ДИПЛОМНА РОБОТА
БАКАЛАВРА
ДРБ. ІПз - 42.00.00.000 ПЗ
Група ІПз-18-1
Федорчак Василь
2022

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерія програмного забезпечення

УДК **004.4**

ДИПЛОМНА РОБОТА БАКАЛАВРА

Тема: "Побудова архітектури ВЕБ-сервісів авторизації в середовищах заданої конфігурації"

(назва згідно з наказом ректора)

Спеціальність – 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДРБ.ІПз - 42.00.00.000 ПЗ

(позначення)

Студент

ІПз-18-1 **/В.М.Федорчак/**
(шифр групи) (підпис) (дата)(розшифровка підпису)

Керівник роботи

Консультанти:

доц., к.ф.-м.н. **/Р.І.Храбатин/**
(посада) (підпис) (дата) (розшифровка підпису)

з питань нормоконтролю

Допускається до захисту
Завідувач кафедри

доц., к.т.н **/В.Я. Піх/**
(посада) (підпис) (дата)(розшифровка підпису)

проф., д.т.н. **/В.І.Шекета/**
(посада) (підпис) (дата) (розшифровка підпису)

2022
(рік)

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітньо-кваліфікаційний рівень бакалавр

Напрям підготовки 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
проф. В.І. Шекета
“ ____ ” _____2022 р.

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ БАКАЛАВРА СТУДЕНТОВІ

Федорчаку Василю Миколайовичу

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) " Побудова архітектури ВЕБ-сервісів авторизації в середовищах заданої конфігурації "

керівник проекту (роботи) Храбатин Роман Ігорович, к.т.н., доцент

затверджені наказом вищого навчального закладу від “ 25 ” травня 2022 р. № 130/7

2. Строк подання студентом проекту (роботи) 15 червня 2022 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1 Огляд існуючих програмних рішень та сервісів в предметній області

2 Алгоритмічні та математичні рішення для задач даного класу

3 Власне оригінальне алгоритмічне та програмне рішення задачі

4 Тестування розробленого програмного продукту

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1 Доступ до вмісту Bing різними способами (рис. 1.1, ст. 9)

2 Структура підходу на основі вибірки(рис. 1.6, ст. 19)

3 Процедура Greedy Algorithm (рис. 2.3, ст. 26)

4 Сторінка відповіді з веб-служби Bing (рис. 3.2, ст. 36)

5 Інтерфейс користувача нашого сканера (рис. 3.3, ст. 37)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання **14 травня 2022 р.**

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Визначення та обгрунтування теми роботи	15.05.2022	виконано
2	Аналіз предметної області та огляд подібних програмних рішень	20.05.2022	виконано
3	Побудова алгоритму власного рішення	06.06.2022	виконано
4	Тестування програмного проукту	08.06.2022	виконано
5	Оформлення пояснювальної записки дипломної роботи бакалавра	12.06.2022	виконано
6	Затвердження пояснювальної записки дипломної роботи бакалавра завідувачем кафедри	15.06.2022	виконано

Студент _____
(підпис)

Керівник _____
(підпис)

АНОТАЦІЯ

В дипломній роботі досліджуються декілька методів вибору запитів у веб-сервісі Microsoft Bing, особливо вплив рейтингу результатів у реальних джерелах даних. Дані веб-сервісів архітектури глибокої мережі загалом зберігаються в базі даних або файловій системі, до якої можна отримати доступ лише через форми веб-запитів або через інтерфейси веб-служб. Однією з проблем глибокого сканування Інтернету є те, як вибрати значущі запити для отримання даних. Результати показують, що для неранжованих джерел даних зважений метод працював трохи краще, ніж алгоритм покриття незваженого набору. Для ранжованих джерел даних необхідна часта оцінка документів для ефективнішого збору даних.

КЛЮЧОВІ СЛОВА: ВЕБ-СЕРВІС, ГЛИБОКА МЕРЕЖА, ПРОБЛЕМА ПОКРИТТЯ МНОЖИН, ЖАДІБНИЙ АЛГОРИТМ, ВАГОВІ ЗНАЧЕННЯ, ЧАСТОТА ДОКУМЕНТУВАННЯ, РАНЖУВАННЯ.

ANNOTATION

The thesis examines several methods of query selection in the Microsoft Bing web service, especially the influence of the ranking of results in real data sources. Deep web architecture web services data is generally stored in a database or file system that can only be accessed through web request forms or through web service interfaces. One of the challenges of deep web crawling is how to select meaningful queries to retrieve data from. The results show that for unranked data sources, the weighted method performed slightly better than the unweighted set coverage algorithm. For ranked data sources, frequent evaluation of documents is necessary for more efficient data collection.

KEY WORDS: WEB SERVICE, DEEP NETWORK, CROWD COVERAGE PROBLEM, GREEDY ALGORITHM, WEIGHT VALUES, DOCUMENTATION FREQUENCY, RANKING.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 6
ВСТУП 7

1 ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ТА СЕРВІСІВ В ПРЕДМЕТНІЙ ОБЛАСТІ 9
1.1 Пошукові системи 9
1.2 Аналіз пов'язаних робіт та проектів 13
1.3 Підхід на основі вибірки 17

2 АЛГОРИТМІЧНІ ТА ФОРМАЛЬНІ МОДЕЛІ РІШЕННЯ ДЛЯ ЗАДАЧ ДАНОГО КЛАСУ 20
2.1 Встановлення проблеми покриття мережі 20
2.2. Використання жадібного алгоритму 24
2.3. Зважений алгоритм 28
2.4. Дослідження проблеми ранжування та рейтингу 32

3 ВЛАСНЕ ОРИГІНАЛЬНЕ АЛГОРИТМІЧНЕ ТА ПРОГРАМНЕ РІШЕННЯ ЗАДАЧІ АРХІТЕКТУРИ ВЕБ-СЕРВІСІВ АВТОРИЗАЦІЇ 36
3.1. Експериментальне середовище проведення імітаційного моделювання .. 36
3.2. Критерії оцінки 39
3.3. Експериментальні дослідження та тестування 41

ВИСНОВКИ 55
СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА 57
БІБЛІОГРАФІЧНА ДОВІДКА

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

СА – сервіс авторизації

БД – База даних

СУБД - Система управління базами даних

ПЗ - Програмне забезпечення

ВСТУП

Дані Deep Web відносяться до вмісту, який динамічно генерується з баз даних або файлових систем. Інформація, яка подається в Deep Web, доступна через інтерфейси запитів, такі як html-форми або веб-сервіси. Багато організацій, наприклад «Arxiv.org», «Bing.com» або «Amazon.com» надають інтерфейси веб-сервісів для доступу до їх глибоких веб-даних. Оскільки дані приховані за інтерфейсами запитів, глибоку мережу також називають прихованою мережею або невидимою мережею. Цей веб-сайт містить велику кількість академічних документів. У більшості випадків користувачі шукають потрібний документ за допомогою форми запиту html у верхній частині.

Глибока мережа часто містить велику кількість документів, які часто мають високу якість і цінність для користувачів. Оскільки статичних посилань на ці глибокі веб-документи немає, контент глибокого веб-пошуку недоступний традиційним пошуковим системам. Щоб отримати доступ до такого вмісту, користувачі повинні ввести одне або кілька ключових слів у HTML-форми та надіслати запит. Згідно з дослідженням, вміст глибокої мережі приблизно в 500 разів перевищує вміст, який бачать звичайні пошукові системи. Отже, як використовувати глибокий веб-контент стає серйозною проблемою для спільноти пошуку інформації.

Представлена дипломна робота бакалавра зосереджена на задачі завантаження даних глибокого Інтернету з реальних веб-сервісів. Ми розробили сканер веб-сервісу, реалізований і експериментуючи з чотирма алгоритмами вибору запитів для глибокого сканування веб-сайтів. Наша мета — оцінити їх ефективність для отримання даних з різних реальних джерел даних через веб-сервіс.

Щоб поділитися своїми даними з користувачами, деякі глибокі веб-сайти надають веб-сервіс для клієнтської програми для доступу до своїх онлайн-баз даних. Веб-сервіс — це технологія, яка забезпечує взаємодію програми з програмою через мережу — незалежно від платформи, мови чи форматів даних.

Додаючи веб-API (інтерфейси програмного забезпечення) у мережі, функціональні можливості веб-сервісу можна активувати за допомогою HTTP-запитів. За допомогою цих API клієнтська програма може отримати доступ до віддаленого вмісту.

1 ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ТА СЕРВІСІВ В ПРЕДМЕТНІЙ ОБЛАСТІ

1.1 Пошукові системи

Переваги використання веб-сервісів включають: немає необхідності заповнювати форму запиту html і не потрібно витягувати відповідні дані зі сторінки результатів html.

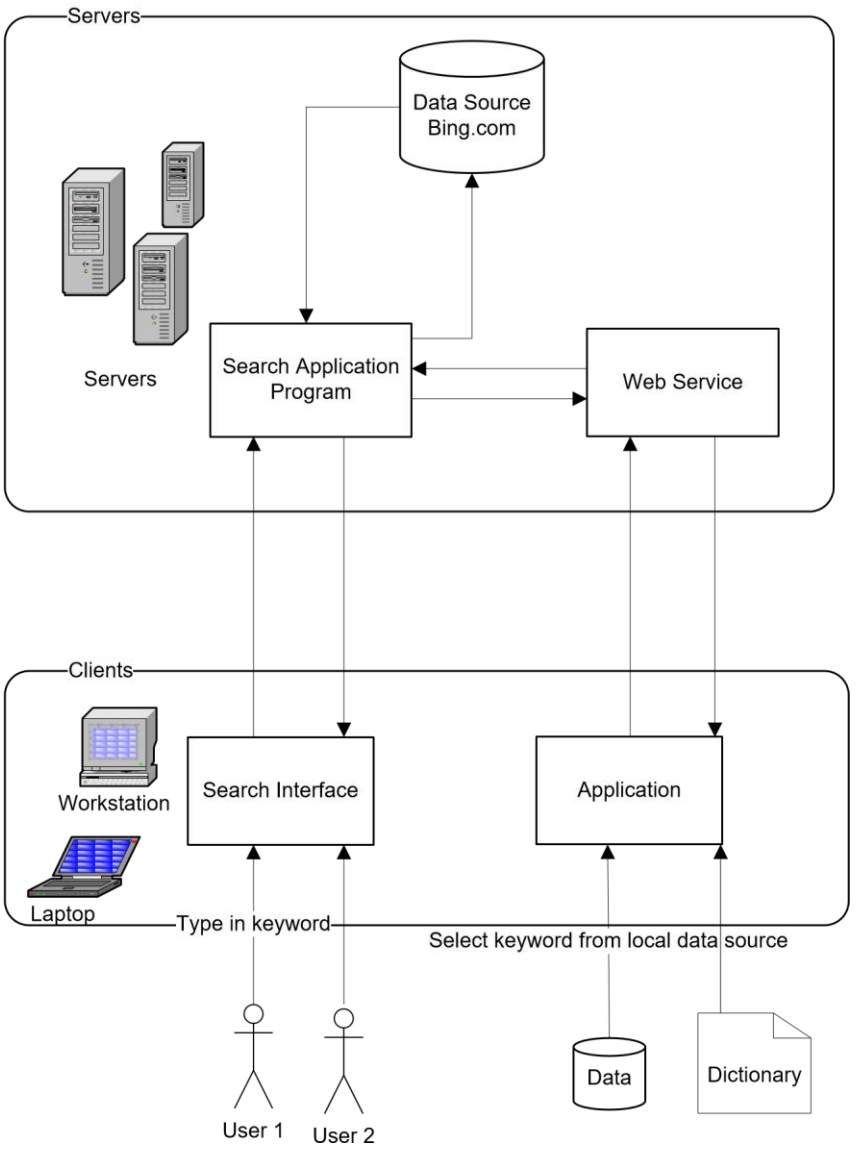


Рисунок 1.1 - Доступ до вмісту Bing двома способами: інтерфейсом пошуку або веб-сервісом

Вміст глибокого Інтернету загалом зберігається в базі даних. За типом бази даних їх можна класифікувати як неструктуровані (текстові) бази даних, так і як структуровані [4].

Неструктурована база даних — це сайт, який переважно містить документи у відкритому тексті (наприклад, юридичні документи). На відміну від цього, структурована база даних — це сайт, який часто містить реляційні дані, наприклад, інтернет-магазин книг, який може мати кілька полів, наприклад назву, автора, ISBN тощо. Для текстової бази даних інтерфейс пошуку зазвичай містить просте текстове поле з ключовими словами.

І навпаки, інтерфейс до структурованої бази даних може дозволити користувачам подавати кілька атрибутів (наприклад, пошук автомобілів за назвою компанії, брендом або роком виробництва). Інтерфейс може містити комбінацію текстового поля, перемикача, спадного меню тощо.

Текстова база даних в основному містить звичайні текстові документи, такі як документи, юридичні документи, новинні статті тощо. Форма запиту Html у текстовій базі даних зазвичай містить лише одне текстове поле для заповнення ключових слів, як показано на рисунку 1.2. Це пошук у форматі html. форму з arXiv.org. База даних arXiv є текстовою. Структурована база даних в основному містить реляційні дані, такі як база даних онлайнового магазину. Форма запиту Html до структури бази даних часто містить кілька текстових полів для заповнення ключових слів.

Ви можете шукати книгу за автором, назвою, ISBN тощо. Наше дослідження пов’язане з текстовою базою даних.

Оскільки мільйони баз даних підключені до Інтернету, ми не можемо ігнорувати дані, приховані за інтерфейсом пошуку. Для використання глибокого веб-контенту двома основними додатками є віртуальна інтеграція та вихід на поверхню.

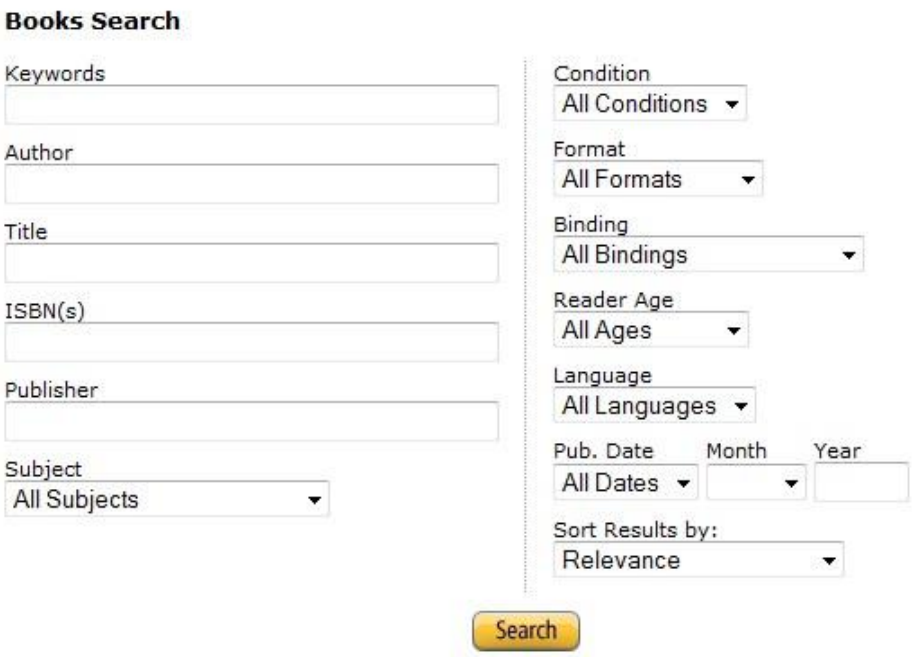


Рисунок 1.2 - Форма пошуку Html книжкового магазину AmazonHere is an example з онлайнового книжкового магазину Amazon.

Підхід віртуальної інтеграції полягає в тому, щоб забезпечити єдиний інтерфейс для доступу до певного типу даних з різних джерел глибокої мережі. Щоб створити таку програму, нам потрібно визначити домен (наприклад, книга, авіаквиток або нерухомість) кожної глибокої мережі та проаналізувати інтерфейс пошуку html. Таким чином, система автоматичної інтеграції часто містить систему автоматичної ідентифікації та семантичну систему. Система ідентифікації покликана аналізувати інтерфейс запитів або вміст глибокого веб-сайту та ідентифікувати домен, до якого він належить. Наприклад, ми знаємо велику кількість глибоких веб-сайтів. Зараз нас цікавлять лише сайти книжкових інтернет-магазинів. Отже, перший крок, нам потрібно знайти ті сайти, які пов’язані з нашою бажаною інформацією про книгу.

