МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна робота бакалавра

ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДОДАТКУ ДЛЯ ПОРІВНЯЛЬНОГО АНАЛІЗУ СУЧАСНИХ АЛГОРИТМІВ ПОШУКУ ІНФОРМАЦІЇ

Здобувач освіти гр. ІН – 82 Артем ЦИБУЛЬНЯК

Науковий керівник,
кандидат фізико-математичних наук,
доцент Сергій ШАПОВАЛОВ

Завідувач кафедри
доктор технічних наук, професор Анатолій ДОВБИШ

СУМИ 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Затверджую_____
Зав. кафедрою Довбиш А.С.
"_____" _____2022 р.

ЗАВДАННЯ

до кваліфікаційної роботи

здобувача вищої освіти четвертого курсу, групи ІН-82 спеціальності «122 – Комп'ютерні науки» денної форми навчання Цибульняка Артема Олександровича.

Тема: " ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДОДАТКУ ДЛЯ ПОРІВНЯЛЬНОГО АНАЛІЗУ СУЧАСНИХ АЛГОРИТМІВ ПОШУКУ ІНФОРМАЦІЇ"

Затверджена наказом по СумДУ

№_____ від _____2022 р.

Зміст пояснювальної записки: 1) аналітичний огляд методів пошуку підрядка в текстовій інформації; 2) постановка завдання й формування завдань дослідження; 3) опис основних положень, математичних алгоритмів і критеріїв, що використовуються інформаційно-екстремальною інтелектуальною системою; 4) розробка інформаційного й програмного забезпечення інтелектуальної системи; 5) аналіз статистичних результатів тестування.

Дата видачі завдання «_____»_____2022 г.

Керівник випускної роботи _____ Шаповалов С.П.

Завдання приняв до виконання _____ Цибульняк А.О.

ЗМІСТ

РЕФЕРАТ

Записка: 59 стор., 22 рис., 2 табл., 49 джерел.

Об’єкт дослідження — сучасні алгоритми пошуку інформації (підрядка).

Мета роботи — розробка сучасних алгоритмів пошуку інформації (підрядка) та їх порівняльний аналіз за допомогою статистичних даних для тестування кожного з алгоритмів.

Методи дослідження — метод функціонально-статистичних випробувань.

Результати — розроблені алгоритми та програмне забезпечення системи сучасних алгоритмів пошуку інформації (підрядка). При цьому задача порівняльного аналізу алгоритмів розв’язана в рамках інформаційно-екстремальної інтелектуальної технології з використанням датасетів. Розроблений алгоритм реалізовано у формі програмного забезпечення, створеного за допомогою інструментального програмного середовища Python 3.10.4, Jupyter Notebook та Pycharm.

Ключові слова — ІНФОРМАЦІЙНІ СИСТЕМИ, ПОШУК ІНФОРМАЦІЇ, АЛГОРИТМИ, МЕТОД ФУНКЦІОНАЛЬНО-СТАТИСТИЧНИХ ВИПРОБУВАНЬ, ІНФОРМАЦІЙНО-ЕКСТРЕМАЛЬНА ІНТЕЛЕКТУАЛЬНА СИСТЕМА.

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ІПС – інформаційно-пошукова система.

PRF – Pseudo relevance feedback, це метод обробки пошукового запиту інформаційно-пошукових систем.

Показник релевантності – це міра відповідності результатів пошуку завданню, поставленому в пошуковому запиті.

API – Application Programming Interface, це опис способів, набір класів, процедур, функцій, структур або констант, за допомогою яких одна комп'ютерна програма може взаємодіяти з іншими програмами.

Кешування – швидкісна пам'ять або частина оперативної пам'яті, де зберігаються копії часто використовуваних даних. Забезпечує швидкий доступ до них. Кеш-пам'ять зберігає вміст і адреси даних, до яких часто звертається процесор.

ПМ – пошукова машина, це програмно-апаратний комплекс, сервер, API, на якому виконується пошуковий рушій.

SEO – Search Engine Optimization, виконує внутрішню і зовнішню оптимізацію сайту з метою підвищення позиції сайту в списку сторінок, знайдених пошуковими системами за конкретними запитами.

Метадані – це дані, що характеризують або пояснюють інші дані.

SERP – Search Engines Result Page, це сторінка пошукового сайту, яка відкривається у відповідь на певний пошуковий запит, тобто результат роботи пошукової системи.

Вартість пошуку – це термін, що означає ефективність пошуку, складається із швидкості надання інформації, релевантності та повноти цієї інформації.

Релевантність – це ступінь відповідності знайденого документа або набору документів інформаційним потребам користувача.

IP-адреса – це унікальний числовий номер мережевого рівня, який використовується для адресації комп'ютерів чи пристроїв у мережах.

Ранжування – це сортування сайтів в пошуковій видачі.

DSMS – Data Stream Management System, являє собою програмну систему для управління безперервними потоками даних.

SLA – Service Level Agreement, це угода між постачальником послуг і користувачем, яка містить кількісні та якісні характеристики наданих послуг, такі як їх доступність, підтримка користувачів, час виправлення та інше

CSRF – Cross Site Request Forgery це тип веб-атаки, що призводить до виконання певних дій від імені користувача на веб-сторінці.

Праймінг – це явище роботи пам'яті, яке представляє собою зміну швидкості або точності рішення задачі, що спостерігається після пред'явлення інформації.

Polyfill – бібліотека, яка реалізує підтримку стандартів, де замовчуванням підтримка цих стандартів частково або повністю відсутня.

DOM – Document Object Model це специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами.

REST – Representational State Transfer, це підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів.

ВСТУП

Актуальність теми дослідження. Завдяки зростанню кількості клієнтів та кількості інформації, яка створюється та доступна, кількість джерел інформації у світі зросла в геометричній прогресії. Внаслідок особливостей взаємодії людини та машини, з одного боку, та семантичної неоднорідності джерел інформації з іншого, успішний пошук інформації стає дедалі складнішим. Рішення цієї проблеми полягає в персоналізації методів пошуку інформації, тобто в адаптації процесу пошуку до унікальних особливостей користувачів, що дозволяє швидко та з найменшими зусиллями знаходити відповідну інформацію. Використання користувацьких моделей для розширення запиту під час адаптивного пошуку інформації, що значно зменшує тривалість інтерактивного взаємодії та витрати користувача, оскільки уточнення запиту обробляється на стороні клієнта, є найбільш перспективним з точки зору зниження часу та вартості пошуку.

Об’єктом дослідження є процес програмної організації пошуку інформації в мережі Інтернет з використанням алгоритмів пошуку підрядка.

Предметом дослідження є алгоритми інформаційного пошуку на основі адаптації процесу пошуку до індивідуальних особливостей користувачів інформаційно-пошукових систем.

Мета роботи: провести порівняльний аналіз релевантності використанням алгоритмів пошуку підрядка для пошуку інформації в пошуковій видачі.

Ключові слова: інформаційно-пошукова система, пошуковий сервіс, користувач, пошуковий запит, релевантність інформації, ранжування, вартість пошуку, індексування веб-сторінок, метод BOM, метод BNDM.

РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ ОРГАНІЗАЦІЇ ПОШУКУ ІНФОРМАЦІЇ В СУЧАСНИХ ІНФОРМАЦІЙНО-ПОШУКОВИХ СИСТЕМАХ

1.1. Загальні алгоритми пошуку інформації в мережі Інтернет

Google, Baidu та низка інших пошукових систем тепер добре відомі. Алгоритми кожної пошукової системи унікальні, і вони так само важливі, як і ключові слова.

Алгоритм пошукової системи — це набір правил або формул, які пошукова система використовує для оцінки релевантності вмісту, і кожна пошукова система має свій набір правил [1]. Ці принципи оцінюють, чи є документ релевантним або корисним для користувача, чи містить він важливу інформацію, яка буде цінною для користувача, а також ряд інших факторів, які впливають на ранжування та перерахування результатів для кожного пошукового запиту пошуку інформації. Є деякі критерії, які сумісні з усіма алгоритмами пошукових систем. Алгоритми пошуку є для кожної пошукової системи та ретельно класифіковані.

Перше – це питання актуальності. Релевантність перевіряє цінну інформацію або вміст в алгоритмі пошукової системи. Це може бути сканування ключових слів або дослідження використання ключових слів. Алгоритм визначає, чи є на цій сторінці релевантна інформація для ключового слова запиту. Розміщення ключових слів також має важливу роль у визначенні релевантності пошуку. Ключові слова в заголовку або перших кількох рядках тексту поставлять публікації на вищу позицію за цим ключовим словом, ніж документи без цих елементів. Частота ключових слів також важлива для релевантності. Статті буде дано кращу оцінку, якщо ключові слова з’являються часто, але не є наслідком ключових слів [2].

Наступним критерієм пошуку є індивідуальні фактори. Елементи, які здійснюють пошук цієї конкретної системи з будь-якої іншої пошукової системи, становлять другу частину критерію для алгоритмічної системи. Окремі частини

алгоритмів кожної пошукової системи пояснюють, чому пошуковий запит Google дає інші результати, ніж пошуковий запит Yandex або Baidu. Кількість сторінок, проіндексованих пошуковою системою, є одним з найважливіших індивідуальних критеріїв. Вони можуть індексувати додаткові сайти або їхні системи, але результати можуть відрізнятися залежно від пошукової системи. Деякі пошукові системи карають спам, а інші його не карають.

Є кілька додаткових елементів, які є частиною алгоритмів, і це фактори позиції сторінки, які є унікальними для кожної пошукової системи. Вимірювання кліків і посилань використовується для визначення місця розташування сторінки. Кількість кліків і посилань може вказати, чи важлива сторінка для реальних користувачів і відвідувачів, і це може сприяти підвищенню алгоритму рейтингу документа.

Алгоритм PageRank (PR) Google використовується для ранжування веб-сайтів на основі результатів пошуку. Ларрі Пейдж, один із засновників Google, надихнув алгоритм PageRank. Найкраще значення важливості веб-сторінок — це PageRank. За словами Google, підтримує кількість і якість з'єднань на сторінці, щоб запропонувати точну оцінку цінності веб-сайту. Основна передумова полягає в тому, що більше зареєстрованих веб-сайтів може підтримувати більше з'єднань з інших веб-сайтів [4].

PageRank тепер не єдиний алгоритм, який використовує Google для категоризації результатів пошуку; тим не менш, він є першим і найбільш відомим.

PageRank — це аналіз посилань, який надає числове значення кожному елементу в колекції документів із гіперпосиланнями, наприклад у Всесвітній мережі, щоб «виміряти» його релевантність у наборі [5]. (рис. 1.1).

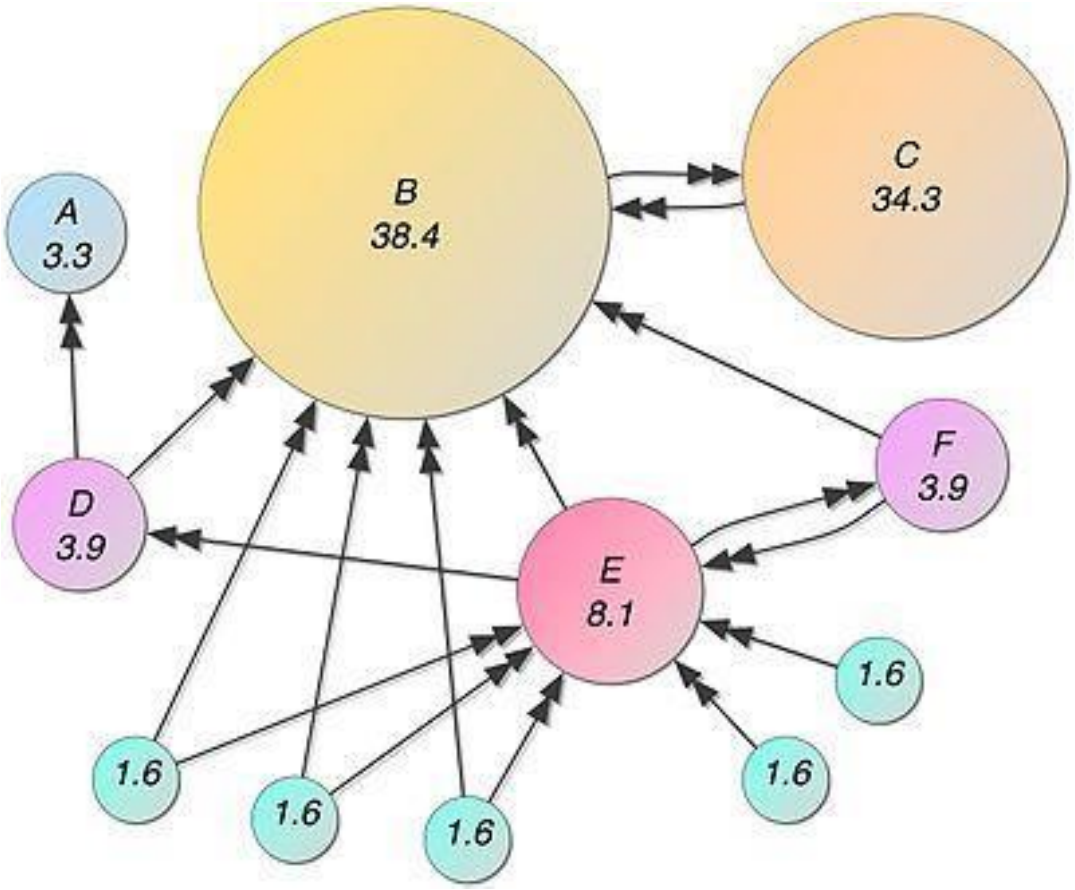


Рис. 1.1. Графічне зображення роботи алгоритму PageRank [6]

За допомогою цього підходу можна шукати будь-яку колекцію елементів, що мають взаємні цитати та посилання. PageRank E — це числова вага, яку він призначає будь-якому елементу E, і записується як PR (E). Інші елементи, наприклад, оцінка, можуть впливати на важливість об'єкта [7].

Він розроблений за допомогою математичного процесу на основі графіків, який застосовується до всіх сайтів всесвітньої мережі, включаючи коледжі та зв’язки, а також центри влади, такі як CNN.com і USA.gov. Значення рейтингу відображає релевантність певної сторінки. На сторінці результатів підтримки [8,] є гіперпосилання.

Після Пейдж і Бріна [9] було опубліковано безліч наукових статей, заснованих на PageRank. На практиці поняття PageRank піддається маніпуляції.

Згідно з дослідженнями, для великого рейтингу PageRank було виявлено неправильний запит [10]. Мета полягає в тому, щоб з’ясувати, як не враховувати зв’язки з паперів, які мають неправильні наслідки.

Наступні елементи впливають на PageRank веб-сторінки [11].

1. Частота ключових слів і розміщення на веб-сторінці. Якщо ця сторінка містить лише один термін, вона має низький бал для цього ключового слова.

2. Тривалість веб-сторінки: щодня створюються нові веб-сайти, і не всі вони залишаються онлайн. Сторінкам історії надається більша вага.

3. На цю сторінку пов’язано кілька інших веб-сайтів: щоб зробити веб-сайт релевантним, подивіться, скільки інших веб-сайтів підключаються до нього.

