ДИПЛОМНА РОБОТА
БАКАЛАВРА

ДРБ. ІП - 66.00.00.000 ПЗ

Група ІП-19-1К

Шекета Тарас

2022

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

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

УДК 004.4

ДИПЛОМНА РОБОТА БАКАЛАВРА

Тема: " Імплементація схем видобування даних в ВЕБ-просторі "

(назва згідно з наказом ректора)

спеціальність – 121 "Інженерія програмного забезпечення"

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДРБ.ІП - 66.00.00.000 ПЗ

(позначення)

Консультанти:

Студент

ІП-19-1К_____/Т.В.Шекета/
(шифр групи) (підпис) (дата) (розшифровка підпису)

Керівник роботи

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

Допускається до захисту
Завідувач кафедри

з питань нормоконтролю

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

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

2022
(рік)

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерія програмного забезпечення

Освітньо-кваліфікаційний рівень бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

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

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ БАКАЛАВРА СТУДЕНТОВІ

Шекеті Тарасу Володимировичу

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) “ Імплементація схем видобування даних в ВЕБ-просторі”

керівник проекту (роботи) Піх Володимир Ярославович, доцент, к.т.н.
затверджені наказом закладу вищої освіти від **“** ___ ” __травня__ 2022 р. № _____

2. Строк подання студентом проекту (роботи) 20 червня 2022 р.

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

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

1. Аналіз предметної області видобування даних з веб-ресурсів
2. Аналіз існуючих моделей та рішень в схемах видобування даних
3. Побудова програмної системи видобування даних у web та її оцінка
4. Проведення імітаційного моделювання
5. Пропонована архітектура та алгоритм видобування даних

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

1. Зразок початкової сторінки (рис. 1.1)
2. Зразок структурованої сторінки (рис. 1.2)
3. Зразок багаторівневої сторінки (рис. 1.3)
4. Зразок вихідного коду HTML (рис. 1.4)
5. Посилання на CSS stylr за допомогою атрибута Class (рис. 1.5)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання **25 травня 2022 р.**

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Аналіз предметної області видобування даних з веб-ресурсів	26.05.2022	виконано
2	Аналіз існуючих моделей та рішень в схемах видобування даних	29.05.2022	виконано
3	Побудова програмної системи видобування даних у WEB та її оцінка	30.05.2022	виконано
4	Проведення імітаційного моделювання	04.06.2022	виконано
5	Пропонована архітектура та алгоритм видобування даних	10.06.2022	виконано
6	Оформлення пояснювальної записки дипломної роботи завідувачем кафедри	20.06.2022	виконано

Студент – дипломник _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

В дипломній роботі використовуються автоматичні екстрактори даних (обгортки), такі як системи Веб-майнінгу як методи вилучення даних, засновані на аналізі вихідного коду HTML веб-сторінки в дерево об’єктної моделі документа, проходячи через модель для виявлення шаблонів для розпізнавання та вилучення різних типів веб-даних. Виконано виявлення потенційно корисних і раніше невідомих знань із вмісту гетерогенного веб-сайту електронної комерції вимагатиме складного завдання пошуку схеми веб-документів з різних веб-сторінок, вилучення цільової інформації та виконання інтеграції даних веб-контенту, створення їхнього віртуального або фізичного сховища даних та видобутку з нього. Пропонується побудувати систему яка використовує веб-структуру та підходи до аналізу вмісту на HTML-коді дерева документів, щоб спростити та зробити процес вилучення системних даних легше розширюваним.

КЛЮЧОВІ СЛОВА: ВИДОБУВАННЯ ДАНИХ, ДЕРЕВО РІШЕНЬ, АНАЛІЗ ВЕБ-КОНТЕНТУ, АВТОМАТИЧНЕ ВИЛУЧЕННЯ ВЕБ-ДАНИХ, ІНТЕГРАЦІЯ, ПОШУК, СХОВИЩЕ ДАНИХ.

