп-листу (короткі слова і розділові знаки, що не несуть смислового навантаження для подальшого аналізу), що призводить до скорочення обсягу тексту і підвищенню його смислової цінності.

Важлівість модулю попередньої обробки запиту полягає у нормалізації тексту. Нормалізація має під собою приведення слів до нормальної форми, де нормальна форма слова – це канонічна форма слова. Наприклад, для іменників початкова форма слова – це форма однини, що стоїть в називному відмінку, для прикметників – прикметник чоловічого роду, однини і стоїть в називному відмінку без прийменника. Дане перетворення не тягне особливих втрат, так як конкретна форма слова рідко володіє корисною інформацією (сенс слова залишається тим же). Часто зустрічаються завдання з великим обсягом вихідних даних, в зв'язку з чим бажано зменшити число ознак. Приводячи ж слова до початкової формі, ми зменшуємо кількість унікальних слів.

При здійсненні нормалізації можна виділити два підходи: стеммінг і лематизації.

Стемінг – це грубий евристичний процес, який відрізає «зайве» від кореня слів, часто це призводить до втрати словотворчих суфіксів. Стеммінг займається пошуком основи слова, з огляду на морфологію вихідного слова.

Таким чином, знаходячи спільну для всіх граматичних форм слова - основу, відсікаючи суфікси і закінчення, стеммінг здійснює морфологічний розбір слова. Алгоритм стеммінга - це конкретний спосіб вирішення завдання пошуку основи слів, а стеммер - конкретна реалізація. Стеммінг має недолік: різні форми слова можуть мати різні основи. Тому після стеммінга результати обробки цих слів будуть різні. У зв'язку з цим застосовується інша техніка – лематизації.

Лематизація – це більш тонкий процес, який використовує словник і морфологічний аналіз, щоб в результаті привести слово до його канонічної формі - лемми. На відміну від стеммінга вона працює на основі словника, де і зберігаються дані про слова і їх початкових формах. Це дозволяє уникати помилок, описаних вище. У разі якщо слова немає в словнику, то відбувається побудова гіпотези про спосіб зміни слова.

Отриманий після виконання лематизації набір слів уже може використовуватися для проведення машинного навчання і вирішення конкретних завдань класифікації, маршрутизації і т.д.

Задачею модулю пошуку слова у БД є визначення чи входить шукане слово (терм, образ), яке складаєтьс з деякої кількості символів, до дексту (рядку, корпусу). Алгоритми та їх обчислювальна складність наведені у розділі 1.4. У разі знаходження шуканого слова, до модулю повертається вказівка на документ, у якому є це слово, а також його анотування.

Модуль пошуку ключових слів є вкрай важливим та необхідним етапом, що передує безпосередній класифікації нових документів для наповнення каталогу. Важлівість модулю полягає у зменшенні розмірності простору ознак, за рахунок чого можна знизити ефект перенавчання - явище, при якому класифікатор орієнтується на випадкові або помилкові характеристики навчальних даних, а не на важливі і значущі. Даний етап дозволяє значно скоротити розмірність рішення задачі і точність класифікації. Для цього можна використовувати метод TF-IDF [29], на вхід якого документ надходить в проіндексований вигляді (числова модель

тексту). Для індексації можуть використовуватися моделі мішка слів, N-грам або Word 2VEC [16, 21].

Згідно дистрибутивній гіпотезі Харріса, слова зі схожим змістом будуть зустрічатися в схожих контекстах.

3.2 Накопичення інформації при організації сховища ІО

Логічне представлення текстових документів обмежується використанням доступної після попередньої обробки інформації. Для цього послідовність методів попередньої обробки включає рівні графемного аналізу (виділення токенів – окремих слів в тексті), морфологічного (визначення граматичних форма та категорій слів) та синтаксичного (рисунок 3.3).

Під декодуванням розуміємо перетворення послідовності байт у послідовність символів (розпакування файлу, аналіз кодування, приведення до одного з форматів csv/xml/json/doc...)

