Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи та алгоритми семантичної обробки веб-документів

(тема)

Виконав:

студент II курсу, групи СПм-20-1

Кулєшов Д.О.
(прізвище, ініціали)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

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

Освітня програма Системне програмування
(повна назва освітньої програми)

Керівник: доц. Лебедєв О.Г.
(посада, прізвище, ініціали)

Допускається до захисту

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

2021 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

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

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Кулєшову Дмитру Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи Методи та алгоритми семантичної обробки веб-документів

затверджена наказом по університету від “ 5 ” листопада 2021 р. № 1657 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 13 грудня 2020 р.

3. Вхідні дані до роботи
використання колекції документів
методи семантичної обробки
використання Java

4. Перелік питань, що потрібно опрацювати у роботі
Сучасні методи виявлення схожих документів
Оцінка схожості документів
Алгоритм виділення семантичних блоків з веб-документів
Реалізація методу

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 14 слайдів ф.А4

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

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання. Пошук літератури	08.11.2021–15.11.2021	
2	Порівняльний наліз сучасних методів	16.11.2021–25.11.2021	
3	Розробка моделі оцінки подібності	26.11.2021–28.11.2021	
4	Розробка алгоритму виявлення схожих	29.11.2021–02.12.2021	
	документів		
5	Реалізація методу	03.12.2021–05.12.2021	
6	Аналіз результатів	06.12.2021–08.12.2021	
7	Оформлення ПЗ	09.12.2021–12.12.2021	

Дата видачі завдання 8 листопада 2021 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Лебедєв О.Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 81 с., 11 рис., 1 табл., 1 дод., 6 джерел.

СЕМАНТИКА, СХОЖІСТЬ, ВЕБ-ДОКУМЕНТ, ПЛАГІАТ, ПОШУК, КОНТЕНТ, БЛОК, НЕЧІТКІСТЬ.

Метою кваліфікаційної роботи є аналіз методів та алгоритмів семантичної обробки веб-документів.

У ході виконання кваліфікаційної роботи проведено аналіз сучасних методів виявлення схожих текстів в цілому та їх застосування до області веб-документів зокрема. Розглянуто задачу виявлення схожості веб-документів, використовуючи інформацію про схожість їх структурно-семантичних блоків. Запропоновано метод виділення блоків на підставі моделі представлення веб-документа. Розроблено програмні засоби, які дозволяють експериментальну перевірку ефективності розглянутих методів і алгоритмів.

ABSTRACT

Master’s thesis: 81 pages, 11 figures, 1 tables, 1 appendices, 6 sources.

SEMANTICS, SIMILARITY, ВЕБ-DOCUMENT, PLAGIARISM, SEARCH, CONTENT, BLOCK, FUZZINESS.

The major goal of this thesis is to analyze the methods and algorithms of semantic processing of веб documents.

In the course of qualification work the analysis of modern methods of identifying similar texts in general and their application to the field of веб documents in particular. The problem of similarity of веб documents, using information about the similarity of their structural and semantic blocks, is considered. A method for selecting blocks based on a веб document presentation model is proposed. Software tools have been developed that allow experimental verification of the effectiveness of the considered methods and algorithms.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 8
ВСТУП 9
1.1 Розпізнавання схожих документів 12
1.2 Термін схожості документів 14
1.3 Існуючі джерела 15
1.4 Метрики подібності документів 16
1.5 Виявлення схожих документів 19
1.5.1 Методи шинглюювання 20
1.5.2 Лексичні методи 24
1.6 Методи кластеризації 26
2 ОЦІНКА СХОЖОСТІ ДОКУМЕНТІВ 31
2.1 Модель представлення веб-документа 31
2.2 Метод виділення блоків з веб-документа 34
2.3 Метод оцінки схожості блоків 36
2.4 Підходи до формалізації нечіткості знань про схожість документів 39
2.5 Метод оцінки 40
3 АЛГОРИТМ ВИДІЛЕННЯ СТРУКТУРНО-СЕМАНТИЧНИХ БЛОКІВ З ВЕБ-ДОКУМЕНТІВ 45
3.1 Алгоритми розбиття веб-сторінок на блоки 45
3.1.1 Практичне застосування 45
3.1.2 Поняття структурно-семантичного блоку 46
3.1.3 Алгоритм виділення блоків 47
3.1.4 Пошук оптимальних значень коефіцієнтів 52
3.1.5 Оцінка обчислювальної трудомісткості 55

3.2 Алгоритми створення єдиного відбитка на основі локальних параметрів документа 56
3.2.1 Виділені параметри веб-документів 58
3.2.2 Опис набору тестів 59
3.2.3 Експериментальна база 61
3.2.4 Обговорення результатів 62
4 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДУ ОЦІНКИ СХОЖОСТІ ВЕБ-ДОКУМЕНТІВ 65
ВИСНОВКИ 72
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 73
ДОДАТОК А 74
Графічний матеріал кваліфікаційної роботи 74

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ТК – набір тестових документів, що використовується для проведення експерименту

ССБ – зв’язний і логічно цілісний з точки зору семантики текстовий елемент, що має конкретне місцезнаходження в структурі веб-документа

HTML – (англ. Hypertext Markup Language – мова розмітки гіпертексту). Стандартна мова розмітки веб-документів в Інтернеті, являється додатком SGML

SGML – (англ. Standard Generalized Markup Language – стандартна узагальнена мова розмітки). Метамова, що визначає мову розмітки для документів.

XML – (англ. extensible Markup Language – розширювана мова размітки). Підмножина SGML, розроблена для спрощення процесу машинного розбору документа.

ВСТУП

Експонентне зростання інформації, що спостерігається останніми роками, призводить до швидкого розвитку різних методів пошуку інформації, що означає виявлення низки документів, що задовольняють заздалегідь заданим умовам пошуку. Необхідність знаходити потрібну інформацію в гігантських наборах даних викликає потреба в пошукових системах, створення яких потребує безлічі конкретних завдань. Одне з таких завдань - визначити схожість різних веб-документів. Відомо, що понад 30% всього інтернет-контенту дублює існуючі сторінки. Виявлення дублікатів дозволяє вирішувати такі завдання:

- усунення дублювання документів під час видачі пошукової системи;
- стиснення даних;
- розпізнавання масових несанкціонованих електронних листів;
- автоматична класифікація;
- Пошук схожих за змістом документів.

У базах даних найбільших пошукових систем є значна кількість записів, що повторюються. Успішне копіювання документа, що вже знаходиться в базі даних, заощаджує місце та покращує якість пошуку. Стиснення даних, яке досягається за рахунок кластеризації повторюваних документів, може значно зменшити розмір масивів документів. Застосування цієї техніки до індексу пошукової системи дозволяє виключити до 30% документів та, як наслідок, збільшити швидкість роботи з індексними даними. Розпізнавання масових несанкціонованих листів; (Спам), на які припадає більше 90% всіх потоків електронної пошти, також є важливим завданням. Ідентифікація схожих документів як таких поштових відправлень дозволяє ефективно знаходити та знищувати величезну кількість небажаної пошти. Завдяки якісній класифікації документів з’являється можливість більш простої навігації за великими масивами документів (наприклад,

результати пошуку), а також оптимізація роботи з непередбачуваними інформаційними потоками контенту. Функція пошуку за запитом документів, схожих на зазначені, дозволяє визначати їхню версію, а також виявляти цитати та плагіат.

Метою кваліфікаційної роботи є аналіз методів та алгоритмів семантичної обробки веб-документів для підвищення ефективності виявлення схожих документів шляхом розробки методу оцінки подібності веб-документів на основі їх локальних параметрів з використанням попереднього розбиття на структурно-семантичні блоки. Основне завдання сформульовано як підвищення якості розпізнавання схожих веб-документів за рахунок оцінки схожості блоків, компонентів веб-документів, а також використання алгоритмів створення відбитків на основі локальних параметрів документів. Для досягнення мети роботи необхідно вирішити такі дослідницькі завдання:

- проаналізувати сучасні методи виявлення подібних текстів, їх застосування у сфері веб-документів, виявити основні метрики схожості та метрики оцінки ефективності;

- розробити математичну модель оцінки схожості документів на рівні складових їх блоків. Створити методику оцінки подібності веб-документів на основі їх структурних та семантичних блоків;

- розробити алгоритм вилучення структурних та семантичних блоків із веб-документів. Оцінити ефективність алгоритмів створення стабільних відбитків документів на основі їх локальних параметрів;

- на основі отриманих результатів розробити структуру спеціального програмного забезпечення, що реалізує методи та алгоритми оцінки схожості веб-документів шляхом розбиття їх на структурні та семантичні блоки. Провести експериментальні дослідження для перевірки розроблених методів та алгоритмів.

1 СУЧАСНІ МЕТОДИ ВИЯВЛЕННЯ СХОЖИХ ДОКУМЕНТІВ

Термін «інформаційний пошук» (information retrieval) був вперше введений Кельвіном Муром в 1948 році [7]. Під пошуком інформації розуміється виявлення в деякій множині документів (текстів) тих з них, які задовольняють заздалегідь визначеним умовам пошуку (пошуковому запиту) або містять відповідні інформаційним потребам дані. У загальному випадку пошук інформації складається з чотирьох етапів:

- визначення інформаційної послідовності і формулювання інформаційного запиту;
- визначення множини можливих джерел інформації;
- отримання інформації з виділених джерел;
- знайомство з отриманою інформацією і оцінка результативності пошуку.

Головне завдання інформаційного пошуку – задовольнити інформаційні потреби користувача. Так як формалізувати побажання користувача (які часто і сам користувач неясно представляє) вдається не завжди, частіше за все ці потреби формуються у вигляді набору ключових слів – запиту. Незважаючи на те, що найбільш поширеним об’єктом запиту є текстовий документ, ніяких принципових обмежень не існує. Наприклад, можна розглядати пошук зображень, музики та інших видів інформації. Таким чином, класичну задачу інформаційного пошуку можна сформулювати як пошук документів, що задовольняють запит. На даний момент, крім цієї основної задачі, досліджується безліч суміжних завдань: фільтрація, класифікація і кластеризація документів, вилучення метаданих, ефективне ранжування, побудова мов запитів, пошук як в локальних реляційних базах даних, так і в гіпертекстових базах даних на зразок Інтернету, і багато іншого [1].

Колосальна кількість документів в Інтернеті не тільки дає можливість

знайти інформацію практично з будь-якого питання, а й ставить завдання пошуку потрібного документа серед незліченної кількості інших. Крім того, Інтернет характерний високою часткою тимчасової інформації, її неконтрольованим якістю та різнорідністю. Неможливість створення виразного каталогу вмісту Інтернету породила особливу категорію сайтів – пошукових систем, що дають можливість пошуку необхідної інформації. Комплекс програм, що забезпечує функціональність пошукової системи, називають пошуковою машиною. Основними критеріями якості роботи пошукової машини є – релевантність пошуку та повнота бази. Поліпшення роботи пошукових систем – це одне з пріоритетних завдань сьогоденного Інтернету. Індексація інформації для пошукової машини здійснюється спеціальними пошуковими роботами. Постійно скануючи вміст мережі і поповнюючи власний індекс, пошукова машина надає спосіб пошуку посилання на потрібний документ шляхом вказівки пов’язаного з його вмістом набору ключових слів. При цьому на кожному етапі пошуковій машині доводиться використовувати безліч специфічних алгоритмів з області інтелектуального аналізу даних (data mining), від особливостей виділення значущих елементів веб-сторінок до ранжирування документів у видачі (списку-результаті запитів).

1.1 Розпізнавання схожих документів

Щоб уникнути появи дублікатів документів у видачі пошукової машини, в процесі поповнення її індексу необхідно вміти визначати схожість даного документа з вже наявними в базі. Коректне визначення схожих документів стосовно веб-пошуку переслідує численні цілі [10]:

- стиснення даних;
- поліпшення ранжирування;
- автоматична класифікація;
- розпізнавання масових поштових розсилок (спаму);

- розпізнавання плагіату;
- знаходження схожих за змістом документів.

Шляхом кластеризації документів-дублікатів можна значно зменшити розмір великих масивів документів. Ще більшого стиснення можна досягти, поєднуючи не тільки дублікати, а й просто схожі документи. Наприклад, застосування такої техніки до індексу пошукової машини дозволяє виключити до 30% документів, і як наслідок, збільшити швидкість роботи з даними індексом.

В видачах найбільших пошукових машин є 5.5% записів-дублікатів [8]. Фільтрація у видачі схожих документів дозволяє підвищити інформативність пошуку для користувача.

За умови якісної класифікації документів виникає можливість більш простої навігації по колекціях документів (наприклад, за результатами пошуку), а також потенціал для оптимізації роботи з непередбачуваними по вмісту інформаційними потоками.

Перевірка документа на наявність запозичень з різних джерел мережі дозволяє з високим рівнем достовірності визначати ступінь оригінальності документа.

Функція знаходження по запиту документів, подібних заданому, дозволяє знаходити подібні один одному документи. У перспективі можлива побудова дерева версій документа, відстежуючи паралельний розвиток з одного спочатку джерела.

Застосування техніки виявлення схожих документів не обмежується вищеописаними цілями. Вона також може застосовуватися в задачах краулінга (веб-crawling), виявленні ссилочного спаму (веб-spam), визначенні авторства тексту за стилем його написання і в інших областях. Крім того, ця задача тісно пов’язана із завданням кластеризації документів [6]. Метою в даному випадку є розбиття заданої вибірки документів на непересічні підмножини (кластери) таким чином, щоб кожен кластер складався з схожих об’єктів, а об’єкти різних кластерів істотно розрізнялися.

1.2 Термін схожості документів

Під висловом «схожі документи» можна розуміти безліч речей. З одного боку, документи з незначними відмінностями, наприклад, з різним оформленням, безсумнівно, будуть схожими. З іншого боку, існують тексти, написані абсолютно різним мовою, незначно перетинаються навіть по використаних словах, але мають дуже велику семантичну схожість. Такі документи також можна назвати схожими. Крім того, між двома вищеописаними крайнощами існує безліч серединних положень. Тому без чіткого визначення того, що саме можна назвати схожими документами подальше обговорення вести безглуздо. Власне поняття нечіткості можна інтерпретувати по різному [5]. Схема, що відображає види і форми нечіткості, наведена на рисунку 1.1. Через неоднозначності визначення поняття схожих документів (так званих майже-дублікатів) воно, в залежності від необхідності, по-різному трактується різними авторами. Існує «парадокс вимірювання» [10]: намагаючись дати об’єктивну оцінку, дослідники в ході роботи роблять багато суб’єктивних припущень. В результаті, незначні розбіжності у визначеннях «дублікатів» і замовчування частини параметрів експериментів авторами призводять до серйозних розбіжностей результатів, і часто складно визначити цінність і місце проведених досліджень.