Після цього ми вже маємо набір глибоких веб-сайтів у домені, що цікавить. Потім нам потрібно створити уніфікований інтерфейс запитів для пошуку на цих сайтах одночасно. Щоб створити уніфікований інтерфейс запитів, необхідна семантична система для створення семантичних відображень і керування ними в

інтерфейсах пошуку цих глибоких веб-сайтів. Коротше кажучи, це відображення запитів на інтерфейси пошуку відмінностей. Тоді система інтеграції витягне, об’єднає та ранжує отримані результати з джерел різниць даних. Нарешті, надайте користувачам результати регенерації. Як правило, віртуальна інтеграція надає користувачеві більше досвіду, крім пошуку. Наприклад, ми шукаємо книгу в пошуковій системі віртуальної інтеграції. Результати отримані з різних книжкових інтернет-магазинів, таких як “amazon.com”, “ebay.com” та “indigo ”. Окрім результатів пошуку, пошукова система також пропонує найкращу ціну на книгу. З цієї причини віртуальна інтеграція більше підходить для структурних баз даних.

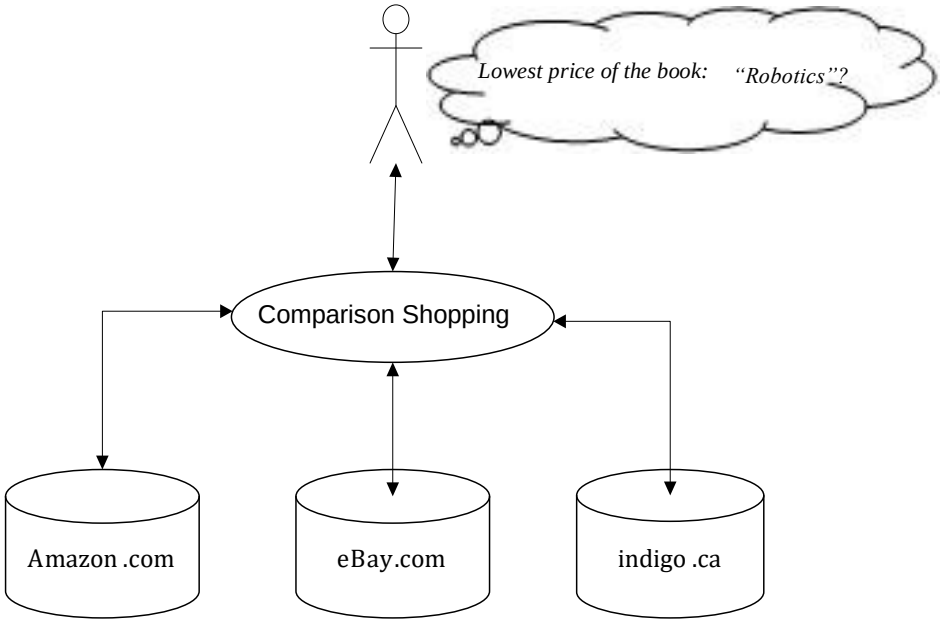


Рисунок 1.3 - Віртуальна інтеграція – Порівняння Покупок

Цей підхід також називається глибоким скануванням Інтернету, який завантажує прихований вміст шляхом надсилання набору запитів. Комерційні пошукові системи, скажімо Google, почали з’являтися в глибокій мережі. Отримані результати вже додано до індексу пошукової системи Google сьогодні. Загалом, проблеми глибокого сканування Інтернету включають: як обробити

форму запиту html ; як отримати відповідні дані зі сторінок результатів; і як вибрати набір запитів. Відмінний сканер повинен автоматично відкривати глибокі веб-сайти. Таким чином, сканеру потрібен підхід до обробки форми запиту html та автоматичного вилучення відповідних даних зі сторінок результатів. У наших експериментах ми використовуємо веб-сервіс, тому нам не потрібно обробляти форму запиту html, а результатом є формат файлу XML. З цієї причини ми можемо зосередитися на проблемі вибору запиту.

Реальне середовище зазвичай встановлює обмеження повернення для максимальної кількості результатів. Більшість попередніх методів призначені для завантаження глибокого веб-сайту без обмеження повернення. Тому вони не можуть добре працювати, коли існує обмеження повернення. Ми представляємо DF-зважений Greedy алгоритм, щоб впоратися з цією проблемою.

Експерименти проводяться на трьох джерелах даних. Розмір першого джерела даних невеликий, який містить близько 30 000 веб-сторінок. Це означає, що проблема обмеження повернення не є серйозною в цьому джерелі даних, оскільки кількість.

1.2 Аналіз пов'язаних робіт та проектів

Ключова проблема глибокого сканування в Інтернеті полягає в тому, як вибрати набір запитів для подання у форму запиту. Існує багато способів вибору ключових слів.

Примітивним рішенням може бути випадковий вибір деяких слів зі словника. Однак це рішення неефективне, оскільки велика кількість рідкісних запитів може не відповідати жодній сторінці, або може бути багато повернень, які перекриваються. Замість вибору ключових слів зі словника було розроблено кілька алгоритмів вибору ключових слів.

Наразі більшість розроблених підходів полягають у аналізі та виборі запитів із документів, завантажених із попередніх запитів, поданих до глибокої веб-бази

даних. Їх можна класифікувати як: графічний підхід, інкрементальний підхід та підхід на основі вибірки. Графовий підхід використовується для завантаження структурованої бази даних, тому він не обговорюється в моїй дипломній роботі.

Інкрементальний підхід вибирає запити із завантажених документів. Кількість документів збільшується в міру надсилання більшої кількості запитів, тому такий підхід називається інкрементальним.

Такий підхід вибирає запит, що повертає більшість нових документів на одиницю ітераційно. Оскільки немає попередніх знань про всі частоти запитів документів, цей метод вимагає оцінки частоти документів на основі вже завантажених документів. Виходячи з цієї оцінки та кількості зустрічей запитів у завантажених документах, можна оцінити кількість збіганих нових документів. Вони пропонують два способи оцінки. Перший метод, який називається оцінкою незалежності, передбачає, що ймовірність появи терміна в підмножині документів дорівнює ймовірності у всьому наборі документів. На основі частоти терміна в підмножині документів N(qi | колекція підмножин) метод може оцінити, скільки разів певний термін зустрічається у всьому наборі документів N(qi). Тоді ми можемо оцінити кількість нових документів за: Nnew(qi) = N(qi) - N(qi | підмножина/колекція).

Метод оцінки Zipf [3] полягає в оцінці частоти термінів у колекції документів, дотримуючись розподілу степеневого закону. Тобто частота терміну в колекції документів визначається за формулою:

$$N(qi)= \alpha (r +\beta)-Y, \quad (1)$$

де r – ранг терміна, а α, β та Y – константи, які залежать від колекції документів. На основі підмножини документів, які ми завантажили, ми можемо оцінити α , β і Y. Враховуючи ранжирування r терміна всередині підмножини колекції документів, N(qi) можна обчислити за формулою (1). Вони порівнюють три політики вибору ключових слів: випадковий (Ключові слова випадково

вибираються зі словника), загальну частоту (Ключові слова вибираються з 5,5-мільйонного корпусу веб-сторінок на основі їхнього зменшення частоти) та їх адаптивний алгоритм. Результати експерименту показують, що адаптивний алгоритм (ключові слова вибираються з підмножини документів) працює надзвичайно добре у всіх випадках. Підходи, запропоновані в [20], вибирають запити з інкрементальної колекції документів. Тому такий підхід ми називаємо інкрементальним. Інкрементальний підхід вибирає запити з інкрементальної колекції документів. Це означає, що вам потрібно проаналізувати кожен документ після його завантаження і знову обчислити частоту документа для кожного терміна. Цей крок буде дуже трудомістким, якщо ми підраховуємо частоту документа для кожного запиту в кожному раунді. Щоб ефективно обчислювати частоту документів, рішення Ntoulas обчислює частоту документів, оновлюючи таблицю статистики запитів після того, як ми надаємо новий запит і завантажуємо більше документів. Однак підтримувати цю таблицю досі важко.

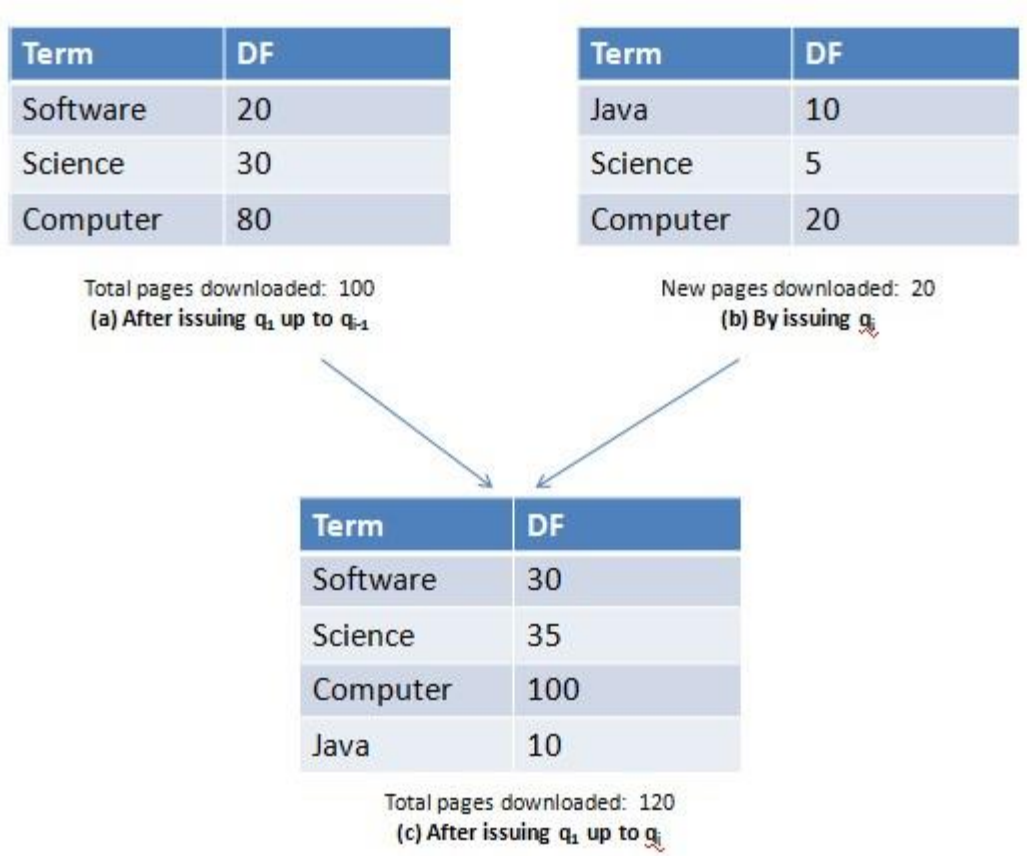


Рисунок 1.4 - Статистична таблиця: частота використання термінів документів

Підхід на основі вибірки спочатку створює зразок бази даних і створює набір запитів із зразкової бази даних, а не ітераційно вибирає ключові слова з додаткової підмножини колекції документів до завершення сканування.

Мадхаван та ін. розробили систему сканування глибокої мережі. Оскільки система є промисловим продуктом, їй потрібно розглянути, як вибрати початкові запити. Їх система виявляє особливості інтерфейсів запитів. Оскільки їм потрібно обробляти різні мови, їхній підхід не вибирає запити зі словника. Замість цього вони вибирають початкові запити з форми запиту html. Після цього ітераційне зондування та підхід до вибору ключових слів подібний до запропонованого.

Їх політика вибору запитів заснована на TF-IDF, який є популярним заходом для пошуку інформації. TF-IDF вимірює важливість слова за наведеною нижче формулою.

$$tfidf(w,p) = tf(w,p) \times idf(w) \qquad (2)$$

Ця формула складається з двох частин: tf(w, p) — термін частоту використання терміна w на сторінці p і вимірює важливість слова w на сторінці p $tf(w,p) = n_{w,p}/N_p$,

де nw,p представляє кількість разів, коли слово w зустрічається на веб-сторінці p; – загальна кількість термінів на сторінці p.

idf (w) (зворотна частота документа) вимірює важливість слова серед усіх веб-сторінок і обчислюється за допомогою де D — загальна кількість веб-сторінок, а dw — кількість веб-сторінок, на яких з’являється термін w.

Метод Madhavan et al додає 25 найпопулярніших слів на кожній веб-сторінці, відсортованих за значеннями TF-IDF, до пулу запитів. З пулу запитів вони видаляють наступні два види термінів.

Виключіть високочастотні терміни, такі як терміни, які з’являлися на багатьох веб-сторінках (наприклад, понад 80%), оскільки ці терміни можуть бути з меню чи реклами.

Видаліть терміни, які зустрічаються лише на одній сторінці, оскільки багато з цих термінів є безглуздими словами, які не входять до вмісту веб-сторінок, наприклад, безглузді або ідіосинкратичні слова, які не можуть бути проіндексовані пошуковою системою.

Решта слів надходять у глибоку мережу під час завантаження запитів і нового набору веб-сторінок. Потім це повторюється знову в новій ітерації. Крім того, їхній підхід акцентує увагу на повзанні, орієнтованому на ширину, що значно відрізняється від попередніх досліджень. Вони спостерігали за статистичними даними на Google.com і виявили, що результати, які повертаються користувачам, більше залежать від кількості глибоких веб-сайтів. Вони проаналізували 10 мільйонів глибоких веб-сайтів. Вони виявили, що на 10 000 найкращих глибоких веб-сайтів припадає 50% результатів Deep-Web, в той час як навіть на 100 000 найбільших глибоких веб-сайтів припадає лише 85%. Це спостереження змушує їх зосередитися на скануванні якомога більшої кількості глибоких веб-сайтів, а не на певних глибоких веб-сайтах.

1.3 Підхід на основі вибірки

У роботі Barbosa et al. запропонувати підхід до перекачування глибокої мережі, вибравши ключове слово з найвищою частотою з колекції зразків документів. Цей алгоритм вибирає ключове слово з найвищою частотою зі списку потенційних ключових слів і, як очікується, забезпечить високе охоплення. Він складається з двох етапів: етап 1 вибирає набір слів із форми пошуку html і випадковим чином видає їх, поки не буде повернута не порожня сторінка результатів. Витягуючи високочастотні слова зі сторінки результатів, їхній алгоритм створює початковий список ключових слів. Потім він ітераційно оновлює частоту слів у списку та додає нові високочастотні слова до списку шляхом випадкового випуску слова у списку, поки кількість подання не досягне порога. На етапі 2 підхід вибирає ключове слово з найбільшою частотою зі

списку ключових слів для створення нового запиту в кожному раунді, доки кількість поданих запитів не досягне максимальних разів.