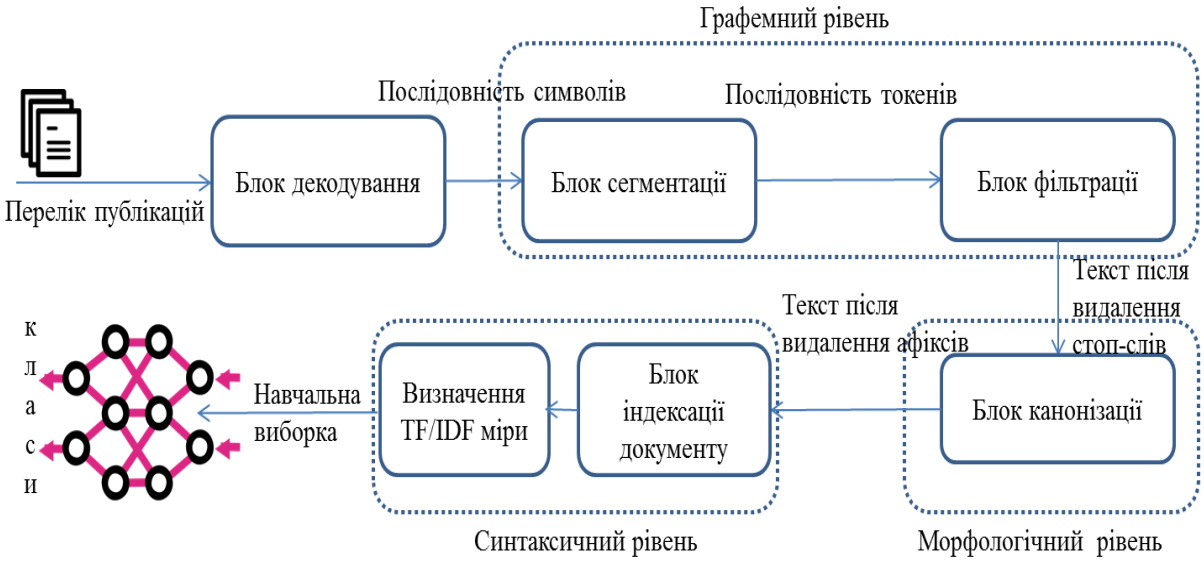

Рисунок 3.3 – Етап накопичення інформації при організації сховища ІО

Сегментація тексту (рисунок 3.4) передбачає розподіл тексту на речення. У найпростішому випадку сегментація виконується на основі маркерів кінця речення – крапки (три крапки), окличного знаку або знаку питання. У роботі токенізація виконується із урахуванням того, що крапка із комою також свідчать про кінець речення, оскільки цей знак часто служить для відокремлення окремих незалежних частин речення. Окрім того, виділення простих речень відбувається також у разі протиставних сполучників – а, але, однак, все ж, проте, хоча. Це є важливим аспектом сегментації, оскільки прості речення, які розділені протиставними сполучниками, скоріш за все, будуть мати різну тональність. Проблема омонімії крапки також врахована – окрім завершення речення вона може виконувати функцію скорочення слів (т.д., т.ч., т.е., ін.). Ці варіанти винесені у словник виключень.

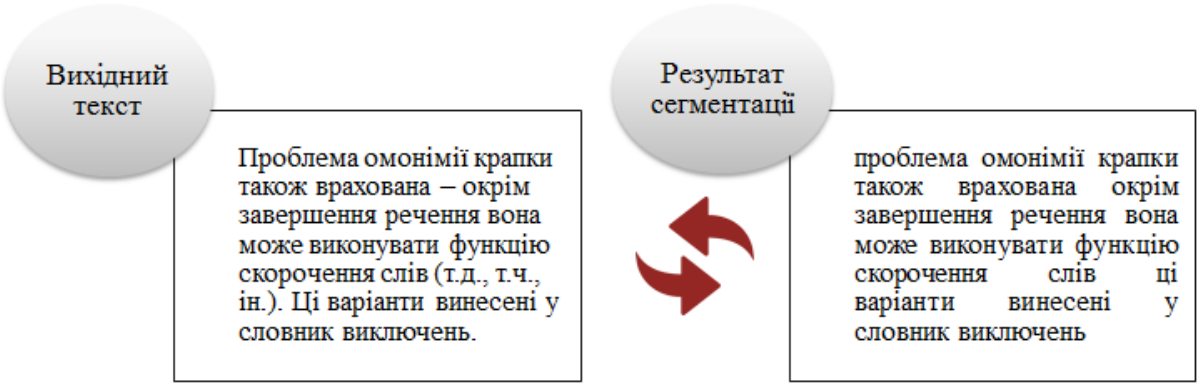

Рисунок 3.4 – Результат сегментації

Подальша фільтрація (рисунок 3.5) полягає у видаленні стоп-слів як слів, які не мають ніякого інформативного навантаження про зміст тексту. До таких слів відносяться функціональні слова (семантично нейтральні, такі як сполучники, артиклі, предлоги…).