ANNOTATION

The thesis uses automatic data extractors (wrappers) such as Web mining systems as data extraction methods based on the analysis of the HTML source code of a web page in the object tree of the model model passing through the model to identify patterns for recognition and derivation of various types of web data. Performed discovery of useful and previously unknown knowledge from the content of a heterogeneous e-commerce website will require the complex task of finding schemas of web documents from different web pages, extracting target information and performing data integration of web content, creating their virtual or physical data warehouse and mining . with him. It is proposed to build a system that uses web framework and content analysis approaches on the HTML code of the document tree to simplify and make the system data extraction process more easily extensible.

KEYWORDS: DATA MINING, DECISION TREE, WEB CONTENT ANALYSIS, AUTOMATED WEB DATA EXTRACTION, INTEGRATION, SEARCH, DATA WAREHOUSE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 7
ВСТУП 8

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ВИДОБУВАННЯ ДАНИХ З ВЕБ-РЕСУРСІВ 9
1.1. Сутність веб-майнінгу 9
1.2. Типи веб-сторінок для веб-майнінгу 11
1.3. Опис переваг сторінок списку 15
1.4. Побудова дерева DOM 16
1.5. Синтаксичний аналізатор Xml і Xpath 18
1.6. Структура очікуваних результатів 19

2. АНАЛІЗ ІСНУЮЧИХ МОДЕЛЕЙ ТА РІШЕНЬ В СХЕМАХ ВИДОБУВАННЯ ДАНИХ 21
2.1 Граматичні підходи видобування даних 21
2.2 Підходи на основі процесів порівняння 36
2.3. Процес видобування інформації з веб-сторінок 42
2.4. Підходи на основі інтерпретації 45

3. ПОБУДОВА ПРОГРАМНОЇ СИСТЕМИ ВИДОБУВАННЯ ДАНИХ У WEB ТА ЇЇ ОЦІНКА 51
3.1 Вирішені проблеми видобування даних 51
3.2 Представлення проблемної області 52
3.3 Пропонована архітектура та алгоритм видобування даних 55
3.4. Емпіричні оцінки пропонованого рішення 73
3.5. Результати імітаційного експерименту 73

3.6. Порівняння пропонованого алгоритму з WOM..... 75

ВИСНОВКИ 78

СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА 79

БІБЛІОГРАФІЧНА ДОВІДКА

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ІМ – імітаційне моделювання

ТЗ – Технічне завдання

БД – База даних

ПЗ - Програмне забезпечення

ШІ – штучний інтелект

ВСТУП

Веб-майнінг – це застосування методів аналізу даних до вмісту, структури та використання веб-ресурсів. Таким чином, це «нетривіальний процес визначення дійсних, раніше невідомих і потенційно корисних шаблонів» у величезній кількості цих веб-даних, шаблонів, які описують їх у стислому для людини керованому порядку величини. Як і інші програми для аналізу даних, Web Mining може отримати результат від заданої структури даних (як у таблицях бази даних), але його також можна застосувати до напівструктурованих або неструктурованих даних, як-от текст довільної форми. Це означає, що веб-майнінг є неоціненною допомогою в трансформації від зрозумілого людині вмісту до зрозумілої машині семантики.

Всесвітня мережа є найбільшим і найбільш широко використовуваним джерелом інформації, до якого легко отримати доступ. Важливо забезпечити інструменти для ефективного відкриття знань. Він містить велику кількість веб-сторінок, які постійно оновлюються, а деякі також видаляються. Дані неоднорідні за своєю природою, у мережі є кілька типів даних, таких як текст, зображення, відео та аудіо.

Інтернет став одним із важливих засобів для бізнесу. Електронна комерція стала найбільш потенційною сферою бізнесу. Величезний обсяг продукції продається в Інтернеті, де кожен постачальник має свій власний веб-сайт, щоб продемонструвати свою продукцію. Ми визначили необхідність порівняльного агента для таких веб-сторінок продуктів, який не лише порівнює поточні дані, але й історичні. Такі агенти вже існують у літературі, але жоден з них не є повністю автоматичним, навіть якщо деякі з них є автоматичними, кожен з них має свої недоліки.

1 . АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ВИДОБУВАННЯ ДАНИХ З ВЕБ-РЕСУРСІВ

1.1. Сутність веб-майнінгу

Веб-майнінг можна стверджувати як застосування методів аналізу даних в Інтернеті. Це виконується для виявлення невідомих знань, які існують у веб-даних. Також для надання деяких послуг з доданою вартістю користувачам.