Доступні алгоритми – це алгоритми, які використовуються разом з основним алгоритмом пошуку; їх реалізації відрізняються, але всі вони вирішують одні й ті ж проблеми [12]. Погляньте на пару з цих алгоритмів [13].

1. Відповіді графіка знань. У 2012 році Google оголосив про календар знань, який містить понад мільярд справжніх людей, місць і речей, а також понад 50 мільярдів фактів і стосунків. Реальні об’єкти, а не просто текстові рядки, складають світ. У результаті вам потрібно буде створити розклад знань, щоб пояснити, як все пов’язано. За допомогою цього алгоритму можна отримати швидкі відповіді.

2. Напрямки руху. Коли люди шукали адреси за посиланнями, завжди було очевидно, що вони не заходили на сайти, де вказані вулиці. Вони напевно знають, як туди дістатися. В результаті був розроблений алгоритм визначення напрямку руху на з’єднанні.

3. Прямі відповіді. Вам можуть знадобитися прямі відповіді на конкретні запитання, і в цьому випадку вам слід зв’язатися з організаціями, які можуть надавати інформацію та послуги, ліцензувати їхні матеріали та надавати змістовні відповіді на сторінці результатів пошуку. Цей алгоритм, наприклад, дозволяє використовувати постачальників із впорядкованою релевантністю та довідковою інформацією про те, скільки фільмів виставлено в певній місцевості, а також постачальників послуг з продажу квітів, щоб визначити розклад кінопоказів у місцевих кінотеатрах.

4. Вибір фрагментів. Коли людям дають запитання, мета цього алгоритму — швидко та без зусиль знайти відповідь. Вибрані фрагменти допомагають швидко відповідати на запити, зосереджуючи увагу на програмно створених фрагментах

веб-сайтів і алгоритмах, які підтримують певний сайт для певної теми. Зразок інформації, згаданий сторонній веб-сайт і посилання на сторінку, заголовок сторінки та URL-адресу включено в запропоновані фрагменти.

5. Розширені списки. Окремий об'єкт, список чи група пов'язаних осіб, місць чи речей не обов'язково є найкращою відповіддю на запитання. У результаті, незалежно від того, чи шукає користувач «маяки Каліфорнії» чи «відомі жінки-астрономи», «цей алгоритм відобразить список цих елементів у верхній частині сторінки.

6. Спочатку відповідає, потім запитує. Користувачі хочуть, щоб інформація була доступна в будь-який час. Ось чому ви повинні почати з відображення мінімуму даних, які клієнт бачить щодня. Наприклад, поточний час, погода в місті, курс валюти тощо. Це дозволяє отримувати інформацію, не звертаючись до запиту, що дійсно корисно [14].

1.2. Аналіз сучасних інформаційно-пошукових систем

Пошукова система Google – це складна технологія, яка робить пошук інформації в Інтернеті майже важким. Для створення результатів пошуку Google використовує унікальний алгоритм. Google просто надає загальну інформацію про їхній алгоритм. Google має продовжувати залишатися конкурентоспроможною пошуковою системою [15]. Павуки або сканери — це автоматизовані програми Google. Google, як і інші пошукові системи, має великий індекс термінів. Відповідь полягає в тому, що Google ранжує результати пошуку, що визначає порядок, у якому Google представляє результати на сторінці результатів пошуку. Google використовує алгоритм PageRank для визначення релевантності кожної веб-сторінки [16].

Кількість інших веб-сторінок, які підключаються до цього веб-сайту, є дуже корисним компонентом в алгоритмі пошуку Google: він оцінює, скільки веб-сайтів посилаються на певний сайт, щоб відповідати його релевантності.

Це більш простий приклад. Розгляньте слово «планета Земля» під час пошуку. Рейтинг сторінок Discovery Earth покращується, оскільки до нього підключається більше веб-сторінок. Інші сайти відображаються вгорі сторінки в результатах пошуку Google, коли сторінка Discovery займає вищий рейтинг. Важко обдурити Google, оскільки він містить посилання на веб-сторінку як голосування, але це можливо [17].

У 2008 році Google почав експериментувати зі своєю пошуковою системою. Спочатку Google дозволяє невеликому набору бета-тестерів змінювати рейтинг результатів пошуку. Тестери бета-версії можуть рекламувати або мінімізувати результати пошуку в цій пробній версії, а також можуть змінити свій пошук, щоб зробити його більш релевантним [18].

Подумайте, як Google знайшов інформацію, яка вам потрібна для відображення в результатах пошуку. Перед тим, як користувач напише свій запит в області введення, алгоритм почне працювати (рис. 1.2).

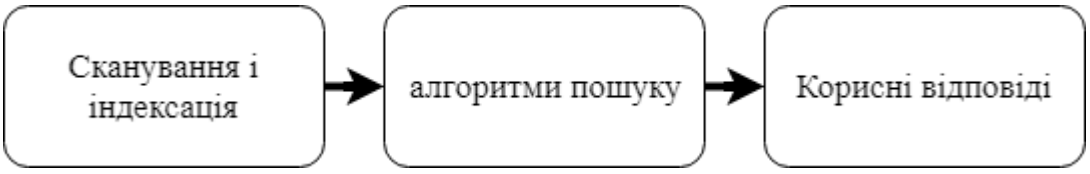


Рис. 1.2. Схема роботи пошукового сервісу

Щоб упорядкувати інформацію з онлайн-сторінок та іншого загальнодоступного вмісту в індексі пошуку, використовуються алгоритми веб-сканерів та павуки. Після цього запускаються алгоритми пошуку. Алгоритми рейтингу Google за лічені секунди просіюють мільярди веб-сторінок в індексі пошуку, щоб отримати корисні та релевантні результати.

Алгоритми пошуку є наступним етапом. Системи рейтингу складаються з ряду алгоритмів, які оцінюють, що шукає користувач і які дані потрібно повернути. Метод ранжування інформації, який Google використовує для надання цінної інформації з Інтернету, зображений на рис. 1.3.

Розуміння значення пошукового запиту має вирішальне значення для відповідних результатів пошуку. У результаті, перш ніж ви зможете визначити сторінки з корисною інформацією, ви повинні спочатку з’ясувати, що означають слова у вашому пошуковому запиті. Мовні моделі створюються, щоб спробувати з’ясувати, які рядки слів шукати в індексах. Це охоплює такі речі, як розуміння орфографічних помилок, розпізнавання типу запитання та використання деяких з останніх досліджень розуміння природної мови.

Рис. 1.3. Схема процесу пошуку та отримання інформації в мережі Інтернет

Наприклад, система синонімів — це пошук кількох запитів, навіть якщо термін є значущим.

Шукайте в Інтернеті сторінки, які містять інформацію, яка відповідає запиту. Коли ви робите простий пошук, алгоритм шукає пошукові фрази в індексах, щоб

визначити сайти, які є релевантними для вас. Система перевіряє, скільки разів і де ці терміни зустрічалися на сторінці, у заголовках та в тексті. Окрім надання користувачам того, що вони шукають, алгоритм шукає докази, щоб оцінити успіхи та можливості. Наприклад, під час пошуку ключового слова «RAM», користувач навряд чи захоче, щоб наступна розробка включала слово «пам'ять». Замість того, щоб просто повторювати запитання, алгоритм намагається перевірити, чи надає сторінка відповідь на нього. В результаті алгоритм пошуку шукає сторінки з релевантними матеріалами, такими як фотографії літаків, фільми або список архітекторів. Нарешті, переконайтеся, що сторінка написана тією ж мовою, що й запит про найкращий алгоритм сторінки вибраною мовою.

Корисні сторінки оцінюються. Для звичайного пошуку можуть бути сотні, якщо не мільйони веб-сторінок, які містять потенційно корисну інформацію. Використовуйте алгоритм, щоб проаналізувати ці веб-сторінки, щоб визначити, які з них найкращі. Система сканує сайти та оцінює людей із подібними запитами, щоб визначити легітимність та довіру до суб'єкта. Якщо інші сайти за темою посилаються на сторінку, це ознака високої якості вмісту. В Інтернеті існує багато спам-сайтів, які намагаються завершити результати пошуку, використовуючи такі стратегії, як повторення ключових слів або посилання, яке проходить через PageRank. Ці сайти пропонують поганий досвід користувачів і потенційно можуть завдати шкоди чи дезінформації [19].

Алгоритм використовує огляд контексту, а також таку інформацію, як місцезнаходження користувача, попередня історія пошуку та параметри пошуку, щоб змінити результати, щоб інформація була найбільш корисною та релевантною для користувача на даний момент.

Для розподілу матеріалу за його регіоном використовується алгоритм розташування та розташування користувача. Наприклад, якщо людина шукає «матч» у Чикаго, він, швидше за все, знайде результати з американського футболу та «матчів у Чикаго». Якщо ви шукаєте «матч» у Лондоні, алгоритм буде враховувати футбольні результати. У деяких випадках ви можете налаштувати результати, використовуючи дані з пошукового запиту попереднього користувача.

Наприклад, якщо людина шукає "Барселона", а потім шукає "Барселона проти Арсенала", це може означати, що вона шукає інформацію про футбольну команду, а не місто.

Перш ніж алгоритм відобразить результати, отримайте найкращі результати пошуку. Алгоритм враховує, як поєднується весь релевантний матеріал: чи є одна тема серед результатів пошуку або їх велика кількість, а також чи занадто багато сторінок, присвячених одній обмеженій інтерпретації. Алгоритм спрямований на надання різноманітного діапазону даних у формах, які є найбільш вигідними для певного типу пошуку.

Google має можливість пропонувати результати пошуку в широкій мережі форматів, щоб допомогти користувачеві швидко знайти потрібну інформацію, але це часто важко зробити через пошук релевантної інформації [20]. З більшою кількістю вмісту в розмові, ніж будь-коли раніше, Google має можливість пропонувати результати пошуку в широкій мережі форматів, щоб допомогти користувачеві швидко знайти потрібну інформацію, але це часто важко зробити дуже часто через пошук релевантних інформація [20].

Інтернет постійно змінюється; кожну секунду нові веб-сайти публікуються у великій кількості. Це відображається в результатах пошуку Google, оскільки компанія постійно аналізує дані в мережі, щоб індексувати свіжі матеріали.

Останнім етапом алгоритму є надання відповідних відповідей. Деякі сторінки результатів швидко змінюються у відповідь на запити користувачів, а інші стають більш надійними. Наприклад, коли користувач шукає останній спортивний результат, йому або їй потрібні останні зміни, але історичні результати можуть залишатися незмінними роками.

Щороку Google обробляє мільярди пошукових запитів. Щодня 15% запитів, які ми обробляємо, є такими, яких ми ніколи раніше не бачили. Розробка алгоритмів пошуку, які можуть забезпечити найбільш релевантні результати для всіх цих запитів, є складним процесом, який вимагає постійного тестування якості та інвестицій.

Тисячі інженерів і студентів намагаються вдосконалити алгоритми та відкрити нові методи збору інформації [21].

У 2021 році Google розширив понад 1600 пошукових рішень, які використовують наступні алгоритми пошуку інформації.

1. Відповіді з графа знань.
2. Напрямки за допомогою трафіку.
3. Прямі відповіді.
4. Вибрані фрагменти.
5. Багаті списки.
6. Відповіді, перш ніж запитати [22].

Пошукова система Яндекс повертає відповідний онлайн-контент з Інтернету у відповідь на пошук користувача. Сучасний Інтернет, навпаки, вимірюється в екзабайтах – квінтильйонах або мільярдах мільярдів байтів даних. Оскільки він відповідає новому пошуковому запиту [23], очевидно, що пошукова система не бачить таку величезну кількість даних щоразу.

Пошукова система Яндекс повертає відповідний онлайн-контент з Інтернету у відповідь на пошук користувача. Сучасний Інтернет, навпаки, вимірюється в екзабайтах – квінтильйонах або мільярдах мільярдів байтів даних. Оскільки він відповідає новому пошуковому запиту [23], очевидно, що пошукова система не бачить таку величезну кількість даних щоразу.

Пошукова система Яндекс використовує пошуковий індекс, який є базою даних, яка містить усі терміни та їх розташування, а також розпізнаний пошук. Зв'язок між позицією слова на веб-сторінці та адресами веб-сайтів в Інтернеті відомий як його розміщення.

Індекс пошуку фіксує кожне слово, яке коли-небудь було знайдено під час пошуку, відповідно до глосарію, який містить лише вибрані фрази. Пошуковий індекс, який відокремлений від телефонної книги, що містить імена та адреси, має більше однієї зареєстрованої адреси для кожного терміна.

Пошукова система Інтернету працює в два етапи. Він починається зі сканування мережі та збереження копії на її серверах. По-друге, він відображає відповідь від своїх серверів у відповідь на пошуковий запит користувача [24].

Перш ніж розробити пошукову систему, необхідно спочатку зібрати інформацію з Інтернету. Індексація — це термін для цієї процедури. Веб-сканер – це спеціальна комп'ютерна система, яка переглядає Інтернет, завантажує нові веб-сторінки та обробляє їх на регулярній основі. Це призводить до того, що «точна копія» Інтернету зберігається на серверах пошукових систем і оновлюється після кожного сканування [25].

Основний сканер сканує всі онлайн-сторінки, які зустрічає, а інший, відомий як Orange, виконує швидке індексування, щоб гарантувати, що найактуальніші документи, включно з документами, що містяться в Інтернеті, або навіть за секунди до сканування, доступні в індекс пошукової системи. Веб-сайти, які необхідно проіндексувати, знаходяться в списках очікування обох сканерів. Нові посилання, виявлені сканерами на сторінках відвідувачів, регулярно додаються до списків. Після того, як власники сайтів використовують систему обслуговування для додавання своїх сторінок до індексу, нові посилання можуть з'являтися в списках очікування.

Додаткову інформацію, наприклад, як часто оновлюється їхній веб-сайт, може додати адміністратор веб-сайту [26].

Перед початком процесу сканування використовується спеціальний інструмент під назвою «планувальник» або «павук-мандрівник», щоб встановити розклад відвідування веб-сайтів. Плануйте дослідити різноманітні елементи, які допоможуть у пошуку інформації, як-от популярність посилань і частота оновлення сторінки. Після побудови розкладу в картинку входить інший компонент пошуку Spider (рис. 1.4). Павук регулярно відвідує сторінки. Додаток завантажує сторінки веб-сайту згідно з розкладом, якщо веб-сайт доступний для павуків і функціонує. Павук перевіряє код і мову завантаженого документа, перш ніж відправити цю інформацію на сервери для зберігання [27].

Інша серверна програма видаляє всю зайву інформацію розмітки зі сторінки, залишаючи лише вміст. Потім він витягує інформацію про цей період часу та надсилає адміністратору всі слова в цьому документі. До наступного сканування вихідний документ також зберігається на сервері. Це дозволяє алгоритму надавати доступ до документів своїм користувачам, навіть якщо сайт зараз не працює. Алгоритм звільняє документ зі своїх серверів або замінює його більшою версією, якщо сайт закривається або документ знищено чи змінено.

Пошукова база створюється шляхом об'єднання пошукового індексу з копіями всіх індексованих документів, включаючи їх тип, код і мову (рис. 1.5).