Через відсутність еталонної міри для оцінки результатів визначення якості порівняння на сьогоднішній день дуже важко. На даний момент спеціальні методики оцінки якості визначення ступеня подібності документів відсутні, і найближчим часом їх поява не передбачається [4]. Різними авторами створюються різні визначення подібності, і, відповідно, різні міри схожості [6]. У зв’язку з цим в даний час формування методик і засобів для оцінки якості порівняння інформації являє собою відкриту проблему.

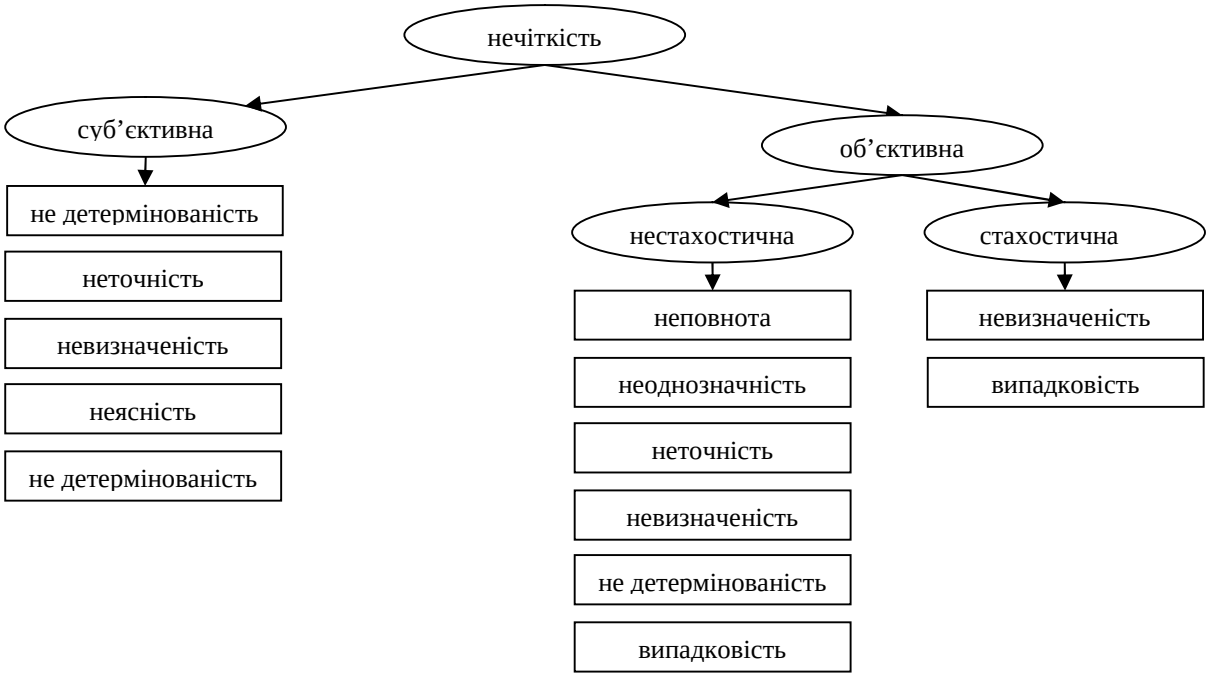

Рисунок 1.1 – Види і форми нечіткості

У наступному тексті будуть використані кілька понять. Під документом в більшості випадків буде матися на увазі типовий веб-документ, отриманий з Інтернету (що включає в себе html-код, елементи розмітки, заголовки і т.п.). Документом-дублікатом (або просто дублікатом) буде називатися ідентична оригіналу копія документа. Схожими документами (або майже-дублікатами) будуть називатися документи, маючі один і той же зміст, за винятком окремих, відносно невеликих, відмінностей.

1.3 Існуючі джерела

Джерела схожих документів в мережі можуть бути самими різними. Якщо виходити з того, що дублікат з’являється шляхом правки деякого вихідного документа, то дублікати документів можна розбити на кілька видів:

- виникаючі при створенні дзеркал (копій) сайтів і документів:
- зміна заголовка і підпису документа (header, footer);

- розташування та наповнення навігаційних елементів;
- різні HTML-адреси, що ведуть до одного документу;
- виникаючі при перетворенні документа:
- зміна формату документа (наприклад, DOC, PDF і HTML);
- різні версії та копії документа;
- шаблонні тексти (наприклад, ліцензійні угоди, опис товарів в Інтернет-магазинах);
- виникаючі при редагуванні документа:
- вставка та видалення абзаців і речень;
- перестановка речень і абзаців місцями;
- форматування тексту;
- незначні зміни тексту, помилки оператора;
- усічення в менший об’єм;
- застосовувані спамерами в масових розсилках:
- вставка в текст наборів випадкових символів;
- довільна вставка і видалення пробілів;
- заміна символів однієї розкладки на символи іншої, мають аналогічне написання;
- вставка значущих слів в текст у випадкових позиціях.

В залежності від передбачуваних видів модифікацій розглянутих документів ефективність використовуваних для розпізнавання дублікатів методів може серйозно відрізнятися.

1.4 Метрики подібності документів

Так як однозначно визначити, які саме документи є майже-дублікатами, неможливо, різними дослідниками були розроблені різні метрики, що дозволяють судити про ступінь подібності двох документів [5]. Метрика схожості, яка зветься схожістю (resemblance) (рідше – коефіцієнтом перекриття), визначається як відношення числа співпадаючих елементів двох

документів до загального числа складових їх елементів, що відображено у формулі 1.1:

$$\text{Resemblance}(A,B) = \frac{|S(A) \cap S(B)|}{|S(A) \cup S(B)|} \quad (1.1)$$

де S(A) – множина елементів документа A.

В якості елементів можуть виступати будь-які частини тексту – речення [4], слова [8], послідовності приголосних [6] або байтів [6]. Отримане значення порівнюється з певним порогом, при перевищенні якого документи вважаються схожими. Знаходження величини цього порога є однією з головних проблем методу.

Поширеною мірою близькості для текстових документів є коефіцієнт Дайса у формулі 1.2:

$$\text{Dicce}(A,B) = \frac{2|S(A) \cap S(B)|}{|S(A)| + |S(B)|}. \quad (1.2)$$

Альтернативний підхід використовує як метрики схожості косинуса кута між векторами, що представляють документи [7]. Спочатку документ представляється у вигляді вектора в n-мірному просторі, після чого коефіцієнт подібності двох документів можна отримати, знайшовши відстань між двома їх векторами за формулою 1.3:

$$\text{Cosine}(A,B) = \frac{\vec{A}\,\vec{B}}{|\vec{A}||\vec{B}|}. \quad (1.3)$$

З метою оптимізації під конкретні завдання були розроблені кілька модифікацій даного підходу, що додали аналіз частоти термів, довжини

документа і т.п. Типова проблема векторного представлення полягає в складності роботи з простором, що містить велику кількість вимірювання. Частково можна вирішити це питання скороченням числа термів в ході спрощення та підготовки документа.

Серед методів нечіткого зіставлення рядків можна виділити підходи, засновані на зіставленні символів, на зіставленні лексем, а також фонетичні методи.

До першого підходу відноситься відстань Левенштейна [2]. Вона розраховується для двох текстових рядків як кількість перестановок і вставок елементів, необхідних для редагування першого рядка до другого. Це визначення можна розповсюдити і для текстів, представляючи абзаци або речення як слова, а слова як символи. Метрика Левенштейна часто використовується в якості базової, інформації про пару документів, що елементарно розраховується.

До другого можна віднести схожість Джаккарда, одержувану аналогічно формулі 1.1. Крім того, існує поняття відстані Джаккарда, що представляє собою дистанцію між двома множинами лексем, представлену формулою 1.4:

$$J_\delta(A,B) = \frac{|S(A) \cup S(B)| - |S(A) \cap S(B)|}{|S(A) \cup S(B)|}. \quad (1.4)$$

Різні фонетичні алгоритми (наприклад, Soundex [12] і Phonix [6]), встановлюють однаковий індекс для рядків, що мають схожу звучання. Ефективність таких алгоритмів сильно залежить від мови, слів які порівнюються. Алгоритми, що використовуються для побудови систем перевірки орфографії (spell checker), також можуть бути адаптовані для визначення схожості текстів.

1.5 Виявлення схожих документів

Методи виявлення документів-дублікатів можна розділити на два класи. Повнотекстові використовують для порівняння весь текст документа. Найпростішим рішенням у цьому випадку є обчислення контрольної суми (сигнатури, хешу) кожної сторінки і попарне порівняння отриманих рядків. При наявності швидкої хеш-функції такий підхід може застосовуватися навіть в умовах великих обсягів документів.

Очевидно, що подібний повнотекстовий метод не може бути використаний при пошуку схожих документів, так як найменша зміна в тексті призводить до появи нового хешу. Для отримання стійкості до незначних змін тексту необхідно вміти отримувати стійку сигнатуру цього документа. Подібну функціональність можуть забезпечити методи порівняння на основі відбитків документа (fingerprints).

Основною ідеєю цього класу методів є те, що для опису документа береться не весь текст документа, а лише підмножина, що описує його ознаки, яка становить так званий відбиток документа. За умови вдало підібраної функції перетворення документа в відбиток такий підхід дозволяє виділити суттєві з точки зору визначення дублікатів елементи документа, і відкинути випадкові фактори, що і призведе до створення стійкого до невеликих змін відбитка документа. Можна показати, що зменшення розміру документа при фіксованій довжині відбитків цього документа веде до підвищення їх змістовності. Нехай ω – множина ознак документа d. Усічення документа d в документ d_{sub}, очевидно, веде до зменшення множини ознак ω:

$L(d_{sub}) < L(d) \rightarrow |\omega_{sub}| < |\omega|$, де $L(d_{sub})$ і $L(d)$ – довжина документів d_{sub} і d відповідно.

Відбиток S визначається його кінцевою підмножиною фіксованого розміру N, завідомо меншою $|\omega|: \omega_d \subset \omega, |\omega_d| = N, N < |\omega_d|$. Зі зменшенням загальної множини ознак ω частка його елементів, задіяних у створенні відбитка S, збільшується. Відповідно, із зменшенням розміру документа

відбиток описує все більшу його частину.

Зберігання відбитків документів, якісь можуть досягати значних розмірів, недоцільно. Замість них в базі даних пошукової машини зазвичай зберігаються контрольні суми, отримані в результаті застосування певної хеш-функції до відбитку документа. Найбільш часто використовуються хеш-функції MD2, MD5 і SHA. Причина їх вибору полягає в тому, що вони задовольняють трьом вимогам: можуть бути розраховані на підставі будь-яких даних, невимогливі до обчислювальних ресурсів і мають дуже низьку ймовірність випадкового збігу. Хеш-методи в принципі можуть бути узагальнені для використання їх при вирішенні найрізноманітніших завдань: як джерело можливий розгляд не тільки набору текстових документів, а й аудіо-файлів, колекцій зображень і т.д. за умови наявності адекватної хеш-функції для джерел кожного типу. В рамках цього класу існує кілька методів. Розглянемо найбільш поширені з них.

1.5.1 Методи шинглюювання

Одним з найбільш відомих методів, що використовуються для виявлення схожих документів, є метод, ідея якого запропонована в 1994 році Уді Манбером [6], і який був остаточно сформульований в 1997 році Андрієм Бродером [8]. Документ перетвориться в безліч ланцюжків елементів певної довжини. Кожному такому ланцюжку ставиться у відповідність хеш-код, так званий шингл (від англ. Shingle, «лусочка»). Два документа будуть вважатися схожими, якщо достатньо велика кількість шинглів, складених з їхніх текстів, співпаде. Таким чином, схожість документів можна висловити відношенням числа однакових шинглів до загальної кількості їх шинглів (по аналогії з формулою 1.1).

Для полегшення аналізу подібності документів проводиться їх спрощення, наприклад: вилучаються оператори розмітки html, прописні букви перетворюються в рядкові, видаляються знаки пунктуації і т.д. У

наступних розділах ця процедура буде розглянута більш докладно.

Результативність класичного алгоритму шинглювання може бути серйозно модифікована зміною ряду його параметрів. Розглянемо основні з них.Елементом ланцюжка в більшості випадків вважається слово, хоча існують варіації алгоритму з використанням в якості елемента окремих букв, або навпаки, цілих речень. Далі, необхідно визначити, чи повинні ланцюжки йти встик або внакладку. У першому випадку вставка навіть одного символу в початок тексту зруйнує ідентичність контрольних сум. Цього можна уникнути, якщо використовувати не фіксовану довжину шингла, а динамічну (наприклад, від точки до точки), що дозволить у разі невеликої зміни втратити тільки той шингл, де воно сталося. Таким чином, виділення шинглів встик доцільно використовувати у випадках, коли наперед відомо, що документ змінюється в малій кількості місць.

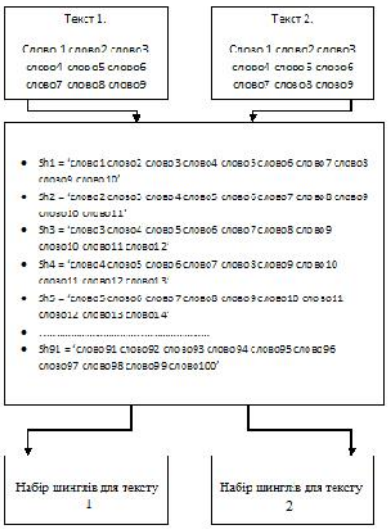

Рисунок 1.2 – Приклад шинглювання

У другому випадку елементи для шингла вибираються внакладку (рисунок 1.2). Це дозволяє не пропустити жодного підрядка тексту, що в кінцевому підсумку позитивно позначиться на якості визначення дублікатів. Однак, якщо не вжити жодних заходів для фільтрації шинглів, цей метод

значно збільшить обсяг даних для порівняння. Довжина шингла w вибирається таким чином, щоб, з одного боку, він був досить великим, щоб звести до мінімуму випадкові повтори, і був досить маленьким, щоб невелика зміна тексту не змінило значення більшості контрольних сум. Як показують експерименти [4], найбільша ефективність зазвичай досягається при довжині, рівній 10 словам. У випадку, якщо документ занадто короткий і не вдається обрати ні одного відповідного шинглу, існують дві можливості: або закільцювати текст документа, що дозволить довести його до необхідної довжини, або використовувати вибірку, розмір якої має логарифмічну залежність від розміру документа.