Містячи великі обсяги даних і їх універсальність, багато дослідників цікавлять виконання кількох методів видобутку для виявлення знань.

Веб-майнінг поділяється на 3 області, як показано нижче

1) Майнінг веб-структур

2) Майнінг використання веб-сайтів

3) Майнінг веб-контенту.

Видобуток структурованих гіперпосилань у мережі називається Web Structure Mining. Розробка веб-структур включає аналіз структури веб-документу, аналіз деревоподібної структури структур сторінок для опису використання тегів HTML або XML.

Ця область дослідження також включає вивчення структури всесвітньої павутини в цілому, як-от соціальні мережі, аналіз цитування тощо. Розглядаючи веб як орієнтований графік, аналіз внутрішніх і вихідних посилань на сторінку є ключем до пошукових систем. Традиційні методи аналізу даних не виконують аналіз посилань, оскільки вони зберігаються як реляційні бази даних, структура посилань не існує. Структуру веб-сайту можна отримати, аналізуючи структуру посилань між сторінками. Це також включає категоризацію веб-сторінок та виявлення спільнот в Інтернеті.

Майнінг використання Інтернету фокусується на техніках, які можуть представляти поведінку користувача під час взаємодії користувача з Інтернетом. Такі дані зберігаються на веб-серверах і у файлах журналів. Ця інформація може бути отримана для визначення інтересів користувачів і подальшого створення

рекомендацій відповідно до. Але проблема виникає в ідентифікації окремого користувача та його сеансів.

Є 3 основні завдання для виконання аналізу використання веб-сайтів, попередньої обробки, виявлення шаблонів та аналізу шаблонів. Дані можуть бути вбудовані в веб-сторінку будь-хто без обмежень, ця гнучкість поступається місцем неповним і відповідним даним, що ускладнює завдання виявлення шаблонів. Тому перед виконанням будь-якого завдання необхідно попередньо обробити веб-дані. Виявлення шаблонів виконується на даних про використання Інтернету за допомогою кількох методів машинного навчання та статистичних методів, розпізнавання шаблонів разом із аналізом даних. Деякі з основних підходів включають аналіз правил асоціацій, кластеризацію, класифікацію, послідовні шаблони та моделювання залежностей. Такі виявлені шаблони можна проаналізувати на предмет цікавих шаблонів, використовуючи деякі вже існуючі форми, такі як sql і olap, залежно від даних.

Видобуток веб-контенту:

Видобуток фактичного вмісту веб-сторінки для отримання знань називається аналізом веб-контенту. Ця область веб-майнінгу включає кілька методів аналізу даних. Дані в традиційному аналізі даних зазвичай доступні в реляційних таблицях даних, але дані в Інтернеті не є однорідними, і вони не існують в одному сховищі, вони поширені у великих масштабах на численних серверах по всьому світу. Завдання аналізу веб-контенту полягає в зборі відповідної інформації зі всесвітньої мережі.

Загальні проблеми в розробці веб-контенту:

1) Веб-дані є неоднорідними за своєю природою, у мережі існує кілька
2) видів даних, таких як текст, зображення, аудіо та відео.

2) Веб-дані є динамічними, щоразу, коли зараз виконується дороге завдання з майнінгу веб-даних, або потім дані в Інтернеті оновлюються, і в

результаті те саме завдання потрібно виконувати знову. Цей сценарій погіршується, коли дані часто оновлюються, на жаль, це поширений сценарій в Інтернеті.

1.2. Типи веб-сторінок для веб-майнінгу

Дані всесвітньої павутини багатьма дослідниками поділяють на 3 типи. Неструктуровані дані, напівструктуровані та структуровані. Оскільки Інтернет не є суворим, будь-хто може представляти свій вміст у будь-який спосіб. Однак на основі домену, якому належить веб-сайт/веб-сторінка, він має подібну структуру з іншими веб-сторінками, що належать до того ж домену. Наприклад, усі веб-сторінки, які належать до колонок новин, показуватимуть свій вміст читачам у подібному форматі. Тип, до якого належить веб-сторінка, має значення, оскільки обгортки, створені для вилучення вмісту веб-сторінки, дуже залежать від типу веб-сторінки та її структури. Тут ми обговорюємо різні типи веб-сторінок.