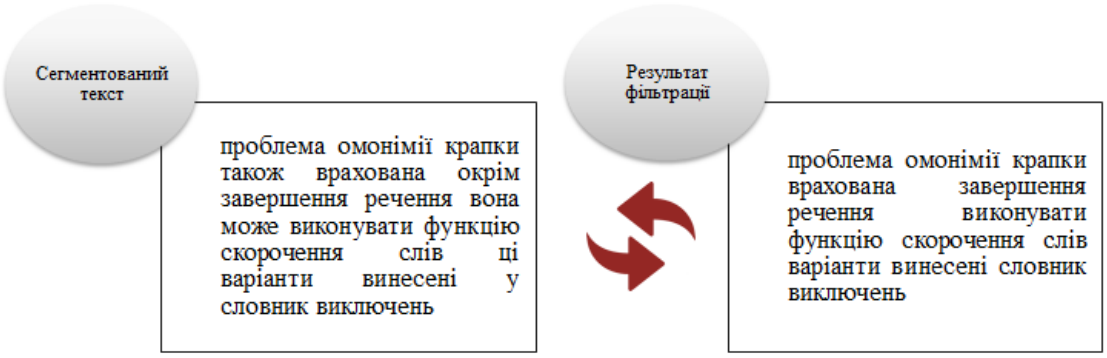

Рисунок 3.5 – Результат фільтрації

Канонізація (нормалізація, лематизація) полягає у приведені токенів до єдиного вигляду, що дозволяє позбутися різниці у написанні слова. Етап виконується на базі сформованого набору правил (алгоритм Пейса/Хаска): слово = основа+аффікси. Цей алгоритм націлений на ітеративне видалення закінчення слова. У алгоритмі застосовується таблиця правил для заміщення закінчень і суфіксів. Алгоритм спирається на останню букву в слові, що робить ефективним пошук правив в загальній таблиці. Видалення йде до тих пір, поки не існує правил в таблиці, відповідних для цього слова.

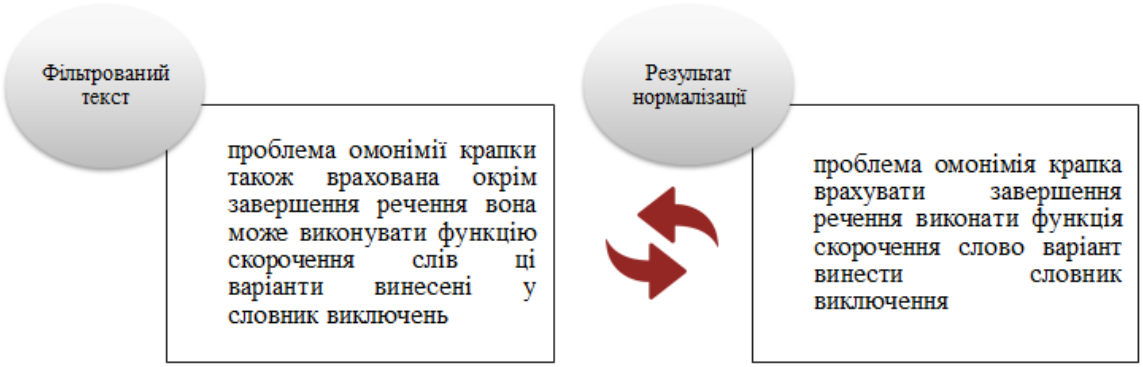

Рисунок 3.6 – Результат нормалізації

Видаляючи афікси, слово отримує вигляд лемми, достатньої для подальшої обробки, а саме – побудови частотного словнику (Vector Space Model, рисунок 3.7) на основі методу «мішок слів» та визначення важловості слів на основі алгоритму TF/IDF.