Отже, можна розрахувати значення шинглів для кожної сторінки, а потім порівняти їх попарно, отримавши ступінь схожості для кожної пари документів. Проте подібний процес при скільки-небудь великих обсягах вибірки (порядку тисяч документів і вище) пред'являє жорсткі вимоги до продуктивності системи. Виникає необхідність якимось чином відбирати необхідні послідовності.

Рішенням проблеми в класичному алгоритмі Бродера є або відбір деякої фіксованого кількості мінімальних за значенням шинглів, або використання шинглів, числові значення яких кратних якомусь числу m (зазвичай від 10 до 40). Другий підхід дозволяє аналізувати вкладеність документів один в одного і відсоток їх перетину за допомогою формули 1.5:

$$\mathrm{Nest}(A,B) = \frac{|S(A) \cap S(B)|}{|S(A)|}. \qquad (1.5)$$

Крім того, так як значення контрольних сум для різних документів розподілені рівномірно, критерій вибірки не прив'язаний до якихось особливостей тексту.

Для ряду практичних завдань отримана навіть після фільтрації кількість шинглів все одно є дуже великою і виникає завдання подальшої

оптимізації алгоритму. Цю проблему можна вирішити за допомогою методу супершинглювання: над розбитим на групи набором шинглів документа розраховується ще деяка кількість (зазвичай 1-6) контрольних сум, які називаються супершинглами. Експериментально було запропоновано такі, близькі до оптимальних параметри [9]: для всього документа обчислюється 84 шингли, вони розбиваються на 6 груп по 14 шинглів в кожній, і тексти вважаються дублями при збігу хоча б двох супершинглів з шести. Очевидно, що в даному випадку можливість виявити схожі документи, у який не повністю збігся набір шинглів, втрачається. Крім того, при роботі з невеликими документами (більшістю листів, наприклад), такий алгоритм неефективний. Проте в ряді завдань він знаходить своє застосування. Приміром, судячи по патенту [9], він використовується в пошуковій машині Google.

Відмінності між методами на основі шинглювання можна укласти в рамки чотирьох умов [6]:

- розмір підрядка, що вирізується з документа;
- функція, що обчислює хеш на основі заданої підрядка;
- число хешей, з яких формується відбиток документа;
- алгоритм вибору (фільтрації) хешей.

Ефективність методу в додатку до конкретної задачі може значно змінюватися в залежності від набору використовуваних параметрів. Було відзначено [10], що досліджень, присвячених систематичному аналізу взаємного впливу різних параметрів алгоритму шинглювання на якість розпізнавання, відсутньо. Особлива увага приділяється відбору шинглів, і показано, що ретельний підбір алгоритму фільтрації дозволяє помітно поліпшити точність.

Крім того, істотним недоліком методів, які використовують відбитки, є їх нездатність відокремити значущі частини текстів від незначущих. В результаті, при повному розбитті документів на фрагменти величезний об’єм (до 98% [5]) збережених даних є надлишковим, так як більшість фрагментів

зустрічається тільки один раз. Це буває дуже накладно в реальних умовах. При використанні стратегії вибіркового зберігання відбитків неминуче виникнення втрат, викликаних відкиданням значущих частин інформації. Алгоритм SPEX, запропонований в [5], дозволяє відкидати незначущі частини документів, зберігаючи при цьому прийнятну якість детектування дублікатів.

1.5.2 Лексичні методи

Методи виявлення схожих документів, умовно звані лексичними, розглядають тексти з точки зору набору слів (термів). На відміну від методів шинглювання, вони не враховують їх послідовність. Основна ідея полягає у відборі деякого безлічі представницьких слів, виходячи з показників значущості цих слів.

Лексичні методи можна розбити на дві групи: локальні та глобальні лексичні сигнатури. У разі застосування локальних сигнатур показником значущості зазвичай служать частотні характеристики слів. Частота слова (term frequency, TF) дозволяє оцінити важливість слова в межах окремого документа і розраховується за формулою 1.6:

$$\mathrm{TF} = \frac{\eta_t}{\sum_k \eta_k}, \tag{1.6}$$

де η_t – число входжень слова t в документ.

Для отримання представницької множини, відповідно до закону Ципфа (в кожній мові слова, що зустрічаються рідше, мають більше смислове значення) відбираються ті терми, які, по-перше, не є стоп-словами (тобто лексичні одиниці, що часто зустрічаються, які пошукова система виключає з розгляду для підвищення точності пошуку), а по-друге, терми, частоти яких лежать в деякому інтервалі між високочастотними (і, отже,

неінформативними в належному ступені) і низькочастотними (які часто є друкарськими помилками) термами (рисунок 1.3).

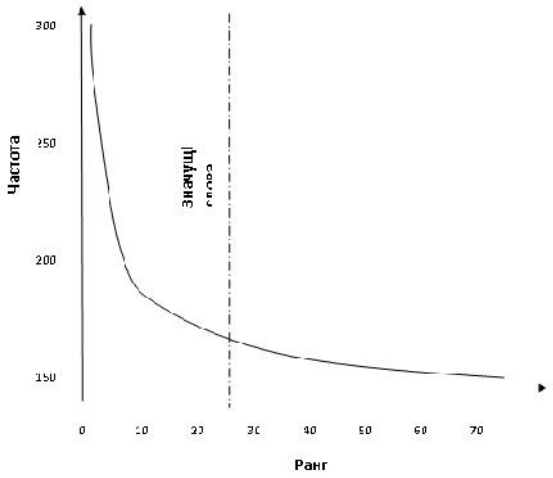

Рисунок 1.3 – Застосування закону Ципфа для відбору значущих слів

При використанні глобальних сигнатур враховується інформація про глобальну статистику слова (IDF, inverse document frequency), яка розраховується за формулою 1.7:

$$\mathrm{IDF} = \log\frac{1+|\mathrm{D}|}{|\mathrm{D_t}|}, \tag{1.7}$$

де D – загальна кількість документів, а $\mathrm{D_t}$ – число документів, що містять слово t.

Для отримання представницького множини в цьому випадку необхідний словник. Він містить набір представницьких слів, який повинен бути, з одного боку, не надто великим, а з іншого – добре описувати колекцію, причому присутність кожного з слів в документі має бути стійко по відношенню до невеликих змін документа.

Одним з найпоширеніших лексичних методів є I-Match [4].

Використовуючи глобальну статистику IDF для відбору множини представницьких слів, він демонструє хороші результати (особливо для невеликих документів, в яких методи шинглювання традиційно програють). У разі використання єдиної сигнатури документа алгоритм вельми нестійкий до невеликих змін документів (зокрема, до застосовуваної спамерами модифікації листів). Для підвищення такої стійкості створюється додатковий набір з K сигнатур, кожна з яких виходить випадковим видаленням частини термів з вихідної сигнатури [6]. В такому разі два документи будуть вважатися схожими, якщо їх набори з K + 1 сигнатур мають схожість хоча б по частині з них (тут є деяка аналогія з методом супершинглювання, в якому подібність також визначається як збіг частини супершинглів). Тим самим, однак, збільшується складність порівняння через появу безлічі відбитків для кожного документа.

I-Match показує найвищі результати на коротких документах. Це дозволяє використовувати його в широкому колі завдань, оскільки, як показують дослідження [5], розміри документів в Інтернеті відповідають закону степеневого розподілу. Звідси випливає, що в загальному випадку в розгляд значно частіше потрапляють документи невеликого розміру, а об’ємні документи зустрічаються значно рідше. Запропонований метод описових слів [9] використовує інвертований індекс пошукової системи для визначення представленості слів документа. Володіючи порівнянною точністю, він перевершує метод каскаду по ефективності, і застосовується в будь-якій пошуковій машині, основаної на інверсному індексі.

1.6 Методи кластеризації

Можливість дати оцінку подібності документів в числовому вигляді дозволяє використовувати великий ряд числових методів кластеризації [10].

Необхідно відзначити різницю між кластеризацією і класифікацією документів. Класифікація є віднесенням кожного документа до одного з

фіксованих числа класів із заздалегідь відомими параметрами (зазвичай отриманими на етапі навчання). Кластеризація – розбиття множини документів на кластери – підмножини, параметри яких заздалегідь невідомі. Кількість таких кластерів може бути довільним або фіксованим.

Можливі два способи проведення кластеризації: зверху вниз і знизу вгору. У першому випадку проводиться послідовний поділ єдиної множини документів до заданого порогу, у другому кожен документ розглядається як окремий кластер, і процес кластеризації полягає в агрегування подібних.

Загальна ідея кластеризації зводиться до поступового агрегуванню схожих документів навколо центроїдів. При цьому необхідно мати інформацію про ступінь подібності розглянутих документів. Для цього використовується матриця близькості документів в просторі термінів. Визначатися близькість документів може по різному: косинусом кута між векторами документів, або евклідовою відстанню між ними.

Під кластером розуміється група документів, що має схожі ознаки. Крім того, більшість методів так чи інакше використовують у своїй роботі поняття центроїда. Центроїд – це вектор, що отримується обчисленням середнього арифметичного векторів всіх документів кластера за формулою 1.8:

$$\vec{c} = \frac{1}{|S|} \sum \vec{d}, \tag{1.8}$$

де S – кластер, що містить безліч документів d.

Найбільш відомі методи кластеризації:

- LSA / LSI;
- CI;
- STC;
- Single Link, Complete Link, Group Average;
- K-means;

- Scatter/Gather.

LSA / LSI – Latent Semantic Analysis / Indexing. Шляхом факторного аналізу безлічі документів виявляються латентні (приховані) фактори, які надалі є основою для утворення кластерів документів. Особливості методу:

- ступеневий зріст часу обчислення може приводити до дуже довгої роботи системи (N2xk, де N = Ndocs + Nterms, k – розмірність простору факторів);

- використовувана модель представлення документа tf – idf зрівнює значимість ознак документа, знижуючи якість кластеризації;

- кластери не перетинаються.

CI – Concept Indexing. Розбиває безліч документів методом рекурсивної бісекції, розділяючи безліч документів на дві частини на кожному кроці рекурсії. Може використовувати інформацію, отриману на етапі навчання.

STC – Suffix Tree Clustering. Кластери утворюються у вузлах суффіксного дерева, яке будується з слів і фраз вхідних документів.

Методи Single Link, Complete Link, Group Average розбивають множину документів на кластери, розташовані в деревоподібній структурі, одержуваної за допомогою ієрархічної агломеративної кластеризації.

K-means. Відноситься до не-ієрархічним алгоритмам. Кластери представлені у вигляді центроїдів, які є «центром маси» всіх документів, що входять в кластер.

Scatter/Gather. Представляється як ітеративний процес розбиття множини документів на групи і представленні потім цих груп користувачеві для подальшого аналізу. Далі процес повторюється знову над конкретними групами.

1.7 Попередня обробка документа

Для поліпшення якості роботи алгоритмів порівняння на першому етапі документ проходить через ряд перетворень, що спрощують його структуру.

Попередня обробка текстів в залежності від мети може передбачати лексичний і морфологічний аналіз. Наведені нижче операції найбільш поширені.

- лексичний аналіз;
- видалення стоп-слів;
- стеммінг.

До лексичного аналізу входить очистка документа від службової інформації та сміттєвого змісту: віддаляється HTML і XML-розмітка, зайві пробіли, переклади рядків. Крім того, якщо алгоритм це передбачає, видаляється пунктуація, цифри, а також всі символи документа наводяться до єдиного реєстру.

Відповідно до закону Ципфа [7] існує емпірична закономірність розподілу частоти слів природної мови. Якщо всі слова мови (або просто досить довгого тексту) впорядкувати за спаданням частоти їх використання, то частота n-го слова в такому списку виявиться приблизно обернено пропорційною його порядковому номеру n (так званому рангу цього слова). Інакше кажучи, слова, що зустрічаються рідше, мають більше смислове значення. На основі законів Ципфа слова, що зустрічаються найбільш часто (наприклад, вигуки, прийменники, суфікси і т.д.), вважаються шумовими словами (стоп-словами) і не приймаються до розгляду.

Під стеммінгом розуміється виділення значущої частини слова, тобто приведення до єдиного кореня. При цьому відкидаються закінчення і суфікси. Існує безліч варіацій стеммінга [5], але найбільш поширений з них алгоритм Портера [7]. Його реалізації на даний момент існують для всіх широко використовуваних мов.

1.8 Висновки до розділу 1

В ході вивчення предметної області було відзначено, що традиційна процедура підготовки документа до створення відбитка полягає в

конкатенації всього текстового вмісту та подальшій обробці, незважаючи на приналежність тексту до якогось логічного сегменту документа.

Шляхом проведення аналогії з роботою [2], автори якої домоглися деякого поліпшення результатів суміжного завдання інформаційного пошуку шляхом відділення навігаційної частини веб-сторінки від змістовної, і спираючись на показане вище збільшення змістовності відбитків при зменшенні розміру документів було сформульовано припущення: попереднє розбиття тексту на семантичні блоки відповідно до структури веб-документа дозволить поліпшити якість розпізнавання схожих документів.

Таким чином, метою даного дисертаційного дослідження є розробка методу оцінки схожості веб-документів з використанням попереднього розбиття на семантичні блоки і опорою на локальні параметри веб-документів.

Основна задача формулюється як підвищення якості розпізнавання схожих веб-документів шляхом оцінки схожості блоків, складових веб-документа, і застосуванні алгоритмів створення відбитків на основі локальних параметрів документів.

Таким чином, у цій роботі пропонується метод оцінки схожості веб-документів на рівні складових їх блоків, а також алгоритм виділення структурно-семантичних блоків різних типів з веб-документів і дослідження алгоритмів створення стійких відбитків документів на основі їх локальних параметрів. Крім цього, розроблена і представлена структура програмного комплексу, що реалізує розроблені методи і алгоритми і дозволяє перевірити їх ефективність експериментально.