Неструктурована веб-сторінка.

Неструктурований тип веб-сторінок звертається до сторінок, які не відповідають жодній структурі, також називаються вільними текстовими документами. Ці типи веб-сторінок публікують свій вміст, наприклад текст, зображення, мультимедіа, безкоштовно. Прикладом цих документів є блоги новин та персональні блоги тощо. Автори блогів розміщують свій вміст відповідно до свого бажання, і не обов’язково дотримуватись структури.

На таких веб-сторінках не можна застосувати вилучення веб-даних на основі шаблонів, і ці сторінки зазвичай представлені як пакет слів або фраз, які можна видобути за допомогою кількох методів аналізу даних.

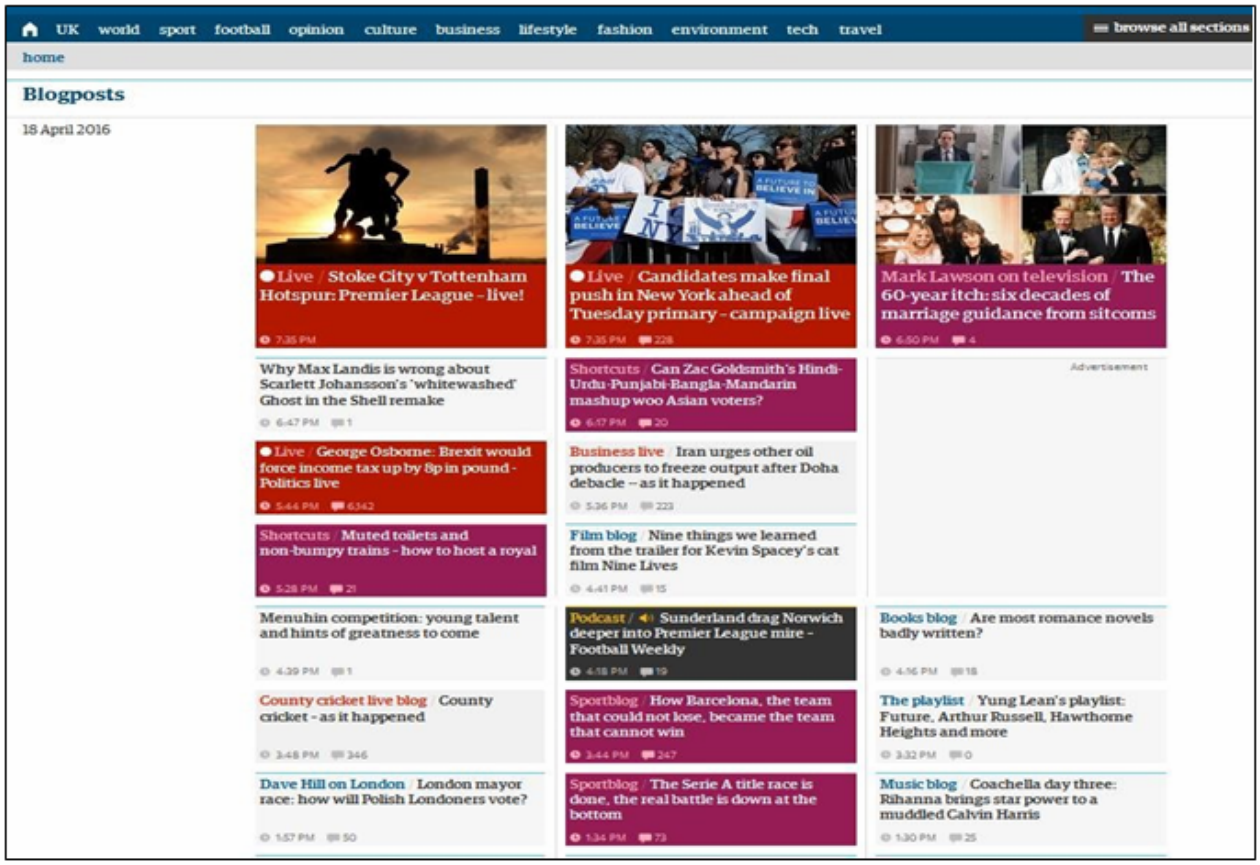


Рисунок 1.1 - Зразок початкової сторінки

Напівструктуровані/структуровані веб-сторінки.