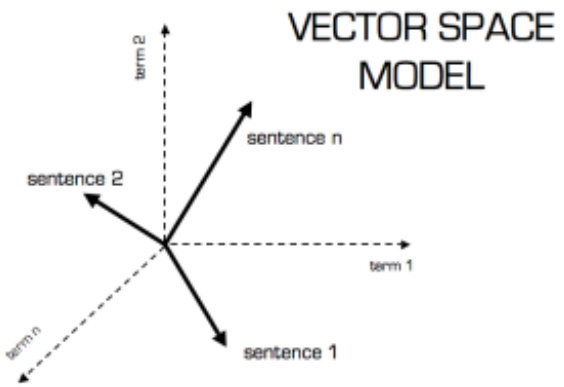

Рисунок 3.7 – Векторна модель тексту

TF (term frequency - частота слова) — відношення числа появи деякого слова до загальної кількості слів у документі. Таким чином, оцінюється важливість слова у межажх деякого документу. IDF (inverse document frequency — зворотня частота документа) — інверсія частоти, з якою деяке слово зустрічається у документах колекції. Облік IDF зменшує вагу загальновживаних слів. Інверсна частота терміна IDF вводиться для зниження значущості слів, які зустрічаються майже у всіх документах. Значення параметра IDF тим менше, чим частіше слово зустрічається в документах колекції. Тобто, якщо слово має високе значення параметру, воно із більшою вірогідністю може бути віднесено до колекції, аніж слово із малим значенням параметру. Але, слід брати до уваги, що слова, у яких значення близьке до нуля, відносяться до загальноуживаних слів, які не можуть дати обєктивну характеристику документу.

Нехай є колекція із трьох документів:

- Проблема омонімії крапки врахована;
- Проблема омонімії, омонімії слова;
- У тексті виникла проблема з крапкою.

Частотний словник (лексичні одиниці характеризуються з точки зору ступеня їх вживаності в сукупності текстів або для мови в цілому, або для окремої тематики, або для одного документу) із включеною мірою IDF буде мати вигляд таблиці 3.1.

Таблиця 3.1– Частотний словник

Слово	Усього	Зустрілося у документах	IDF
Проблема	3	3	0
Омонімія	3	2	0,18
Крапка	2	2	0,18
Врахувати	1	1	0,47
Слово	1	1	0,47
Текст	1	1	0,47
Виникати	1	1	0,47

Частотні словники широко застосовуються як у комп'ютерній лінгвістиці (наприклад, для класифікації текстів), так і в лінгвістиці традиційній (наприклад, для порівняння лексики різних авторів, аналізу зміни лексики з плином часу і т. п.).

Подальша робота полягає у класифікації вихідних передоброблених документів на базі одного з класифікаторів.

Наведений функціонал та можливості системи обумовлюють актуальність даної розробки, а також висувають ряд вимог:

- масштабованість сховища даних;
- забезпечення цілісності, достовірності, конфіденційності та відмово стійкості системи;
- реалізація обробки різного типу пошукових запитів (голосових, текстових, сканованих, фото) та швидкого пошуку даних у системі, спираючись на вказану область інтересів користувача;
- досягнення високого ступеню повноти та точності поділу документів на класи із визначенням найактуальніших задач;
- забезпечення глибокого аналізу завантажених робіт для групування можливих команд дослідників.

4 РІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ. ПРАКТИЧНІ РЕЗУЛЬТАТИ

4.1 Визначення часу упорядкування частотного словнику у послідовній реалізації

Серед методів векторизації, результати якої можуть бути використані для подальшої класифікації тексту, було проаналізовано метод «мішку слів» сумісно із моделлю TF-IDF. Часто в документі починають домінувати слова, які дуже часто зустрічаються, але вони можуть містити не стільки «інформаційного змісту» моделі, скільки більш рідкісні, але, можливо, специфічні для предметної області слова. Для уникнення цієї проблеми зявляється необхідність визначення IDF міри. За замовченням результатом роботи моделі TF-IDF є побудований словник із вагою кожного слова – частотний словник. Ключовими в даному випадку будуть слова, що мають міру IDF у середньому діапазоні значень (для слів, які зустрічаються у великій кількості документів, IDF буде близький до нуля (якщо слово зустрічається у всіх документах IDF дорівнює нулю), що говорить про семантичну важливість слова). Найменше IDF характерно для загальновживаних слів, найбільше для унікальних слів документу.