2 ОЦІНКА СХОЖОСТІ ДОКУМЕНТІВ

У цьому розділі описується модель представлення веб-документа і пропонується метод виділення блоків з веб-документів. На їх основі описуються метод виділення блоків з веб-документів і метод оцінки схожості документів шляхом використання інформації про схожість їх блоків.

2.1 Модель представлення веб-документа

HTML (від англ. Hypertext Markup Language – «мова розмітки гіпертексту») – це стандартна мова розмітки документів в Інтернеті. Мова HTML інтерпретується браузером і відображається у вигляді документа, зручного для людини. Стандарт HTML, який є додатком SGML, розробляється і підтримується консорціумом W3C з 1995 року.

Відповідно до стандарту DOM [9] (від англ. Document Object Model – «об’єктна модель документа»), які розробляються і підтримуваним консорціумом W3C з 1998 року, формалізовано опис веб-сторінок у вигляді дерева HTML-тегів (рисунок 2.1). Це дозволяє в автоматичному режимі будувати DOM-дерево для заданого веб-документа.

Як показано в [3], будь-яка класифікація з ієрархічною структурою має деревоподібну структуру, яку графічно можна представити кількома способами: у вигляді дерева, вкладених множин, вкладених дужок або списку з відступами. В даному випадку найбільш цікавими та наочними відображеннями є перші два: дерево дозволяє візуально оцінити складність і глибину дерева, а вкладені множини за умови коректної інтерпретації HTML-тегів дають шукане відображення сторінки в «браузерному» вигляді.

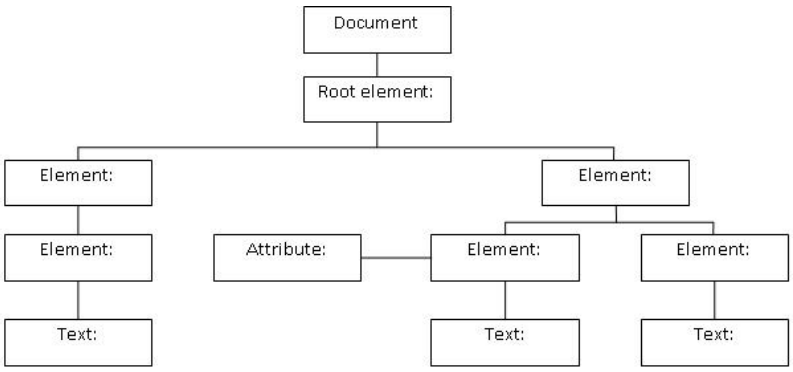

Рисунок 2.1 – Приклад DOM-дерева HTML-сторінки

Для вирішення завдання виділення блоків необхідно здійснити перетворення вихідного веб-документа D_w в множині блоків B_w допомогою функції f_b у формулі 2.1.

$$f_b = D_w \to B_w,\ B_w = \{b_1, b_2, \dots, b_n\}, n > 0. \tag{2.1}$$

Інформаційною структурою I [3] назвемо впорядковану послідовність знаків кінцевої довжини l, представлену у вигляді рядка:

$$I = \{i_1, i_2, \dots, i_l\}. \tag{2.2}$$

Вихідний веб-документ D_w відображається на інформаційну структуру I. Надалі такий рядок будемо також називати вихідним текстом.

Інформаційним вмістом I_f назвемо текстове вміст веб-документа, що безпосередньо відображається браузером на екрані. На відміну від I, I_f не містить HTML-тегів та іншої службової інформації веб-документа, необхідної для коректної інтерпретації сторінки браузером.

Надалі інформаційний зміст будемо також називати контентом.

Для перетворення I в I_f введемо функцію Filt через формулу 2.3:

$$L(I_f) = |I_f|. \tag{2.3}$$

Введемо поняття довжини контенту L, позначимо її через довжину множини I_f у формулі 2.4:

$$L(I_f) = |I_f|. \tag{2.4}$$

Оскільки I_f відповідає I з відкинутою службовою інформацією, очевидно випливає формула 2.5:

$$L(I_f) \leq L(I). \tag{2.5}$$

Стандарт HTML описує уявлення HTML-документів у вигляді DOM-дерева. Враховуючи відповідну специфікацію, інформаційна структура I вазаємнооднозначно відображається на граф T = (S,U), де S = {s} – множина вузлів, U = {u} – множина з'єднуючих дуг, що з'єднують вузли. Таким чином DOM-дерево представляється у вигляді ациклічного графа, що має безліч рівнів $S = \{S^0, S^1, \dots, S^{N-1}, S^N\}$ з коренем у вузліS_i^N Кожен рівень містить упорядковану множину вузлів $S^N = S_k^n$

Кожному вузлу дерева $S_i^n\ (n \leq N)$ поставлена у відповідність інформаційна структура $I(S_i^n)$. Позначимо через T(s) піддерево дерева T з коренем у вузлі s.

Визначимо операцію видалення із заміщенням вузла з графа T, позначимо її символом δ. Операція полягає в наступному:

- у графі T = (S,U) у наміченого для видалення з заміщенням вузла $s \in S$, розташованому на рівні n виділяється множина $S_c = \{s_i\}$вузлів, які є ієрархічними нащадками вузла s і розташованими на рівні n + 1;
- з графа T виділяється вузол s_p, по відношенню до якого видаляється вузол s є ієрархічним нащадком (інцидентним вузлу s і розташованим на рівні n-1);
- з графа T видаляється вузол s і всі інцидентні йому дуги;.
- кожен вузол $s_i \in S_c$ з'єднується дугою з вузлом s_p.

Необхідно відзначити, що дана операція не може бути проведена на кореневому вузлі графа$T: \delta: S^n, n > 1$. Крім того, для подальших міркувань буде корисно виділити той факт, що якщо проводилося видалення вузола s, що не містить контенту $I_f(s) = \emptyset$, то загальна довжина контенту графа $L(I_f(T))$ не змінюється.

2.2 Метод виділення блоків з веб-документа

Приступимо до розробки методу виділення блоків з інформаційним змістом з веб-документа на основі моделі представлення веб-документа, описаної вище.

Суть методу полягає в послідовному виділенні блоків, що представляють собою піддерева вихідного дерева ієрархічної структури HTML-документа, спираючись на значення певної вагової функції вузла.

Блоком b будемо називати множину вузлів S деякого піддерева T_s вихідного дерева T з коренем в певному S_i^n. Метою методу є трансформація документа D_w в множину блоківB_w, що буде реалізацією функції f_b (2.1).

Для оцінки якості пропонується використовувати наступний підхід: кількість отриманих блоків має укладатися в заздалегідь заданий бажаний

інтервал за формулами 2.6 і 2.7:

$$|B_w| \in |n_g^-, n_g^+|, \tag{2.6}$$

$$n_g^- = n_g - r,\ n_g^+ = n_g + r, \tag{2.7}$$

де n_g– бажане число блоків, r – допустиме відхилення.

Послідовність кроків методу буде виглядати наступним чином:

- обробка веб-документа D_w і формування графа $T = (S, U)$;
- видалення вузлів, що не роблять впливу на інформаційний зміст: $\forall s \in S, |I_f(s)| = 0 : \delta(T, s)$;
- розрахунок значення вагової функції W(s) для кожного вузла $s \in S$;
- виділення блоку b в множину B_w шляхом вирізання піддереві T(s) з коренем у вузлі s, що мають максимальне значення вагової функції $Max(W(s))$;
- у разі, якщо інформаційний зміст решти вузлів $|I_f(T)| > 0$, повернення до кроку 3.

Зазначимо, що у випадку, коли потрібно виділяти не інформаційний зміст I_f, а інформаційну структуру I, послідовність кроків методу повинна бути модифікована. Зокрема, крок 2 буде відсутній, оскільки інформаційна структура властива всім вузлам дерева T; крім того в останньому кроці повернення буде проводитися за умови $|I(T)| > 0$, що, очевидно, в даному випадку відповідає умові $|S| > 0$.

Розглянемо кроки методу більш докладно.

Перший крок описувався в попередньому розділі і не вимагає

подальших коментарів. Другий крок служить для мінімізації загального числа вузлів |S| з метою прискорення роботи операції розрахунку ваги, розглянутої далі.

На третьому кроці проводиться розрахунок значення вагової функції $W(s)$ для кожного вузла дерева. Більш докладно процедура побудови необхідної ваговій функції описана в наступному розділі.

Нарешті, четвертий крок методу виділення блоків передбачає трансформацію виділяється піддереві $T(s)$ в інформаційний зміст, що становить блок b у формулі 2.8:

$$b = I_f(T(s)). \tag{2.8}$$

2.3 Метод оцінки схожості блоків

На даному етапі ми маємо множину блоків $B_w = \{b_1, b_2, \dots, b_n\}, n > 0$. Кожен блок $b \in B_w$ містить деякий інформаційний зміст I_f – контент. Необхідно оцінити ступінь схожості контенту блоків між собою, побудувавши функцію $Sim(b_i, b_j)$.

Стандартні методи оцінки схожості документів працюють за допомогою визначення відносини подібності на текстах документів (до яких відносяться і блоки $b \in B_w$).

Зазвичай для цього використовується деяка метрика подібності, що зіставляє двом документам число в інтервалі $[0,1]$, яке характеризує подібність і деякий параметр p – поріг, перевищення якого свідчить про велику схожість документів.

Для цього з документа d, представленого множиною синтаксичних або лексичних ознак Y, вибирається підмножина ознак $y \in Y$, якf утворює

короткий опис документа – його відбиток. В залежності від обраного методу перетворення принципи цього відбору, як і; число відбитків, можуть істотно різнитися.

Опишемо формально роботу методів оцінки подібності об’єктів на основі відбитків. Множина об’єктів Ω допомогою функції створення відбитка f_s відображається в простір R^k (параметр k в даному випадку служить довжиною відбитка) у формулі 2.9:

$$f_s : \Omega \to s = \{0,1\}^k \tag{2.9}$$

Функція f_s при цьому повинна мати такі властивості [8]:

- $f_s(A) \neq f_s(B) \Rightarrow A \neq B$.
- $\Pr\left(f_s(A) = f_s(B) | A \neq B\right) = \frac{1}{2^k}$

Таким чином, виділення під відбиток всього чотирьох байт пам’яті зменшить ймовірність некоректного збігу відбитків через занадто малоу довжину до $1/2^{32}$. У стандартних реалізаціях методів на основі відбитків зазвичай виділяється не менше восьми байт: k = 264.

Відзначимо, що в разі однозначного відображення документа d на єдиний відбиток s результати порівняння відбитків матимуть бінарний характер. Справді, документи будуть вважатися схожими тільки в тому випадку, якщо їх відбитки збігаються. В іншому випадку, ніякого зв’язку між документами виявлено не буде.

В залежності від вимог до якості детектування схожості та особливостей конкретного методу створення відбитків функція f_s може створювати для документа d більше одного відбитка: $f_s(d) = S_d, S_d = \{s_1, s_2, \ldots, s_n\}, n > 0$. В цьому випадку для оцінки ступеня

схожості використовується будь-яка метрика схожості.

Згідно метриці перекриття (1.1) блоки $b \in B_w$ яким відповідають множини відбитків блоків вважаються схожими з нечіткою схожістю, рівною по формулі 2.10:

$$Sim(b_i, b_j) = \frac{|S_{b_i} \cap S_{b_j}|}{|S_{b_i} \cup S_{b_j}|}. \quad (2.10)$$

Дана метрика має такі властивості:

- $Sim(b_i, b_j) \in [0,1]$;
- $Sim(b_i, b_i) = 1$;
- $Sim(b_i, b_j) = Sim(b_j, b_i)$.

Блоки будуть вважатися схожими, якщо чисельне вираження їх схожості вище деякого порога p_b у формулі 2.11:

$$Sim(b_i, b_j) > p_b, p_b \in [0,1]. \quad (2.11)$$

Отримавши метрику схожості пари блоків, можна визначити матрицю попарної схожості блоків у формулі 2.12:

$$M = [m_{ij}], m_{ij} = Sim(b_i, b_j), i, j \in [1, |B|]. \quad (2.12)$$

Матриця M є квадратною, унітрикутною і симетричною:

- $m_{ij} = 1, i = j = \overline{1,n}$;
- $m_{ij} = m_{ji}, i, j = \overline{1,n}$.

Необхідно зазначити, що відповідно до властивості №3, вираз (2.10)

має на увазі загальний ступінь схожості двох блоків, незалежну від порядку їх розгляду. У тому випадку, якщо необхідно отримати ступінь входження блоку b_i в блок b_j, вона буде виражатися по формулі 2.13:

$$Sim_s(b_i, b_j) = \frac{|S_{b_i} \cap S_{b_j}|}{|S_{b_i}|}. \quad (2.13)$$

Властивості метрики (2.13) аналогічні (2.10), за винятком властивості №3.

Аналогічно, на підставі (2.13) визначається матриця входження блоків за формулою 2.14:

$$M_s = [m_{ij}], m_{ij} = Sim_s(b_i, b_j). \quad (2.14)$$

У цьому випадку матриця M_s також є квадратною і унітрикутною, але не симетричною.

2.4 Підходи до формалізації нечіткості знань про схожість документів

Поняття схожих документів, як було показано в розділі 1, саме по собі має на увазі нечіткість. Схожість документів можна вважати абсолютною тільки в тому випадку, якщо вони повністю збігаються між собою. У всіх інших випадках чисельне вираження схожості знаходиться в інтервалі [0,1]. Незалежно від обраної метрики схожості документи вважаються майже-дублікатами, якщо схожість перевищує деякий, наперед заданий поріг.

Крім того, нечіткість виявляється і в ступені достовірності оцінки схожості. Оскільки пряме попарне порівняння документів в умовах великих їхніх обсягів проводити не представляється можливим через недоліки

машинних ресурсів, всі існуючі реалізації методів виявлення схожості документів працюють на основі якоїсь хеш-функції. Досягаючи таким чином значного зменшення обчислювальної складності порівняння, вони жертвують точністю і повнотою, оскільки так чи інакше значна частина документа відкидається в процесі перетворення.

Таким чином, очевидною є потреба висловлювати наявні знання про схожість документів через нечіткі знання.

Основні підходи [11] до моделювання нечіткості можна систематизувати за допомогою таких критеріїв:

- загальний підхід до представлення нечіткості (нечіткі множини, імовірнісні множини тощо);
- область значення функції приналежності (стандартні нечіткі множини, нечіткі числа тощо);
- область значень функції приналежності ([0, 1]-нечіткі множини, [-1, 1]-нечіткі множини, інтервальнозначні нечіткі множини та ін);
- тип відповідності між областю визначення і областю значень функції приналежності (взаємооднозначне, невзаємооднозначне та ін.)