Пошукова база даних має оновлюватися на регулярній основі, щоб не відставати від постійно мінливих матеріалів Інтернету та гарантувати, що пошукова система може знайти правильний баланс і найновішу інформацію у відповідь на певний пошуковий запит. Кожне оновлення бази даних спочатку керується на головному сервері пошуку, де може бути розташована перша пошукова система та результати використання результатів. Без дзеркал, дзеркальних сайтів або інших документів, які не є документами, базові пошукові сервери містять лише основну частину інформації. Це розділ пошукової бази даних, який відповідає на запити користувачів [28].

Кожні кілька днів «пакети» даних пошукової бази доставляються з серверів зберігання на головний пошуковий сервер.

Пошукова база даних має оновлюватися на регулярній основі, щоб гарантувати, що пошукова система зможе знайти належний баланс і отримати інформацію у відповідь на певний пошуковий запит. Кожне оновлення бази даних починається на головному сервері пошуку, де вона може бути розміщена як початкова пошукова система та результати її використання. Базові пошукові системи мають лише більшість інформації без дзеркал, сайтів-дзеркал та інших документів, які не є документами. Це частина пошукової бази даних, яка відповідає на запити користувачів [28].

«Пакети» даних пошукової бази надсилаються з серверного сховища на головний пошуковий сервер кожні кілька днів.

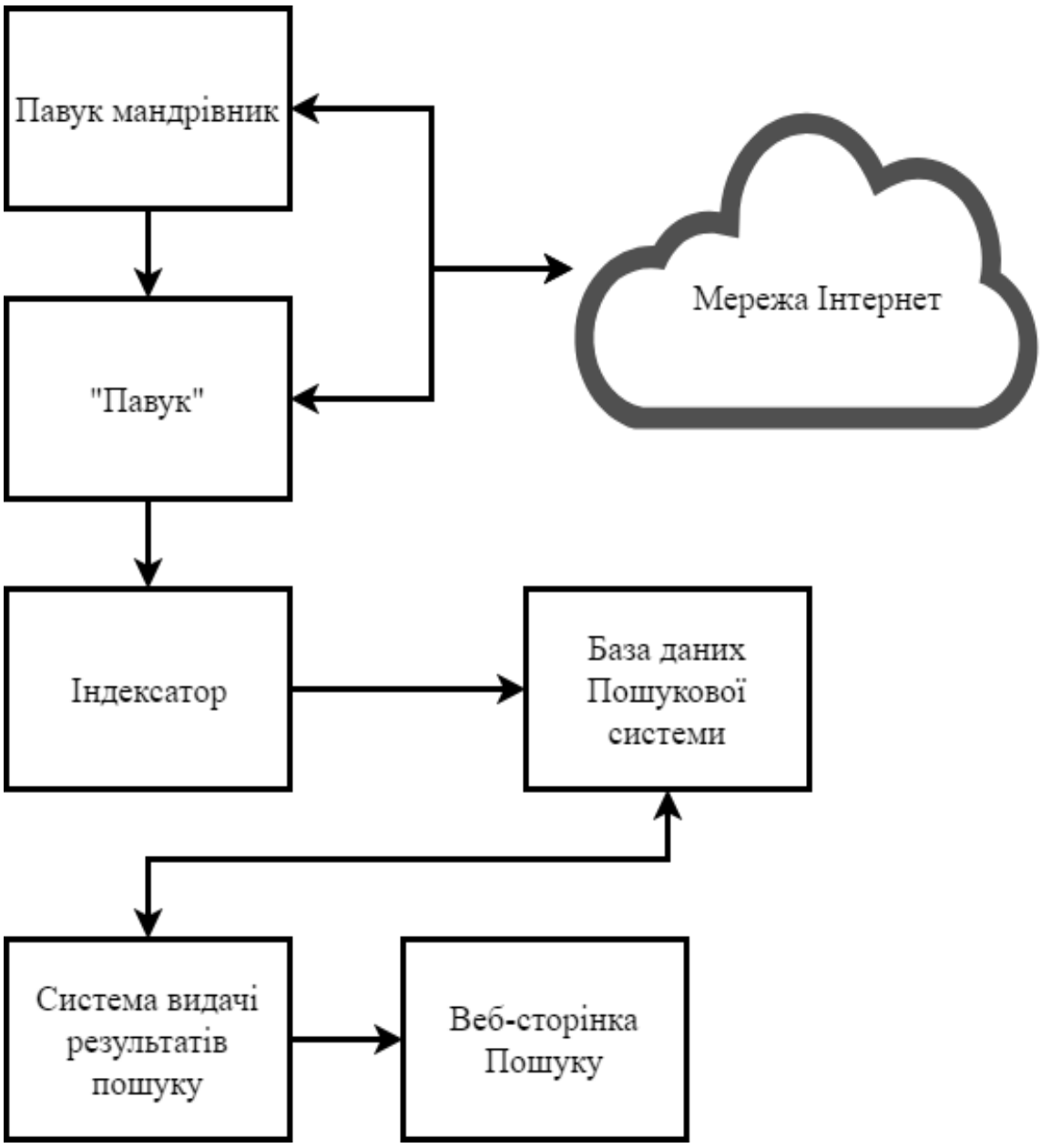


Рис. 1.4. Схема роботи алгоритму «Павук»

Orange scanner — це пристрій пошуку інформації в режимі реального часу. Його підхід, як і підхід Spider's, налаштований на пошук останніх статей, а також вибір великої кількості сайтів, які були б цікавими для споживачів. Ці файли негайно аналізуються та пересилаються на основні пошукові сервери. Через невеликий розмір цих даних зміни можуть вноситися в режимі реального часу, навіть протягом дня, не викликаючи перевантаження сервера [30].

До того, як працює пошукова система, потрібно виконати два кроки. Сканування мережі, індексація сторінок і підготовка їх до пошуку на ранніх етапах розробки. Іншим етапом є пошук у раніше створеній пошуковій базі даних для відповіді на конкретний запит користувача [31].

Рис. 1.5. Фрагмент бази даних індексів з копіями усіх проіндексованих документів

Знайти цінну інформацію стає все важче через поширення інформації в Інтернеті. Основна складність пошукової системи полягає в тому, як ефективно отримувати та використовувати цю інформацію. Система в першу чергу відповідає за збір даних, зберігання та оновлення інформації в Інтернеті як джерела інформації для всієї пошукової системи. Його називають «Павуком», тому що він рухається по павутині, як павук. Baiduspyder, Googlebot і Sogou Web Spyder [32] є прикладами поширених пошукових павуків.

Якщо мережа даних використовується як цільовий графік, процес Spyder може розглядатися як обхід цільового графіка.

Починаючи з кількох ключових вихідних з’єднань, користувач може постійно відкривати нові джерела посилань і сканувати гіперпосилання на документи, щоб зібрати якомога більше корисних документів. Для такої величезної компанії, як Baidu, оновлення бібліотеки документів і посилань щоразу, коли

можливі зміни, наприклад видалення нового посилання на сторінку документа, який система раніше відсканована, є складним процесом.

Алгоритм пошуку Baidu базується на наведених нижче ідеях. Система сканування повинна використовувати пропускну здатність якомога ефективніше, щоб зібрати якомога більше важливих ресурсів з обмеженими сервісними ресурсами та пропускною здатністю через велику кількість Інтернет-ресурсів.

Це викликає ще одну проблему, яка спричинена пропускною здатністю документа. Це посилання порушить звичайний доступ користувачів, якщо рівень встановлений занадто високий. Як наслідок, потрібен певний контроль протягом усього процесу сканування, щоб досягти мети не лише впливати на доступ користувача до документа, а й охопити якомога більше важливих ресурсів.

Baidu підтримує ряд різних кодів повернення [33].

1. Найбільш загальний код 404 вказує на те, що не вдалося знайти документ. Веб-сторінка видаляється з бібліотеки після закінчення терміну її дії. Якщо система найближчим часом знову ідентифікує посилання, документ не буде відсканована.

2. Код повернення 503 вказує на те, що послуга недоступна, і я вважаю, що документ тимчасово недоступний, тобто веб-сайт закритий і доступ обмежено. Це посилання не буде видалено алгоритмом, і воно буде відвідано кілька разів за короткий проміжок часу. Якщо документ було відновлено, його буде знову відскановано в звичайному режимі; якщо документ продовжує повертати 503, буде створено з’єднання з базою даних.

3. Код повернення 403 вказує на те, що сторінка була забанена, і я думаю, що тепер вона обмежена. Якщо це нове посилання, алгоритм не буде переходити за ним тривалий час і відвідає його кілька разів за короткий проміжок часу; якщо він увімкнено, алгоритм не публікує його і посилатиметься на нього кілька разів.

4. Репрезентативний код 301 вказує, що користувач буде відправлений на іншу URL-адресу на невизначений термін. Алгоритм підраховує кількість сторінок, які були перетворені в нове посилання.

Наступним кроком алгоритму є визначення кількості перенаправлень посилань. З ряду причин деякі веб-сторінки в Інтернеті мають статус переспрямування посилань. Система повинна виявляти та аналізувати переспрямування посилань, а також запобігати шахрайству, щоб належним чином керувати цими ресурсами.

Мережевий протокол сканується за допомогою алгоритму Baidu. Пошукова система Baidu має розширений підхід до сканування. Насправді пошукові системи та постачальники ресурсів мають взаємопов'язані системи. Інші пошукові системи не можуть задовольнити критерії пошуку користувача через налаштування конфігурації пошукової системи [34].

Створення системи гарантує, що ваш матеріал буде просуватися через пошукові системи, дозволяючи охопити ширшу аудиторію. Система сканування стикається з інтересами інтернет-ресурсів.

Обидві сторони даних повинні відповідати певним вимогам у процесі сканування, щоб оптимізувати обробку та стикування між двома сторонами, щоб налаштувати пошукову систему та безпрограшну систему. Мережні протоколи пов'язані з нормами, які підтримуються в цій процедурі.

Згідно зі статистикою, Google обробляє в середньому понад 40 000 пошукових запитів щосекунди, і очевидно, що велика кількість людей використовує Google у своєму повсякденному житті. Однак це не єдиний великий гравець у цій галузі. Пошукова система, така як Baidu, є привабливою альтернативою для споживачів Інтернету в багатьох країнах Східної Європи, де Google стикається з жорсткою конкуренцією

Baidu набуває популярності серед китайських споживачів, пропонуючи різноманітні послуги, такі як Wiki, обмін файлами та навіть власну платформу соціальних мереж. Зараз він домінує на понад 80% попиту на ринку Китаю і має понад 70 мільйонів активних користувачів щоденних мобільних розширень.

Baidu, звичайно, розуміє китайську мову та граматику краще, ніж Google, і, отже, оптимізує китайський матеріал ефективніше, ніж Google. Завдяки законам

про Інтернет у багатьох країнах для покращення вмісту для користувачів Baidu використовуються фахівці з оптимізації матеріалів [35].

Хоча Google як і раніше вважається чемпіоном, коли мова йде про пошукові системи, варто зазначити, що існують сотні мільйонів пошуків, які проходять через інші платформи, такі як згадані вище, і вони не повинні бути знехтувані, коли мова йде про створення глобального пошукової системи в Інтернеті. Розуміння мови та культури має безперечну перевагу для успіху пошукових систем у всьому світі.

1.3. Аналіз проблем існуючих методів пошуку інформації

Сканування, створення та індексація — це дві основні операції пошукових систем, які надають користувачам відсортований список сайтів, які вони визначили як найбільш релевантні. Пошукова система повинна сканувати свій індекс веб-сторінок на наявність матеріалів, пов’язаних із пошуком користувача, щоб знайти відповідну інформацію. Для створення цього індексу пошукова система використовує веб-сканер. Проблема в тому, що пошук системного алгоритму повинен постійно передаватися в Інтернет, тому що я не можу дізнатися, коли інформація на веб-сторінці змінилася, і він завжди сканує веб-сайти.

Через обмежені можливості обробки пошукові системи не можуть оновлювати та підтримувати всі сторінки в Інтернеті. Ця програма сканує Інтернет і зберігає дані про сайти, які відображає веб-сканер. Веб-сканер копіює веб-сторінку та додає її адресу до індексу щоразу, коли звертається до неї. Потім веб-сканер повторює операцію копіювання та індексування для всіх попередніх адрес на сторінці. Потім він повертається до посилань на проіндексовані сторінки. Це проблема масштабованості; алгоритм споживає значну кількість ресурсів комп'ютера. Він продовжує це робити, збираючи величезний індекс з кількох онлайн-сторінок.

Деякі веб-сайти не можуть їх сканувати, що також є важливою проблемою. Ці сторінки, а також ті, до яких ніхто не підключається, буде вилучено з покажчика.

Це індекс для пошукової системи. Кожна веб-сторінка, рекомендована пошуковими системами, перевіряється.

Пошукові системи відсіюють результати, щоб відобразити найкорисніші для них.

Найвідомішим методом покращення результатів онлайн-пошуку є PageRank. Простіше кажучи, PageRank – це конкурс популярності веб-сторінок. Веб-сторінка буде більш цінною, якщо на ній буде більше посилань. Це говорить про те, що результати будуть кращими. Існують такі проблеми, як відсутність можливості знайти щось цінне або необхідність довго чекати, оскільки пошукова система повертає найпопулярніші відповіді, які не є релевантними.

PageRank вважає сторінки на першій сторінці результатів найкращими. Визначаючи порядок представлення результатів користувачам, пошукові системи враховують безліч різноманітних «сигналів». Наприклад, як часто сторінка оновлюється і чи походить вона з авторитетного сайту. Існує кілька пошукових систем, з яких можна вибрати. Різні алгоритми використовуються різними пошуковими системами. В результаті деякі з їхніх сайтів відображатимуть результати в іншому порядку або навіть зовсім інші результати [36].

Через різноманітність алгоритмів пошуку важко правильно організувати веб-сторінку, щоб вона стала популярною в пошукових системах; проблема в тому, що пошукова система не розуміє, чого хочуть клієнти.

Щоб відповісти на це питання, ви повинні спочатку подивитися на різні типи даних, отриманих і збережених під час пошуку. Як сервер кожного онлайн-сервісу, пошукові системи в Інтернеті автоматично та методично реєструють дані, включаючи кожен запит сторінки. Журнал пошуку містить таку інформацію, як IP-адреса користувача, тип і мова браузера, дата і час запиту, ідентифікаційний файл, встановлений у користувача браузера, і сам пошуковий запит.

IP-адреса (Internet Protocol) — це номер, який однозначно ідентифікує пристрій (комп’ютер, мобільний телефон тощо) у комп’ютерній мережі, що спілкується за допомогою Інтернет-протоколу.

Ваш постачальник послуг Інтернету (постійно) (статична IP-адреса) призначає IP-адресу пристрою, підключеного до Інтернету (динамічна IP-адреса).

Постачальники пошукових систем можуть знаходити провайдерів поблизу географічного розташування пристрою, який ви використовуєте, на основі IP-адрес, але ви не можете ідентифікувати людей, які шукають. Цю позицію можна порівняти з положенням людини, яка знає місцезнаходження дому іншої людини (місто, вулицю та номер будинку), але не може знайти його на місцевій карті.