У роботі прийнято наступні пороги для класифікації (таблиця 4.1).

Таблиця 4.1 – Обрання кількості слів, які будуть брати участь у класифікації у залежності від кількості слів у вихідному тексті

Кількість слів вихідного тексту	Слова частотного словнику, що беруть участь у класифікації
До 1000	Усі слова у словнику
До 10000	З сотого + 40%
До 100000	З сотого + 20%
Вище 100000	З сотого + 20000слів

Такі показники є обгрунтованими, оскільки у текстах невеликого обсягу достатньою кількістю слів упорядкованого частотного словнику, які подаються на вхід класифікатора є 40%. Дотримання такого відсотку для текстів великого обсягу не є доцільним, оскількиз начно збільшується час роботи алгоритму класифікації; при цьому зростання точності класифікації не відбувається.

Виділення значущих слів в упорядкованому масиві веде до необхідності швидкого сортування та упорядкування великого об'єму даних – слів у тексті. Таким чином, послідовність дій буде наступна – рисунок 4.1.

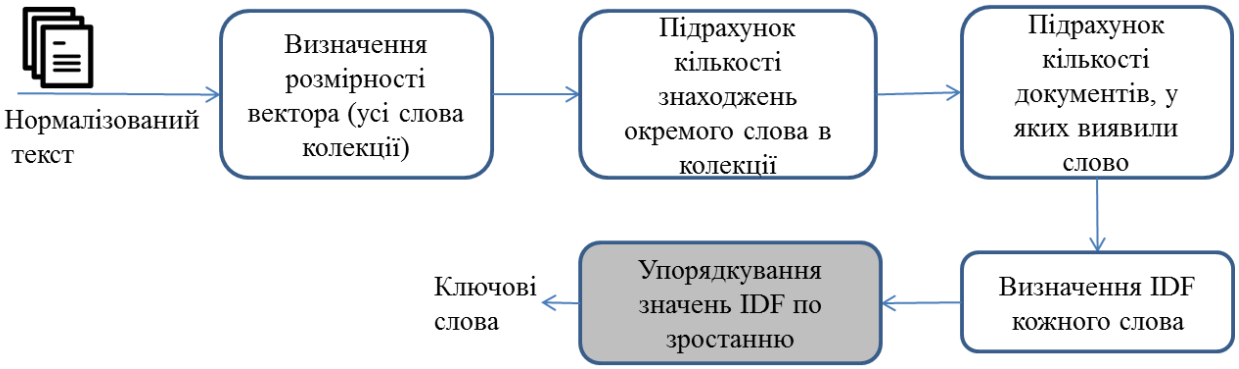

Рисунок 4.1 – Деталізація роботи блоків синтаксичного рівня запропонованої моделі

Упорядкування значень IDF великої кількості термів є операцією довготривалою, тому що за замовчуванням використовується послідовний алгоритм. У роботі проведено експериментальне дослідження впливу використання різних алгоритмів сортування на час виділення ключових слів.

Результати виділення ключових слів для колекцій різного розміру із стандартним упорядкуванням значень IDF наведена у таблиці 4.2.

Із наведеної таблиці результатів видно, що час сортування визначений ваг термів для великих словників складає більше 8% часу. Саме тому скорочення часу упорядкування може дати гарні результати.

Таблиця 4.2 – Час виконання послідовної синтаксичної обробки колекції документів у відповідності з рисунком 4.1

Розмір колекції	Кількість ключових слів	Час упорядкування, *сек*	Загальний час роботи алгоритму, *сек*
10	10	0	4,8
100	100	0,9	17
1000	1000	4,6	200
10000	З 100 по 4100 слово	26,5	509
100000	З 100 по 20100 слово	57	2124
1000000	З 100 по 20100 слово	407	3654
10000000	З 100 по 20100 слово	923	7521

4.2 Визначення часу упорядкування частотного словнику у паралельній реалізації на системах із загальною памяттю

Результати виділення ключових слів для колекцій різного розміру із паралельним упорядкуванням значень IDF на основі алгоритму Qsort наведена у таблиці 4.3. Реалізація виконана на системі із загальною памятю.