В даному випадку необхідності у використанні розширених трактувань нечіткості не виявилося, тому використовувався математичний апарат стандартних нечітких множин.

На користь використання нечіткої логіки проти теорії ймовірностей свідчать такі міркування. Нечітка логіка описує інший рід невизначеності – не ймовірність деякого результату, як теорія ймовірності, а невизначеність наявного результату. З поняттям схожості документів, розглянутих в даній роботі, це співвідноситься краще.

2.5 Метод оцінки

На даний момент ми маємо сукупність (D, B, S, M), що представляє множину веб-документів, блоків і відбитків відповідно і матрицю схожості

блоків. Є такі відповідності:

- $d \in D$;
- $f_b: d \to B_w, B_w = \{b_1, b_2, \dots, b_n\}, n > 0$;
- $f_s: b \to S_b, S_b = \{s_1, s_2, \dots, s_m\}, m > 0$;
- $M = m_{ij}, m_{ij} = Sim(b_i, b_j)$.

Необхідно побудувати метод, що дозволяє оцінити ступінь подібності документів $d \in D$. Іншими словами, необхідно розробити метрику подібності пари документів $Sim_d(d_i, d_j)$.

Введемо додаткові позначення:

- $n(b) = |S_b|$ – кількість відбитків блоку b;
- $n_d = |B_d|$ – кількість блоків документа d;
- $l(b) = Length(b)$ – довжина блоку b (символів);
- $l(d) = Length(d)$ – довжина документа d (символів);

Очевидні наступні властивості:

- $l(d) = \sum_{i=1}^{n(d)} l(b_i)$;
- $\forall b \in B, 0 < l(b) \leq l(d_b)$;

Звідси зрозуміло, що у випадку, коли $l(d) = l(b_d)$, множина B_d складається з одного елемента, ідентичного d.

Введемо нечітку підмножину $\tilde{b}_i$ множини B, визначену як множина впорядкованих пар у формулах 2.15 і 2.16:

$$\{(b, \mu_{b_i}(b))\}, \forall b \in B, \tag{2.15}$$

$$\mu_{b_i}(b) = Sim(b_i, b), b \in B. \tag{2.16}$$

Визначимо нечітку підмножину $\tilde{b_i}$ для всіх блоків $b \in B, i \leq |B|$. Зробимо об'єднання всіх нечітких підмножин блоків кожного документа у формулі 2.17:

$$\mu_{b_i}(b) = \bigcup_{j=1}^{|B|} \mu_{\tilde{b_j}}(b), b \in B_{d_i}, i \leq |D|. \tag{2.17}$$

Отриманий набір нечітких множин $\mu_{d_i}(b)$ відображає сукупну інформацію про схожість наявних в кожному документі $d \in D$ блоків $b \in B_d$. Застосовуючи різні операції нечітких множин над цим набором, можна отримувати різноманітну інформацію про схожість окремих документів і блоків. Розглянемо основні операції.

Ступінь рівності двох нечітких множинd_i, d_j визначається через операцію еквівалентності [7] у формулі 2.18:

$$A \leftrightarrow B = MIN(MAX(1 - A, B), MAX(1 - B, A)), \tag{2.18}$$

формулою 2.19 такого вигляду:

$$Sim(d_i, d_j) = \bigcap_{b \in B} (\mu_{d_i}(b) \leftrightarrow \mu_{d_j}(b)). \tag{2.19}$$

Вираз 2.19 може використовуватися в якості метрики блокової схожості – він видає значення в інтервалі [0, 1] і в цілому відображає ступінь

збігу блоків документів.

Ступінь включення двох нечітких множин d_i, d_j визначається через операцію імплікації у формулі 2.20:

$$A \to B = MAX(1 - A, B), \tag{2.20}$$

формулою 2.21 такого вигляду:

$$Sim(d_i, d_j) = \bigcap_{b \in B} (\mu_{d_i}(b) \to \mu_{d_j}(b)). \tag{2.21}$$

Ступінь включення відображає в інтервалі $[0,1]$ нечітку величину включення блоків документа d_i в документ d_j.

Слід зазначити, що $Sim(d_i, d_j)$ може бути виражена через $Sim_s(d_i, d_j)$ у формулі 2.22:

$$Sim(d_i, d_j) = MIN\left(Sim_s(d_i, d_j), Sim_s(d_j, d_i)\right). \tag{2.22}$$

Таким чином, ступінь рівності нечітких множин дорівнює мінімальній зі ступенів їх взаємного включення. Доказ цього твердження наведено в [7].

Для отримання множини блоків d_d, що є майже-дублікатами блоків розглянутого документа d (тобто ступінь подібності яких перевищила поріг p) проводяться наступні операції:

- отримання звичайної підмножини p-рівня нечіткої підмножини d_i документа $d: d_p = \{b | \mu_{d_i}(b) \geq p\}, p \in [0,1]$;

- віднімання з отриманої множини d_p блоків, що належать документу $d: d_p = d_p \setminus B_d$.

В якості метрики, що відображає число незпівпалих блоків, можна використовувати узагальнену відстань Хеммінга за формулою 2.23:

$$d(d_i, d_j) = \sum_{k=1}^{|B|} |\mu_{d_i}(b_k) - \mu_{d_j}(b_k)|. \quad (2.23)$$

Очевидно, що $d(d_i, d_j) \in [0, |B|]$. Для зручності роботи отримане значення можна нормалізувати формулою 2.24:

$$d_n(d_i, d_j) = \frac{d(d_i, d_j)}{n}, \quad (2.24)$$

де $n = |B|$. Вираз (2.24) назвемо метрикою відносного блочної відстані.

Відзначимо також, що існує можливість описувати підсумкову міру близькості двох документів також іншими способами. Приймемо в розгляд довжину контенту блоків і документів L, тоді отримаємо формулу 2.25

$$Sim(d_a, d_b) = \sum_{i=1}^{n(d_a)} \sum_{j=1, j \neq i}^{n(d_b)} \frac{Sim(b_i, b_j)(l(b_i) + l(b_j))}{l(d_a) + l(d_b)}. \quad (2.25)$$

На основі стандартної метрики перекриття описаний метод оцінки схожості блоків, що дозволяє отримати нечітке значення схожості двох блоків в інтервалі [0, 1].

3 АЛГОРИТМ ВИДІЛЕННЯ СТРУКТУРНО-СЕМАНТИЧНИХ БЛОКІВ З ВЕБ-ДОКУМЕНТІВ

У цьому розділі на основі описаних раніше методів оцінки схожості документів на рівні складових їх блоків розробляються відповідні алгоритми. Зокрема, детально розглядається алгоритм виділення структурно-семантичних блоків з веб-документів, а також описується процедура побудови дерева детальності. Крім того, вивчається можливість застосування локальних параметрів тексту при створенні стійких відбитків веб-документів і демонструється ряд алгоритмів.

3.1 Алгоритми розбиття веб-сторінок на блоки

Завдання добування інформації з веб-сторінок не втрачає своєї актуальності з моменту появи перших веб-сайтів. В Інтернеті доступна величезна кількість інформації, яка має певну структуру і можливість отримувати дані з таких структур, безсумнівно, корисна. Однак відсутність апріорних знань про структуру зберігання даних, необхідність фільтрації елементів оформлення та інші проблеми роблять це завдання досить складнмим.

У цьому параграфі розглядається задача виділення структурно-семантичних блоків з веб-сторінок. Описано два підходи: лінійного виділення блоків та створення дерева детальності.

3.1.1 Практичне застосування

Задача виділення блоків зі значущою інформацією є різновидом задач аналізу структури HTML-сторінок і співвідноситься з багатьма іншими завданнями [6], [2]:

- трансформація сторінок в вид, зручний для відображення на пристроях з невеликим екраном;

- виділення значущої інформації для людей з обмеженими можливостями, що страждають фізичними вадами;

- створення анотацій (фрагментів) до документів;

- підготовка даних для лінгвістичного аналізу;

- відстеження змін і кешування сторінок.

Пропонований в [7] метод використання автоматичних засобів вилучення інформації (так званих «посередників», wrappers) маючи на увазі відносну регулярність набору документів, тобто наявність у них схожої структури. Показуючи високу якість вилучення при роботі з однотипною колекцією, даний підхід програє в умовах «хаотичних» відвідин веб-сторінок пошуковим роботом.

Крім того, існує ряд методів [2], заснованих на виділенні повторюваних фрагментів веб-сторінок. Вони виходять з припущення про те, що в межах одного веб-сайту сторінки мають повторювані фрагменти в частині навігаційних блоків. В результаті відкидання таких повторюваних блоків залишається значуща, змістовна частина документа. У даній роботі ці методи не розглядаються, оскільки вимагають багатократного проходу по колекції документів для виділення змістовної частини.

3.1.2 Поняття структурно-семантичного блоку

Для подальшого обговорення необхідно визначити сутність поняття семантичного блоку для веб-сторінки. У даній роботі під ним мається на увазі частина вмісту веб-сторінки, яка природним чином виділяється людиною при погляді на неї. Іншими словами, логічні сегменти, на які людина розіб’є сторінку при необхідності, і будуть вважатися семантичними блоками.

Оцінка належності тексту до візуального сегменту, що відображається

браузером сторінки вельми складна, оскільки вимагає попереднього рендеринга і подальшого звірення відображуваного тексту та «сирого» вмісту документа разом з розміткою. Це призводить до необхідності виділяти блоки, керуючись тільки ієрархічною структурою HTML-коду сторінки. Однак блоки, що виділяються таким чином не можуть повною мірою вважатися семантичними, оскільки безпосередньо залежать від структури документа.

Таким чином у даній роботі було сформульовано нове визначення: структурно-семантичним блоком називається по можливості логічно і семантично цілісний сегмент веб-сторінки, побудований з точки зору і з використанням її ієрархічної структури.

3.1.3 Алгоритм виділення блоків

Блок-схема пропонованого алгоритму виділення структурно-семантичних блоків з веб-документів представлена на рисунку 3.1.

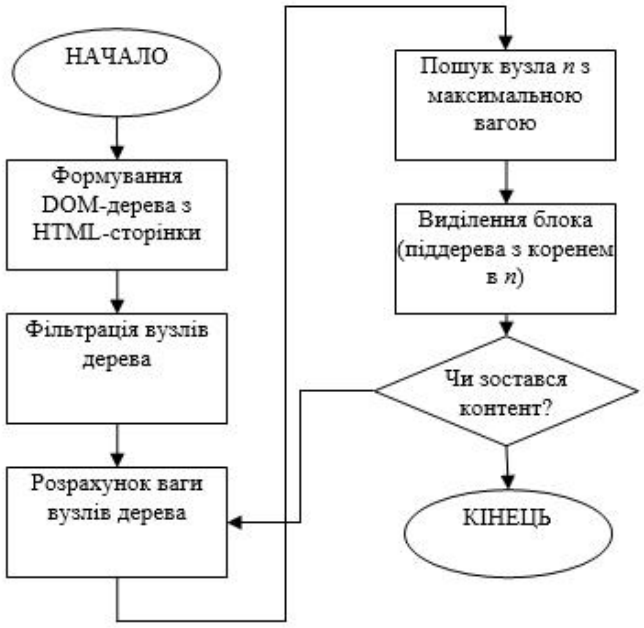

Рисунок 3.1 – Блок-схема алгоритму виділення блоків А1

Крок 1: побудова DOM-дерева G = (V, E) із заданого веб-документа.

Крок 2: фільтрація дерева.

Крок 3: розрахунок ваги w кожного вузла n дерева.

Крок 4: пошук вузла з максимальною вагою Max(w).

Крок 5: виділення структурного блоку шляхом вирізання з дерева G піддереві з коренем в n.

Крок 6: оцінка обсягу тексту в останньому дереві. Якщо більше мінімального порога m – перейти до кроку 3.

Крок 7: виділення вузлів дерева, що залишилися в останній структурний блок.

Результатом виконання алгоритму є набор виділених із заданої веб-сторінки структурних блоків.Розглянемо етапи алгоритму більш докладно.

Відповідно до стандарту DOM формалізовано опис веб-сторінок у вигляді дерева html-тегів (рисунок 3.1). Це дозволяє в автоматичному режимі будувати DOM-дерево для заданого веб-документа за умови, що він написаний у відповідності зі стандартом.

Розглянемо вузли побудованого з веб-сторінки DOM-дерева. Враховуючи наші потреби в розбитті, можна розбити вузли на кілька типів відповідно до специфікації HTML:

- батьківські – знаходяться у вершині дерева (HTML, BODY);

- проміжні – мають і батька, і нащадків:

структурні – необхідні для опису структури документа (TABLE, TR, TD, FORM, DIV, SPAN, P);

візуальні – необхідні для задач візуалізації (A, B, I, STRONG,U);

- кінцеві – не мають нащадків:

- теги (IMG, BR, HR, SCRIPT);
- фрагменти тексту;
- коментарі.

Детальна класифікація HTML-тегів за типами наведена в додатку А.

Підсумкової метою даної задачі розбиття є рознесення тексту

документа текстового типу по блокам, виділеним з його структури. Таким чином, найбільш важливі для нас два типи вузлів: кінцеві вузли, що містять безпосередньо текст документа, а також проміжні вузли структурного типу, що дозволяють оцінити його структуру.

Інші вузли (за винятком батьківських) можуть видалятися цілком, як ті, що не роблять впливу на результат і що ускладнюють розбиття. Насправді, значна частина елементів сторінки не впливає на текст, служачи лише для оформлення сторінки, коментарів та інших завдань. До таких елементів в загальному випадку належать:

- коментарі;
- зайві пробіли, переводи рядків;
- скрипти;
- вузли візуального типу;
- проміжні вузли, не мають контенту у нащадків.

Крім того, залежно від типів і вмісту блоків, які необхідно отримати, набір елементів, що підлягають видаленню в ході спрощення, може різнитися.

Робити видалення проміжних вузлів можна двома способами: з видаленням нащадків і без нього. У першому випадку, що застосовується при видаленні проміжних вузлів, які не мають контенту у нащадків, з дерева вирізається тег разом з піддеревом, для якого він служив коренем. У другому випадку, що застосовується при видаленні проміжних вузлів візуального типу, проводиться стягування вузлів дерева, тобто перепризначення нащадків вузла, що видаляється його предку.