Цю картку бачить лише провайдер користувача, і провайдер може ідентифікувати абонента Інтернету, який зробив запит, переглянувши IP-адресу, дату та час запиту. Однак, як більше однієї людини може проживати в одному будинку, так і більше одного комп'ютера та іншого пристрою, тобто більше користувачів, можуть бути підключені до однієї IP-адреси. У результаті поєднання даних, постачальників пошукових систем і даних, оброблених провайдером, навряд чи дозволить надійно ідентифікувати користувача.

Через широке використання динамічних IP-адрес їхня здатність співставляти запити військового пошуку із запитами веб-сторінок від однієї й тієї ж особи суттєво ускладнена. Ідентифікаційні файли використовуються постачальниками пошукових систем, а також багатьма іншими постачальниками веб-послуг, оскільки вони вимагають таких навичок, як збереження налаштувань і переваг, а також покращення своїх послуг шляхом вивчення поведінки користувачів.

Коли ви спочатку відвідуєте сторінку користувача, ідентифікатор файлу — це величезний файл даних, якого немає на веб-сервері користувача. Веб-браузер на сервері перебуває на наступних сторінках відвідувачів у цьому домені, оцінюючи їх використання як повторних відвідувачів. Без індексування файлів кожна запитувана веб-сторінка буде незалежною від усіх інших, що призведе до відключення таких веб-сервісів, як веб-пошта та інтернет-магазини [37].

Термін дії деяких ідентифікаційних файлів закінчується після закінчення кожного сеансу, тоді як постійні файли зберігаються на жорсткому диску до закінчення терміну їх дії.

У багатьох випадках елементи веб-сторінки, такі як реклама, відрізняються від інших регіонів, і їх можна розмістити на жорсткому диску комп'ютера користувача за допомогою файлів аутентифікації сторонніх розробників.

Пошукові системи можуть використовувати їх для відстеження переміщень користувачів на веб-сайтах, дозволяючи рекламодавцям створювати профілі користувачів на основі їх моделей перегляду. Ці файли використовуються для ідентифікації користувача веб-браузера, а не особи.

Ідентифікаційний файл, розміщений пошуковою системою, у поєднанні з IP-адресою може з'єднувати різні запити з конкретною машиною, але в деяких ситуаціях, хто стояв за нею під час виконання запитів. Пошукові системи зберігають пошукові запити, а також IP-адресу та ідентифікатор користувача.

Пошукові запити виявленого користувача можуть бути пов'язані з його IP-адресою та перевіркою справжньої особистості користувача, що може означати загрозу конфіденційності.

Для швидкого та надійного пошуку інформації пошукова система ідентифікації збирає, аналізує та зберігає дані. Щоб мати справу з пошуком, дизайн ідентичності включає багатодисциплінарні принципи з мов, когнітивної психології, математики та інформатики. Веб-індексування [38] — це інша назва процедури в контексті пошукових систем, призначених для пошуку веб-сторінок в Інтернеті.

Популярні пошукові системи зосереджені на індексації повнотекстових веб-документів рідною мовою. Типи даних для пошуку включають медіа, відео, аудіо та графіку.

Хоча метадані пошукової системи повторно використовують індекси інших служб або сторінки і не зберігають локальний індекс, кешування PM постійно зберігає індекс разом із текстовим компонентом сторінки. Часткові текстові служби обмежують індексовані глибини, щоб мінімізувати розмір індексу, вибираючи з повнотекстових індексів. У той час як системи агентства індексуються в режимі реального часу, більше служб здійснюють пошук та індексацію в певний період часу після необхідного часу обробки.

Зберігання індексу використовується для підвищення швидкості та продуктивності пошуку під час пошуку відповідних документів для пошукового запиту. Він буде сканувати кожну сторінку в базі даних без індексу пошукової системи, що займе багато часу і багато обчислювальних ресурсів.

Хоча до індексу тисяч сторінок можна отримати доступ за мілісекунди, сканування кожного слова в тисячах величезних текстів може зайняти години. В обмін на час, заощаджений під час пошуку інформації, для зберігання індексу потрібно більше пам’яті комп’ютера, а також значне збільшення часу, необхідного для його оновлення (рис. 1.6).

Функція Search Facets, яка дозволяє фільтрувати дані за заданим значенням атрибута, є важливим компонентом системи. «Бренди» на веб-сайті електронної комерції є чудовою ілюстрацією цього.

Відповіддю на цю складність є отримання додаткової вартості та результатів; проте цього може бути недостатньо, і користувацькі продукти повинні шукати значення вручну (рис. 1.7).

Індекс структури даних архітектури пошукових систем варіюються в залежності від способу виконання індексації та методів зберігання індексу, щоб відповідати наступним чинникам проектування.

Рис. 1.6. Схема роботи пошукової системи на основі функцій ідентифікації контенту

При проектуванні архітектури інформаційно-пошукової системи необхідно враховувати наступні міркування.

1. Злиття – як оцінювати факти в покажчику, або як слова чи тематичні елементи додаються до покажчика під час обходу текстового корпусу, і якщо кілька індексаторів можуть працювати асинхронно.

2. Методи зберігання - наприклад, зберігання індексів, яке визначає, чи потрібно стискати дані чи фільтрувати.

3. Розмір індексу - обсяг пам'яті, необхідний комп'ютеру для підтримки індексу.

4. Швидкість пошуку – це час, необхідний для пошуку терміна в інвертованому індексі. Основною складністю в інформатиці є швидкість, з якою запис може бути розташований у структурі даних, на відміну від того, як швидко його можна змінити або видалити.

5. Підтримка - оскільки індекс оновлюється протягом тривалого періоду.

6. Відмовостійкість – важливість надійності сервісу.

Пошкодження індексу, оцінка того, чи можна обробляти погані дані ізольовано, впоратися з пошкодженим апаратним забезпеченням, розділами та схемами, такими як розділи на основі хешування або складені розділи, а також реплікація — усе це проблеми пошуку.

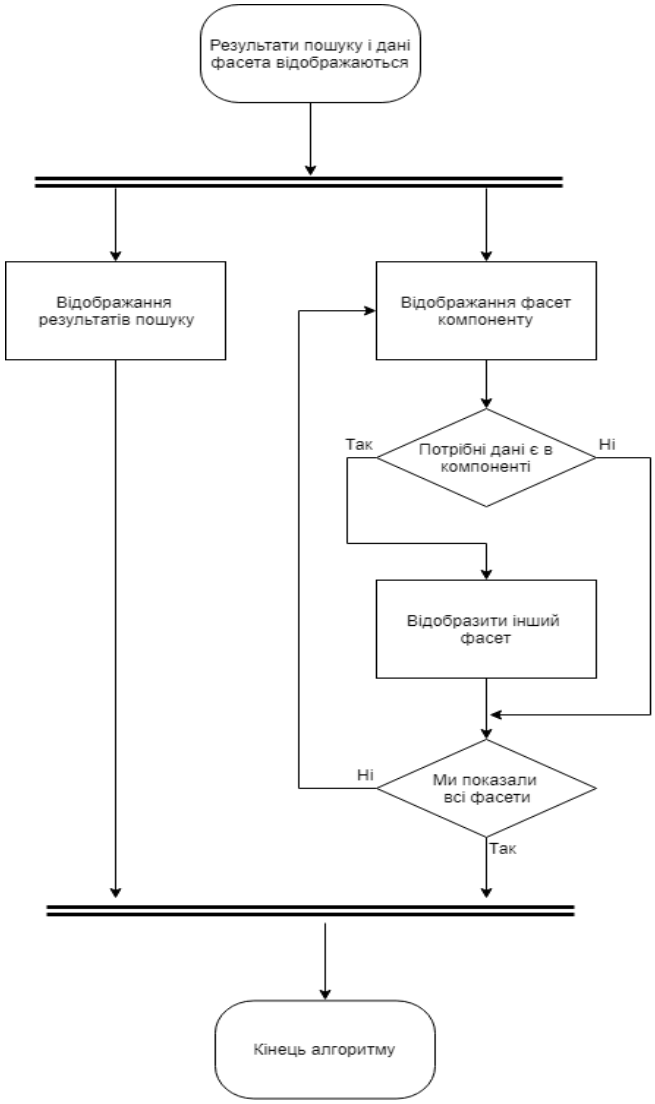
1. Суфіксальне дерево – деревоподібна структура, що дозволяє здійснювати лінійний пошук за часом. Для побудови структури використовуються суфікси слів. Суфіксальне дерево — це лінійна форма дерева. Розширюване хешування підтримується Attempts, що корисно для індексації пошуковими системами. Використовується для пошуку закономірностей і групування в послідовностях ДНК. Найбільшим недоліком є те, що зберігання слова в дереві може зайняти більше місця, ніж саме слово.

2. Інвертований індекс - хеш-таблиця або дерево, що містить список усіх входжень кожної умови атомарного пошуку.

3. Індекс цитування - база даних, яка фіксує зв'язки або зв'язки між текстами для використання в аналізі цитування, який є вивченням бібліографічних даних.

4. Цитування Ngram – зберігає дані певної довжини для використання в різних видах текстового пошуку чи аналізу.

5. Матричний документ – двовимірна розріджена матриця, яка використовується в прихованому семантичному аналізі для підтримки присутності слів у документах [40].



1.7. Алгоритм роботи пошуку на основі метода Фасета

Для виконання критеріїв розробки IRS індекс структури даних архітектури пошукової системи змінюється на основі техніки індексування та способів зберігання індексу.

РОЗДІЛ 2. АЛГОРИТМИ ПОШУКУ ПІДРЯДКА В РЯДКУ

2.1 Загальні теоретичні відомості про алгоритми пошуку підрядка в рядку

У обробці тексту пошук підрядка в рядку є вирішальним питанням. Алгоритм узгодження рядків є найважливішим компонентом у побудові програм операційної системи. Крім того, вони розробляють методи програмування, які використовуються як парадигми в інших галузях інформатики. Вони також важливі в теоретичній інформатиці, де вони ставлять складні проблеми.

Текст завжди буде основним засобом спілкування. Текст окремого або вхідного підрядка, який називається більшим шаблоном або шаблоном, можна шукати для пошуку підрядка в рядку. Обидва рядки засновані на алфавіті, який складається з обмеженої кількості літер.

Залежно від того, чи надано рядок першим, зразок чи текст, існує два види рішень. Для шаблонів попередньої обробки використовуються і повинні використовуватися алгоритми, засновані на використанні автоматів або комбінаторних ознак рядків. У рішеннях типу [43] використовується ідея індексів, які можуть бути реалізовані деревами або автоматами.

Текст сканується за допомогою вікна за алгоритмами відповідності рядків. По суті, вони вирівнюють ліві кінці вікна та текст, потім порівнюють символи вікна з шаблоном символів — це відомо як спроба — і зміщують вікно праворуч, якщо є шаблон або невідповідність.

Підхід грубої сили виявляє всі екземпляри підрядка x у тексті y за O (m * n) секунд, де m — довжина підрядка, а n — довжина тексту (див. рис. 2.1).

Послідовність, у якій порівнюється порівняння між символами шаблону та текстовими символами при кожній спробі, може бути використана для класифікації кількох кращих підходів грубої сили. Цей підхід використовує лише невеликий обсяг пам’яті і не вимагає ніякої підготовки. Більшість порівнянь персонажів є непотрібними й непотрібними [44].

Є чотири категорії: найбільш природний спосіб порівняння - зліва направо, що є напрямком читання; виконання порівнянь справа наліво, що призводить до найкращих алгоритмів на практиці; найкращі теоретичні результати досягаються при виконанні порівняння в певному порядку; і деякі алгоритми, для яких порядок порівняння не підходить, наприклад, алгоритм «грубої сили» [45].

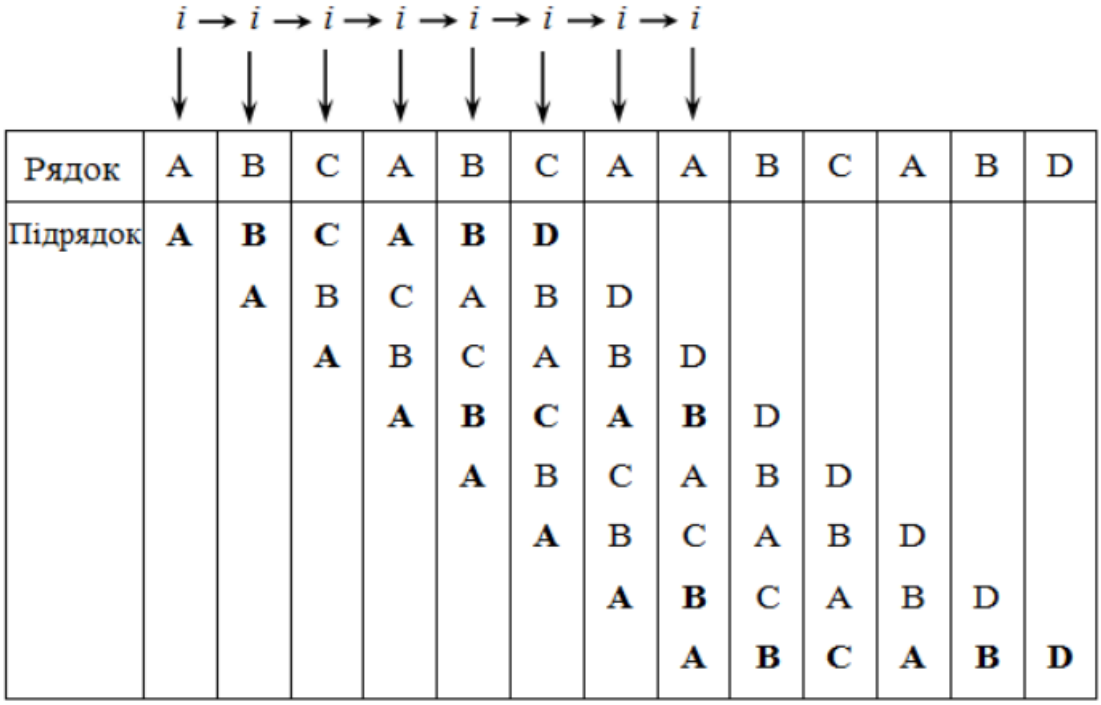


Рисунок 2.1. Алгоритм грубої сили

У обговорюваних реальних випадках хешування забезпечує просте рішення, яке дозволяє уникнути порівняння квадратичних символів і завершується за лінійний час з відповідною ймовірністю. Гаррісон представив його, а Карп і Рабін ретельно його оглянули.

Метод Shift є або ефективною технікою для вирішення конкретної проблеми зіставлення рядків, або він легко адаптується до широкого діапазону завдань зі збігом рядків, припускаючи, що довжина шаблону не перевищує розмір слова пам’яті машини.

Морріс і Пратт створили перший лінійно-часовий метод формування струн. Кнут, Морріс і Пратт були тими, хто його доопрацював. Пошук поводиться як процес розпізнавання автоматом, а символ тексту порівнюється з символом шаблону не більше ніж $log_{\Phi}(m + 1)$ разів, де $\Phi = \frac{1+\sqrt{5}}{2}$ – значення золотого перетину.