Напівструктуровані/структуровані веб-сторінки – це тип сторінок, які відповідають визначеній структурі, яка досягається завдяки структурованому джерелу даних, такому як база даних або таблиця. Поки дані на веб-сторінці публікуються, вони матимуть формат і вирівнюються. Прикладом структурованої веб-сторінки є сторінка з багатим продуктом на веб-сайті електронної комерції, як-от bestbuy.ca, оскільки вся інформація, така як відомості про продукт, по суті, зберігається в базі даних, яка публікується на веб-сторінці за суворим шаблоном. Такі дані можна витягти за допомогою обгорток, створивши правила вилучення вручну або автоматично на основі цільових даних. Знову в структурованій веб-сторінці є дві класифікації, сторінка списку та сторінка деталей. Які формуються завдяки типу шаблону, який використовується для їх створення.

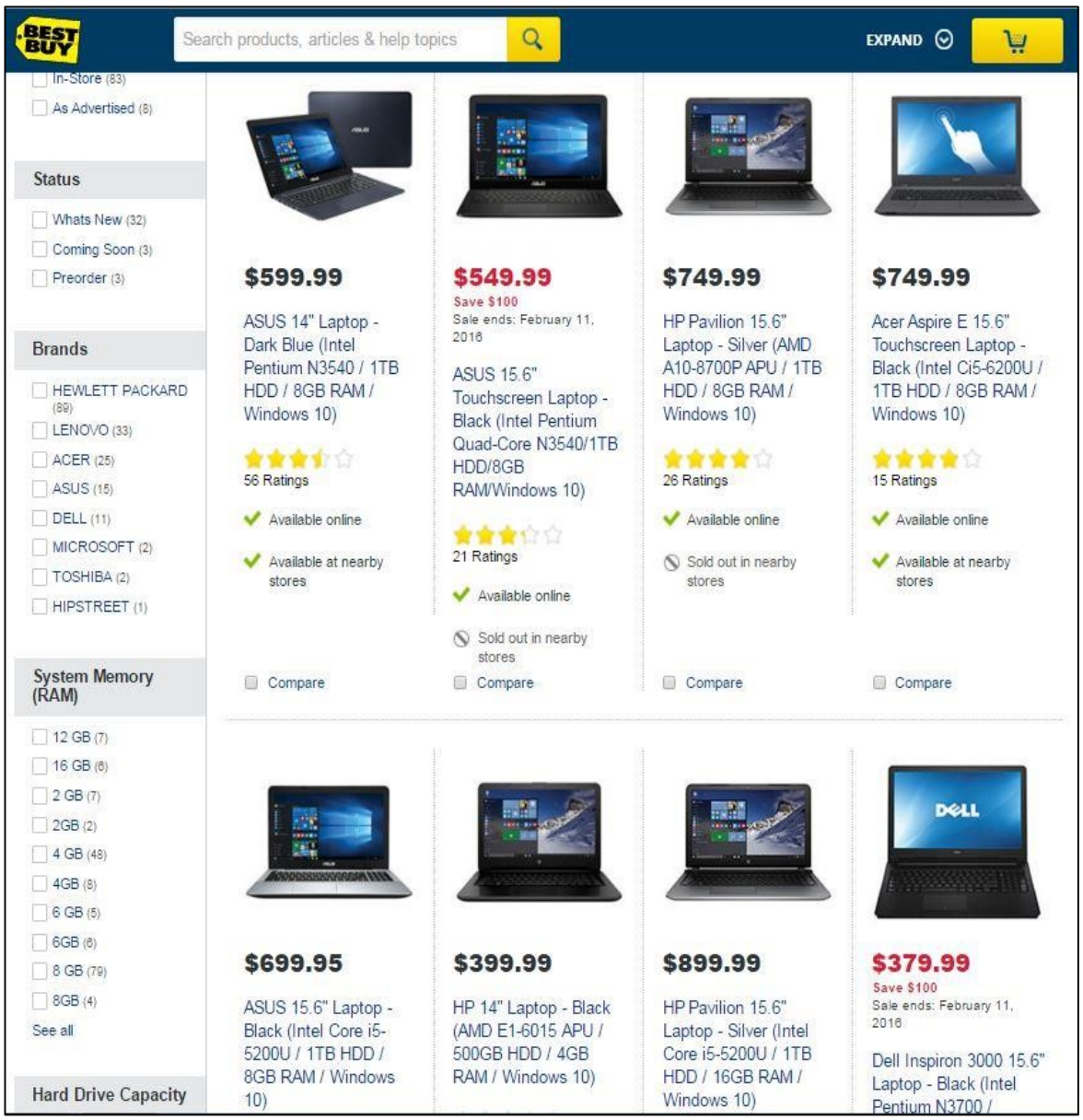


Рисунок 1.2 - Зразок структурованої сторінки

Веб-сторінка списку

Цей тип веб-сторінки складається з кількох структурованих записів даних. Наприклад, веб-сайти електронної комерції відображають усі продукти, що належать до однієї категорії, у списку або у форматі сітки на своїй веб-сторінці. Враховуючи об’єктно-орієнтовану модель даних, яка розглядає один продукт як об’єкт, веб-сторінка зі списком матиме набір об’єктів, які належать до одного типу з набором атрибутів, з якими всі об’єкти можуть бути пов’язані чи ні. На малюнку зображена веб-сторінка з bestbuy.ca, це веб-сторінка

продукту, яка являє собою сторінку типу списку, що містить 8 об’єктів, які відображаються з кількома атрибутами, такими як ціна, марка, розмір, колір, процесор, жорсткий диск, оперативна пам’ять та операційна система. Також об’єкти 2, 8 мають атрибутну знижку, якої не має жоден з інших об’єктів на сторінці. Це метадані продукту, які можна зберігати в сховищі