Вихідною передумовою, що пояснює ідею методу виділення блоків з веб-документів, було наступне спостереження: основна частина контенту веб-сторінки міститься, як правило, на нижніх рівнях DOM-дерева. Враховуючи обумовлені стандартом HTML, а також історично складені особливості побудови веб-сторінок, можна стверджувати, що елементи тексту, які знаходяться на нижніх рівнях, структуровані виходячи з логіки

побудови HTML-ієрархії, можуть служити хорошим наближенням до початково необхідним семантичним фрагментами веб-документа.

Таким чином, виникає задача побудови вагової функції вузла s графа T, який адекватно відображає величину інформаційного змісту піддереві вузла s відповідно до глибиною дерева N.

В залежності від вимог до виділених блоків функція обчислення ваги вузла може виглядати по різному. Враховуючи базове завдання даного дослідження (виділення блоків для подальшої генерації на їх основі відбитків документа), основним значущим параметром вузла було вирішено вважати обсяг контенту (тут і далі під контентом мається на увазі текстова інформація, що безпосередньо виводиться на екран в результаті відтворення веб-сторінки браузером) його піддерева.

Будемо вважати, що вага W вузла s прямо пропорційний довжині контенту піддереві цього вузла T(s) у формулі 3.1:

$$W \sim L(I_f(T(s))) \tag{3.1}$$

Також припустимо, що вага W вузла s обернено пропорційний довжині контенту піддереві вузла s_p, що є батьківським по відношенню до вузла s – чим менше зміна, тим відносно більший контент зосереджений в цьому вузлі і його нащадків, і тим більший інтерес для нас має вузол у формулі 3.2:

$$W \sim \frac{1}{L(I_f(T(s_p)))} \tag{3.2}$$

Нарешті, вага вузла пропорційний рівню n, на якому знаходиться вузол, оскільки, виходячи з вищеописаних міркувань, краще виділяти блоки по можливості на нижніх рівнях за формулою 3.3:

$$W \sim n. \tag{3.3}$$

Виходячи з цих міркувань і після проведення низки експериментальних перевірок, вагова функція вузла W(s) була визначена наступним чином згідно з формулою 3.4:

$$W_s = \frac{l * \log(l * k_1)\log(d * k_2 + k_3)}{p + k_4}, \tag{3.4}$$

де $l = L(I_f(T(s)))$ – довжина контенту піддерева з коренем в s, $p = L(I_f(T(s_p)))$ – довжина контенту піддерева з коренем в s_p, $d \in [1, N]$ – глибина вузла (рівень вкладеності), K_{1-4} – коефіцієнти. Введення логарифма пояснюється бажанням зменшити вплив відносно високих «чистих» значень параметрів, які можуть досягати значень порядку 105 для великих веб-документів.

Оскільки в процесі формування DOM-дерева з веб-сторінки виконується видалення вузлів, що не впливають на результат, а так само вузлів з l = 0, вагова функція приймає позитивні значення на всіх вузлах дерева.

З іншого боку, може виникнути завдання класифікації виділяються блоків за типами. Набір типів блоків буде відрізнятися в залежності від вимог, але можна попередньо позначити базовий набір, який логічно випливає з структури та вмісту веб-документа.

Такий набір буде включати в себе наступні типи блоків: текстові, структуровані, посилальні, утилітарні, мультимедійні (більш докладно можливі типи блоків описані в Додатку Б).

Приміром, класифікація блоку як посилального (тобто містить відносно велику кількість посилань на інші сторінки за допомогою тегів типу

<A HREF ...>) дозволяє з високою часткою ймовірності вважати, що цей блок належать до навігаційної частини веб-сторінки. В цьому випадку є розумним введення в вагову функцію W (s) додаткового параметра A, що визначає число посилальних тегів, що зустрілися в піддереві T (s).

Детальний аналіз можливих параметрів функції W(s) залежно від вимог до типів блоків, що виділяються виходить за рамки даної роботи.

Як показано в роботі [7], використання інформації про атрибути тегів не призводить до поліпшення результатів, тому в даній роботі атрибути не враховувалися.

Зазначимо, що алгоритм A1, спочатку проектувався для роботи з HTML-сторінками, в принципі може бути використаний з будь-яким форматом документів, що зберігають текстову інформацію і передбачають ієрархічну структуру. В цілому, для адаптації алгоритму під новий формат необхідно переглянути лише кроки 1-2, які формують дерево на основі документа і відкидають зайві гілки.

Наприклад, для використання даного алгоритму з документами у форматі XML, широко поширеному в Інтернеті і вельми схожому з HTML, по структурі і логіці побудови, необхідно лише модифіковані набір врахованих при обробці вихідного коду документа тегів (крок 1).

3.1.4 Пошук оптимальних значень коефіцієнтів

Задача пошуку оптимальних значень коефіцієнтів ваговій функції (3.4) може вирішуватися по різному [6]. Зокрема, до неї можуть бути застосовані методи ітераційного глобального пошуку.

До таких методів можна віднести генетичні алгоритми, мурашиний алгоритм, а також інші комбіновані глобально-локальні методи пошуку, що володіють пам’яттю.

Для застосування цих методів на ЕОМ потрібно задати максимально точну імітаційну модель, визначити змінні, значення яких потрібно знайти,

області їх визначення, а також цільову функцію рішення від цих змінних.

Генетичні алгоритми [8, 64] відносяться до еволюційних методів пошуку. Вони є результатом узагальнення спостережень за процесом природного відбору в біології, забезпечуючи поступове наближення до екстремуму досліджуваної функції.

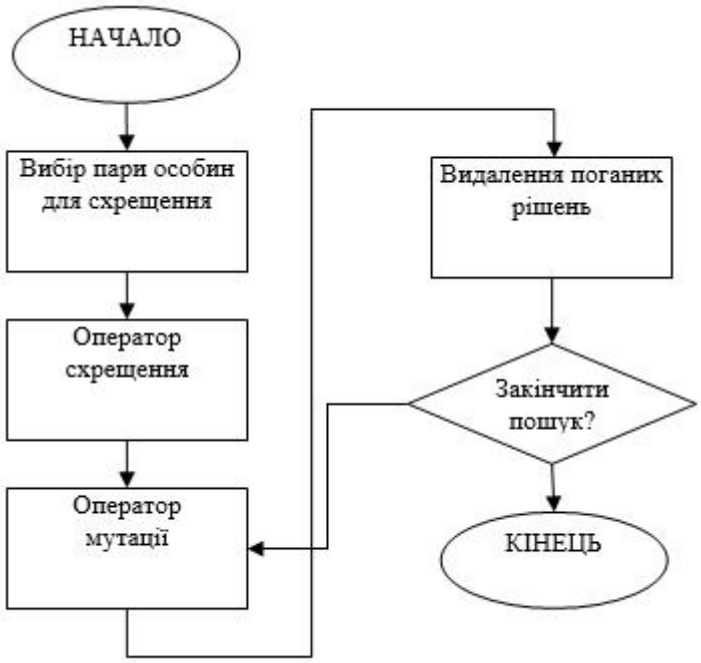

Рисунок 3.2 – Структурна схема простого генетичного алгоритму

У розглянутій задачі пошук оптимальних значень коефіцієнтів вагової функції 3.4 здійснювалася з використанням простого генетичного алгоритму. Його структурна схема приведена на рисунку 3.2. Кожна особина в популяції описувалася вектором коефіцієнтів k(1-4). Цільова (фітнес) функція розраховуються на основі бажаного числа блоків, що отримуються з веб-сторінки, і описувалася наступною формулою 3.5:

$$\text{Fitness} = \begin{cases} \frac{70}{\text{min} - \text{c}} \text{ для c} < min, \\ \frac{70}{\text{c} - \text{max}} \text{ для c} > max, \\ 100 \text{ інакше} \end{cases} \quad (3.5)$$

де c – сумарне число блоків всіх сторінок тестової колекції, отримане в результаті розбиття с функцією ваги, коефіцієнти якої були визначені відповідною хромосомою;

min і max – мінімальне і максимальне бажане число блоків відповідно.

Для пошуку коефіцієнтів використовувалися наступні параметри генетичного алгоритму: розмірність популяції – 30, кількість поколінь – 30. Найсильніше ймовірність залежить від числа особин в популяції. Проте в даному випадку завдання пошуку глобального екстремуму не настільки актуальна, оскільки для виконання основної мети роботи – перевірки припущення про поліпшення якості розпізнавання дублікатів веб-документів при використанні поблочного порівняння, що буде виконуватися шляхом проведення експерименту – точності, що показується алгоритмом з використанням знайдених параметрів достатньо .

Таблиця 3.1 – Значення коефіцієнтів для різноманітного бажаного числа блоків

Число блоков	k_1	k_2	k_3	k_4	fitness	wrong
2-4	0,21	1,08	7,80	1	0,88	5
3-7	0,15	1,93	4,07	1	0,89 ,	4
8-12	0,50	1,94	0,26	1	0,49	11
13-17	25,74	136,90	23,61	-12,7	0,36	13

У таблиці 3.1 наведені значення коефіцієнтів для різного бажаного числа блоків. Стовпець fitness демонструє нормоване на максимально

можливе значення фітнес-функції, стовпець wrong – число сторінок, розбивка на блоки яких не вклалася в заданий інтервал.

3.1.5 Оцінка обчислювальної трудомісткості

Застосування методу виділення блоків з веб-сторінок до реальних задач залежить від його продуктивності, яка визначається його обчислювальною трудомісткістю.

Обчислювальні ресурси, що витрачаються при використанні даного методу, можна розділити на два класи: одноразово необхідні для проведення підготовчої роботи та використовувані для безпосереднього пошуку і виділення блоків.

Підготовча робота полягає в побудові дерева з веб-документа. Це включає в себе парсинг вихідного HTML-коду сторінки і формування ієрархічної структури вузлів. На цьому ж етапі проводиться фільтрація необхідних вузлів.

Після цього відбувається основна частина роботи – пошук і виділення блоків. Аналіз алгоритму A1 показує, що максимальне число виконань блоків основного циклу (кроки 3-6) дорівнює кількості вузлів дерева N.

Число операцій з розрахунку ваг вузлів дорівнює добутку числа вузлів і кількості параметрів вагової функції 3.4 – 3N.

Операція з розрахунку довжини контенту вузла вимагає злиття всіх вузлів піддерева з коренем в заданому вузлі – розмір N_L змінюється від 1 до N.

Таким чином, в гіршому випадку трудомісткість вилучення блоків з веб-документа пропорційна кубу числа його вузлів, що розраховується за формулою 3.6:

$$O(N * 3N * N_L) = O(N^3). \quad (3.6)$$

З іншого боку, вагова функція 3.4, як правило, забезпечує виділення фіксованого числа блоків, свідомо меншого, ніж загальне число вузлів (в іншому випадку втрачається сенс вилучення агрегуючих блоків). Враховуючи це, трудомісткість вилучення блоків пропорційна квадрату числа вузлів веб-документа за формулою 3.7:

$$O(3N * N_L) = O(N^2). \tag{3.7}$$

Така трудомісткість є досить високою в разі обробки сторінок з великим числом вузлів. Однак, враховуючи закон степеневого розподілу в застосуванні до веб-сторінок (рисунок 1.4), а також позиціонуючи даний алгоритм як засіб перевірки вихідного припущення даної роботи, така обчислювальна складність є прийнятною. Виходячи з цього, подальшу оптимізацію описаного алгоритму було вирішено залишити за кордоном даної роботи.

3.2 Алгоритми створення єдиного відбитка на основі локальних параметрів документа

Ще одним потребуючим розгляду завданням є завдання формування сталого відбитка документа.

Раніше було розглянуто ряд класичних методів детектування схожих документів. Відмінною рисою більшості з них є створення кількох відбитків одного документа. У лексичних методах це зазвичай досягається рандомізацією набору термів, що описують документ, в синтаксичних – вибором різних ділянок документа для аналізу.

Наявність безлічі відбитків одного документа, безумовно, збільшує якість порівняння та надає додаткові можливості по оцінці рівня подібності документів. З іншого боку, коли мова йде про мільйони документів, значну роль грає швидкість застосовуваних алгоритмів. В цьому випадку логічним

способом прискорення порівняння стає перехід від множини відбитків, що ідентифікують документ, до одного «універсального».

Це призводить до помітного «огрублення» результатів, так як ступінь схожості визначити вже не вдається. Однак виграш у швидкості і генерації відбитків, і виконання порівняння може бути пріоритетніше.

Важливою властивістю методів, що ставлять у відповідність кожному документу єдиний відбиток, є «бінарність» їх суджень. Справді, документи будуть вважатися схожими тільки в тому випадку, якщо їх відбитки збігаються. В іншому випадку, ніякого зв'язку між документами виявлено не буде. При зменшенні величини порога, вище якого документи вважаються дублікатами, кількість документів, очевидно, збільшується. Таким чином, неможливо зберегти високі значення точності і повноти для всього діапазону: досягнення високої повноти викликає зниження точності, і навпаки.

Виходячи з того, що створення відбитка документа, розбитого на множину блоків, само по собі має на увазі наявність множини відбитків не меншого розміру $|S_w| \geq |B_w|$, було вирішено провести перевірку ефективності не тільки на основі методу шинглювання, але і з використанням якого-небудь алгоритму створення єдиного відбитка.

В роботі [3] автором був запропонований метод побудови відбитка документа, що будується на основі певного набору статистичних параметрів документа, підібраних виходячи з міркувань стійкості до певних форм модифікацій документа.

Основна ідея полягала в припущенні, що такі параметри можуть послужити своєрідною альтернативою текстовому змісту документа. Приміром, кількість крапок, ком і заголовних букв в тексті дозволяє приблизно зафіксувати кількість речень у тексті; загальна довжина тексту в символах з виключенням пробілів і стоп-слів дає загальну оцінку обсягу; оцінка кількості і відносини різночастотних слів може служити деякою заміною семантичному аналізу документа і т.д.

Пропонується наступний метод побудови відбитка: відбиток документа

будується на основі певного набору статистичних параметрів документа, а саме: кількість певних символів в тексті (точок, пробілів, спецсимволів); загальна довжина тексту (з виключенням ряду елементів і без); кількість і співвідношення високо-, середньо- і низькочастотних слів в документі; середня довжина речення і слова, і т.п. Набір параметрів підбирається, виходячи з міркувань стійкості до різних видів модифікацій документа.