У роботі подано подальше покращення методу на основі вибірки. Ключові слова вибираються з фіксованого зразка бази даних за набору алгоритму покриття. Очікується, що ті запити, які можуть охопити більшість документів у зразковій базі даних, охоплять більшість даних у всій базі даних. Основу цього підходу показано на рисунку нижче. Для підходу на основі вибірки запити вибираються із вибіркового набору документів із загальної бази даних. Цей підхід складається з трьох етапів:

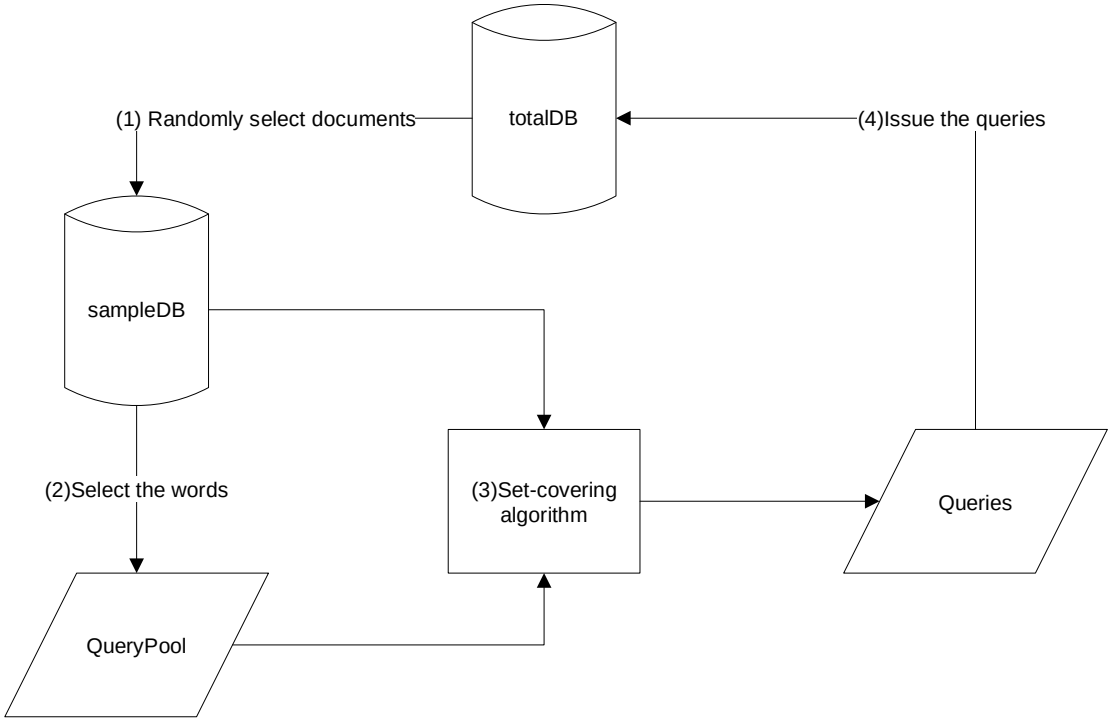


Рисунок 1.5 - Структура підходу на основі вибірки

Створіть зразок БД: видайте початкові ключові слова до загальної БД, отримайте відповідні документи, а потім побудуйте зразок бази даних; Побудуйте пул запитів: проаналізуйте всі документи у зразковій БД, застосуйте алгоритм покриття наборів для вибору ключових слів і створіть пул запитів;

Надішліть запити до Total DB для отримання документів.

Перевага цього методу полягає в тому, що потрібно завантажувати лише невелику частину документів, тому що сканеру іноді може знадобитися лише URL-адреса, а не всі документи. Ми зосереджуємось на підході на основі вибірки. Отже, більш детально про підхід на основі вибірки буде описано.

2 АЛГОРИТМІЧНІ ТА ФОРМАЛЬНІ МОДЕЛІ РІШЕННЯ ДЛЯ ЗАДАЧ ДАНОГО КЛАСУ

2.1 Встановлення проблеми покриття мережі

Однією з ключових проблем сканування глибокої мережі є вибір набору значущих ключових слів. Вибір запитів із колекції документів є популярним методом. Ntoulas та інші вперше використовують проблему покриття множин для представлення проблеми вибору запиту.

Проблема покриття множин є типовою проблемою NP . Його можна описати наступним чином: за умови скінченної множини U і сімейства X підмножин U, рішення полягає в тому, щоб знайти покриття C, об'єднання якого дорівнює U, і воно є підсімейством X. Проблему покриття множин можна розділити на дві проблеми. Одним з них є множина, що покриває проблему рішення, тобто задана пара (U,X) і ціле число k; питання полягає в тому, щоб вирішити, чи є кришка розміру k чи менше. Задача рішення про покриття множин є NP-повною. Іншим є множина, що покриває задачу оптимізації, тобто для пари (U,X) метою є знайти підмножини мінімального розміру.

O, елементи якого охоплюють всю U. Задача оптимізації, що покриває множини, є NP-складною .

Більш формально, задані U та X наступним чином:

$$\mathrm{U} = \bigcup_{\mathrm{S}\in\mathrm{X}} \mathrm{S}$$

Задача оптимізації множини полягає в тому, щоб знайти сімейство множин O таких, що

$$U = \cup_{S\in O} S \ \ where\ O \subseteq X$$

Вхідні дані проблеми покриття множини часто представлені матрицею запиту-документа, як показано на рисунку . На рисунку (a) матриця представляє зв'язок між трьома запитами (q1, q2, q3) і чотирма документами (d1, d2, d3, d4). Якщо клітинка (i,j) дорівнює 1, запит в і-му рядку (qi) міститься в документі в j-му стовпці (dj). Це матричне представлення можна проілюструвати на рисунку (b). Прямокутник на рисунку (b) представляє весь набір документів. Кожен документ представлений у вигляді чорної точки всередині прямокутника. Кожен овал на рисунку (b) представляє набір документів, охоплених запитом.

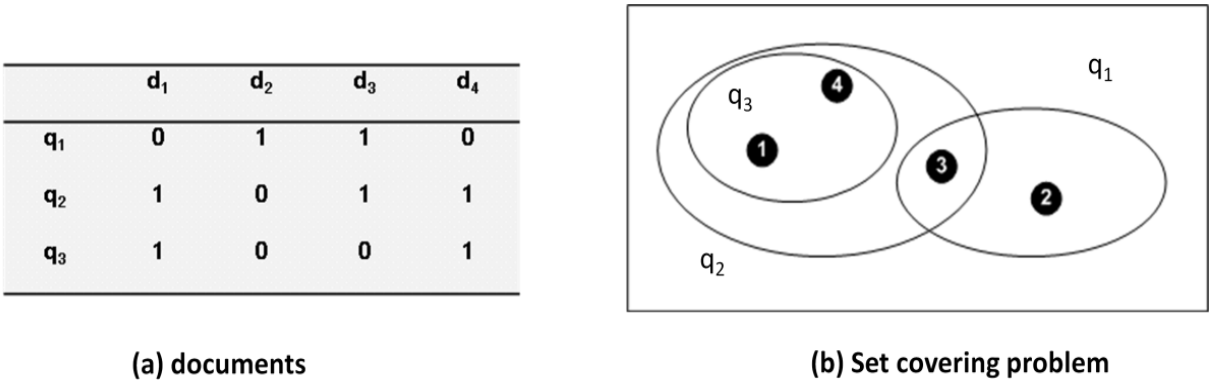


Рисунок 2.1 - Формалізація проблеми вибору запиту

Проблема покриття мінімальної множини може бути сформульована як цілочисельна лінійна програма. Відповідно до формулювання цілочисельної лінійної програми, ми формалізуємо проблему покриття множини для вибору запиту так: Нехай A є матрицею m*n з 0 і 1, що представляє колекцію документів, як на рисунку. Проблема покриття множини полягає в тому, щоб знайти рішення m-вектора S, Si = 0 або 1 (i = 1,…,m), який показує, чи вибраний запит і, чи ні. Ci — це m-вектор цілого додатного числа, який представляє вартість кожного запиту, і Ci = 1,…,m. E — це n-вектор з одиниць, який представляє кожен документ у матриці A. Більш формально це можна сформулювати так:

$$minimize(C^T S) \qquad (3)$$

$$Subject\ to\ :\ A^T \times S \geq E\ ,\qquad (4)$$

де Si = {0,1| i = 1,…,m}; Ei = {1|i=1,…,n}; Ci = $\{\sum_{j=1}^{n} A_{ij} \mid i = 1,\ldots,m\}$.

Наприклад, є два рішення: {q1, q2} і {q1, q3} для проблеми в

З малюнка ми можемо дізнатися:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

У матриці А є три підмножини q1, q2 і q3. За визначенням С:

$$C = \{2, 3, 2\}$$

Перший крок, ми перевіряємо, що обидва рішення задовольняються умовою.

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}^T \times S \geq E, where\ E = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Для рішення {q1, q2}: $S = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$

$$\text{Thus, } LHS = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

$$\text{LHS} = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 1 \end{bmatrix} > \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = E$$

Для рішення {q1, q3}: $S = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

$$\text{Thus, } LHS = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\text{LHS} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = E$$

Отже, обидва рішення за умови: $A^T \times S \geq E$

На наступному кроці ми розраховуємо вартість для обох рішень.

Для рішення {q1, q2}:

$$\text{Total cost} = \qquad C^T S$$

$$= \begin{bmatrix} 2 \\ 3 \\ 2 \end{bmatrix}^T \times \qquad \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

$$= 5$$

Для рішення {q1, q3}:

$$\text{Total cost} = C^T S$$

$$= \begin{bmatrix} 2 \\ 3 \\ 2 \end{bmatrix}^T \times \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$= 4$$

За формулою цільової функції (3) ми знаємо, що рішення 2 краще, ніж рішення 1.

Було доведено, що задача покриття множин є NP-повною. Оптимальне рішення важко отримати за поліноміальний час. Розробляються різні алгоритми оптимізації, такі як Greedy, Weighted Greedy, Genetic, Clustering тощо. Традиційні Greedy і Зважені алгоритми Greedy будуть реалізовані в розділі імітаційного експерименту.

2.2. Використання жадібного алгоритму

Одним із популярних алгоритмів для вирішення проблеми покриття множини є жадібний алгоритм, який обирає запити за одним правилом: на кожному раунді завжди вибирає запит, який охоплює найбільшу кількість нових документів за одиницю вартості. Жадібний алгоритм робить локально «найкращий» вибір на кожному етапі, але це не найкращий вибір у всьому світі. Припускаючи, що ми побудували пул запитів із набором запитів, жадібний алгоритм полягає у пошуку найбільш ефективного запиту з пулу запитів.

Жадібний алгоритм:

Вхідні дані: m * n Матриця A;

Вихід: Розв'язок m-вектора S;

$$dfi = \sum_{j=1}^{n} A_{ij};$$

B = A;

E є n-вектором і ініціалізує кожен елемент рівним 1; S і c є m-вектором і ініціалізують кожен елемент 0;

поки $A^T \times S < E$ {

for(i=1; i<= m; i++){

$$newi = \sum_{j=1}^{n} B_{ij};$$

ci = dfi / newi;

}

Find a k which minimizes ck ;

Sk =1;

Remove the kth row and jth column from B, if cellij =1;

} for (j = 1; j <=m; j++){ if(Sj = 1 and Aj is redundant) Sj = 0;

} return S; For example, the matrix below represents a sample database which contains nine documents (d1, d2,

d3, …, d9). Suppose our query pool includes six queries (q1, q2, q3, q4, q5, and q6).

Таблиця 2.1 - Приклад жадібного алгоритму (1)

	d1	d2	d3	d4	d5	d6	d7	d8	d9
q1	0	0	0	0	0	0	1	0	1
q2	0	0	0	0	1	0	1	0	1
q3	1	0	0	0	0	1	0	0	0
q4	0	0	0	0	0	1	1	1	0
q5	1	1	0	1	1	0	0	0	1
q6	1	1	1	1	0	0	0	0	0

За правилом, згаданим вище, ми можемо перетворити матрицю в задачу покриття множини, як показано на рисунку нижче. Ми вибираємо набори (запити) за жадібним алгоритмом, щоб охопити всі документи. Вся процедура перелічена таким чином:

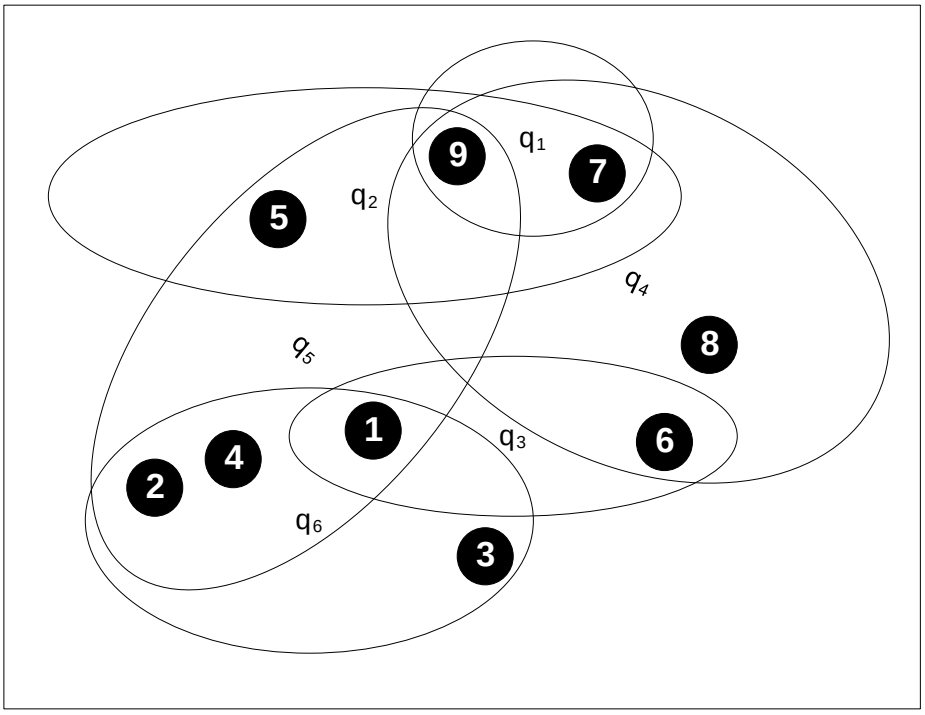


Рисунок 2.2 - Формалізація набору (приклад)

Раунд 1: додайте q5 до пулу запитів, значення new/df для кожного запиту дорівнює 1.

В результаті ми випадковим чином вибираємо запит з найбільшим df.

Таблиця 2.2 - Приклад жадібного алгоритму (2)

	d1	d2	d3	d4	d5	d6	d7	d8	d9	df	new	new/df
q1	0	0	0	0	0	0	1	0	1	2	2	1
q2	0	0	0	0	1	0	1	0	1	3	3	1
q3	1	0	0	0	0	1	0	0	0	2	2	1
q4	0	0	0	0	0	1	1	1	1	4	4	1
q5	1	1	0	1	1	0	0	0	1	5	5	1
q6	1	1	1	1	0	0	0	0	0	4	4	1

Раунд 2: Додайте q4 до пулу запитів, оскільки new/df(q4) = ¾ – максимальне значення в останньому стовпці.

Таблиця 2.3 - Приклад жадібного алгоритму (3)

	d1	d2	d3	d4	d5	d6	d7	d8	d9	df	new	new/df
q1 q2 q3	0 0 1	0 0 0	0 0 0	0 0 0	0 1 0	0 0 1	1 1 0	0 0 0	1 1 0	2 3 2	1 1 1	1/2 1/3 1/2
q4	0	0	0	0	0	1	1	1	1	4	3	3/4
q5	1	1	0	1	1	0	0	0	1	5	--	--
q6	1	1	1	1	0	0	0	0	0	4	1	1/4

Раунд 3: додайте q6 до пулу запитів, оскільки new/df(q6) = ¼ – максимальне значення в останньому стовпці.

Таблиця 2.4 - Приклад жадібного алгоритму (4)

	d1	d2	d3	d4	d5	d6	d7	d8	d9	df	new	new/df
q1 q2 q3	0 0 1	0 0 0	0 0 0	0 0 0	0 1 0	0 0 1	1 1 0	0 0 0	1 1 0	2 3 2	0 0 0	0 0 0
q4	0	0	0	0	0	1	1	1	1	4	--	--
q5	1	1	0	1	1	0	0	0	1	5	--	--
q6	1	1	1	1	0	0	0	0	0	4	1	1/4

Отже, Розв’язком цього прикладу є {q5, q4, q6}.

Всю процедуру вибору наборів за жадібним алгоритмом можна перенести на перегляд набору, як на рисунку нижче.