даних на основі історичних запитів та порівняльного аналізу. Ця сторінка також містить інші дані, такі як заголовок, навігаційні дані, рекламні оголошення (наразі не видно) та нижній колонтитул (наразі не видно), усі цю інформацію слід ігнорувати, а обгортка має витягувати лише цільові об’єкти. Цей малюнок є прикладом нашої вхідної веб-сторінки для нашої системи WOM_S, яка виконує вилучення даних.



Рисунок 1.3 - Зразок багаторівневої сторінки

Веб-сторінка Detail

Детальна веб-сторінка відображає лише один об’єкт з усіма його атрибутами. На малюнку показана детальна сторінка з веб-сайту електронної комерції chaptersindigo.ca. Об’єкт є книгою, а на сторінці з детальною інформацією відображаються всі атрибути цього об’єкта, такі як назва книги, ціна, формат, розміри, опубліковано , мова тощо. Все це можна витягти для збереження в сховище даних для послуги з доданою вартістю.

1.3. Опис переваг сторінок списку

Ми вибрали сторінку зі списком замість сторінки з детальною інформацією з таких причин:

1. Сторінки зі списком містять найважливіші атрибути об'єкта з точки зору клієнта, тоді як сторінки з детальною інформацією містять усі атрибути об'єкта. Нашою основною метою є розробка системи, яка може виконувати порівняльний аналіз і історичні запити як послугу.

2. Наше спостереження полягає в тому, що детальна сторінка містить більше шуму, ніж сторінка списку, враховуючи рекомендації та рекламу. Оскільки ми працюємо над усуненням шумних даних у нашому сховищі даних, ми віддаємо перевагу копіювати менш шумні веб-сторінки.

3. Витягування даних об’єкта та його атрибутів з веб-сторінки, а потім перехід на іншу сторінку, щоб витягти інший об’єкт та його атрибути, вимагає великих ресурсів, коли всі об’єкти належать до одного типу і ми можемо витягти кілька об’єктів з однієї сторінка в типі списку.