Таким чином, необхідно провести експериментальну перевірку можливості використання локальних параметрів тексту документа для створення стійкого відбитка документа.

Очікувані переваги такі:

- побудова локального відбитка дозволяє ефективно обробляти великі обсяги документів, оскільки не вимагає попереднього пробігу за наявними для побудови глобальними колекціями;

- пропонований метод досить простий і може бути реалізований з високою ефективністю, що часто є критичним в умовах великих масивів документів;

- незалежність від мови, якою написаний документ, дозволяє обробляти будь-які веб-документи без попередньої настройки.

3.2.1 Виділені параметри веб-документів

Частково питання побудови локального відбитка документа розглядалися в [9, 79]. Найбільш цікавим в плані високих показників точності і повноти алгоритм LongSent [9], що працює дуже просто: відбиток створюється шляхом зчеплення двох найдовших речень документа в порядку убування їх довжини.

Виділимо параметри і характеристики текстів, які будуть брати участь в побудовах відбитків:

а) з точки зору статистичних параметрів:

- послідовності;

- загальна кількість;
- середнє.

б) з точки зору елементів тексту:

- абзаци;
- речення;
- слова (терми);
- символи.

в) з точки зору розмітки:

- зі збереженням HTML-розмітки;
- зі збереженням текстової розмітки;
- зі збереженням спецсимволів;
- відфільтровані.

Алгоритми створення відбитків, що розглядаються далі, складені на основі комбінацій вищенаведених параметрів з урахуванням відсутності подібних алгоритмів в роботах інших авторів.

3.2.2 Опис набору тестів

Відповідно до ідеєю дослідження, був розроблений ряд тестів, у кожному з яких перевірялася ефективність оригінального алгоритму створення відбитків (за винятком тестів № 8-9, узятих з інших робіт для порівняння).

Тест №l: кількість спецсимволів і розділових знаків.

Підраховувалося число входжень у текст документа d кожного елемента з набору p, що включав в себе наступні символи: .,–:;!?() і символ пробілу.

Документ представлявся у вигляді вектора розмірністю 11 елементів, компонентами якого було число входжень кожного символу в документ.

$$s = (c_1, c_2, \ldots, c_{11}), \text{де } c_i = \text{count}(d, p_i).$$

Тест №2: кількість слів різних довжин.

Підраховувалася число входжень слів різних довжин в текст документа (тут і далі під словом мається на увазі безперервна алфавітно-цифрова послідовність). Документ представлявся у вигляді вектора розмірністю, рівно. довжині найдовшого слова документа L. Компонентами вектора був рядок виду: «[довжина слова] – [число входжень]».

$$s = (c_1, c_2, \ldots, c_L), \text{де } c_i = d_i + \text{Count}(d, d_i), L = \text{Max}(\text{Length}(d_i))$$

Тест № 3: кількість і середня довжина слів і речень.

Підраховувалася середня довжина терма і речення в документі, а також загальна кількість тих і інших. Відбиток складався шляхом конкатенації отриманих значень рядком виду: «[середня довжина слова] – [середня довжина речення] – [число слів] – [загальне число речень]».

Тест № 4: популярні слова.

Підраховувалося число повторень кожного слова в тексті документа. Документ розбивався на терми і представлявся у вигляді вектора, розмірність якого дорівнює розміру словника документа n. Компонентами вектора був рядок виду: «[слово] – [число входжень]».

$$s = (c_1, c_2, \ldots, c_n), \text{де } c_i = d_i + \text{Count}(d, d_i)$$

Використовуваний в даному експерименті алгоритм має віддалену схожість з алгоритмом I-Match, проте є два істотні відмінності: використовуються дані виключно з оброблюваного документа, без залучення зовнішніх джерел даних у вигляді глобальної колекції термів; а також враховується частота терма в документі.

Тест № 5: послідовність довжин слів.

Документ розбивався на терми, після чого проводилася конкатенація

довжин всієї послідовності термів. Відбиток будувався на основі вектора.

Тест № 6 агрегатний.

Сукупний тест, що збирає всі попередні тести в один. Відбиток збирався конкатенацією відбитків тестів 1-5.

Тест № 7: послідовності спецсимволів.

З тексту документа віддалялися буквено-цифрові символи, залишаючи спецсимволи, пробіли і переводи рядків.

Тест № 8: два найдовших речення.

З тексту документа виділялися два найдовших речення та зчіплювалися один з одним. Тест проводився для порівняння результатів з аналогічним тестом з [9], що показав там найкращі результати.

Тест № 9: хеш тексту.

Над текстом документа обчислювалася хеш-функція MD5. Тест служить для надання базових результатів визначення дублікатів.

3.2.3 Експериментальна база

Для виконання дослідження було використано набір даних. Ця колекція включає близько 750 тисяч сторінок.

Дані про дублікати цього набору даних, надані разом з ним, складалися на основі функції Левенштейна. Були відомі пари документів, коефіцієнт подібності яких, після видалення тегів, становив не менше 85% і складаються не менше, ніж з 20 слів. Відповідно, результати експериментів порівнювалися саме з цим «ідеальним» списком дублікатів.

Для видалення HTML-розмітки використовувалася відкрита Java-бібліотека HTML Parser, що надає дану функціональність. Частка документів з некоректно розпізнаною html-розміткою становила не більше 4%, причиною чого в більшості випадків служили логічні помилки в початковому тексті документа. В кінцевому підсумку кожним документом ставилося у відповідність 128-бітний MD5-хеш, отриманий з згенерованого відбитка.

Відповідно до [10], MD5 дає значно менше число помилкових спрацьовувань, ніж 40-бітний хеш Рабина, використаний в ранніх дослідженнях, при збереженні досить високої швидкості генерації.

Програмна реалізація описаних методів включала в себе реалізацію наступних етапів.

Підготовка тестових даних: парсинг заголовків та індексів документів з наданого набору даних у власну базу даних; заповнення відомостей про відомі пари документів-дублікатів.

Набір експериментів, для кожного з яких виконувалися:

- генерація хешів для кожного документа;
- видалення HTML-розмітки (якщо потрібно);
- побудова відбитка відповідно до конкретного алгоритмом тесту;
- хешування.
- аналіз результатів;
- розрахунок ступеня відповідності результатів різних експериментів один одному.

3.2.4 Обговорення результатів

У зв’язку з низкою обмежень набору апаратного забезпечення, на якому проводилися експерименти, тести проводилися двома наборами.

Перший набір включав в себе тести 1-6. Результати тестів першого набору наведено в додатку Г. Графічна інтерпретація результатів тестів приведена на рисунках 3.3 і 3.4.

Виняток стоп-слів (для тих тестів, в яких це б мало смисл) не дало помітного результату: відхилення від базових результатів не перевищували 1% (як в позитивну, так і в негативну сторону).

Були також виконані модифікації тесту № 3, що використовують окремо слова і речення (середню довжину і загальна кількість). Вони показали на 20-25% гіршу точність при збереженні тієї ж повноти, що і в

об’єднаному тесті.

Крім того, вивчені результати ряду модифікацій тесту № 4:

- обрізка 20-35% з кожного боку відсортованого за кількістю входжень списку слів – не зробила помітного впливу (менше 1%);

- використання в якості одиниць не слів, а речень – не справила помітного впливу (менше 1%);

- відсортований за кількістю входжень список слів, нормалізований і округлений до першого знака після коми – дав виграш в точності в розмірі 1-2%.

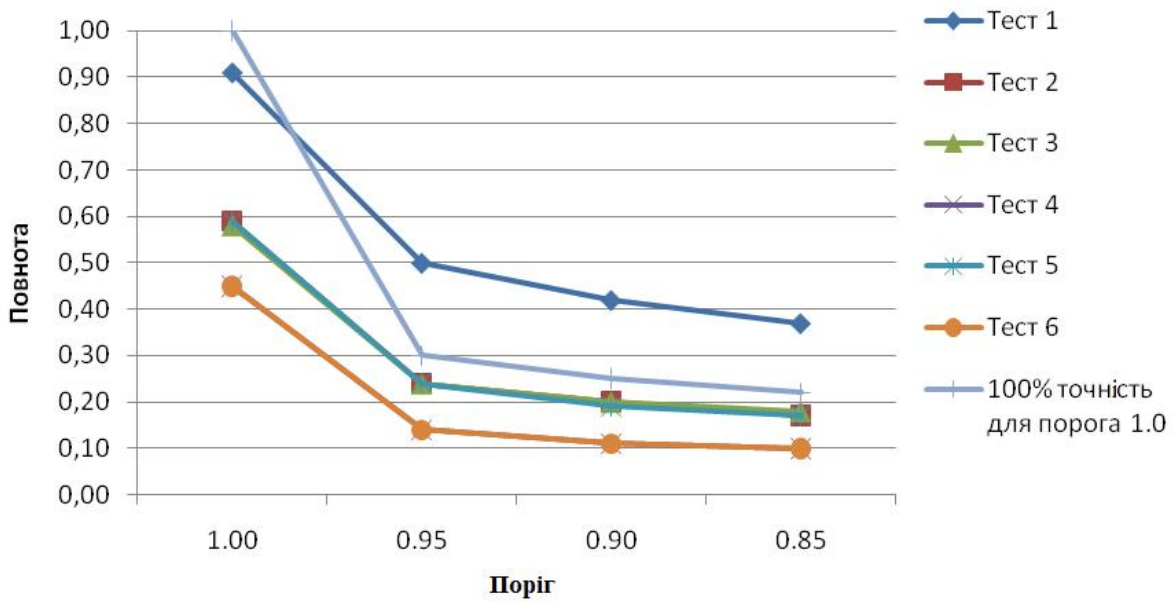

Рисунок 3.3 – Графік повноти для тестів першого набору

Розроблено алгоритм виділення структурно-семантичних блоків з веб-документів, що спирається на текстовий вміст і HTML-структуру сторінок. Передбачена можливість налаштування числових характеристик одержуваних блоків шляхом зміни коефіцієнтів вагової функції. Експерименти показали високу якість виділення за умови вимоги низького (2-7) числа блоків, і задовільну якість – при виділенні високого (8-12 і більше) числа блоків. Крім того, описана процедура побудови дерева детальності, що дозволяє отримати ієрархічну структуру для деталізації

блоків на потрібному рівні.

Досліджено ряд алгоритмів створення відбитків на основі локальних параметрів тексту документа. В цілому результати тестів можна вважати задовільними. Вони дозволяють підтвердити припущення про прийнятність ефективності використання таких параметрів при створенні відбитка, однак спектр застосування отриманих відбитків обмежується низкою умов. Їх застосування резонно в разі низьких (щодо класичних методів порівняння) вимог до якості розпізнавання, переважному числі коротких документів і високих вимогах до продуктивності алгоритму створення відбитків.

Невисокі показники на низьких порогах схожості документів (у порівнянні, наприклад, з [9]) є наслідком «бінарності» суджень, обумовлених єдиним відбитком, і хороших показниках на високих порогах. «Пом’якшення» відбитка можна домогтися збільшення кількості виявлених пар дублікатів (як в тесті № 4), що дозволить, змістити ефективну кордон на більш низький рівень.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДУ ОЦІНКИ СХОЖОСТІ ВЕБ-ДОКУМЕНТІВ

У цьому розділі описана програмна реалізація програми, що реалізує метод оцінки схожості веб-документів на рівні блоків, що містяться в них, алгоритм виділення структурно-семантичних блоків веб-документів, і що дозволяє експериментально оцінити їх ефективність на довільно підібраних тестових колекціях веб-документів.

4.1 Структура програмного забезпечення

Відповідно до етапів рішення вихідної задачі знаходження схожих веб-документів можна виділити наступні структурні елементи необхідного програмного забезпечення:

- користувальницький графічний інтерфейс для оцінки якості розбиття веб-сторінки на блоки;
- користувальницький графічний інтерфейс для аналізу якості детектування схожих веб-документів;
- програмна реалізація алгоритму розбивки веб-сторінок на структурно-семантичні блоки;
- програмна реалізація методу оцінки схожості документів на основі їх блоків;
- модуль розрахунку статистики для оцінки результатів експерименту.

Перераховані елементи програмної системи не вимагають обов’язкового розподілу по окремим модулям програми і можуть бути реалізовані в рамках єдиного програмного модуля, який володіє графічним інтерфейсом користувача. На основі необхідних програмних функцій та обраної структури програмної системи було вирішено реалізувати програму «Модуль оцінки подібності веб-документів на рівні блоків», іменоване надалі

додатком або програмою. Виходячи з означених вище елементів програми була побудована діаграма взаємодії користувача з додатком, наведена на рисунку 4.1.

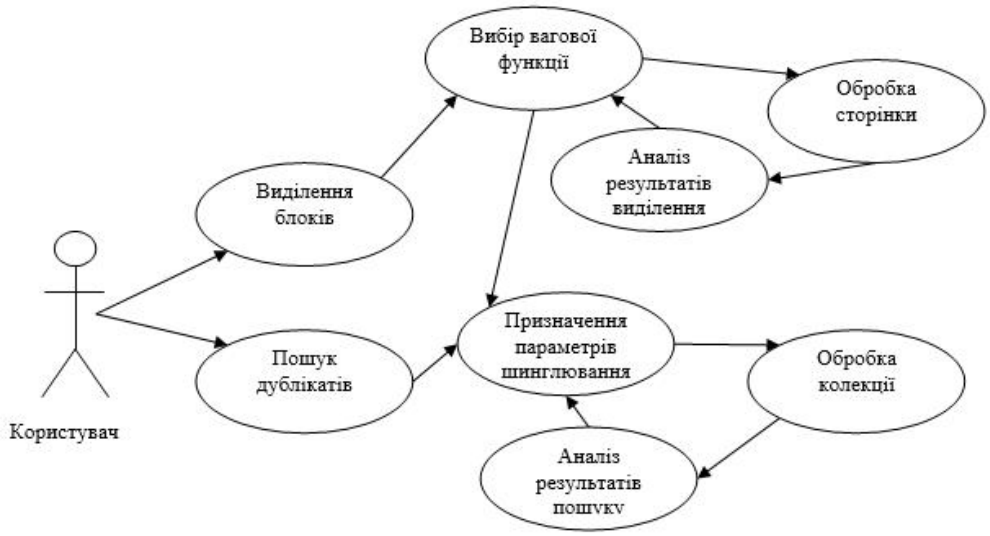

Рисунок 4.1 – Use-case діаграма програми

Крім того, до програмного забезпечення висувався ряд додаткових вимог:

- повинна бути можливість модифікації програми при розширенні діапазону завдань;

- процедура роботи з додатком повинна бути якомога простішою та уніфікованою;

- витрати часу на підготовку вихідних даних і обробку результатів повинні бути мінімізовані.