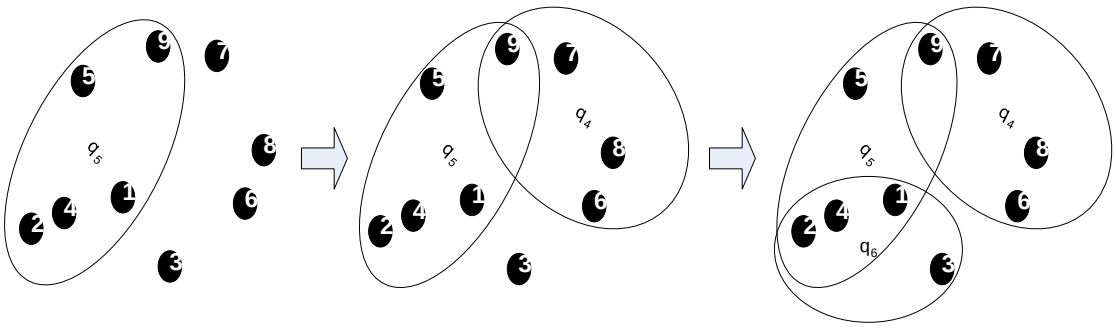


Рисунок 2.3 - Процедура Greedy Algorithm

Перевірте, чи охоплює рішення {q5, q4, q6} усі документи. Таким чином:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \times S \geq E,\ where\ E = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},\ S = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\text{Thus, } LHS = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 \\ 2 \\ 1 \\ 2 \\ 1 \\ 1 \\ 1 \\ 1 \\ 2 \end{bmatrix} > \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = E$$

2.3. Зважений алгоритм

Традиційні алгоритми покриття набору погано працюють при застосуванні до глибокого сканування Інтернету через різні особливі особливості домену програми. Як правило, більшість алгоритмів покриття ігнорують розподіл частот документів. У [4] автори розробили новий набір, що охоплює алгоритм, який націлений на сканування глибокої мережі. Замість простого алгоритму охоплення жадібних наборів, він вводить ваги в жадібну стратегію. Вони використовують частоту документа df (кількість документів, які містять конкретний попередній запит), вагу документа dw (зворотну до кількості термінів у QP, яка зустрічається в документі) і вагу запиту qw (суму ваг документа всіх документів, що містять термін q). Зважений жадібний алгоритм заснований на виборі запиту з найменшим df/qw.

Щоб покращити простий жадібний алгоритм і зменшити перекриття, зважений жадібний алгоритм вводить ваги в жадібну стратегію і пропонує зважений жадібний алгоритм замість простого алгоритму, що покриває жадібний набір. Визначення вводяться таким чином:

Визначення 3 (Вага документа): Нехай D={d1,…,dm} буде SampleDB, а QP={q1,…,qn} — QueryPool. Ми розглядаємо кожен документ як набір термінів і використовуємо позначення qj di, щоб вказати, що термін qj зустрічається в документі di. Вага документа щодо QP і di (1≤i≤m), що позначається dw(di, QP) (або dw для shot), є оберненою до кількості термінів у QP, що зустрічається в документі di. , тобто $\mathbf{dw} = \frac{1}{|di \cap QP|}$

Визначення 4 (Вага запиту): вага запиту qj (1≤j≤n) у QP щодо D, що позначається qw(qj, QP) (або скорочено qw), є сумою ваг документа всі документи, що містять термін qj, тобто $\mathbf{qw} = \sum_{qj \epsilon di, di \epsilon D} \mathbf{dw}$

Зважений жадібний алгоритм:

```
Вхідні дані: m * n Матриця A;
  Вихід: Розв'язок m-вектора S;
```

dfi $= \sum_{j=1}^{n} A_{ij}$;

B = A;

E є n-вектором і ініціалізує кожен елемент рівним 1;

S і c є m-вектором і ініціалізують кожен елемент 0;

поки {

for(i=1; i<= m; i++){

;

ci = dfi / qwi; }

Знайти k, що мінімізує ck;

Sk =1;

Видаліть k-й рядок і j-й стовпець з B, якщо cellij =1;

} for (j = 1; j <=m; j++){ if(Sj = 1 і Aj є зайвим) Sj = 0;

} повернути S;

Зважений жадібний алгоритм завжди вибирає наступний запит із найбільшим “qw/df”. На основі цього правила зважений жадібний алгоритм вибирає ключові слова з SampleDB як запити з нижчою частотою перекриття. Цей алгоритм отримує набагато кращий результат, ніж простий жадібний метод у SampleDB, тому слід очікувати, що він отримає кращий результат у TotalDB.

За зваженим алгоритмом жадібності ми можемо отримати рішення №2(q6, q4, q2), наприклад 2.

1 раунд:

Таблиця 2.4 - Приклад алгоритму Wighted Greedy (1)

	d1	d2	d3	d4	d5	d6	d7	d8	d9	df	qw	qw/df
q1	0	0	0	0	0	0	0.333	0	0.25	2	0.583	0.292
q2	0	0	0	0	0.5	0	0.333	0	0.25	3	1.083	0.361
q3	1	0	0	0	0	0.5	0	0	0	2	1.5	0.75
q4	0	0	0	0	0	0.5	0.333	1	0.25	4	2.083	0.521
q5	0.5	0.5	0	0.5	0.5	0	0	0	0.25	5	2.25	0.45
q6	0.5	0.5	1	0.5	0	0	0	0	0	4	2.5	0.625

2 раунд:

Таблиця 2.5 - Приклад алгоритму Wighted Greedy (2)

	d1	d2	d3	d4	d5	d6	d7	d8	d9	df	qw	qw/df
q1	0	0	0	0	0	0	0.333	0	0.25	2	0.583	0.292
q2	0	0	0	0	0.5	0	0.333	0	0.25	3	1.083	0.361
q3	1	0	0	0	0	0.5	0	0	0	2	0.5	0.25
q4	0	0	0	0	0	0.5	0.333	1	0.25	4	2.083	0.521
q5	0.5	0.5	0	0.5	0.5	0	0	0	0.25	5	0.75	0.15
q6	0.5	0.5	1	0.5	0	0	0	0	0	4	-	-

3 раунд:

Таблиця 2.6 - Приклад алгоритму Wighted Greedy (3)

	d1	d2	d3	d4	d5		d6	d7	d8	d9	df	qw	qw/df
q1	0	0	0	0		0	0	0.333	0	0.25	2	0	0
q2	0	0	0	0	0.5		0	0.333	0	0.25	3	0.5	0.1667
q3	1	0	0	0	0		0.5	0	0	0	2	0	0
q4	0	0	0	0	0		0.5	0.333	1	0.25	4	-	-
q5	0.5	0.5	0	0.5	0.5		0	0	0	0.25	5	0.5	0.1
q6	0.5	0.5	1	0.5	0		0	0	0	0	4	-	-

Всю процедуру вибору множин за зваженим жадібним алгоритмом ми переносимо у вид покриття множини, як на рисунку нижче.

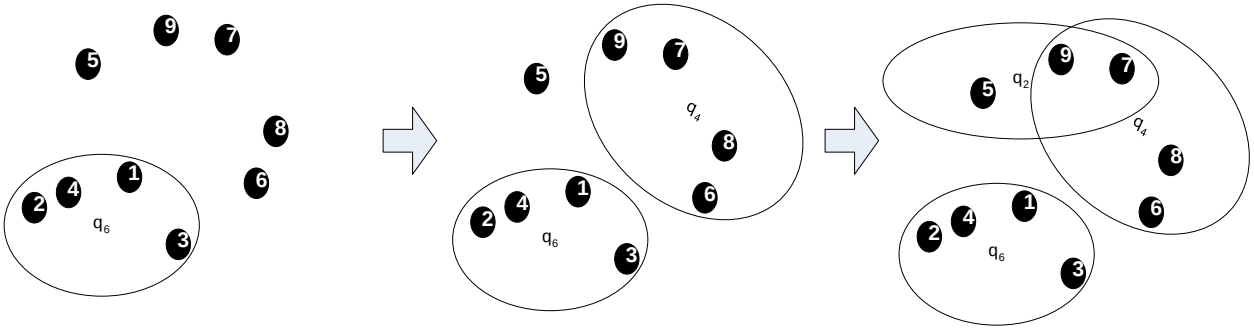


Рисунок 2.4 - Процедура у вигляді набору покриття (за зваженим алгоритмом жадібності).

Перевірте, чи охоплює рішення {q6, q4, q2} усі документи, за допомогою

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \times S \geq E, \qquad where\ E = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \qquad S = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\text{Thus, } LHS = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 2 \\ 1 \\ 2 \end{bmatrix} > \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = E$$

Отже, всі документи охоплюються рішенням {q6, q4, q2}.

Далі обчисліть вартість рішення {q6, q4, q2}: $= \begin{bmatrix} 2 \\ 3 \\ 2 \\ 4 \\ 5 \\ 4 \end{bmatrix}^T \times \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = 11$

Порівнюючи з вартістю в прикладі, ми бачимо, що рішення, задане зваженим жадібним алгоритмом, краще, ніж рішення, створене звичайним жадібним алгоритмом.

2.4. Дослідження проблеми ранжування та рейтингу

Обмеження повернення та політика ранжирування призводить до проблеми ранжирування. Багато прихованих веб-сайтів встановлюють обмеження k для кількості результатів. Коли запит відповідає великій кількості документів, глибокі веб-сайти повертають не більше k документів. Це називається обмеженням повернення. Політика ранжування – це правило сортування результатів.

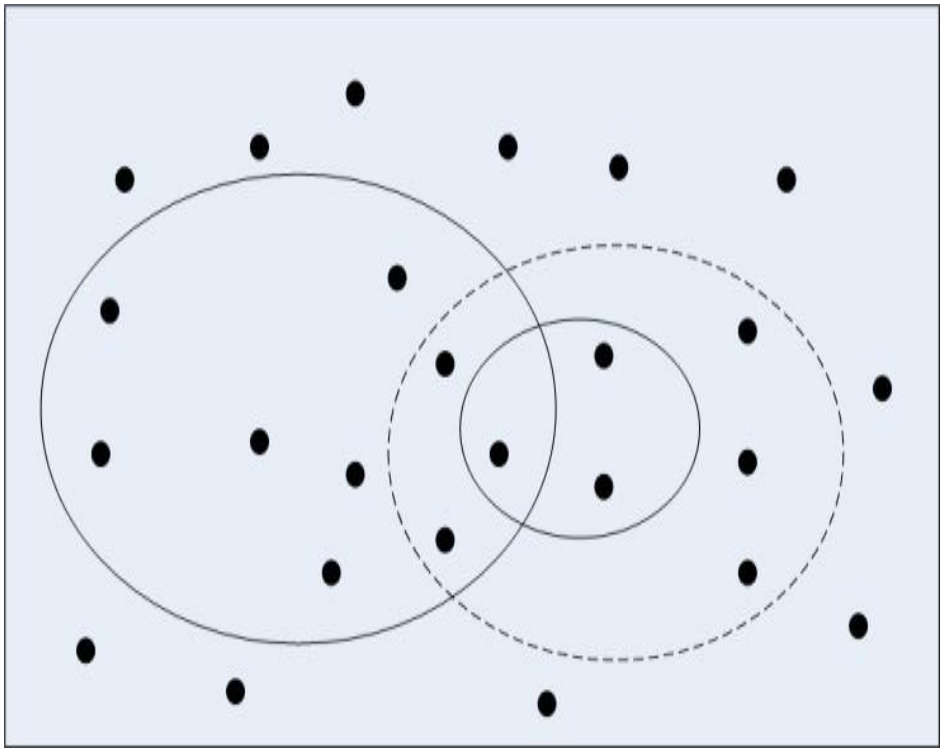


Рисунок 2.5 - Глибокий веб-сайт зазвичай встановлює обмеження на повернення.

Політика рейтингу зазвичай може бути як статичною, так і динамічною. Статичний рейтинг, скажімо веб-сервіс «twitter.com», сортує результати за датою та часом. Динамічне рейтингування може сортувати результати за релевантністю запиту. Ті документи, які дуже пов’язані з пошуковим запитом, будуть перераховані вгорі. Чим релевантніше пошуковому запиту, тим позиція документа ближче до верху. Однак для комерційних пошукових систем політика

рейтингування набагато складніша. Як правило, комерційні пошукові системи ранжують результати в основному за порядком релевантності та важливості. Релевантність веб-сторінок буде оцінюватися за багатьма факторами: такими як кількість разів, позиція появи, чи містить заголовок термін тощо. Важливість веб-сторінок буде вимірятися іншими критеріями, наприклад, кількістю посилань на веб-сторінку з інших веб-сайтів та репутації цих веб-сайтів. Після того, як пошукова система відсортує список документів з їх оцінками, вона вибере найпопулярніші k документів як результати для запиту.

Обмеження повернення дає нам велику проблему для завантаження даних із глибоких веб-сайтів. Наприклад, жадібний алгоритм і зважений жадібний алгоритм, ймовірно, вибирають терміни з високою частотою документа як запити.

Передбачається, що ці запити відповідають великій кількості документів. Однак через обмеження повернення можна завантажити лише максимальну кількість документів k.

Коли глибокий веб-сайт встановлює обмеження повернення, правило рейтингу також є важливою проблемою для глибокого сканування. За Lu et al. У попередньому дослідженні, якщо пошукова система застосовує правило статичного рейтингу, є такий результат:

$$M \leq \frac{k}{df_q} \times N, \qquad (5)$$

де M – кількість документів, які можна завантажити; k - обмеження повернення; dfq – нижня межа частоти документів для всіх надісланих запитів; N – розмір джерела даних.

Ця формула показує, що якщо ми виберемо високочастотні терміни як запити, можна завантажити менше документів. Наприклад, припустимо, що ми продовжуємо надсилати запити, чия частота документів перевищує 200, до пошукової системи глибокого Інтернету, чия k дорівнює 100. Відповідно до

рівняння 5, якщо пошукова система перераховує результат із політикою статичного рейтингування, загальна кількість документів, які можуть повинні бути завантажені:

$$M \leq N \times \frac{100}{200};$$

$$= \frac{1}{2} N;$$

Незалежно від того, скільки запитів надіслано. Незважаючи на те, що політика динамічного рейтингування може полегшити таку проблему рейтингу, ці популярні терміни все ще не є хорошим вибором.

Тому наша ідея полягає в тому, щоб вибрати запити, частота документів яких менше k.

Запропонований нами метод покращує зважений алгоритм, завжди вибираючи запити, частота документів яких менше k. Однак ми не знаємо частоти використання терміна в документах, поки не подамо термін як запит. Отже, ми повинні оцінити частоту документації терміна.

Одним із простих методів є оцінка частоти документа терміну в загальній базі даних за допомогою бази даних зразків. Припускаючи, що ймовірність терміну в вибірковій базі даних і в загальній базі даних однакова, частоту документа термін можна оцінити за допомогою:

$$\widehat{df}_{TotalDB} = df_{SampleDB} \times |TotalDB|/|SampleDB| \qquad (6)$$

Метод, яким ми застосовуємо метод оцінки частоти документів до зваженого жадібного алгоритму для вибору запитів із зразкової бази даних, називається алгоритмом DFWeighted.

DF-Weighted в першу чергу оцінює частоту документації загальної бази даних для всіх термінів у вибірковій базі даних. Терміни, частота документів

яких менше k, вибираються для створення матриці з усіма документами, які вони охоплюють. Після цього матриця обробляється як вхід зваженого жадібного алгоритму. Потім зважений жадібний алгоритм виводить набір запитів.

3 ВЛАСНЕ ОРИГІНАЛЬНЕ АЛГОРИТМІЧНЕ ТА ПРОГРАМНЕ РІШЕННЯ ЗАДАЧІ АРХІТЕКТУРИ ВЕБ-СЕРВІСІВ АВТОРИЗАЦІЇ

3.1. Експериментальне середовище проведення імітаційного моделювання

Завдання наших експериментів — оцінити різні політики завантаження, описані на реальних глибоких веб-сайтах. Ми вибираємо веб-сервіс Bing як наш тестовий майданчик і використовуємо фрагменти даних, індексованих в Bing, як різні джерела даних глибокого Інтернету. Цими фрагментами можуть бути веб-сайти, такі як .edu, доступ до яких можна отримати за допомогою синтаксису пошуку Bing. Такий інтерфейс пошуку та результати від Bing дуже схожі на вікно пошуку, надане самим веб-сайтом. Таким чином, ми можемо змоделювати доступ майже до всіх веб-сайтів як джерел глибоких веб-даних із можливістю пошуку.

Наприклад, якщо ми плануємо тестувати на « », ми можемо багаторазово надсилати запити до пошукової системи Bing, наприклад «site: .edu [запит]», як показано на рисунку.

У цій дипломній роботі замість використання інтерфейсу пошуку HTML-форми ми надсилаємо запити до веб-сервісу Bing. Тому нам не потрібно заповнювати форму запиту html і витягувати дані зі сторінок результатів html.

API Bing надає HTTP Get для реалізації процесу подання запитів. Запит до кінцевої точки HTTP складається із запиту HTTP GET до відповідного URI. Є два URI, один для результатів XML і один для результатів JSON. Формат XML є

використані в нашому експерименті. Тому ми надсилаємо наші запити до URI: «http://api.search.live.net/xml.aspx”. Якщо ми хочемо запитати сайт « *.edu» сторінки, що відповідають терміну «великі», повний запит, надісланий до веб-служби Bing:

http://api.search.live.net/xml.aspx?Appid=<AppID>&query=site: .edu%20large&sources=web.