Перевірки текстових символів використовуються під час пошуку за допомогою визначеного кінцевого автомата, але він займає додатковий простір O (m), де є значення алфавіту. Використовуючи автоматизований шаблон суфіксів, алгоритм прямого зіставлення проводить стільки ж перевірок текстових символів.

Алгоритм Апостоліко-Крошемур є простим алгоритмом, який виконує $\frac{3}{2}\,n$ порівняння символів тексту в найгіршому випадку.

«Не такий простий» метод — це дуже простий алгоритм із квадратичною найгіршою складністю за часом, однак він вимагає постійної фази обробки часу та простору та є дещо сублінійним у середній ситуації.

2.1.2. Порівняння справа наліво

У традиційних програмах метод Бойєра-Мура виявляється найкращим алгоритмом узгодження рядків. Для команд «пошук» і «заміна» їх спрощений аналог часто використовується в текстових редакторах.

Коул продемонстрував, що для неперіодичних шаблонів максимальна кількість порівнянь символів обмежується 3n після подальшої обробки. Для їхніх моделей це найгірший квадрат.

Кілька варіантів Бойєра-Мура дозволяють уникнути квадратичного характеру методу. З точки зору порівняння символів, Apostolic і Duncarlo, Crochemour і Colussi створили найбільш вдалі рішення.

Емпіричні дані показують, що для загальних питань найбільш вдалими є варіанти методу Бойера і Мура, такі як «швидкий пошук» і алгоритм на основі суфіксального автомата Крошемура.

Алгоритми Чжу і Такаоки і Беррі-Равіндрана є варіантами алгоритму Бойера-Мура, які вимагають додаткового простору.

2.1.3. Порівняння у специфічному порядку

Галіль-Сейферас і Крошемур-Перрен були першими двома лінійними алгоритмами для узгодження рядків в оптимальному просторі (двосторонні алгоритми). Вони відрізняються тим, що поділяють шаблони на дві частини і шукають спочатку правий компонент, а потім ліву, якщо права частина не змінюється.

Алгоритми Colus і Galil-Duncarlo поділяють загальну кількість місць шаблону на дві групи. Спочатку вони перевіряють символи в першій підмножині шаблонів зліва направо, а потім шукають інші символи зліва направо, якщо немає відмінностей.

Алгоритм Колуссі – це поліпшення алгоритму Кнута-Морріса-Пратта, який виконує не більше $\frac{3}{2}n$ порівнянь символів тексту в найгіршому випадку.

В одному сценарії метод Галіла-Данкарло покращує алгоритм Колуса, дозволяючи йому порівнювати не більше ніж n текстових символів у найбільшому випадку.

Алгоритми «оптимальної невідповідності» та «максимального зміщення», які використовувалися в неділю, відсортували місця шаблону за частотою символів і найбільшим зміщенням відповідно.

2.1.4. Алгоритм Horspool

Алгоритм Horspool є версією методу Бойєра-Мура, який використовує одну зі своїх функцій зміщення і не дбає про порядок, у якому порівнюються текстові символи. Усі інші варіанти, такі як недільний «швидкий пошук», кращий алгоритм Бойєра – Мура, алгоритм Сміта та алгоритм Райта, також є хибними.

2.2. Алгоритм Бойера-Мура

Одним із перших алгоритмів, які ефективно використовували пам'ять і часові ресурси для вирішення лінійного пошуку та біоінформатики, був алгоритм Бойера-Мура. Його використання обґрунтовано його тимчасовою складністю з низьким використанням пам'яті O (m + σ), що використовується додатковою евристичною системою, що стала основою всіх методів, отриманих на основі алгоритму Бойєра-Мура [46]. Його особливість — дві евристики та відповідні рядки з кінця вікна порівняння. Зазначається, що x – необхідний шаблон (підрядок), y – текст (рядок) у пошуку [47].

1. Евристика стоп-символу.

На стадії препроцесингу створюється таблиця стоп-символів для усього алфавіту, який використовується в зразку. В ній позначена остання позиція кожного с можливих символів алфавіту в підрядку.

Якщо символ y I + j] = b у тексті y відрізняється від символу x I = підрядок x після його заміни в позиції j, вибірка переміщується безпосередньо за допомогою евристики символу зупинки, поки не буде вихідної відповідності між подібними символами. (див. рис. 2.2)

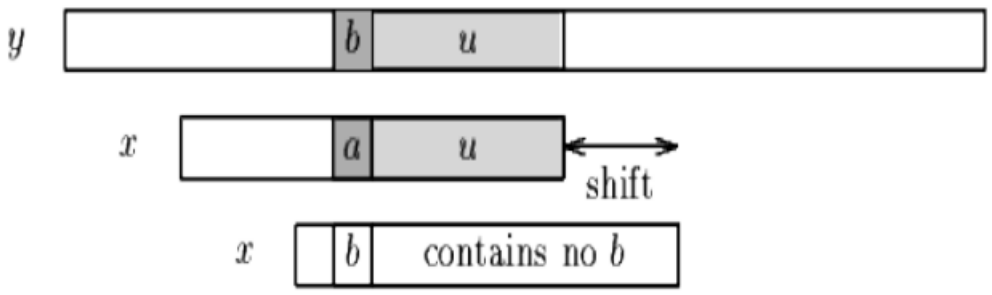


Рисунок 2.2. Зсув за евристикою стоп-символу

2. Евристика збігу суфіксів.

Також створюється таблиця суфіксів (див. табл. 2.2) для підрядка. Для суфіксів, які зустрічаються у зразку, що відображається найменшим значенням для того, щоб, на який потрібно звернути увагу, найвищий суфікс, який дорівнює

сегменту, який був у співвідношенні з текстом, який також дорівнює в, позначається, що для запобігання повторної невідповідності, символ перед v після зсуву не повинен дорівнювати символу, який стоїть перед v до зсуву (див. рис. 2.3).

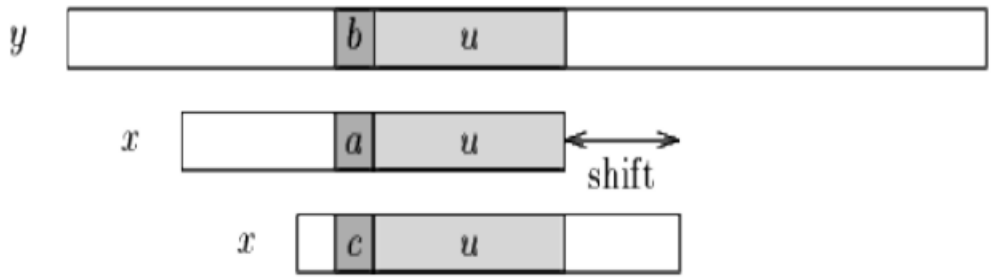


Рисунок 2.3. Зсув за евристикою збігу суфіксів

2.2.1. Таблиці евристик

Мінімальне зміщення, за допомогою якого потрібно змінити шаблон, щоб суфікс знову збігався, а символ, що передує суфіксу, був іншим, щоб продовжити порівняння, вказано в таблиці суфіксів для кожного потенційного символу підрядка запису.

Наприклад, якщо бажаним підрядком є "10000", ми отримуємо:

Таблиця 2.1

Таблиця стоп символів

Символ	0	1	Всі інші
Позиція	1	4	2

Таблиця 2.2

Таблиця суфіксів

№	Суфікс	Зсув
1	10000	3
2	10000	2
3	10000	1
4	10000	5

2.2.2. Алгоритм Райта

Алгоритм Райта є одним із відомих методів оптимізації Бойєра-Мура. Спочатку він порівнюється з правим текстовим символом вікна порівняння, потім, якщо вони збігаються, він порівнює перший символ зразка з лівим текстовим символом вікна порівняння, а потім, якщо вони збігаються, він порівнює серед символів підрядка із середнім символом вікна порівняння. Якщо вони збігаються, перейдіть до наступного символу від другого до останнього, час від часу знову порівнюючи середній символ.

Евристичний знак зупинки був збережений, але евристика збігу суфіксів була вилучена, оскільки на практиці вона суперечила дивам методу Райта.

Райт також сказав, що його підхід чудово підходить для визначення розташування підрядків в англійських текстах, а також опису існування символьних посилань [48].

Порівнюючи кожну букву шаблону в тексті, алгоритм Райта шукає шаблон x у заданому тексті y. Нижче описано, як працює пошук.

1. Останній символ шаблону вирівнюється за крайнім правим символом вікна.

2. Якщо є, початковий символ зразка відповідає крайньому лівому символу вікна.

3. Якщо запустити його знову, він порівнює середній символ підрядка з середнім символом вікна.

Якщо все пройшло належним чином, початкове порівняння почнеться з другого символу і закінчиться останнім.

Якщо все успішно, то початкове порівняння починається від другого символу до останнього. Якщо на будь-якому етапі алгоритму є невідповідність, то виконується зміщення за таблицею стоп символів, яка була обчислена в фазі попередньої обробки.

2.3. Алгоритм Чжу-Такаокі

Алгоритм ZhuTaki ще більше покращує метод Мура. Його розробники зрозуміли, що евристичний символ зупинки не функціонує в алфавіті нижнього регістру, тому вони змінили його так, щоб таблиця була для двох символів замість одного.

Процедура використання двох таблиць зсуву використовується, коли початкові два символи не збігаються або повністю збігаються. Таблиця суфіксів Бойєра-Мура та таблиця символів зупинки, в якій символи виражені парами.

Двовимірний масив може обробляти значення зміщення o.

Час складності попередньої обробки становить O (m + | O $|^2$), тоді як фаза пошуку становить O (m * n).

2.4. Оптимізований алгоритм Чжу-Такаокі-Райта

Створено новий метод для підвищення ефективності алгоритму пошуку рядків шляхом поєднання переваг алгоритмів ZhuTakaoka і Wright.

Попередня обробка виконується за алгоритмом у два кроки.

В алгоритмі використовується таблиця символів зупинки Чжу-Такаокі. Фаза пошуку була змінена, щоб використовувати метод Райта.

Отже, ось де ми знаходимося з точки зору визначення двоетапної конструкції:

1. Після символу підходу Райта алгоритм порівнює останнє вікно порівняння з підрядком, потім порівнює перший символ порівняння віку та підрядок, якщо вони збігаються. Зміщення виконується за допомогою таблиці символів зупинки зміщення для двох послідовних символів, що відповідають двом останнім символам вибірки, якщо обидва ці символи збігаються; в іншому випадку зміщення виконується за допомогою таблиці символів зупинки зміщення для двох послідовних символів, що відповідають двом останнім символам вибірки.

2. Інші символи перевіряються справа наліво на другому кроці, щоб побачити, чи повністю вони збігаються чи не збігаються. Вікно порівняння зміщується, а символи зупинки зміщуються, якщо всі символи збігаються. Зміщення відбувається відразу після таблиці символів зупинки в разі розбіжності.

Метод прискорюється, порівнюючи спочатку кінцевий символ і останній перший символ, а потім продовжуючи порівнювати символи справа наліво.

Крім того, щоб отримати якомога більше зміщення, була використана таблиця символів зупинки, що добре впливає на ефективність. Ці елементи разом узяті є причиною вдосконалення алгоритму.

2.5. Формати даних для алгоритмів пошуку підрядка в рядку

2.5.1. FASTA формат

Формат FASTA є одним із найбільш широко використовуваних форматів для файлів, що містять інформацію про біологічне секвенування [49]. Перший рядок даних завжди починається з літери ">" (більше) або, рідше, із символу, який використовується для коментаря ";" (крапка з комою), а в кінці може стояти знак «*» (зірочка), що позначає кінець послідовності, а все інше – нуклеотидну послідовність [50]. Він найчастіше використовується у відкритих банках геному через його простоту [51].

```
>fgenesh1_pg.C_scaffold_219000006
ATGGTAGATTTGAAAGATACTCTTGCAAAGTTCACATCTGCTCTTAGTTTTCAGGAGAAATGTAAGTTTC
CATCTCAACCTCAACAAAATTCCAAAGGGCAATACAATTCAAGTGCGAGTAGCTCTAGAAGCCAACACAT
GGATCAAGTTAAATCAGTCACTACTCTTTGTCGTGGTAAGGTTATTGAAAAACCTATTCTTGAACCTTGT
AAGAAAGATGATGAGTTAATCTCTGAGGGTAAGGAAATGGTTGAACCTGAACATTGCAAAGAAAAGATTG
ATTTCCCACCAGTACTTCCATTTCCTAATGCCATGACCAAACAAAGAAAAGTCAATCACAATTCTGAAAT
CTTTGGAACTTTCAAACAGGAAAAATCAAAGGCAAATCAGCCCAACACCAAGCGTAATCATGCCAACGAC
CCATTGGAGGTGGCAATTGGGCCAATCACAAGAGCTAGGGCAAAGAAGCATTGA

>eugene3.02190008
ATGTGGGGAATTGTTTTTTACCACCAACCAATTCTTGCAGCTACTATCATGTCAGATCAAAAGAACAAAA
CACCCCCTCCCAATGTAGAACTTAAATTCCAAATGCAAGCCATGACGAAGATGATGGAAAGAATGAATTC
CGTGATGGGGAATGTGTGTGACAGACTTGAGAAAGTGGGAAAACAAGGTAATGTCAGAACATGTACCCAA
GACGTGAGAAAGGTTGGGGCTGAACCAAAATCAAACAATGGCAGAGGGGCTGAAAGGCCAAGGTGGGCTG
ATTATGCGGATTTTGAGGTGGACGTTGATGATATTGTTGATGGTGGTTTTAAGGATGAGACCATAGGCCA
TCAAAAAGGTTTTCAACACCATAGAAACCGAAGGGATTTCATGTATTTTGACGGGGTGTTATGGCAAAAG
AAAATGAGGATTCAAAAGGAGAGGTGTCAAAGGGAGAGAAATAAAGAGATTGGTGTTCTAAAAAATGAAT
CCAAGAGTCTATACCGTATTCTAGGGGAGATGAAGCAAGAACTTGATGTGTTAATGGCAATAGTCAATGC
CCAACAAGCTTCAGAAAGAGAGGAGAAAATACGCTGCAATGATGATATACGAAATAGAATGGATGCTACT
TTCATCAAAAGTTGGTGGCGACGATTGTTGGCAAGAAAAGAACTCCAAAGGCTTCAAAAAGAGGCTAAGg
aatttggtccttaa

>eugene3.02190007
ATGGAAGGCATGGCTAGTTCAGCAGCACAAACAACCCCTACCACGCAGATTCCACCAGCAGCCCCTACGA
CCAGCATGACCATGGATAATGTGGTACCACTGGTGCGACTAGTCAAGAGTATGAGGGAAATGGGCTGTGA
ACCGTATATGGGGGAACAAGATGCAGAGATAGCTGGAAGATTGATCAGGAAGGTAGAAAAGACAATGATT
```

Рисунок 2.4. Приклад даних у FASTA форматі

2.5.2. FASTQ формат

Формат FASTQ – це текстовий файл, до якого також можна отримати доступ із FASTA і, серед іншого, підтримує вимірювання якості даних. Кожен нуклеотид має свій індекс якості, що вказує на можливість помилки [7].

Кожна послідовність представлена чотирма рядками у файлі FASTQ.