Перед створенням будь-якої обгортки для вилучення даних на веб-сторінці розуміння вихідного коду html допомагає створити кращі. HTML означає Hyper Text Markup Language, це мова, яка являє собою набір тегів розмітки, які допомагають у створенні веб-сторінки. Документи HTML описуються за допомогою кінцевого набору тегів HTML, де кожен тег має значення та мету використання, як правило, для відображення вмісту документа

на основі вимог користувача. Відповідальність за обробку HTML-документа та відображення вмісту несе веб-браузер. Типовий html-файл починається з оголошення його <!DOCTYPE>, що допомагає браузеру правильно відображати веб-сторінку. Оголошуючи doctype у HTML, тобто <!DOCTYPE HTML>, він розуміє, що файл є файлом html, і обробляє теги для належного відображення вмісту. Початок html-коду ініціюється тегом<HTML>, а кінець - </HTML>, отже, теги html об'єднані в пари, але також є кілька одиночних тегів. Деякі з тегів html включають <BODY>, <HEAD>, щоб браузер міг зрозуміти, що складається з розділу заголовка і що міститься в тілі. Інші теги допомагають зрозуміти, як має відображатися вміст < FONT>, <COLOR>, < H1>, <H2>, <IMG>, <A>,
 тощо, вони називаються вбудованими тегами. Такі теги, як <DIV>, <TABLE>, <OL>, <FORM>, <LI>, <TR>, <TD> допомагають у групуванні, організації та розділенні веб-сторінки, також звані тегами рівня блоків відповідно до w3consortium. На рисунку показано зразок вихідного коду HTML і те, як він буде відтворюватися браузером у вигляді веб-сторінки.

```
<!Doctype html>
<html>
  <head>
     <title>
sample</title>
   </head>
   <body>
     <p>this is a </p>
      <h1> Sample page
</h1>
    </body>
</html>
```

```
Це зразок сторінки
```

Рисунок 1.4 - Зразок вихідного коду HTML

1.4. Побудова дерева DOM

Dom означає об'єктну модель документа, вона стала необхідним кроком для багатьох алгоритмів вилучення даних. Dom дерево будується з використанням вихідного html коду веб-сторінки. Теги та їх ієрархія формують батьківські дочірні відносини, пояснюючи, як організовано вміст на сторінці. Це розуміння дозволяє дослідникам створювати обгортки для вилучення даних. Дерева DOM можна побудувати за допомогою двох методів.

1. Використання лише тегів
2. Використання як тегів, так і візуальних підказок

Використання лише тегів:

Більшість тегів html зустрічаються в парах з відкритим і закритим тегом. У таких парах можуть бути інші пари тегів, що дають початок вкладеній структурі. Дерево DOM для сторінки HTML створюється, розглядаючи кожну пару тегів як вузол.

HTML не є суворою мовою, тому можуть зустрічатися відсутні або неповні теги, тому потрібно правильно вставляти теги, щоб отримати правильне дерево DOM. Це викликає завдання попередньої обробки, яке називається очищенням даних. Далі створюється дерево DOM з чистого html-файлу.

Іншим методом є опора на візуальну інформацію (розташування тегів на екрані), щоб зробити висновок про структурні відносини між тегами для побудови дерева DOM. Вкладені прямокутники використовуються для інтерпретації структури тегу. Це виконується шляхом знаходження 4 меж прямокутника кожного елемента html шляхом виклику механізму візуалізації браузера. Перевірка вмісту виконується, щоб побачити, чи один прямокутник міститься в іншому, це інтерпретує вкладену структуру тегів HTML.

Каскадні таблиці стилів допомагають веб-розробникам розмістити інформацію на веб-сторінці. Він складається з інформації про те, як елементи html мають відображатися на веб-сторінці. Можна створити каскадну таблицю стилів для визначення стилів для веб-сторінок, включаючи дизайн, макет і варіанти відображення. Усі веб-сайти B2C використовують зовнішній css для

пошуку своїх веб-сторінок. Зовнішній css містить кілька стилів, кожен із яких ідентифікується унікальною назвою класу. Атрибут class посилається в тег html, якому має бути виділений бажаний стиль. На малюнку наведено приклад, зовнішній css та його використання у вихідному коді html. На малюнку показано зразок файлу css з двома стилями, позначеними двома класами prodlook і prodprice. Перший стиль prodlook описує, що вміст будь-якого блоку, який використовує цей клас, має встановлювати жирність шрифту, а вміст має підтримувати ліве поле в 20 пунктів. У той час як другий стиль prodprice описує, що вмісту в будь-якому блоці, який посилається на цей клас, має бути призначений шрифт sans_serif із червоним кольором і зеленим фоном. На другому малюнку показано їх використання в html-коді за допомогою атрибута class.

```
div.prodlook{
font-weight:bold;
margin-left:30px}
div.prodprice{
font: sans-serif;
color:red;
background:green}
```

```
<div
class=”prodlook”>…….</div>
<div
class=”prodprice”>…..</div>
```