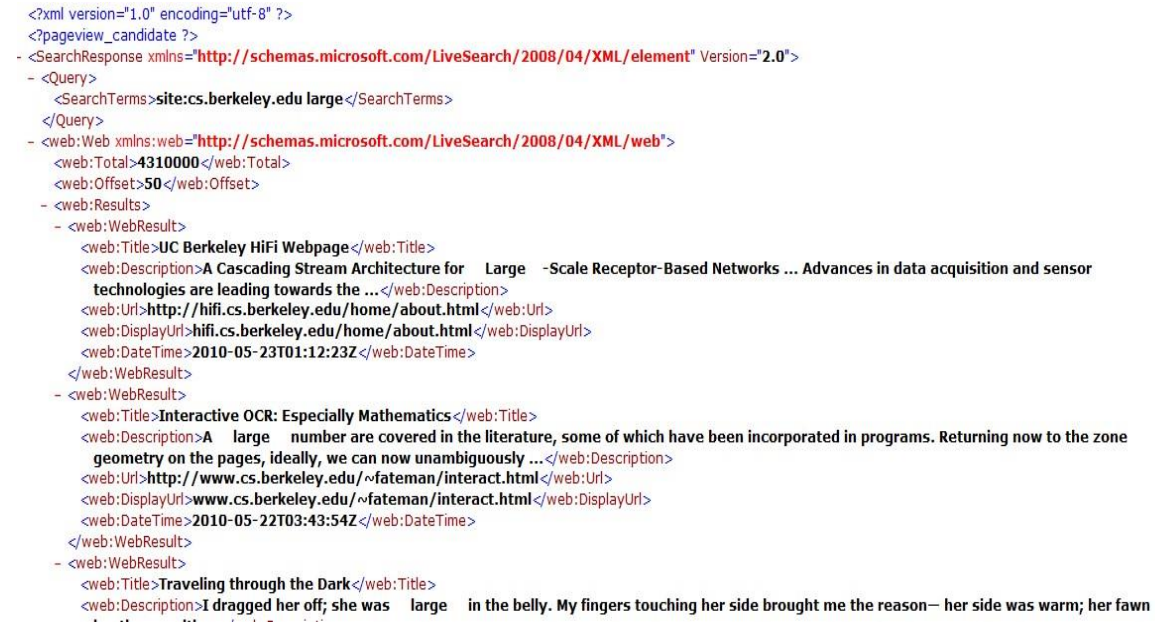


Рисунок 3.1 - Сторінка відповіді з веб-служби Bing

Зображення вище є частиною сторінки відповідей. Нижче описано кілька повернених елементів:

1) Елемент Total, “<web:Total>”, містить приблизну кількість результатів для конкретного запиту. Оскільки веб-сервіс Bing зазвичай надає дуже неточне число, ми використовуємо нашу власну оцінку розміру, вичерпно надсилаючи дуже велику кількість запитів.

2) Елемент Offset, “<web:offset>”, вказує поточну позицію набору результатів, який ви обробляєте. Ви можете змінити зсув за допомогою додаткового параметра Offset. Кожна сторінка відповідей містить не більше 50 результатів. Це означає, що якщо запит відповідає 100 результатам, вам потрібно двічі надіслати цей запит до веб-служб Bing. Наприклад, якщо ви хочете запитати 50 результатів одночасно, ви повинні передати «web.count=50» як частину рядка запиту. Якби ви хотіли отримати наступні 50 результатів після отримання перших результатів, ви б пройшли

“web.offset=51”. Повний URI буде таким: http://api.search.live.net/xml.aspx?

Appid=<AppID>&query=site: .edu%20large&sources=web

&web.count=50&web.offset=51

Веб-сервіс Bing створює деякі проблеми для глибокого сканування Інтернету, наприклад обмеження повернення, рейтинг результатів, розбиті на сторінки та перекриття між сторінками.

Ліміт повернення, лише одна тисяча перших результатів може бути повернута на запит.

Перекриття між сторінками: іноді веб-сервіс Bing може повертати ті самі документи, коли ви надсилаєте запит. У нашому експериментальному результаті ми обрізали цей тип дубліката.

Критерії рейтингу: порівнюючи локальне джерело даних моделювання, ми не знаємо правила пошукової системи Bing для рейтингування результатів. Ця проблема також ставить перед нами новий виклик для завантаження даних глибокого Інтернету.

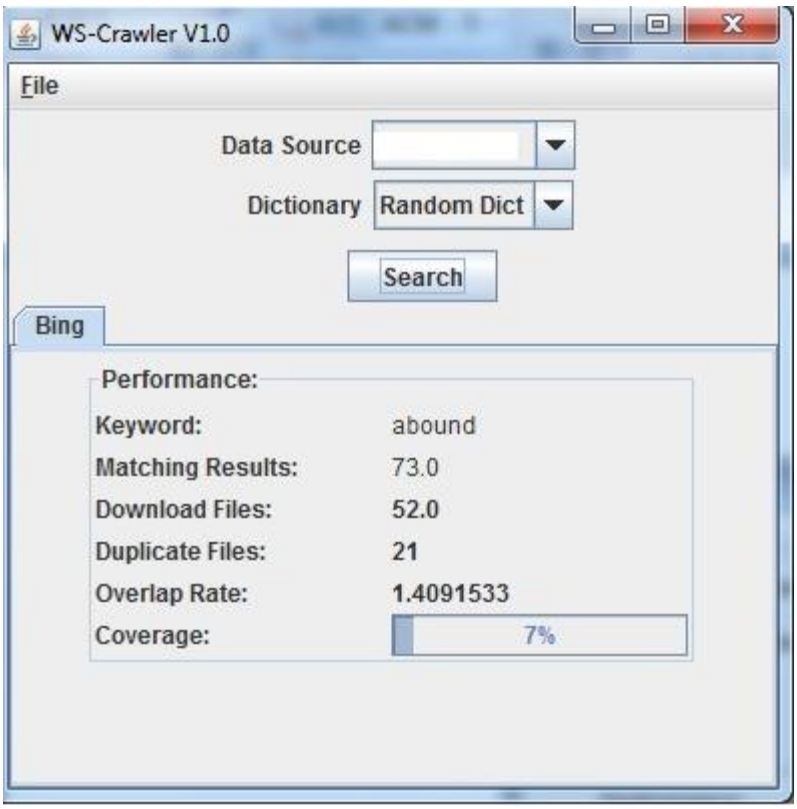


Рисунок 3.2 - Інтерфейс користувача нашого сканера

Щоб полегшити експеримент із веб-сервісом Bing, ми створюємо сканер веб-служби. На рисунку нижче показано графічний інтерфейс користувача та діаграму потоку даних нашого сканера. Щоб зробити сканер більш гнучким, система не залежить від алгоритмів. Алгоритм вибору запиту виводить запити в текстовий файл. І наш WS-Crawler читає запити з текстового файлу та створює пул запитів.

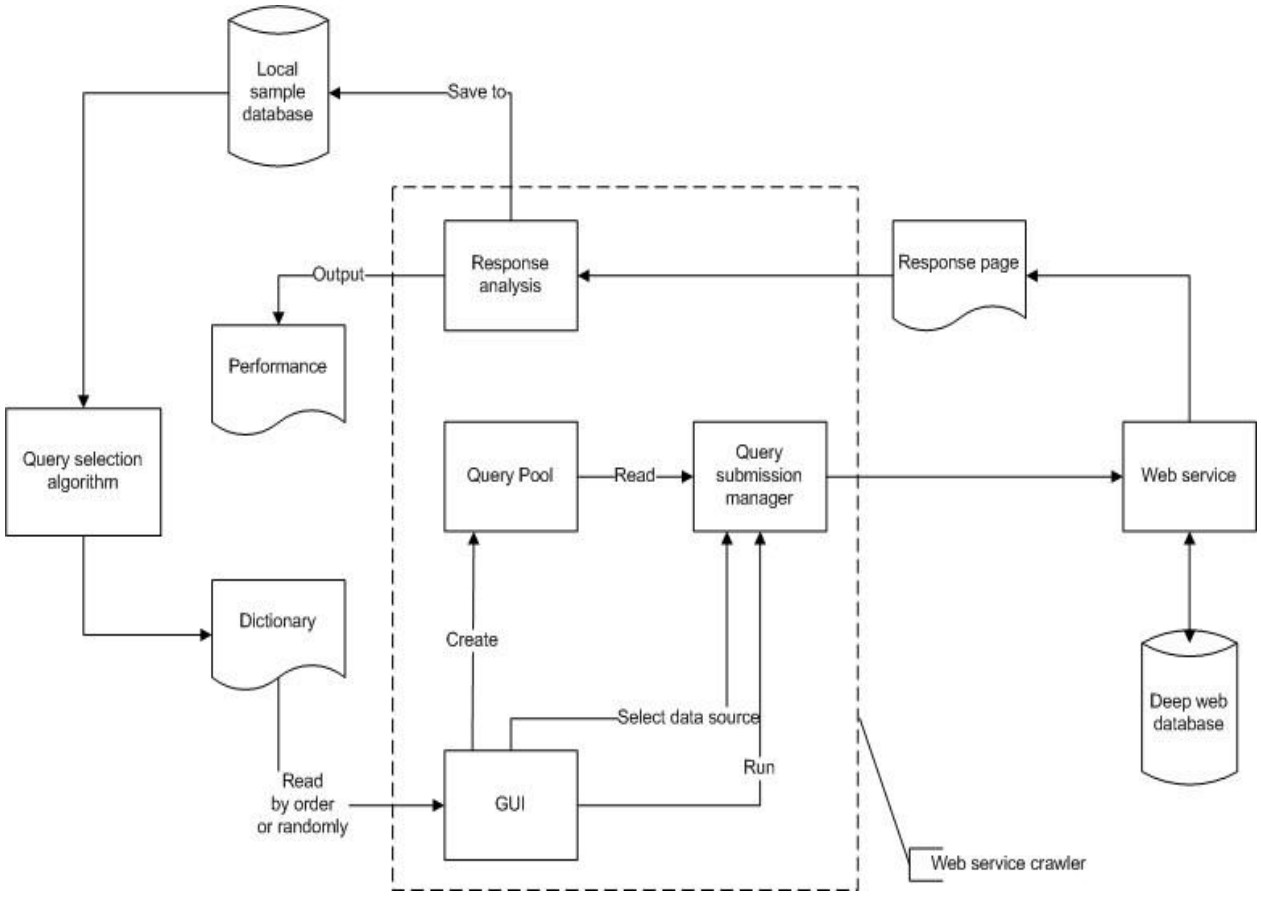


Рисунок 3.3 - Схема потоку даних сканера

3.2. Критерії оцінки

Коли ми вибираємо запити з документів за різними алгоритмами, рішення також мають бути різними. Щоб оцінити, яке рішення є кращим, ми вибираємо Hit Rate і Overlapping Rate як критерії оцінки.

Швидкість звернення – це вимірювання, скільки відсотків документів збирає сканер. Таким чином, показник звернення дорівнює кількості

завантажених унікальних документів, поділеній на загальну кількість документів у веб-базі даних.

$$HR(Q, DB) = \frac{|\cup_{j=1}^{i} S(q_j, DB)|}{|DB|} \quad (7)$$

Швидкість перекриття використовується для вимірювання вартості зв'язку. У формулі (8) коефіцієнт перекриття дорівнює кількості завантажених документів, включаючи дублікати документів, поділених на кількість завантажених унікальних документів.

$$OR(Q, DB) = \frac{\sum_{j=0}^{i} |S(q_j, DB)|}{|\cup_{j=0}^{i} S(q_j, DB)|} \quad (8)$$

Наприклад, у нас є набір документів, який містить 4 документи (d1, d2, d3 і d4). У нашому пулі запитів є 3 запити (q1, q2 і q3). Зв'язок між документами та запитами показано на рисунку 7. У нас є два рішення, які можуть охопити всі документи. Один – {q1, q2}, а інший – {q1, q3}. Solution 1:

$$\mathrm{OR}(\{q_1, q_2\}, \mathrm{DB}) = \frac{2+3}{4} = 1.25$$

$$\mathrm{HR}(\{q_1, q_2\}, \mathrm{DB}) = \frac{4}{4} = 100\%$$

Рішення 2:

$$\mathrm{OR}(\{q_1, q_3\}, \mathrm{DB}) = \frac{2+2}{4} = 1$$

$$\mathrm{HR}(\{q_1, q_3\}, \mathrm{DB}) = \frac{4}{4} = 100\%$$

Швидкість попадання для обох рішень однакова. Однак швидкість перекриття рішення 2 нижча, це означає, що рішення 2 досягає 100-відсоткового

покриття при меншій кількості завантажених документів. Отже, рішення 2 краще, ніж рішення 1.

3.3. Експериментальні дослідження та тестування

Експерименти проводяться на трьох джерелах даних: « .edu», «» та «ctv ». Для кожного джерела даних ми створюємо три вибірки, розміри яких становлять приблизно 5%, 10% і 20% від вихідного джерела даних. Ми створюємо три різні зразки баз даних для кожного джерела даних. Розміри цих зразків баз даних становлять приблизно 5%, 10% і 20%. зразки баз даних будуються таким чином:

1) Довільно виберіть запити зі словника Вебстера, що містить про 59000 термінів;

2) Надайте деякі з цих запитів веб-службі Bing і завантажте понад 20% документів;

3) Усі ці документи можна розділити на багато частин за запитами. Ми випадковим чином розбиваємо ці набори документів приблизно на 5%, 10% або 20% колекції документів;

4) Використовуйте Lucene (інструмент для індексації документів) для створення зразків баз даних.

Таблиця 3.1 - Опис джерел даних та зразків баз даних

Data Source	.edu		ctv
Approximate (N)	30,000	150,000	140,000
Sample Size	1548	8019	6911
	3319	13924	14504
	6066	29690	28568

Спостереження за оцінкою сили на джерелах даних

Проблема рейтингу має великий вплив на продуктивність завантаження даних із глибоких веб-сайтів. Таким чином, необхідно виміряти рівень рейтингу

джерел даних. Потужність рейтингу вимірюється шляхом обчислення відсотка запитів, які знаходяться в межах обмеження повернення. Усі вибрані терміни, частота документів яких перевищує 1, із трьох зразків баз даних (20%N). Усі ці терміни надсилаються до веб-сервісу Bing, і k має значення за замовчуванням 1000. Тоді ми можемо отримати кількість термінів, частота документів яких менше 100 і 200.

Таблиця 3.2 - Відсоток термінів у межах k

Data Source	Approximate (N)	Terms Num (tn)	df100	df100 /tn	df200	df200 / tn	Ranking Strength
.edu	30,000	48,522	36,546	75%	39,627	81%	weak
	150,000	271,530	173,438	64%	184,403	68%	middle
ctv	140,000	116,073	63,638	55%	69,385	59%	strong

З таблиці 3.2 можна зробити наступне спостереження:

Оскільки розмір джерела даних “ .edu” невеликий, збіги більшості запитів не перевищують межу повернення. Тому рейтинг рейтингу в цьому джерелі даних слабкий.

Ми оцінюємо чотири політики вибору запитів для 27 комбінацій експериментальних середовищ. Для кожного джерела даних ми встановили три різні обмеження повернення (100, 200 і 1000) і три різні бази даних зразків (приблизно 5%, 10% і 20%). Ми оцінили чотири політики вибору запитів наступним чином:

 Випадковий: випадковий вибір запитів зі словника Вебстера;

 Політика на основі вибірки: жадібний, зважений жадібний та DF-зважений жадібна політика.

Як згадувалося раніше, ці алгоритми на основі вибірки вибирають запити з матриць. Однак якщо ми згенеруємо матрицю шляхом експорту всіх термінів із зразка бази даних, ця матриця буде настільки великою, що пам’ять нашого комп’ютера не зможе цього дозволити. Згідно з дослідженням ми продовжуємо випадковий вибір термінів із вибіркової бази даних, поки загальна частота

термінів у документі не перевищить розмір вибіркової бази даних у 20 разів. Ми використали ці терміни для створення матриці, як зазначено в розділі . Нарешті, на матриці запускаються три алгоритми на основі вибірки.

Тому що ми оцінюємо ефективність сканування, порівнюючи значення OR і HR. У наших експериментах ми розробляємо діаграму для запису значення OR, HR і вихідних даних у таблиці 3.3.