Рядок 1 – починається зі знака «@», який слід використовувати для позначення наставника та будь-якої додаткової інформації (наприклад, рядок заголовка FASTA).

Рядок 2 – послідовності нуклеотидів.

Рядок 3 – починається з символу "+" і знову за ним слідує ідентифікатор послідовності (і будь-який опис).

Рядок 4 – кодує значення якості для послідовності в рядку 2 і повинен містити ту ж кількість символів, скільки символів зазначено в біологічній послідовності.

Оригінальні файли Sanger FASTQ також дозволяли звести послідовність і якість рядків до мінімуму, тобто розділити на кілька рядків, але це не обов’язково; насправді, поганий вибір «@» та «+» як маркерів може ускладнити аналіз; ці символи також можна вибрати в рядку якості.

Колись це був найбільш використовуваний формат даних для програмних додатків, але через його розмір перейшов формат FASTA.

На ілюстрації наведено приклад (див. рис. 2.5).

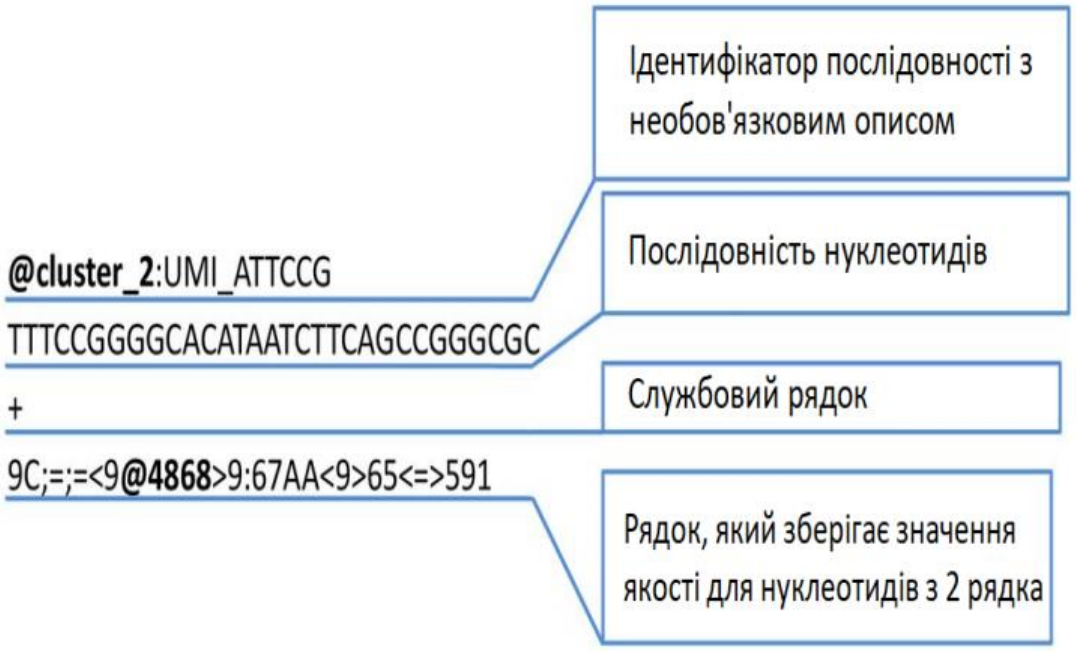


Рисунок 2.5. Приклад даних у FASTQ формат

2.6 Ефективні варіанти алгоритму зворотного збігу Oracle

2.6.1 Алгоритми швидкого пошуку та швидкого пошуку вперед

Алгоритм швидкого пошуку [6] є дуже простим, але ефективним варіантом алгоритму Боєра Мура. Нехай p — шаблон довжини m, а t — текст довжини n над скінченним алфавітом Σ. Алгоритм швидкого пошуку обчислює приріст свого зсуву, застосовуючи правило поганих символів тоді й тільки тоді, коли під час першого порівняння символів виникає невідповідність, а саме під час порівняння символів p[m−1] і t[s+m−1], де s - поточний зсув. В іншому випадку використовується правило доброго суфікса.

Алгоритм швидкого пошуку вперед [7] підтримує ту саму структуру алгоритму швидкого пошуку, але він заснований на модифікованій версії правила доброго суфікса, який називається правилом прямого суфікса, яке використовує символ перегляду вперед для визначення більші зміни. Таким чином, якщо перша невідповідність виникає в позиції i < m − 1 шаблону p, правило хорошого суфікса пропонує вирівняти підрядок t[s+i+1..s+m] з його крайнім правим входом у p перед. символом, відмінним від p[i]. Якщо такого випадку не існує, правило прямого доброго суфікса пропонує приріст зсуву, який дозволяє узгодити найдовший суфікс t[s+i+1..s+m] з префіксом p. Це відповідає просуванню зсуву s на $-\!\!\rightarrow$gsP (i + 1, t[s + m]) позицій, де:

$$\overrightarrow{gs}_P(j,c) =_{\text{Def}} \mathbf{min}(\{0 < k \leq m \mid p[j-k\,..\,m-k-1] \sqsupset p \text{ and } (k \leq j-1 \rightarrow p[j-1] \neq p[j-1-k]) \text{ and } p[m-k] = c\} \cup \{m+1\}),$$

для j = 0, 1, . . . , m і c ∈ Σ.

Правило good-suffix та правило forward good-suffix вимагають таблиці розміру m та m·|Σ| відповідно. Їх можна побудувати за час O(m) і O(m·max(m, |Σ|)), відповідно.

Більш ефективні реалізації алгоритму швидкого пошуку та швидкого пошуку вперед отримують за тими ж напрямками алгоритму Tuned-Boyer-Moore [13] за допомогою швидкого циклу, використовуючи методику, описану в розділі 2.1 і показану на рис. 2.6.

Тоді наступна фаза відповідності може початися з (m−2)-го символу шаблону. Наприкінці фази відповідності алгоритми використовують правило good-suffix для зсуву.

2.6.2 Алгоритм Q-Hash

Алгоритми сімейства q-Hash були введені в [15], де автор представив адаптацію алгоритму зіставлення множинних рядків By та Манбера [18] до задачі відповідності одного рядка.

Ідея алгоритму q-Hash полягає у врахуванні факторів шаблону довжини q. Кожен підрядок w такої довжини q хешується за допомогою функції h на цілі значення в межах 0 і 255. Потім алгоритм обчислює на етапі попередньої обробки функцію shift() : {0,1,...,255} → {0 ,1,...,m − q}. Формально для кожного 0 ≤ c ≤ 255 значення shift(c) визначається як:

$$shift(c) = \mathbf{min}\left(\{0 \le k < m-q \mid h(p[m-k-q\,..\,m-k-1]) = c\} \cup \{m-q\}\right).$$

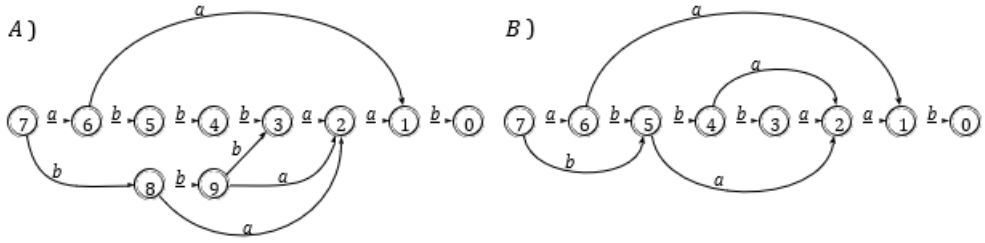


Рис. 2.6 - Фактор-автомат (A) і фактор-оракул (B) зворотного шаблону p = baabbba. Факторний автомат розпізнає всі і тільки фактори зворотного зразка

Фаза пошуку алгоритму складається із зчитування для кожного зсуву s шаблону в тексті підрядка w = t[s+m−q ..s+m−1] довжини q.

Якщо shift[h(w)] > 0, то застосовується зсув довжини shift[h(w)]. В іншому випадку, коли shift[h(w)] = 0, шаблон x наївно перевіряється в тексті.

У цьому випадку застосовується зсув довжини sh, де sh = m − 1 − i з i = max{0 ≤ j ≤ m − q|h(x[j ..j + q − 1]) = h(x[m − q + 1..m − 1]}.

2.6.3 Алгоритми зворотного автоматичного узгодження

Алгоритми, засновані на стратегії Бойєра-Мура, зазвичай намагаються узгодити суфікси шаблону, але можна зіставити деякі префікси або деякі фактори шаблону, скануючи поточне вікно тексту справа наліво, щоб покращити довжину шаблону. зміни. Це можна зробити за допомогою факторних автоматів і факторних оракулів.

Фактор-автомат [4,9,3] шаблону p, Aut(p), також називається фактором DAWG для p (для орієнтованого ациклічного графа слів). Такий автомат розпізнає всі фактори p. Формально мова, яку розпізнає Aut(p), визначається наступним чином

L(Aut(p)) = {u ∈ Σ∗ : існує v,w ∈ Σ∗ такі, що p = vuw}.

Фактор-оракул шаблону p, Oracle(p), є дуже компактним автоматом, який розпізнає принаймні всі множники p і трохи більше інших слів. Формально Oracle(p) є автоматом {Q,m,Q,Σ,δ} таким, що

1. Q містить рівно m + 1 станів, скажімо, Q = {0,1,2,3,...,m}

2. m — початковий стан

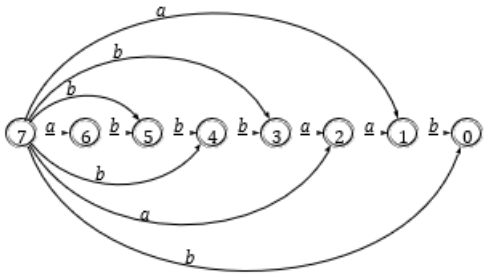
3. всі стани є остаточними

4. мова, прийнята Oracle(p), така, що L(Aut(p)) ⊆ L(Oracle(p)).

Незважаючи на те, що фактор-оракул здатний розпізнавати слова, які не є факторами шаблону, його можна використовувати для пошуку шаблону в тексті, оскільки єдиний фактор p довжини, більшої або рівної m, розпізнається оракул - це сам шаблон. Розрахунок оракула є лінійним у часі та просторі за довжиною шаблону.

На рисунку 2.6 показані факторний автомат і фактор-оракул зворотного шаблону p = baabbba.

Фактор-автомат структур даних і фактор-оракул використовуються відповідно в [10,11] і в [1] для отримання в середньому оптимальних алгоритмів узгодження шаблонів (рис. 2.7).



abbbaab.

Рисунок 2.7 - Недетермінований фактор

BDM (для зворотного зіставлення Dawg), тоді як алгоритм, що використовує фактор оракула, називається BOM (для зворотного зіставлення Oracle). Такі алгоритми переміщують по тексту вікно розміру m. Для кожної нової позиції вікна автомат, зворотний p, використовується для пошуку коефіцієнта p справа наліво вікна. Основна ідея алгоритмів BDM і BOM полягає в тому, що якщо його зворотний пошук не вдався на літері c після прочитання слова u, тоді cu не є фактором p, і переміщення початку вікна відразу після c є безпечним. Якщо розпізнати коефіцієнт довжини m, ми знайшли появу шаблону.

Алгоритми BDM і BOM мають квадратичну складність у найгіршому випадку, але є оптимальними в середньому, оскільки вони виконують O(n(logσm)/m) перевірок текстових символів, досягаючи найкращої межі, показаної Яо [19] у 1979 році.

2.6.4 Алгоритм BNDM

Алгоритм BNDM [16] (для зворотного недетермінованого збігу Dawg) є біт-паралельним моделюванням [2] алгоритму BDM. Він використовує недетермінований автомат замість детермінованого в алгоритмі BDM.

На рисунку 2.7 показана недетермінована версія факторного автомата для зворотного шаблону p = baabbba. Для кожного символу c ∈ Σ на етапі попередньої обробки ініціалізується бітовий вектор B[c]. i-й біт дорівнює 1 у цьому векторі, якщо c з'являється у зворотному шаблоні в позиції i. Інакше i-й біт встановлюється на 0. Вектор стану D ініціалізується на 1m. Такий самий вид сканування справа наліво у вікні розміру m виконується, як і в алгоритмі BOM, тоді як вектор стану оновлюється подібним чином, як і в алгоритмі Shift-And [2]. Якщо після цієї операції оновлення m-й біт дорівнює 1, ми знайшли префікс, що починається з позиції j, де j – кількість оновлень, зроблених у цьому вікні. Таким чином, якщо j – перша позиція у вікні, відповідність знайдено.

Спрощена версія BNDM, яка називається SBNDM, представлена в [12]. Цей алгоритм відрізняється від оригінального основним циклом, який розпочинає кожну ітерацію з перевірки двох послідовних текстових символів. Крім того, він реалізує швидкий цикл для отримання кращих результатів у середньому. Результати експерименту показують, що цей спрощений варіант завжди ефективніший за вихідний.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Реалізація алгоритмів пошуку підрядка

Для проведення порівняльного аналізу були реалізовані наступні методи:

1. Наївний алгоритм - вважається найпростішим алгоритмом пошуку підрядки в рядку. В якості атомарної операції використовується порівняння двох символів.

2. Алгоритм Рабіна-Карпа - в якості атомарної операції було використано порівняння символів, створення нового хеша та порівняння хешей. Для перекладу аркуша із символів у аркуш із чисел (маленьких чисел, що не перевершують потужність використовуваного алфавіту) була використана функція препроцессингу LabelEncoder. Однак складність її роботи не підсумовувалась зі складністю алгоритму, щоб не створити спотворення при порівнянні алгоритму з іншими.

3. Алгоритм Боєра-Моєра - для алгоритму BMH підсумовуються всі операції в препроцессингу, а також кожен випадок порівняння символів.

4. Алгоритм Кнута-Моріса-Пратта - складність алгоритму підсумовується з подвійного проходження текстом під час формування масиву евристики префікса. А також порівняння символів під час роботи основного алгоритму.