Один з елементів масиву призначається опорним. Елементи масиву переставляються таким чином, щоб всі, які менше опорного, виявилися лівіше від нього, а які більше - правіше. Для кожного з підмасивів операція рекурсивно повторюється. Ефективність алгоритму залежить головним чином від того, наскільки вдало буде підбиратися цей опорний елемент. Ідеальний випадок, коли він постійно ділить чергові подмассіви на рівні частини, але якщо кожен раз вираховувати, то час буде губитися на

розрахунок. Різні модифікації в основному відрізняються один від одного способом вибору опрного елементу та розбиттям на підмасиви. У роботі використано розбиття Хоара.

Обчислювальна складність алгоритму Qsort складає (n * log n).

Для обчислень використовується обчислювальний ресурс центрального процесору Intel(R) Core(TM) i5-3210M CPU із завантаженням чотирьох ядер.

Таблиця 4.3 – Час виконання синтаксичної обробки колекції документів із паралельним упорядкуванням ваг термів на ситемі із загальною памятю

Розмір колекції	Кількість ключових слів	Час упорядкування, *сек*	Загальний час роботи алгоритму, *сек*
10	10	0	4,65
100	100	0,6	15,4
1000	1000	3,8	174
10000	З 100 по 4100 слово	18,23	480
100000	З 100 по 20100 слово	31,4	2006
1000000	З 100 по 20100 слово	206,5	3400
10000000	З 100 по 20100 слово	321,8	6354

Із наведеної таблиці результатів видно, що запропоноване сортування скорочує час роботи блоку синтаксичного рівня запропонованої моделі, але все ж займає довго часу. При цьому видно, що для малих обсягів даних прискорення майже відсутнє. Для великих обсягів даних прискорення складає майже 3 рази.

4.3 Визначення часу упорядкування частотного словнику у паралельній реалізації на ситемах із масовим паралелізмом

Результати виділення ключових слів для колекцій різного розміру із паралельним упорядкуванням значень IDF на основі алгоритму поразрядного сортування LSD (least significant digit - спочатку молодший розряд) з підрахунком (Radix Sort) наведена у таблиці 4.4. Реалізація виконана на системі із масовим паралелізмом. Обчислювална чкладність алгоритму становить O (n).

Таблиця 4.4 – Час виконання синтаксичної обробки колекції документів із паралельним упорядкуванням ваг термів на ситемі із масовим паралелізмом

Розмір колекції	Кількість ключових слів	Час упорядкування, *сек*	Загальний час роботи алгоритму, *сек*	Загальний час роботи алгоритму, *сек*
10	10	0	4,8	4,71
100	100	0,8	17	15,5
1000	1000	1,4	200	169
10000	З 100 по 4100 слово	2,06	509	457
100000	З 100 по 20100 слово	2,4	2124	1940
1000000	З 100 по 20100 слово	3,56	3654	2987
10000000	З 100 по 20100 слово	8,48	7521	6129

Основною особливістю алгоритму є висока ефективність для сильно перемішаних або випадкових наборів даних. На майже відсортованих

наборах має сенс застосовувати інші алгоритми, так як виграш буде не такий значний. Погано працює на малих масивах, менше пари сотень елементів.

Для обчислень використовується обчислювальний ресурс графічного процесору NVIDIA GeForce GT 650M із архітектурою Kepler та максимальною кількістю CUDA-ядер 384..

4.4 Аналіз отриманих результатів

Скорочення часу сортування термів на основі послідовного та паралельних алгоритмів наведено на рисунку 4.2.