Таблиця 3.3: Графік запису експерименту

Запит	est mi	mi	ui	di	Mi	Ni	or	OR	HR
q1									
q2									
….									
qi									

Нижче наведено пояснення щодо стовпців у таблиці 9:

1) est mi: збіги, оцінені пошуковою системою Bing;

2) mi: фактичні збіги;

3) ui : кількість нових документів, отриманих за допомогою запиту;

4) di: кількість дублікатів документів, отриманих за допомогою запиту;

5) Mi: загальна кількість збігів (включаючи повторювані документи), отриманих за допомогою {q1 …qi};

6) Ni: кількість загальних унікальних документів, отриманих за допомогою {q1 …qi};

7) або: швидкість перекриття запиту;

8) АБО: швидкість перекриття до qi; 9) HR: частота попадань до qi.

Після виконання 108 (4×27) експериментів з 27 комбінаціями ми створюємо 27 діаграм продуктивності. Значення OR відкладаються на осі x, а значення HR — на осі y. Усі діаграми продуктивності наведені в додатку I. Ми вибираємо деякі репрезентативні діаграми для переліку нижче.

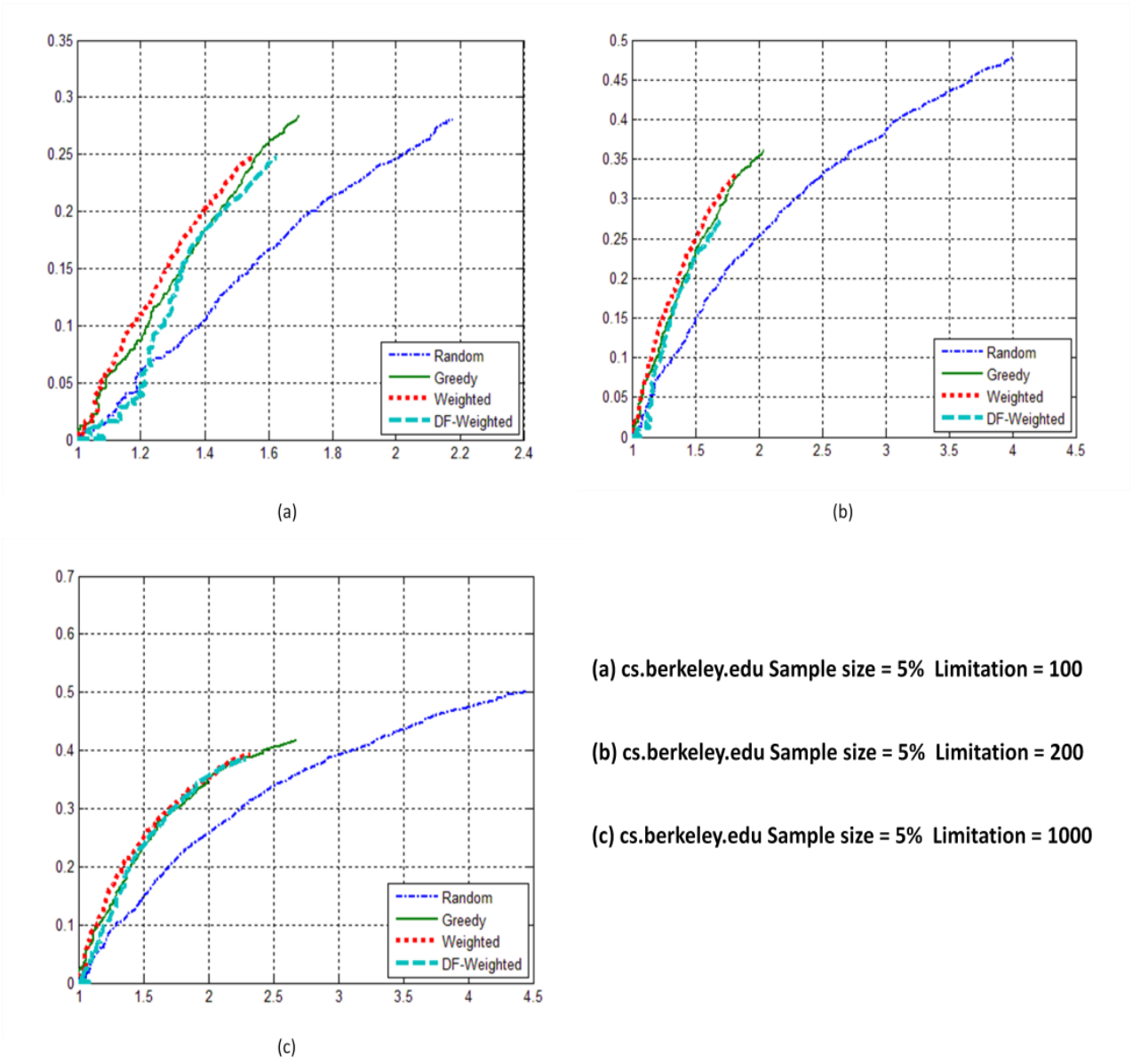


Рисунок 3.5 - Діаграми продуктивності

Перші три діаграми взяті з джерела даних « .edu». Як було сказано раніше, рейтинг цього джерела даних щодо запитів слабкий. Більшість виданих запитів не перевищують обмеження повернення k. Іншими словами, Weighted Greedy схожий на DF-Weighted. Таким чином, запропонований нами метод не має жодних переваг у цьому джерелі даних.

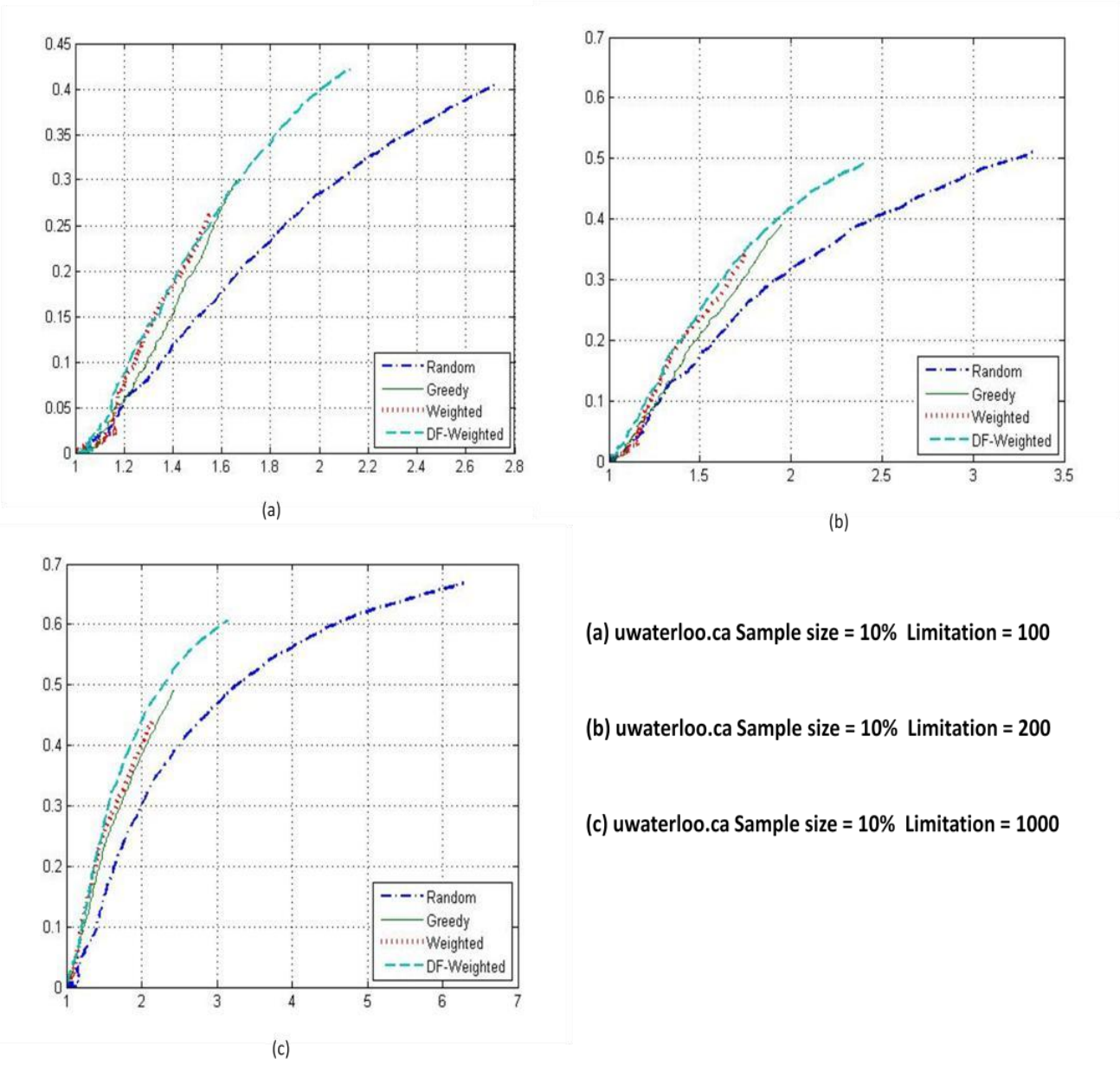


Рисунок 3.6 - Діаграми продуктивності

Проте все змінюється в діаграмах продуктивності. Оскільки рівень рейтингу стає сильніше, DF-Weighted працює краще в джерелах даних.

Це можна побачити на рисунку 3.7. DF-Weighted найкраще працює з точки зору АБО.

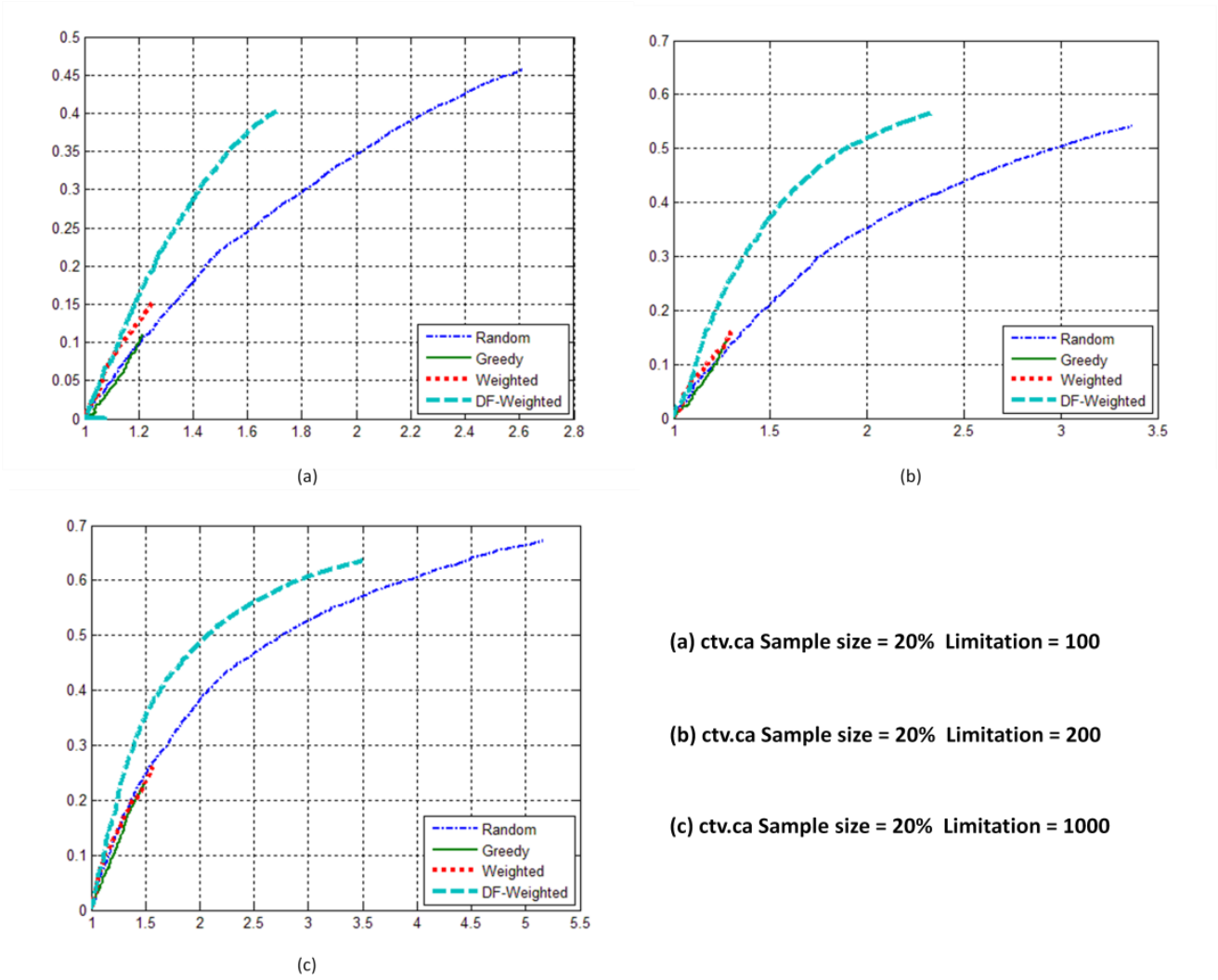


Рисунок 3.7 - Діаграми продуктивності ctv

Жадібний алгоритм і зважений алгоритм спочатку розроблені для вирішення неранжування джерел даних. Але рейтинг «ctv » найсильніший у трьох джерелах даних. На рисунку діаграми показують, що проблема ранжирування створює великі проблеми для обох алгоритмів. Для пояснення візьмемо приклад. «Дім» – це слово з найбільшою частотою документації і з'являється на більшості веб-сторінок «ctv ». Згідно з правилом жадібного алгоритму, слово «дім» має бути обрано жадібним алгоритмом.

Однак він не може отримати стільки документів, скільки очікувалося, через обмеження повернення. Тому важко досягти високого HR для жадібного та зваженого алгоритму. Але наш DF-зважений алгоритм міг би ідеально вирішити цю проблему. Діаграми продуктивності також показують, що DF-Weighted

найкраще працює з точки зору АБО в джерелах даних « » і «ctv ». Щоб було зрозуміліше, побудуємо таблицю (табл. 11). Ми перераховуємо їхній персонал із фіксованим значенням АБО для двох джерел даних: « » і «ctv ». Ми можемо спостерігати, що DF-зважений жадібний алгоритм працює найкраще, за винятком ситуації в першому рядку.

Таблиця 3.4 - Порівняння DF-Weighted та інших

				Випадковий словник	Жадібний	Зважений	DF-Зважений
дані	Обсяг вибірки(%)	k	OR	HR(%)	HR(%)	HR(%)	HR(%)
	5	100	1.4	12	19	20	19
		200	1.4	16	22	23	25
		1000	1.4	15	25	25	30
	10	100	1.4	12	15	19	19
		200	1.4	16	20	23	23
		1000	1.4	15	23	25	25
	20	100	1.4	12	18	21	22
		200	1.4	16	20	22	22.5
		1000	1.4	15	19	22	23
ctv	5	100	1.4	17.5	-	-	26.5
		200	1.4	18	-	-	27
		1000	1.4	21.5	-	-	26
	10	100	1.4	17.5	-	-	29
		200	1.4	18	-	-	30
		1000	1.4	21.5	-	-	31
	20	100	1.4	17.5	-	-	29
		200	1.4	18	-	-	32
		1000	1.4	21.5	-	-	32

Крім того, з усіх діаграм продуктивності ми також виявили: між жадібним алгоритмом і зваженим жадібним алгоритмом останній у більшості випадків трохи перевершує перший. Щоб було зрозуміліше, побудуємо нову таблицю. Перерахуємо їх HR із фіксованим значенням АБО для двох джерел даних: «

.edu» і « ». Ми можемо спостерігати, що зважений жадібний алгоритм завжди працює краще, ніж жадібний алгоритм.