В якості вхідних даних виступає короткий алфавіт із 4 символів та довгий, що складається з усіх англійських літер алфавіту:

```
alphabet_sh = ['a','b','c','d']
alphabet= [chr(i) for i in range(97,123)]
```

Тестування проводилося на масиві довжиною 10,100,1000 символів для кожного з алгоритмів пошуку підрядка. За відображення результатів порівняльного аналізу відповідають функції бібліотеки Pandas (рис. 3.1 – 3.2).

	max	average	min
Naive : 10	16.0	10.406	8.0
Naive : 100	148.0	128.489	110.0
Naive : 1000	1363.0	1309.319	1259.0
Naive : 10000	13307.0	13123.002	12950.0
RK : 10	22.0	16.414	16.0
RK : 100	214.0	200.440	196.0
RK : 1000	2086.0	2042.776	2014.0
RK : 10000	20569.0	20465.575	20356.0
BMH : 10	9.0	3.503	2.0
BMH : 100	36.0	19.165	7.0
BMH : 1000	224.0	177.659	131.0
BMH : 10000	1883.0	1753.102	1583.0
KMP : 10	28.0	26.123	26.0
KMP : 100	212.0	207.527	206.0
KMP : 1000	2036.0	2021.574	2009.0
KMP : 10000	20201.0	20162.556	20115.0

Рис. 3.1 – Тестування алгоритмів пошуку підрядка

	max	average	min
Naive : 10	11.0	8.315	8.0
Naive : 100	110.0	101.872	98.0
Naive : 1000	1057.0	1037.836	1020.0
Naive : 10000	10476.0	10397.450	10327.0
RK : 10	17.0	16.006	16.0
RK : 100	199.0	196.111	196.0
RK : 1000	2003.0	1997.151	1996.0
RK : 10000	20025.0	20007.412	19997.0
BMH : 10	4.0	2.146	2.0
BMH : 100	8.0	3.427	2.0
BMH : 1000	32.0	16.407	6.0
BMH : 10000	191.0	146.120	103.0
KMP : 10	27.0	26.001	26.0
KMP : 100	207.0	206.005	206.0
KMP : 1000	2007.0	2006.048	2006.0
KMP : 10000	20012.0	20006.559	20006.0

Рис. 3.2 – Тестування алгоритмів пошуку підрядка в звичайному англійському алфавіті

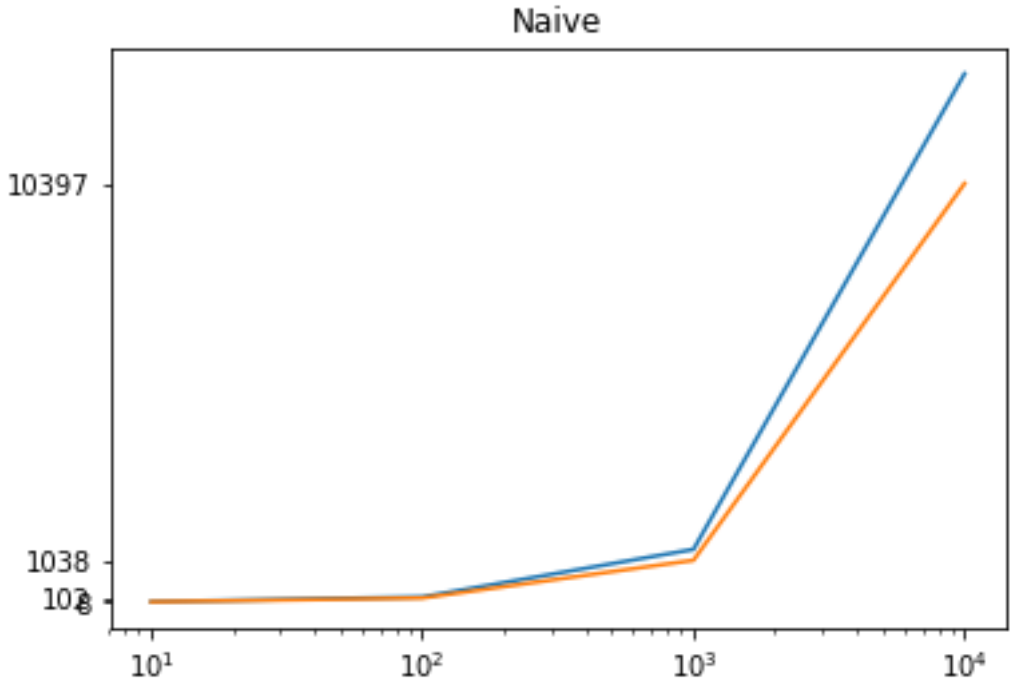


Рис. 3.3 – Залежність кількості операцій від розміру вхідних данних для наївного алгоритму (ось X – кількість слів в алфавіті)

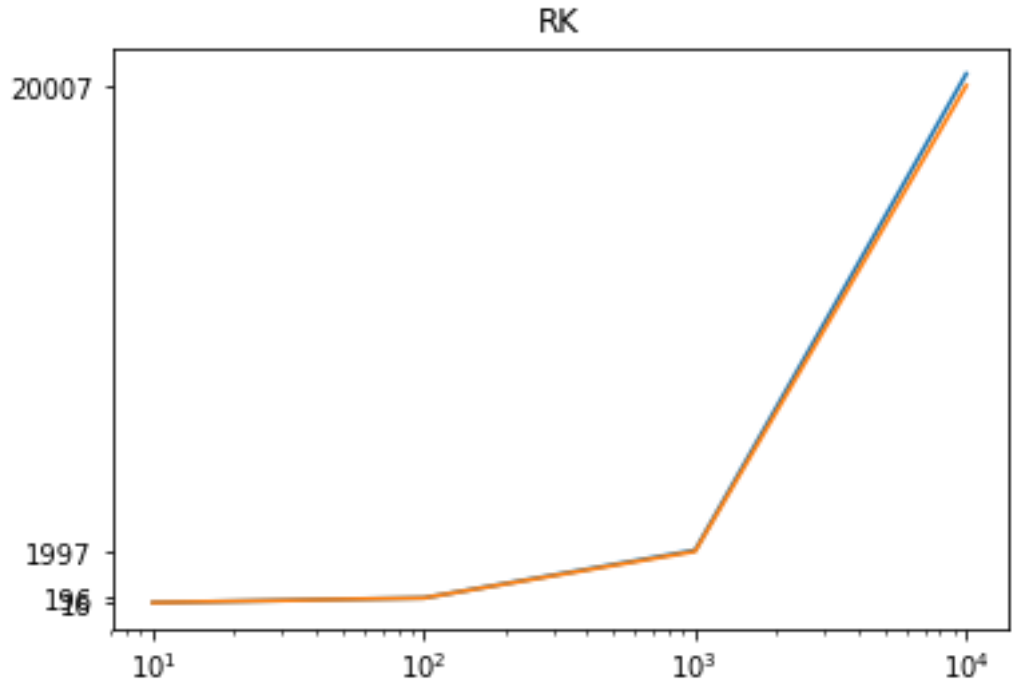


Рис. 3.4 – Залежність кількості операцій від розміру вхідних данних для алгоритму Рабіна-Карпа (ось X – кількість слів в алфавіті)

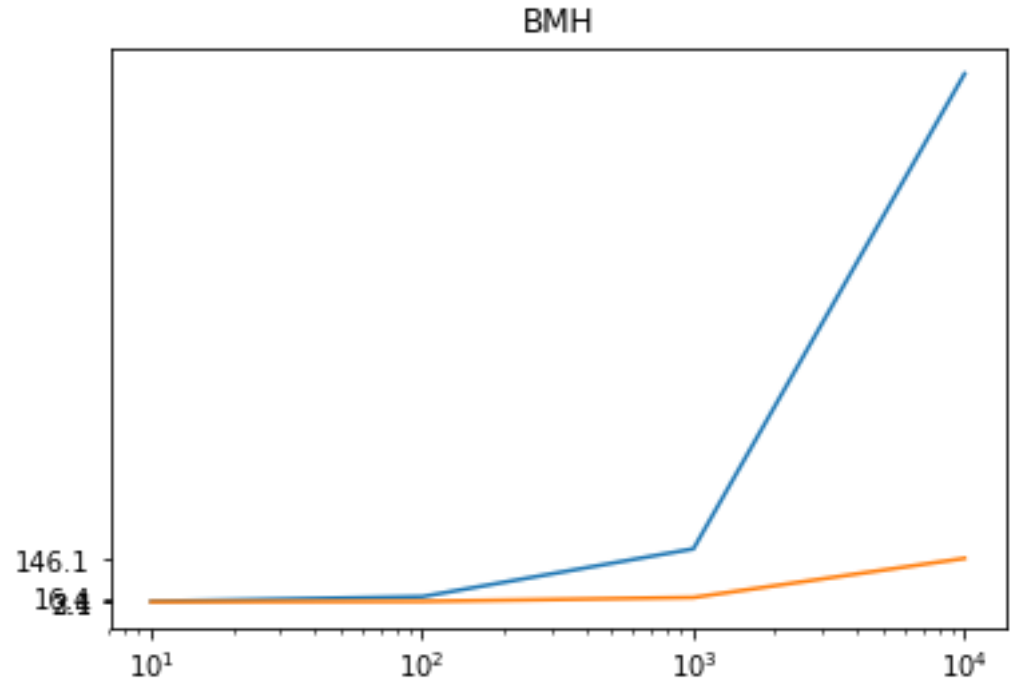


Рис. 3.5 – Залежність кількості операцій від розміру вхідних данних для Боєра-Моєра алгоритму (ось X – кількість слів в алфавіті)

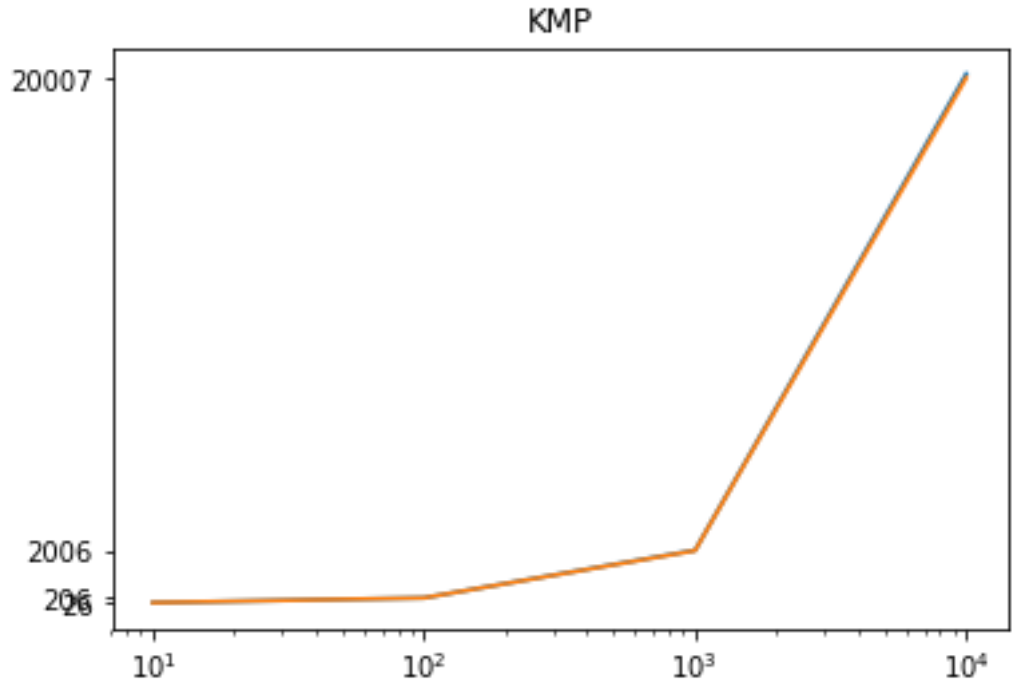


Рис. 3.6 – Залежність кількості операцій від розміру вхідних данних для алгоритму Кнута-Моріса-Пратта (ось X – кількість слів в алфавіті)

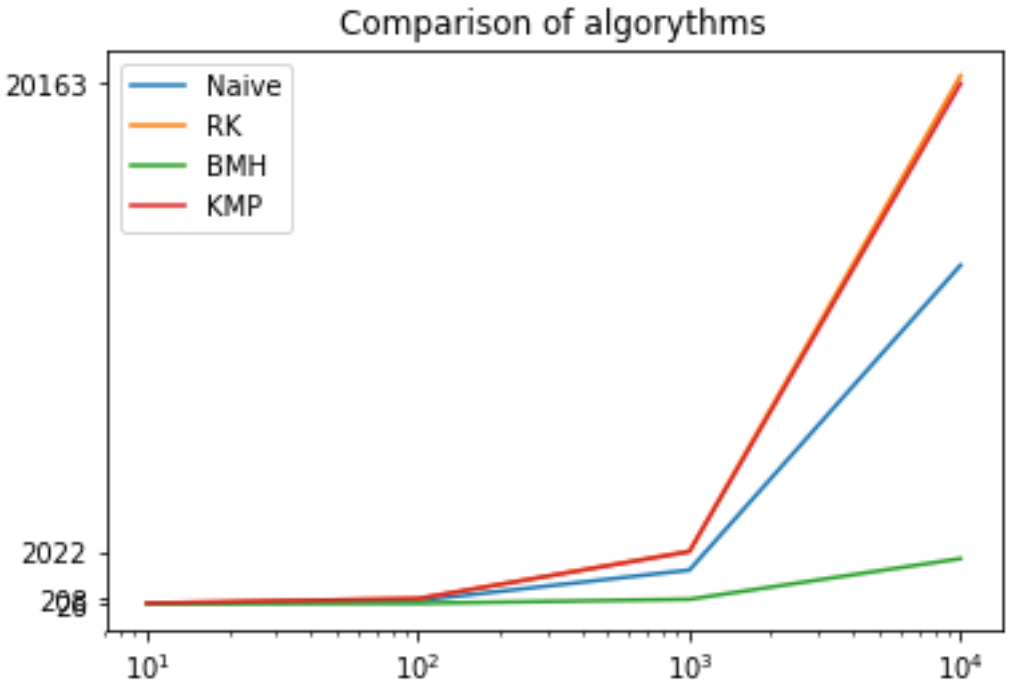


Рис. 3.7 - Порівняльний аналіз роботи алгоритмів

Як можна помітити, не у всіх алгоритмів збільшення довжини алфавіту веде до збільшення кількості операцій. У RK, BMH ці значення майже збігаються, що пов'язано зі специфікою алгоритму, а також з тим, що шуканий патерн не змінюється при зміні алфавіту.

Діаграма порівняння алгоритмів оманлива, тому що для кожного ми брали різні атомарні операції виходячи з логіки алгоритмів (десь додавалася хешування, десь препроцесинг - все це збільшує число операцій, але не говорить про низьку ефективність алгоритму). Так, наприклад, судячи з діаграми, алгоритм BMH є найефективнішим, а друге місце посідає наївний алгоритм. Якщо перше не викликає запитань, то друге є досить сумнівним.

Також з таблиць і графіків можна помітити, що кількість операцій зростає лінійно, тобто збільшення довжини тексту на порядок (у 10 разів) так само збільшує кількість операцій (у 10 разів).

3.2 Сучасні алгоритми пошуку підрядка

Для проведення порівняльного аналізу сучасних алгоритмів з алгоритмами, описаними в пункті 3.1 були реалізовані та порівняні наступні алгоритми:

1. Алгоритм грубої сили полягає в перевірці на всіх позиціях тексту від 0 до n-m, чи починається поява шаблону там чи ні. Потім після кожної спроби він зміщує візерунок рівно на одну позицію вправо.

Алгоритм грубої сили не вимагає фази попередньої обробки та постійного додаткового простору на додаток до шаблону та тексту. Під час фази пошуку порівняння текстових символів можна виконувати в будь-якому порядку. Часова складність цієї фази пошуку O(mn) (наприклад, при пошуку am-1b). Очікувана кількість порівнянь текстових символів становить 2n.

2. Алгоритм BNDM (Backward Nondeterministic Dawg Matching algorithm) використовує таблицю B, яка для кожного символу c зберігає бітову маску. Маска в Bc встановлюється тоді і тільки тоді, коли xi=c.

Стан пошуку зберігається в слові d=dm-1 .. d0, де довжина шаблону m менша або дорівнює розміру машинного слова.