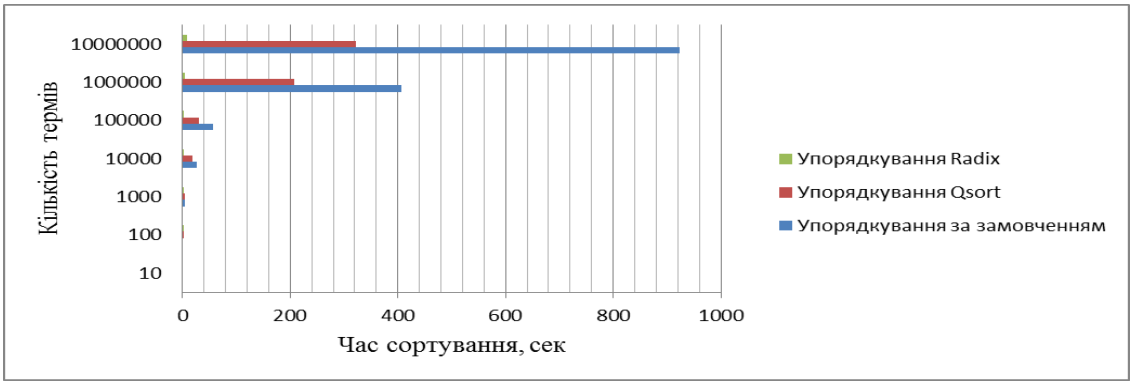

Рисунок 4.2 – Час сортування термів

Прискорення, отримане для блоку синтаксичної обробки наведено на рисунку 4.3.

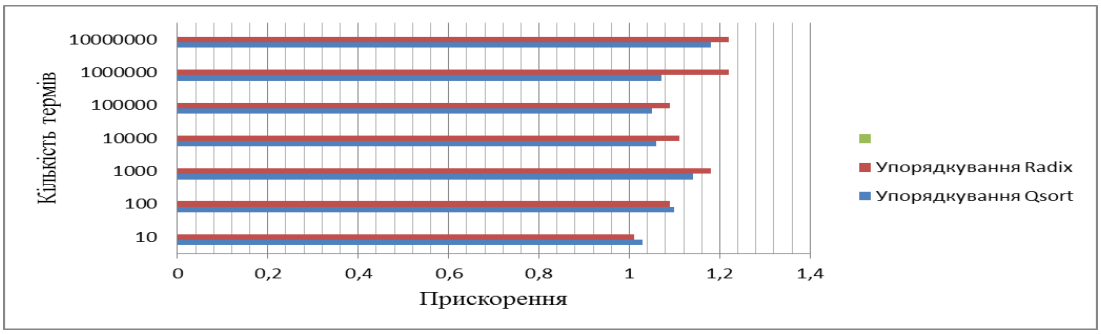

Рисунок 4.3 – Отримане прискорення для блоку синтаксичної обробки

Аналіз отриманих результатів показав, що запропоноване впаровадження сортування вагів термів на системах із масовим паралелізмом у побудованму частотному словнику скорочує час роботи блоку синтаксичного рівня запропонованої моделі майже на 18%. При цьому видно, що для малих обсягів даних прискорення майже відсутнє. Для великих обсягів даних прискорення складає майже 100 разів у порівнянні із послідовним сортуванням, що використовується за замовчуванням.

ВИСНОВКИ

В ході роботи над атестаційною роботою магісра була запропонована модель сховища інформаційних об’єктів із модифікованими та вдосконаленими методами прискореної обробки текстової інформації.

Були вирішені наступні задачі:

- запропонована модель сховища інформаційних об’єктів, яка складається з наступних модулів: модуль попередньої обробки пошукового запиту; модуль інформаційного пошуку образу; модуль пошуку ключових слів у корпусі; модуль формування бази даних;

- проведено аналіз методів попередньої обробки тексту на виявлення можливості реалізації на системах із масовим паралелізмом, який показав можливість та необхідність реалізаціїї методів інформаційного пошуку та побудови частотного словнику на високопродуктивних обчислювальних системах, оскільки мають явний паралелізм даних;

- виконано дослідження впливу характеристик обчисюлвальної системи на час реалізації модифікованого методу визначення ваги слів у корпусі тексту завдяки використанню алгоритмів прискореного упорядкування елементів.