Можливість розширення діапазону задач, що вирішуються додатком, досягається шляхом використання парадигми ООП-програмування. В результаті отримано великий набір програмних класів, що допускають просте модифікування і широкі можливості по повторному використанню програмного коду.

Витрати часу на підготовку вихідних даних зводяться до мінімуму,

оскільки потрібно лише надати додатку директорію з аналізованими на схожість веб-документами (або безпосередньо вказати веб-документ для проведення розбиття на блоки). Результати, отримані після обробки вихідних даних, виводяться у вигляді, підготовленому для простої обробки користувачем, а також передбачають графічну інтерпретацію.

Підсумкова структурна схема програми приведена на рисунку 4.2.

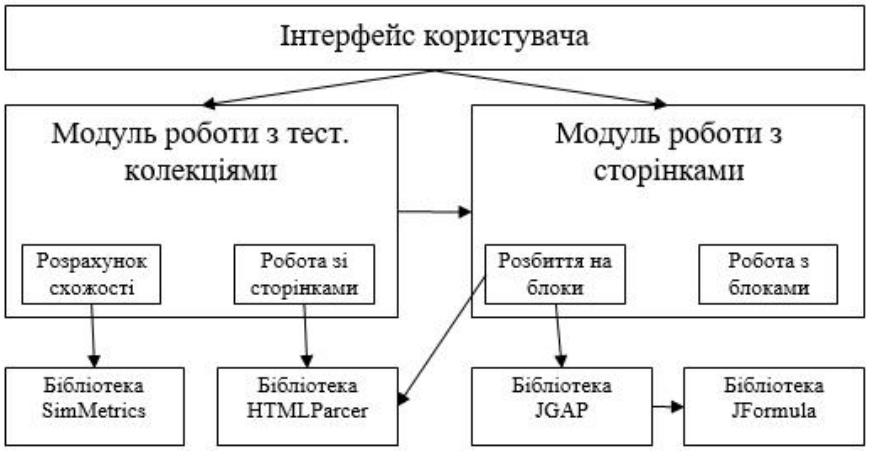

Рисунок 4.2 – Структурна схема програми

4.2 Програмна платформа

В якості програмної платформи для реалізації програми була вибрана програмна платформа Java. Вибір визначався наступними перевагами даної платформи:

- кросплатформеність, що досягається трансляцією програмного коду в байт-код, що виконується на віртуальній Java-машині;
- наявністю і широким вибором різних бібліотек, у тому числі-для задач вилучення даних;
- відкритістю і безкоштовністю специфікацій;
- наявністю великої кількості засобів розробки.

4.3 Програмна реалізація

Отримана в результаті проектування додатку діаграма класів наведена на рисунку 4.3. В процесі проектування набору класів ми виходили з реальних сутностей, використовуваних додатком. Основну множину класів можна розбити на три категорії.

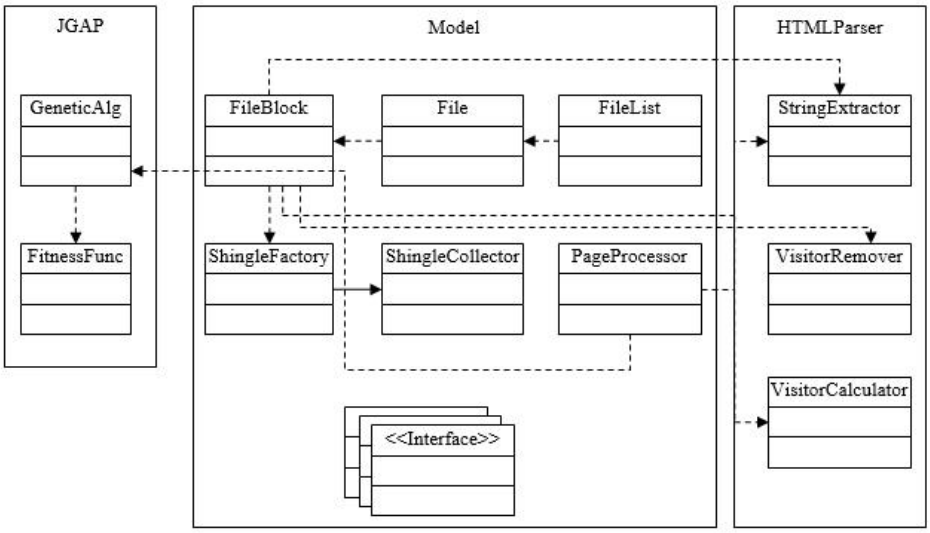

Рисунок 4.3 – Діаграма класів додатку

Перша категорія класів відповідає за створення і обробку відбитків колекцій веб-документів. Ієрархічна структура «колекція веб-документів – веб-документ – блоки веб-документа – відбитки блоку» знайшла своє відображення в класах FileList, File і FileBlock відповідно (відбитки блоку зберігалися в масиві, що належить потрібного екземпляру класу FileBlock). Статичний клас ShingleFactory надає методи генерації відбитка потрібним алгоритмом на основі поданого текстового блоку.

Клас ShingleCollector надає зручні інтерфейси для пошуку однакових відбитків, а також з’ясування приналежності відбитків до блоків і навпаки.

Друга категорія класів призначена для розбиття веб-документів на блоки. Для цього був розроблений ряд класів, які успадковують базовий

функціонал класів бібліотеки HTMLParser. Клас VisitorRemover дозволяє видаляти з дерева тегів вузли, що задаються умовою; клас VisitorCalculator служить для розрахунку ваг вузлів дерева; клас ShingleExtractor перетворює дерево тегів до текстового вигляду, виділяючи безпосередній контент веб-документа. Статичний керуючий клас PageProcessor здійснює обробку заданої веб-сторінки.

Третя категорія класів включає в себе службові класи, які потрібні для роботи з графічним інтерфейсом користувача, а також використання додаткових сторонніх бібліотек, описаних нижче.

Для розрахунку ступеня близькості веб-сторінок різними стандартними метриками використовувалася бібліотека Simmetrics.

В якості інтерпретатора введених користувачем формульних виразів застосовувалася бібліотека JFormula.

Для програмної реалізації генетичних алгоритмів в задачі виділення блоків використовувалася відкрита бібліотека JGAP.

Обчислювальний алгоритм рішення задачі розбиття веб-документа на блоки реалізований на основі алгоритму А1, описаного в розділі 3.

Розроблене в результаті додаток володіє графічним призначеним для користувача інтерфейсом, виконує функції обчислювального рішення задачі, а також володіє функціями імпорту вихідних даних і виведення результатів рішення.

В якості вхідних даних використовуються веб-сторінки, представлені у вигляді HTML-файлів на довільному носії інформації. Кілька веб-сторінок може бути представлено у вигляді тестової колекції. Крім цього, вхідними параметрами є настройки алгоритмів.

Розроблене програмне забезпечення призначене для використання на IBM – сумісних ЕОМ з операцірнной системою Windows XP і пізніших версій. Для функціонування програми необхідно, щоб в операційній системі була встановлена вільно розповсюджувана бібліотека Java SE 1.5 або вище. Програма не має виражених вимог до обсягу встановленого ОЗУ: обсяг

використовуваної програмою пам’яті залежить від розмірності розв’язуваної задачі.

4.4 Графічний інтерфейс

Графічний інтерфейс програми представляє собою набір вікон, кожне з яких служить для вирішення конкретної частини загальної задачі. На верхньому рівні відповідно до use case-діаграми (рисунок 4.1) були виділені три основних вікна: «Виділення блоків», «Пошук дублікатів» і «Звіти». Нижче ці вікна представлені більш детально.

4.4.1 Вікно роботи з алгоритмом розбиття на блоки

Вікно, призначене для вивчення методу розбиття веб-сторінок на блоки, представлено на рисунку 4.4. Воно містить кілька блоків: конфігураційний блок, блок перегляду оброблюваної веб-сторінки, табличний блок списку вузлів і блок дерева детальності.

Блок перегляду оброблюваної веб-сторінки дозволяє працювати з двома її представленнями – у вигляді оригінальної сторінки і фільтрованої (з вилученням ряду вузлів). Крім того, кожне з уявлень можна розглянути у трьох варіантах: «браузерному» вигляді (аналогічно відображенню веб-сторінки в звичайному браузері), в HTML-коді, а також у вигляді DOM-дерева.

Останній варіант наведено на рисунку 4.5. Для зручності ручного аналізу візуалізуються дерево автоматично показує значення відповідних кожному вузлу параметрів вагової функції (глибини вузла, ваги контенту, кількості посилань і т.п.), а також виводить для кожного вузла конкатенований рядок контента його піддерева. Сам контент відображається в дереві жирним шрифтом.

Частина інтерфейсу даного вікна представлена на рисунку 4.6.

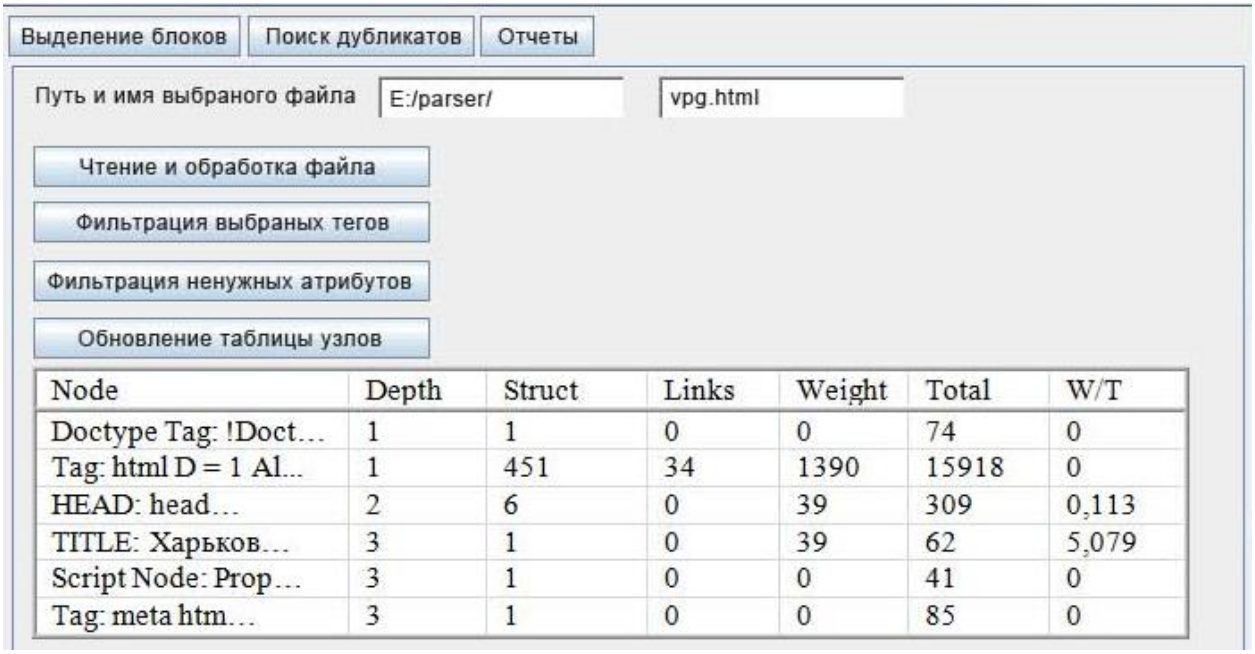

Рисунок 4.4 – Загальний вигляд вікна роботи з алгоритмом розбиття на блоки

Конфігураційний блок, розташований у верхній частині вікна, надає наступний набір функцій:

- завдання імені і шляху до оброблюваної веб-сторінці;
- видалення обраних HTML-тегів (проводиться шляхом вирізання вузлів з подальшим зрощенням їх батьків і нащадків);
- видалення вузлів, які не представляють інтересу (наприклад, вузлів з нульовим вагою контенту);
- виділення блоків (з подальшим заповненням дерева детальності).

Табличний блок списку вузлів демонструє повний список вузлів DOM-дерева та їх атрибутів в табличному вигляді. Клацанням миші на довільному заголовку стовпця здійснюється сортування.

ВИСНОВКИ

В результаті проведеної роботи щодо практичної реалізації запропонованих алгоритмів і методів був реалізовано додаток «Модуль оцінки подібності веб-документів на рівні блоків», що дозволяє на практиці оцінити їх придатність на довільно підібраних тестових колекціях веб-документів.

З його допомогою були проведені експерименти, описані в попередніх розділах, показана працездатність і ефективність методу оцінки подібності веб-документів на рівні блоків, а також підібрані коефіцієнти вагової функції алгоритму A1 для різних вимог до числа блоків.

Результати проведеного експерименту дозволили підтвердити вихідне припущення про підвищення ефективності розпізнавання схожих документів при використанні інформації про схожість складових їх блоків. Показана висока ефективність в плані виділення навігаційної частини сторінок, оскільки подібні навігаційні блоки практично завжди виділяються розробниками сторінок в окрему структурну одиницю DOM-дерева. Показано збільшення якості розпізнавання дублікатів при використанні попереднього розбиття їх на структурно-семантичні блоки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бортманов Б.А. Пошук необхідної інформації на веб-сторінках в задачах інформаційного пошуку [Текст]. / Б.А. Бортманов, К.В. Вершинкова, Л.Т. Добриня; Матеріали конференції Інтернет-Математика 2010, 2010. – 160 с.

2. Ганшик М.Т. Проектування приоритетних ланцюгів на базі територіального розподілу промислових систем [Текст]. дис. к.т.н. /. М.Т. Ганшик, 2009. – 125 с.

3. Леонченко П.Т. Отримання інформаційної статистики при пошуку схожих документів [Текст]. / П.Т. Леонченко Інтернет-Математика 2011, 2011 – 250 с.

4. Комаров Б.А. Нечітке моделювання в середовищі MATLAB та fuzzyTECH. [Текст]. / Б.А. Комаров, Н.Т. Лазарев, 2007. – 283 с.

5. Бойко В.Л. Росширення схожості пошукових пар [Текст]. / Матеріали 16-ї міжнародної конференції Інтернет, 2007. – 560 с.