Таблиця 3.5 - Порівняння жадібних і зважених

				Жадібний	Зважений
дані	Обсяг вибірки(%)	k	OR	HR(%)	HR(%)
.edu	5	100	1.5	21	24
		200	1.5	23	25
		1000	1.5	22	23
	10	100	1.5	19	22
		200	1.5	20	22
		1000	1.5	21	23
	20	100	1.5	23	23
		200	1.5	24	26
		1000	1.5	24	26
	5	100	1.4	19	20
		200	1.4	22	23
		1000	1.4	25	25
	10	100	1.4	15	19
		200	1.4	20	23
		1000	1.4	23	25
	20	100	1.4	18	21
		200	1.4	20	22
		1000	1.4	19	22

Вплив проблеми рейтингу на HDF

Алгоритм Барбози завжди вибирає запити з найвищою частотою в кожному раунді. Щоб очистити вплив проблеми ранжирування на запити з високою частотою документів, ми проводимо набір експериментів. В експерименті з розділу ми вже отримуємо фактичну частоту документів для термінів, надсилаючи всі терміни у зразковій базі даних до веб-служби Bing. Потім ми

витягуємо всі терміни, частота документів яких становить не менше 200, щоб створити новий набір запитів і зберегти їх у текстовому файлі.

Текстовий файл називається словником HDF.

Таблиця 3.6 - Кількість термінів для трьох джерел даних

Зразок бази даних (20%N)	Умови (df > 1)	Умови (df ≥200)
.edu	48,522	5,339
	271,530	25,042
ctv	232,146	20,269

На початку ми встановили k до 100 для всіх експериментів. У першому експерименті ми випадковим чином представляємо вибіркові умови. Ми називаємо цей підхід випадковим зразком. У другому експерименті ми вибираємо ці запити у словнику HDF як запити. Завдяки такому підходу завжди вибираються високочастотні запити, це позначається HDF. Щоб створити досить високу частоту термінів для документів, ми створюємо набір роздільних запитів шляхом випадкового поєднання термінів словника HDF з правилом АБО, скажімо «спочатку АБО Гейдельберг АБО соціальні АБО театри АБО загалом АБО включають». У третьому експерименті ми видаємо диз'юнктивні запити, що містять п'ять термінів.

Такий підхід називається HDF5. Таким же чином підхід, що видає набір диз'юнктивних запитів з 10 або 15 термінами, позначається HDF10 або HDF15. Ми передаємо ці чотири підходи до вибору запитів з високою частотою документів у політику HDF. Діаграми продуктивності п’яти підходів для трьох джерел даних наведені на рисунку 3.8.

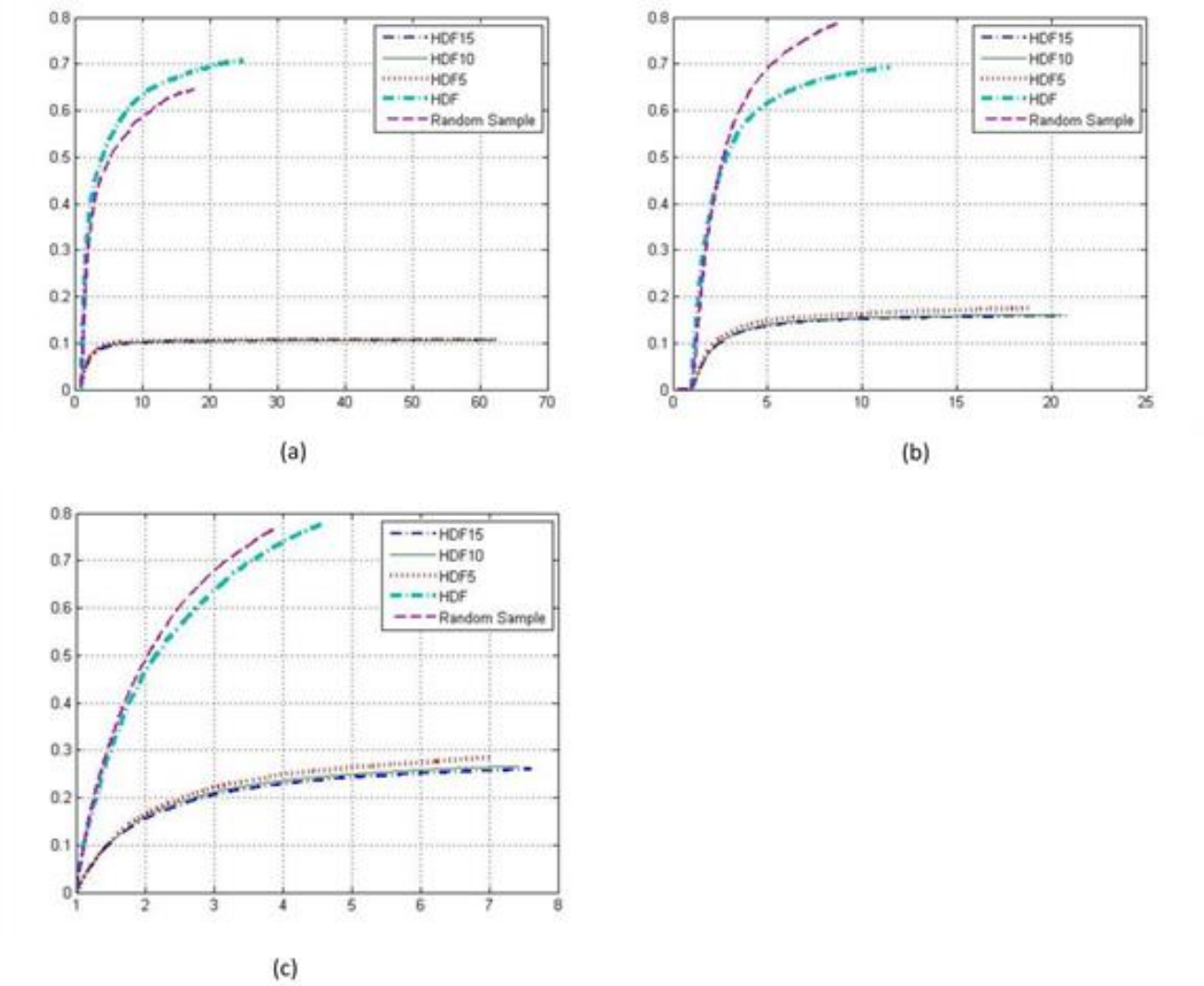


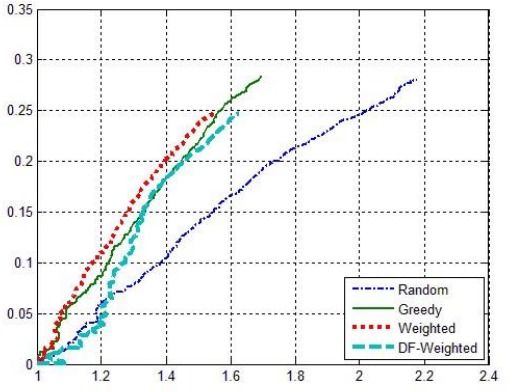
Рисунок 3.8 - Порівняння умов випадкової вибірки та термінів високої DF

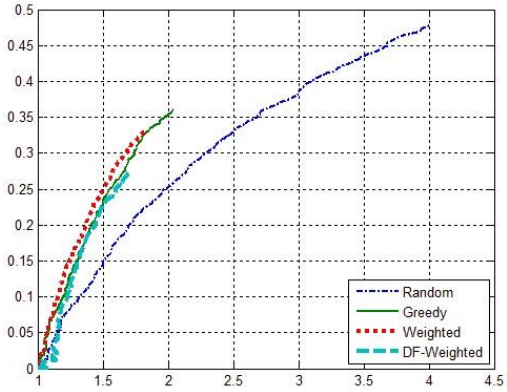
На рисунку 3.8 ми можемо зробити наступне спостереження: для політики HDF частота запитів документів вище, продуктивність гірша. На рисунку 19 (a), коли рівень рейтингу « .edu» слабкий, продуктивність підходу HDF перевищує підхід випадкової вибірки. Однак, оскільки рівень рейтингу стає сильніше в “ ” та “ctv ”, продуктивність підходу HDF навіть гірша, ніж підхід випадкової вибірки. Ці експерименти доводять, що політика HDF не може

досягти хорошої продуктивності для джерел даних рейтингу.

Усі експериментальні результати 27 комбінацій:

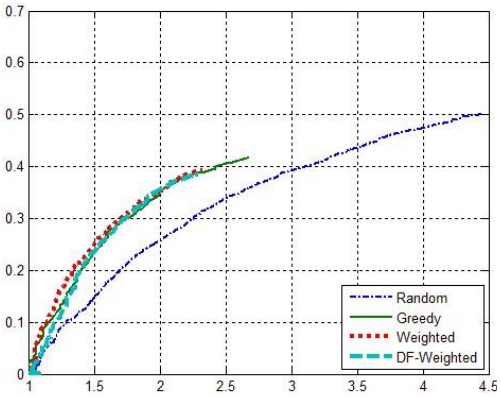
(a) .edu Розмір вибірки = 5% Обмеження = 100 (b) .edu Розмір вибірки = 5% Обмеження = 200





(c) cs.berkelev.edu Sample size = 5% Limitation = 1000

cs.berkeley.edu Sample size = 10% Limitation = 100



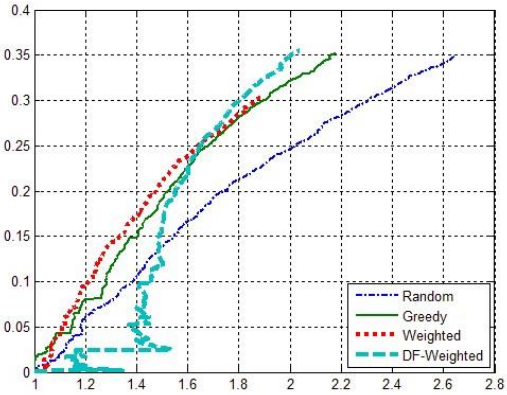
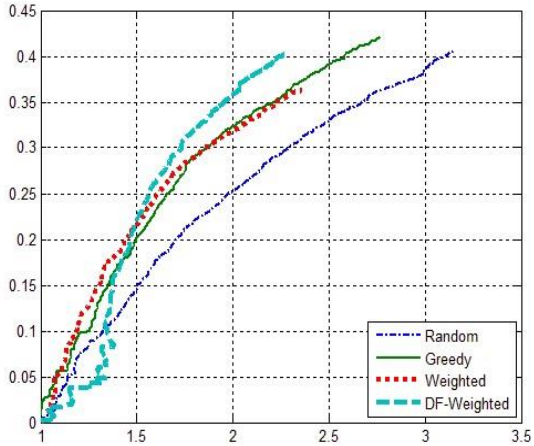


Рисунок 3.9 - Розмір вибірки = 10% обмеження = 200 .edu Розмір вибірки = 10% обмеження = 1000



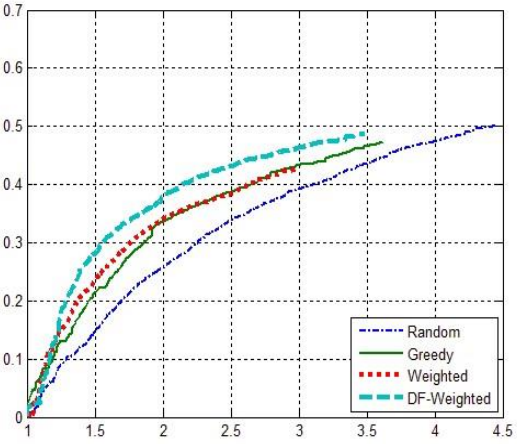
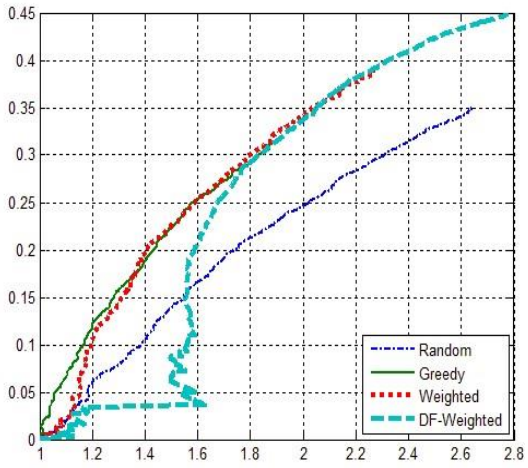


Рисунок 3.10 - Розмір вибірки = 20% обмеження = 100 .edu Розмір вибірки = 20% обмеження = 200



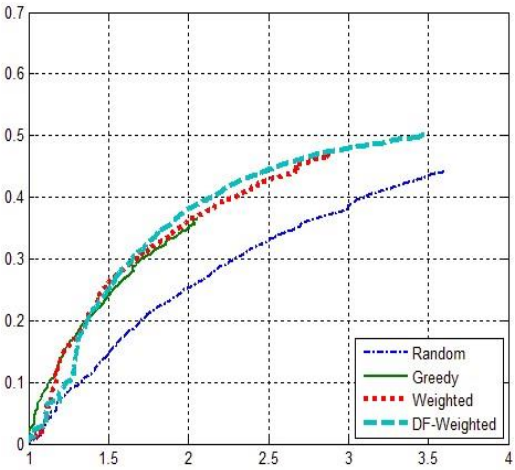
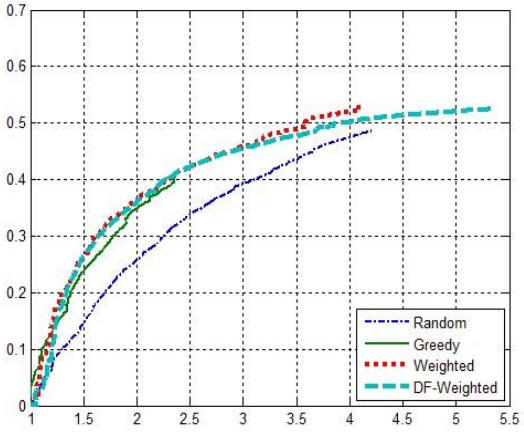
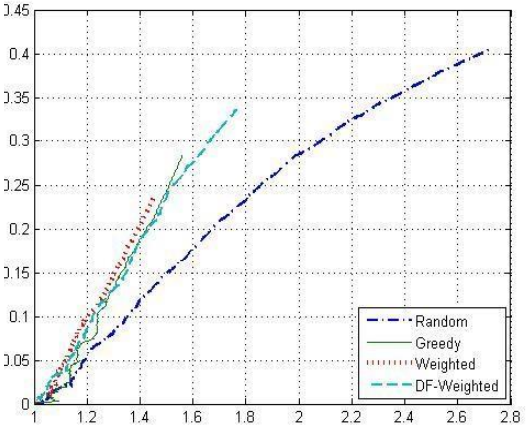


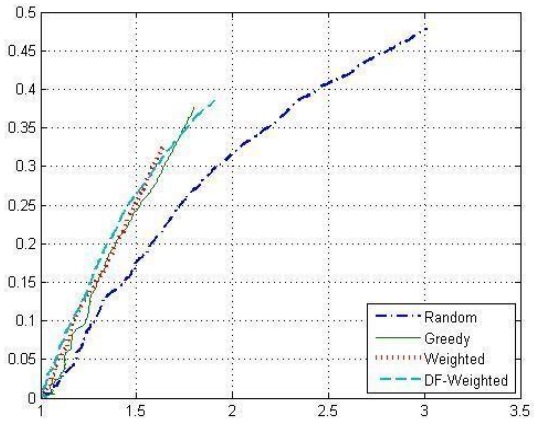
Рисунок 3.11 - Розмір вибірки = 20% обмеження = 1000 Розмір вибірки = 5% обмеження = 100



uwaterloo.ca Sample size = 5% Limitation = 200



uwaterloo.ca Sample size = 5% Limitation = 1000



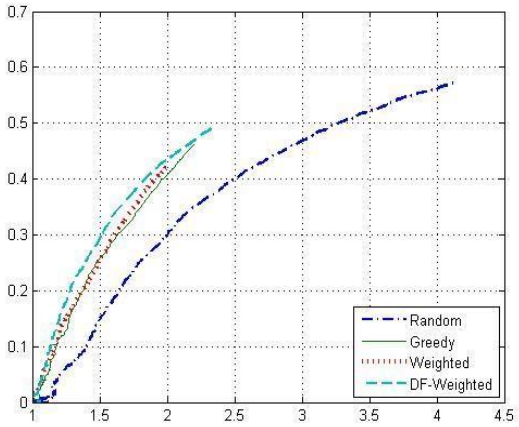
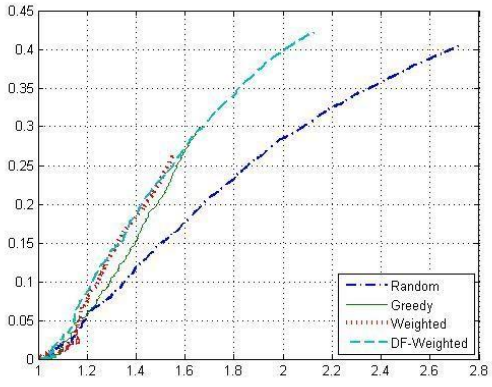
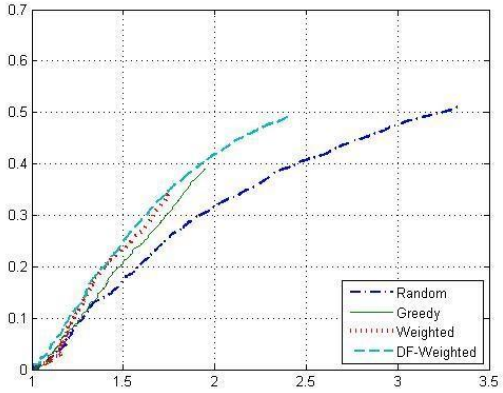