Біт di на ітерації k встановлюється, якщо an, тільки якщо x[m-i .. m-1-i+k]=y[j+m-k .. j+m-1]. На ітерації 0 d встановлюється на 1m-1. Формула для оновлення d відповідає d'= (d & B[yj]) << 1.

Збіг існує тоді і тільки тоді, коли після ітерації m воно виконується dm-1=1.

Коли dm-1=1, алгоритм відповідає префіксу шаблону в поточній позиції вікна j. Найдовший відповідний префікс дає перехід до наступної позиції.

3. *Алгоритм BOM.* Алгоритми типу Бойєра-Мура відповідають деяким суфіксам шаблону, але можна знайти відповідність деяким префіксам шаблону, скануючи символ вікна справа наліво, а потім покращуючи довжину зсувів. Це стає можливим завдяки використанню суфікса оракула зворотного зразка. Ця структура даних є дуже компактним автоматом, який розпізнає принаймні всі суфікси слова і трохи більше інших слів. Алгоритм узгодження рядків, що використовує оракул зворотного шаблону, називається алгоритмом зворотного збігу Oracle.

Суфіксним оракулом слова w є детермінований кінцевий автомат O(w) =(Q, q0, T, E).

Мова, яку приймає O(w), є такою, що {u в Sigma* : існує v в Sigma* такий, що w = vu} в L(O(w)).

Фаза попередньої обробки алгоритму Backward Oracle Matching полягає в обчисленні суфіксного оракула для зворотного шаблону xR. Незважаючи на те, що він здатний розпізнавати слова, які не є фактором шаблону, суфікс оракул може використовуватися для зіставлення рядків, оскільки єдине слово довжини, більшої або рівної m, яке розпізнається оракулом, є сам зворотний шаблон . Обчислення оракула є лінійним у часі та просторі за довжиною шаблону.

Під час фази пошуку алгоритм Backward Oracle Matching аналізує символи вікна справа наліво за допомогою автомата O(xR), починаючи з стану q0. Він триває до тих пір, поки для поточного символу більше не буде визначено перехід. У цей момент довжина найдовшого префікса шаблону, який є суфіксом відсканованої частини тексту, менша за довжину шляху в O(xR) від початкового стану q0 до останнього зустрічного кінцевого стану. Знаючи цю довжину, тривіально обчислити довжину зміни, яку потрібно виконати.

Алгоритм зворотного зіставлення Oracle має квадратичну складність у найгіршому випадку, але він є оптимальним у середньому.

В якості вхідних даних були використані дані секвенування ДНК. За відображення графічного результату відповідає бібліотека для графічної візуалізації Plotly (рис. 3.8).

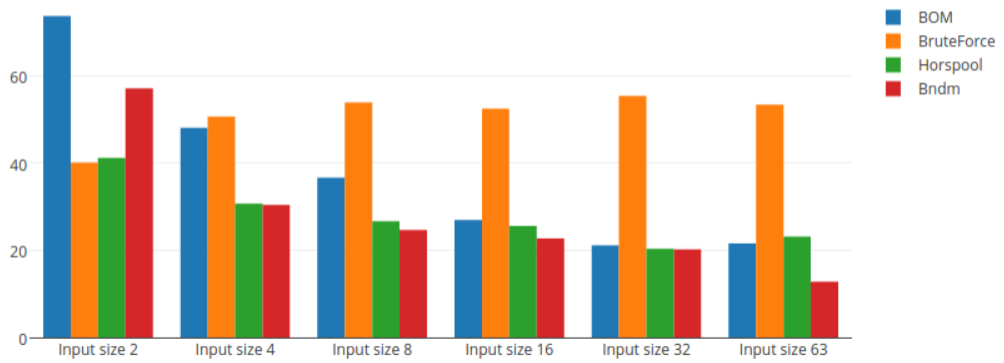


Рис. 3.8 – Порівняльний аналіз сучасних алгоритмів пошуку підрядка

ВИСНОВКИ

У даній кваліфікаційній роботі бакалавра дисертації виділено недоліки сучасних інформаційно-пошукових систем, які використовуються для пошуку та роботи з інформацією у мережі Інтернет.

У першому розділі здійснено аналіз методів пошуку інформації в мережі Інтернет сучасних інформаційно-пошукових систем, зокрема, методів пошуку системи Google, яка в середньому обробляє понад 40 тисяч пошукових запитів за секунду. Виділено переваги та недоліки пошукової системи Baidu, яка має широкий спектр різноманітних послуг, включаючи бібліотеку знань, систему обміну файлами та власну платформу соціальних медіа даних.

У другому розділі на основі виділених недоліків існуючих алгоритмів пошуку інформації (підрядка) запропоновано та описано алгоритм пошуку інформації на основі зведення пошукового запиту до індивідуальних особливостей користувача. Запропонований алгоритм дозволяє збільшити вартість пошуку, замінивши алгоритм ранжування алгоритмом сортування в пошукових системах за категоріями. Інформація заповнюється системою послідовних об'єктів, яка представляє собою список. Об'єкти містять ключову інформацію, за допомогою цих ключів ми можемо налаштувати роботу з інформацією і підвищити актуальність пошуку. Аналіз конфігурації служби пошуку виконується для збільшення вартості пошуку для кінцевого користувача.

Система сортування аналізує запит кінцевого користувача і надає знайдену інформацію, відсортовану за релевантністю.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Zong Woo Geem. Recent Advances in Harmony Search Algorithm// Studies in Computation Intelligence. – 2016. – P. 51– 75.

2. Schütze, Hinrich, Christopher D. Manning, Raghavan, Prabhakar: Introduction to information retrieval, Prabhakar// Cambridge, UK: Cambridge University Press. – 2008. – P. 121– 125.

3. Автоматизовані інформаційно-пошукові мови. [Електронний ресурс]. — Режим доступу: http://ubooks.com.ua/books/00092/inx13.php. — Дата доступу: Травень 2019. — Назва з екрана.

4. Sergey Brin, Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine// Computer Science Department, Stanford University, Stanfor. – 1998. – P. 45– 48.

5. Google PageRank. [Електронний ресурс]. — Режим доступу: https://ahrefs.com/blog/google-pagerank/. — Дата доступу: Травень 2019. — Назва з екрана.

6. Benjamin M. Schmidt & Matthew M. Chingos: Ranking Doctoral Programs by Placement A New Method// Cambridge, UK: Cambridge University Press. – 2007. – P. 523–529.

7. Matthew Richardson, Pedro Domingos: The Intelligent Surfer:Probabilistic Combination of Link and Content Information in PageRank// The Politics of Search. – 2001. – P. 364–374.

8. Bradley C. Love, Steven A. Sloman: Mutability and the determinants of conceptual transformability// Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society. – 2001. – P. 654–659.

9. Brin, S., Page, L.: The anatomy of Web search engine// Computer Networks and ISDN Systems. – 2001. – P. 107–117.

10. Page, Larry: PageRank: Bringing Order to the Web// Stanford Digital Library Project. – 2002. – P. 47–52. 85

11. Taher Haveliwala & Sepandar Kamvar: The Second Eigenvalue of the Google Matrix// Stanford University Technical Report. – 2008. – P. 20– 34.

12. Google Removes Pagerank and Launches Rainbrain Technology. [Електронний ресурс]. — Режим доступу: https://web.archive.org/web/20161223012407/http://seometrics.in/googleremoves-pagerank-launches-rainbrain-technology/. — Дата доступу: Травень 2019. — Назва з екрана.

13. The Periodic Table. [Електронний ресурс]. — Режим доступу: https://searchengineland.com/seotable— Дата доступу : Травень 2019. — Назва з екрана.

14. Matthew Richardson, Pedro Domingos, A. Chingos: The Intelligent Surfer:Probabilistic Combination of Link and Content Information// Cambridge, UK: Cambridge University Press. – 2001. – P. 364–374.

15. The real impact of Google’s RankBrain on search traffic. [Електронний ресурс]. — Режим доступу: https://thenextweb.com/contributors/2017/05/22/real-impact-googlesrankbrain-search-traffic/#.tnw_nYDXqPTo. — Дата доступу: Травень 2019. — Назва з екрана.

16. The Anatomy of a Large-Scale Hypertextual Web Search Engine. [Електронний ресурс]. — Режим доступу: http://infolab.stanford.edu/~backrub/google.html. — Дата доступу: Травень 2019. — Назва з екрана.

17. Google's Tensor Processing Unit could advance Moore's Law. [Електронний ресурс]. — Режим доступу: https://www.pcworld.com/article/3072256/googles-tensor-processingunit-said-to-advance-moores-law-seven-years-into-the-future.html. — Дата доступу : Травень 2019. — Назва з екрана. 86

18. Google: RankBrain. [Електронний ресурс]. — Режим доступу: https://searchengineland.com/library/google/google-rankbrain. — Дата доступу : Травень 2019. — Назва з екрана.

19. What are malicious websites?. [Електронний ресурс]. — Режим доступу: https://us.norton.com/internetsecurity-malware-what-aremalicious-websites.html. — Дата доступу : Травень 2019. — Назва з екрана.

20. Best Search Engines in The World. [Електронний ресурс]. — Режим доступу: https://www.inspire.scot/blog/2016/11/11/top-12-best-searchengines-in-the-world238. — Дата доступу : Травень 2019. — Назва з екрана.

21. What is Google Team Drive. [Електронний ресурс]. — Режим доступу: https://www.systoolsgroup.com/google-drive/team-drive.html. — Дата доступу : Травень 2019. — Назва з екрана.

22. 8 major Google algorithm updates, explained. [Електронний ресурс]. — Режим доступу: https://searchengineland.com/8-major-googlealgorithm-updates-explained-282627. — Дата доступу : Травень 2019. — Назва з екрана.

23. Web Crawlers and User-Agents - Top 10 Most Popular. [Електронний ресурс]. — Режим доступу: https://www.keycdn.com/blog/webcrawlers. — Дата доступу : Травень 2019. — Назва з екрана.

24. How does a search engine work?. [Електронний ресурс]. — Режим доступу: https://www.bigcommerce.com/ecommerce-answers/how-doessearch-engine-work/. — Дата доступу : Травень 2019. — Назва з екрана.

25. Sorting, searching and algorithm analysis. [Електронний ресурс]. — Режим доступу:https://pythontextbok.readthedocs.io/en/1.0/Sorting_and_Searching_Algorithms.html. — Дата доступу : Травень 2019. — Назва з екрана. 87

26. Searching Algorithms. [Електронний ресурс]. — Режим доступу: https://www.geeksforgeeks.org/searching-algorithms/. — Дата доступу : Травень 2019. — Назва з екрана.

27. Shervin Daneshpajouh, Mojtaba Mohammadi Nasiri, Mohammad Ghodsi,: A Fast Community Based Algorithm for Generating Crawler Seeds Set// Computer Science Department, Stanford University, Stanfor. – 2008. – P. 153–178.

28. A comprehensive and scalable database search system for metaproteomics. [Електронний ресурс]. — Режим доступу:

https://www.ncbi.nlm.nih.gov/pubmed/27528457. — Дата доступу : Травень 2019. — Назва з екрана.

29. Search trademark database. [Електронний ресурс]. — Режим доступу: https://www.uspto.gov/trademarks-application-process/search-trademarkdatabase. — Дата доступу : Травень 2019. — Назва з екрана.

30. Scan Algorithms. [Електронний ресурс]. — Режим доступу: http://www.cs.ecu.edu/karl/3300/spr14/Notes/Algorithm/scan.html. — Дата доступу : Травень 2019. — Назва з екрана

31.C. Allauzen, M. Crochemore, and M. Raffinot: *Factor oracle: a new structure for pattern matching*, in SOFSEM'99, J. Pavelka, G. Tel, and M. Bartosek, eds., LNCS 1725, Milovy, Czech Republic, 1999, Springer-Verlag, Berlin, pp. 291–306.

32. R. Baeza-Yates and G. H. Gonnet: *A new approach to text searching*. Commun. ACM, 35(10) 1992, pp. 74–82.

33. A. Blumer, J. Blumer, A. Ehrenfeucht, D. Haussler, M. T. Chen, and J. Seiferas: *The smallest automaton recognizing the subwords of a text*. Theor. Comput. Sci., 40(1) 1985, pp. 31–55.

34. A. Blumer, J. Blumer, A. Ehrenfeucht, D. Haussler, and R. McConnel: *Linear size finite automata for the set of all subwords of a word: an outline of results*. Bull. Eur. Assoc. Theor. Comput. Sci., 21 1983, pp. 12–20.

35. R. S. Boyer and J. S. Moore: *A fast string searching algorithm*. Commun. ACM, 20(10) 1977, pp. 762–772.

36. D. Cantone and S. Faro: Fast-Search*: a new efficient variant of the Boyer-Moore string matching algorithm*. WEA 2003, LNCS 2647(4/5) 2003, pp. 247–267.

37. D. Cantone and S. Faro: *Fast-Search Algorithms: New Efficient Variants of the Boyer-Moore Pattern-Matching Algorithm*. J. Autom. Lang. Comb., 10(5/6) 2005, pp. 589–608.

38. C. Charras and T. Lecroq: *Handbook of exact string matching algorithms*, King's College Publications, 2004.

39. M. Crochemore: *Optimal factor tranducers*, in Combinatorial Algorithms on Words, A. Apostolico and Z. Galil, eds., vol. 12 of NATO Advanced Science Institutes, Series F, Springer-Verlag, Berlin, 1985, pp. 31–44.

40. M. Crochemore, A. Czumaj, L. Ga¸sieniec, S. Jarominek, T. Lecroq, W. Plandowski, and W. Rytter: *Speeding up two string matching algorithms*. Algorithmica, 12(4/5) 1994, pp. 247–267.

41. M. Crochemore and W. Rytter: *Text algorithms*, Oxford University Press, 1994.

42. J. Holub and B. Durian: *Fast variants of bit parallel approach to suffix automata*. Talk given in: The Second Haifa Annual International Stringology Research Workshop of the Israeli Science Foundation, http://www.cri.haifa.ac.il/events/2005/string/presentations/Holub.pdf, 2005.

43. A. Hume and D. M. Sunday: *Fast string searching*. Softw. Pract. Exp., 21(11) 1991, pp. 1221– 1248.

44. D. E. Knuth, J. H. Morris, Jr, and V. R. Pratt: *Fast pattern matching in strings*. SIAM J. Comput., 6(1) 1977, pp. 323–350.

45. T. Lecroq: *Fast exact string matching algorithms*. Inf. Process. Lett., 102(6) 2007, pp. 229–235.

46. G. Navarro and M. Raffinot: *A bit-parallel approach to suffix automata: Fast extended string matching*, in Proceedings of the 9th Annual Symposium on Combinatorial Pattern Matching, M. Farach-Colton, ed., LNCS 1448, Piscataway, NJ, 1998, Springer-Verlag, Berlin, pp. 14–33.

47. D. M. Sunday: *A very fast substring search algorithm*. Commun. ACM, 33(8) 1990, pp. 132–142.

48. S. Wu and U. Manber: *A fast algorithm for multi-pattern searching*, Report TR-94-17, Department of Computer Science, University of Arizona, Tucson, AZ, 1994.

49. A. C. Yao: *The complexity of pattern matching for a random string*. SIAM J. Comput., 8(3) 1979, pp. 368–387.