Аналіз отриманих результатів показав, що запропоноване впаровадження сортування вагів термів на системах із масовим паралелізмом у побудованму частотному словнику скорочує час роботи блоку синтаксичного рівня запропонованої моделі майже на 18%. При цьому видно, що для малих обсягів даних прискорення майже відсутнє. Для великих обсягів даних прискорення складає майже 100 разів у порівнянні із послідовним сортуванням, що використовується за замовчуванням.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. N. Besimi, B. Çiço and A. Besimi, "Overview of data mining classification techniques: Traditional vs. parallel/distributed programming models," 2017 6th Mediterranean Conference on Embedded Computing (MECO), Bar, 2017, pp. 1-4, doi: 10.1109/MECO.2017.7977126.

2. Зайцева С.Г., Барковська О.Ю. Прискорений пошук та заміна тексту в текстовому документі. // Комп'ютерні та інформаційні системи і технології. Збірник наукових праць третьої міжнародної НТК. – Харків: ХНУРЕ, 2019. – 23-24 квітня 2019. – С. 101.

3. N. Andhale and L. A. Bewoor, "An overview of Text Summarization techniques," 2016 International Conference on Computing Communication Control and automation (ICCUBEA), Pune, 2016, pp. 1-7, doi: 10.1109/ICCUBEA.2016.7860024.

4. M. Chistol, "A Comparative Study of Parametric Versus Non-Parametric Text Classification Algorithms," 2020 International Conference on Development and Application Systems (DAS), Suceava, Romania, 2020, pp. 208-213, doi: 10.1109/DAS49615.2020.9108968

5. A.Broder. On the resemblance and containment of documents. Compression and Complexity of Sequences // SEQUENCES'97, IEEE Computer Society, 1998, C. 21-29

6. D. Fetterly, M. Manasse, M. Najork. A Large-Scale Study of the Evolution of Web Pages // WWW2003, 2003, Budapest, Hungary.

7. Diego A. Rodríguez Torrejón, José Manuel Martín Ramos Text Alignment Module in CoReMo 2.1 Plagiarism Detector Notebook for // PAN at CLEF 2013, 2013

8. Feng G. et al. A Bayesian feature selection paradigm for text classification // Information Processing & Management. — 2012. — Т. 48. — №. 2. — P. 283-302.

9. G¨unal S. et al. On feature extraction for spam e-mail detection // International Workshop on Multimedia Content Representation, Classification and

Security. — Springer, Berlin, Heidelberg, 2006. — P. 635–642.

10. Tan S., Wang Y., Wu G. Adapting centroid classifier for document categorization // Expert Systems with Applications. — 2011. — Т. 38. — №. 8. — P. 10264-10273

11. Lovins J. B. Development of a stemming algorithm // Mech. Translat. & Comp. Linguistics. — 1968. — Vol. 11. — №. 1-2. — P. 22–31.

12. Porter M. F. An algorithm for suffix stripping // Program. — 1980. — Vol. 14. — №. 3. — P. 130–137

13. R. Premalatha and S. Srinivasan, "Text processing in information retrieval system using vector space model," International Conference on Information Communication and Embedded Systems (ICICES2014), Chennai, 2014, pp. 1-6, doi: 10.1109/ICICES.2014.7033837.

14. Salton G., Wong A., Yang C. S. A vector space model for automatic indexing // Communications of the ACM. — 1975. — Vol. 18. — No. 11. — P. 613–620.

15. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ, 2002.

16. Классификация М. Флинна [Електронний ресурс]. – Електрон. дан. – [2010] . – Режим доступу вільний.: http://parallel.ru/computers /taxonomy/flynn.html

17. Дацюк В.Н., Букатов А.А., Жегуло А.И. Методическое пособие по курсу «Многопроцессорные системы и параллельное программирование», Ростов-на-Дону, 2000.

18. GPGPU. [Електронний ресурс]. – Електрон. дан. – [2011]. – Режим доступу вільний.: http://ru.wikipedia.org/wiki/GPGPU

19. ATI Stream SDK 2.01 с улучшенной поддержкой OpenCL [Електронний ресурс]. – Електрон. дан. – [2013]. – Режим доступу вільний: http://nvworld.ru/news/amd-ati-stream-sdk-201/.

20. CUDA ZONE [Електронний ресурс]. – Електрон. дан. – [2013]. – Режим доступу вільний.: http://www.nVidia.ru/object/what_is_cuda_new _ru.html