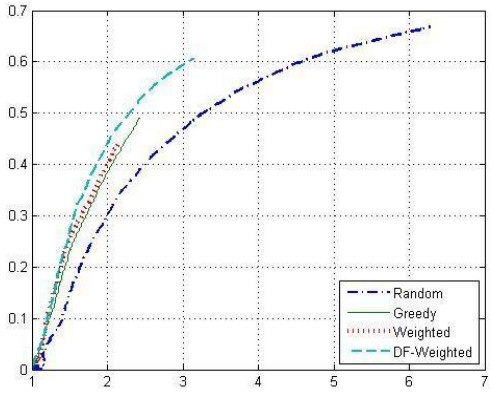
Рисунок 3.12 - Розмір вибірки = 10% обмеження = 100 Розмір вибірки = 10% обмеження = 200



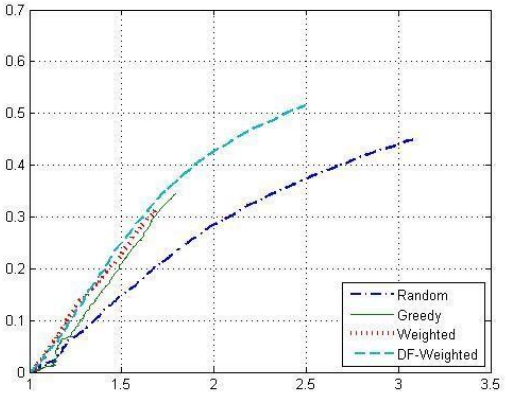
Sample size = 10% Limitation = 1000



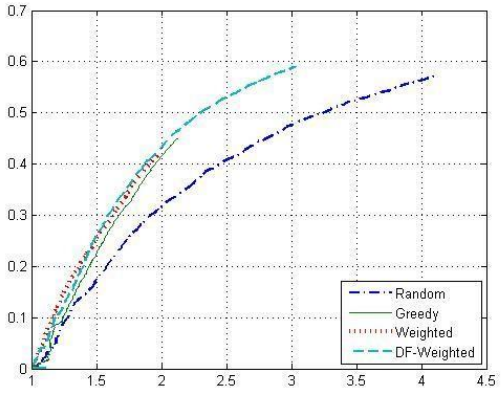
Sample size = 20% Limitation = 100



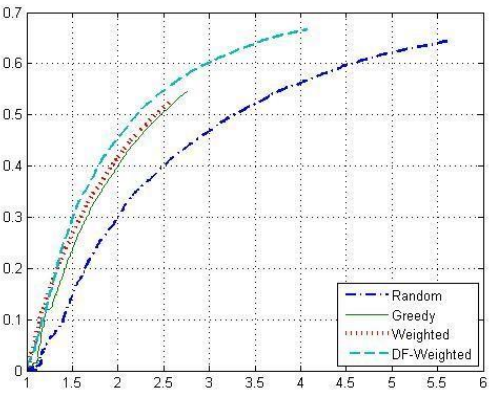
Sample size = 20% Limitation = 200



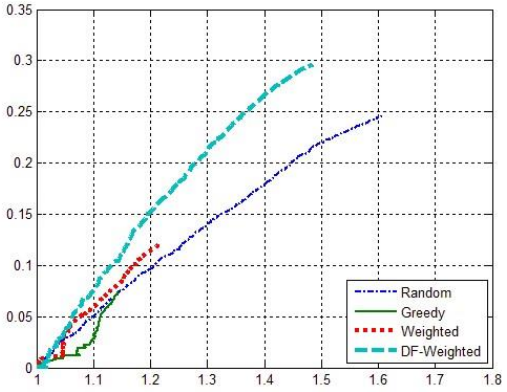
Sample size = 20% Limitation = 1000

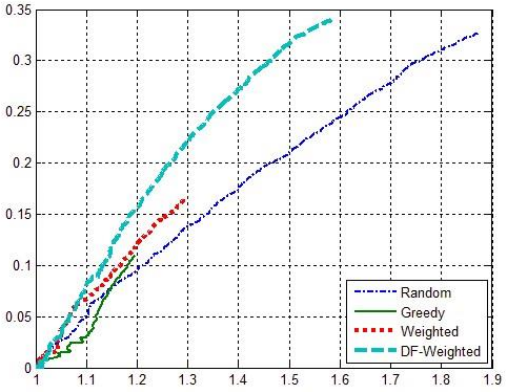


ctv Sample size = 5% Limitation = 100



ctv Sample size = 5% Limitation = 200





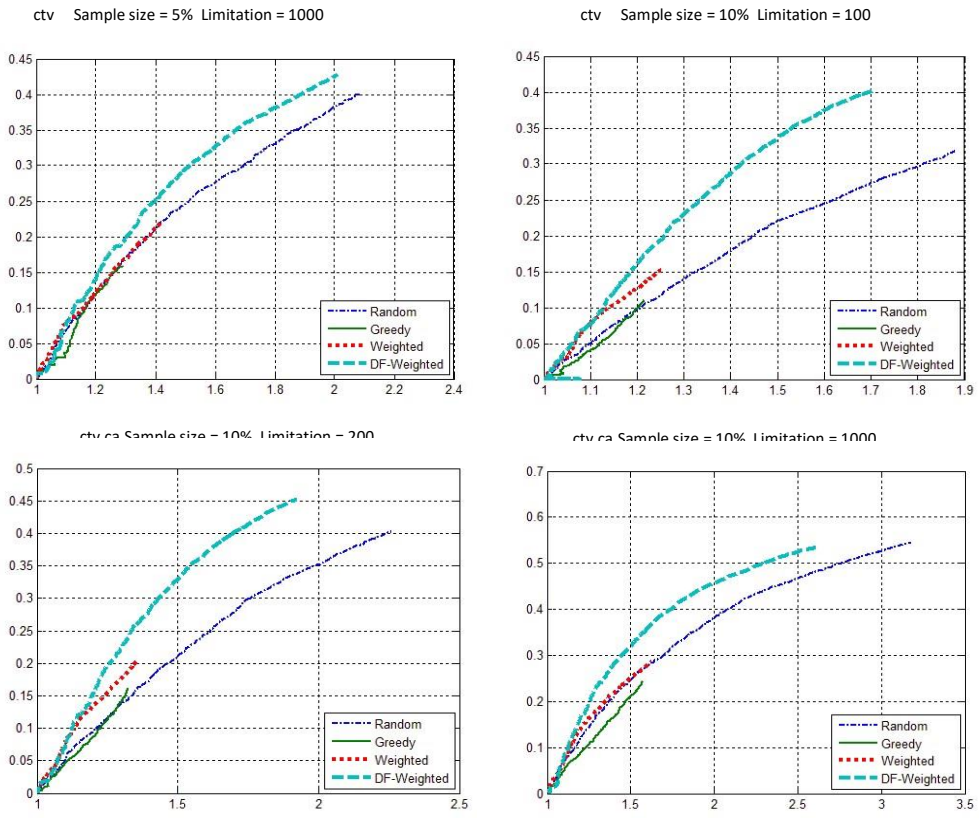


Рисунок 3.13 - Розмір вибірки = 20% обмеження = 100 ctv Розмір вибірки = 20% обмеження = 200

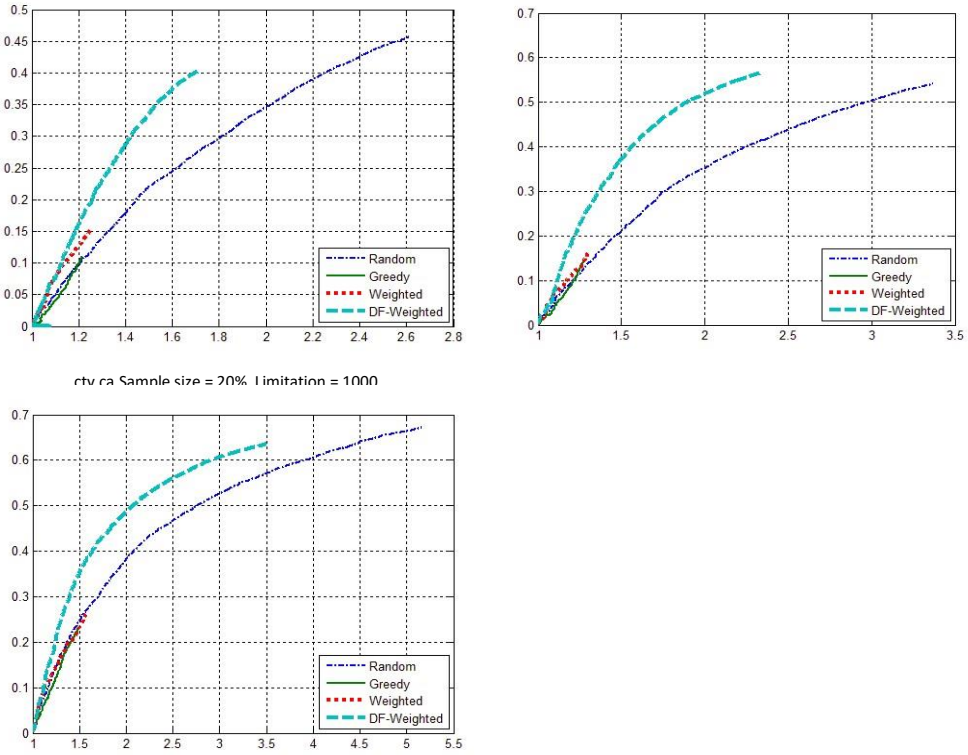


Рисунок 3.14 - Розмір вибірки = 10% обмеження = 100 Розмір вибірки = 10% обмеження = 200

ВИСНОВКИ

У цій дипломній роботі вивчається проблема архітектури ВЕБ-сервісів авторизації вибору запитів, щоб наш сканер ефективно отримував доступ до вмісту глибокої мережі. Щоб досягти цієї мети, ми вибираємо запити-кандидати із зразка бази даних, використовуючи набір алгоритмів покриття.

Звичайним методом вирішення проблеми покриття множин є жадібний алгоритм. І зважений жадібний є різновидом жадібного шляхом введення ваги запиту. Крім того, щоб зосередитися на проблемі вибору запитів, ми отримуємо доступ до глибокої мережі через веб-сервіси. Більшість із цих служб встановлюють обмеження повернення результатів. Щоб підвищити продуктивність сканування, ми розробили алгоритм, запровадивши оцінку частоти документів на основі зразкової бази даних.

Ми проводимо експерименти з веб-сервісом Bing і вибирали джерела даних. Ми оцінювали чотири стратегії вибору запитів: випадкові запити зі словника, жадібний алгоритм, зважений жадібний алгоритм і df-зважений алгоритм. Експериментальна оцінка показує:

1. Зважений жадібний алгоритм у більшості експериментів перевершує жадібний алгоритм.
2. Алгоритм досягає відмінної продуктивності в джерелах даних сильного рейтингу.

Обмеження кількості повернень є великою проблемою для сканування глибокої мережі. Обмеження суворіше, тому більш необхідним є хороший підхід до вибору запитів для веб-сканера. Зважений алгоритм досягає високого результату в експериментах, але метод оцінки частоти документа дійсно дуже наївний. Окрім незалежного методу оцінки максимальної правдоподібності, були запропоновані деякі інші підходи. Якщо ми включимо ці методи оцінки в техніку відбору запитів, ми вважаємо, що це має бути корисним для досягнення кращої продуктивності. Ми також виявили, що коли розмір джерела даних досить великий і обмеження повернення дуже мале, дуже важко досягти

високого рівня. Для вирішення цієї проблеми можливим методом є комбінація кількох ключових слів. Основна проблема полягає в тому, як об’єднати запит з кількома ключовими словами без перевищення обмеження повернення та з низькою вартістю. У цій дипломній роботі ми зосередимося лише на текстовій базі даних. Але як вибрати запити для завантаження реляційної бази даних також цікава тема. Для реляційної бази даних форма запиту html зазвичай також надає інтерфейс із кількома атрибутами.

Ми можемо застосувати ту саму ідею, яку використовували для вибору запиту на обіцянку для кожного атрибута, оцінюючи зразкову базу даних на основі частоти документів. Передбачити частоту документування значень кількох атрибутів також має бути великою проблемою.

СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА

[1] L. Barbosa and J. Freire. Siphoning hidden-web data through keyword-based interfaces. In SBBD, 2004.

[2] M. K. Bergman. The Deep Web: Surfacing Hidden Value. Journal of Electronic Publishing, 2001.

[3] S. Byers, J. Freire, and C. T. Silva. Efficient acquisition of web data through restricted query interfaces. In WWW Posters, 2001.

[4] J. P. Callan and M. E. Connell. Query-based sampling of text databases. ACM Transactions on Information Systems, 19(2):97–130, 2001.

[5] Cars.com FAQ. http://siy.cars.com/siy/qsg/faqGeneralInfo.jsp#howmanyads.

[6] A. Doan, P. Domingos, and A. Y. Halevy. Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach. In SIGMOD, 2001.

[7] L. Gravano, P. G. Ipeirotis, and M. Sahami. QProber: A system for automatic classification of hidden-web databases. ACM Transactions on Information Systems, 21(1):1–41, 2003.

[8] B. He and K. Chang. Automatic Complex Schema Matching across Web Query Interfaces: A Correlation Mining Approach. TODS, 31(1), 2006.

[9] B. He, M. Patel, Z. Zhang, and K. C.-C. Chang. Accessing the Deep Web: A survey. Communications of the ACM, 50(5):95–101, 2007.

[10] P. G. Ipeirotis and L. Gravano. Distributed Search over the Hidden Web: Hierarchical Database Sampling and Selection. In VLDB, pages 394–405, 2002.

[11] J. Madhavan, P. A. Bernstein, A. Doan, and A. Y. Halevy. Corpus-based Schema Matching. In ICDE, 2005.

[12] J. Madhavan, S. Jeffery, S. Cohen, X. Dong, D. Ko, C. Yu, and A. Halevy. Web-scale Data Integration: You can only afford to Pay As You Go. In CIDR, 2007.

[13] A. Ntoulas, P. Zerfos, and J. Cho. Downloading Textual Hidden Web Content through Keyword Queries. In JCDL, pages 100–109, 2005.

[14] S. Raghavan and H. Garcia-Molina. Crawling the Hidden Web. In VLDB, pages 129–138, 2001.

[15] A. Rajaraman, Y. Sagiv, and J. D. Ullman. Answering Queries Using Templates with Binding Patterns. In PODS, 1995.

[16] G. Salton and M. J. McGill. Introduction to Modern Information Retrieval. 1983.

[17] J. Wang, J.-R. Wen, F. Lochovsky, and W.-Y. Ma. Instance-based Schema Matching for Web Databases by Domain-specific Query Probing. In VLDB, 2004.

[18] P. Wu, J.-R. Wen, H. Liu, and W.-Y. Ma. Query Selection Techniques for Efficient Crawling of Structured Web Sources. In ICDE, 2006.

[19] W. Wu, C. Yu, A. Doan, and W. Meng. An Interactive Clustering-based Approach to Integrating Source Query Interfaces on the Deep Web. In SIGMOD, 2004.

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи бакалавра: "Побудова архітектури ВЕБ-сервісів авторизації в середовищах заданої конфігурації"

Обсяг пояснювальної записки: 58 аркушів

Дата закінчення дипломної роботи 15 червня 2022р.

Підпис студента _____