Рисунок 1.5 - Посилання на CSS stylr за допомогою атрибута Class

1.5. Синтаксичний аналізатор Xml і Xpath

Подібно до HTML, XML також є мовою розмітки та є рекомендацією w3c. У той час як HTML має заздалегідь визначений набір тегів, xml дозволяє користувачам визначати власні описи, завдяки чому він є не тільки машиночитаним, але й читаним людиною. XML призначений для зберігання та транспортування даних і використовується виключно в IT-системах. Основна відмінність між HTML і XML полягає в тому, що перший призначений для відображення даних, а пізній — для перенесення даних. XPath (мова XML Path)

— це мова для пошуку інформації в документі XML. XPath використовується для навігації між елементами та атрибутами в документі XML. Він використовує вирази шляху для навігації в документах XML. Крім того, він містить бібліотеку стандартних функцій. XPath використовує вирази шляху для вибору вузлів або наборів вузлів у XML-документі. Ці вирази шляху дуже схожі на вирази, які ви бачите під час роботи з традиційною файловою системою комп'ютера. Деякі з прикладів виразів: /-> вибирає з кореневого вузла//-> Вибирає вузли в документі з поточного вузла, які відповідають виділенню незалежно від того, де вони знаходяться. .-> Вибирає поточний вузол , ..-> Вибирає батьківський вузол для поточного вузла, @->Вибирає атрибути. Він також має підстановочні знаки, підстановочні знаки XPath можна використовувати для вибору невідомих вузлів XML.* ->Збігається з будь-яким вузлом елемента, @* -> Збігається з будь-яким вузлом атрибутів і node()-Збігається з будь-яким вузлом будь-якого типу. Він надає деякі методи для отримання інформації з документа xml, наприклад, метод xpath evaluate() для повернення конкретних даних із файлу xml, які відповідають виразу xpath. Він може приймати як вхідний документ xml і повертає його вузли як список вузлів. getlength(), щоб отримати кількість вузлів, що відповідають виразу шляху в xml.

Ми використали всі ці функціональні можливості в нашій системі, щоб значно полегшити вилучення даних.

1.6. Структура очікуваних результатів

1) Ми пропонуємо замінити використання NFA на кроці 4 системи WOM на алгоритм частого пошуку структури (FSfinder), який використовує як веб-контент, так і структурний аналіз. FSfinder використовує синтаксичний аналізатор Java xpath для підсумовування частоти кожного зустрічальності тегів у веб-коді html, зберігаючи лише теги, які відповідають певному мінімальному появі (наприклад, 3), потім для кожного тегу він використовує дерево DOM веб-

сторінки, щоб знайти блок тегу та отримати найбільш часто використовуваний блок. Цей найчастіше використовуваний блок потім позначається як блок даних, а його область даних/ім’я таблиці витягується разом із його схемою.

2) Схему можна динамічно виявляти для будь-якої веб-сторінки за допомогою нашої техніки, тоді як із WOM, щоб додати та дізнатися про нову або реструктуризовану веб-сторінку B2C, NFAs потрібно оновити, щоб вона могла розпізнавати нові функції, яких раніше не було. в його структурі. Наша запропонована система використовує відповідність регулярних виразів у аналізаторі Java xpath за допомогою таких методів, як compile(), evaluate(), які виявлять найпоширеніший тег рівня структурного блоку (наприклад,< divclass = “ ″ >) у дереві Dom.

3) Наш алгоритм не дотримується жодних жорстких методів відповідності, як-от вирівнювання дерева (наприклад, розпізнавання зон і блоків дерева DOM із спостереженнями серії 1 і 2) або NFA, як це зроблено в . Замість цього ми підсумовуємо та спостерігаємо за появою тегів, які створюють подібну структуру блоків.

4) WOM_Simple розширюється і має потенціал для масштабування до великих веб-сайтів через його просту й ефективну техніку.

5) Він дуже автоматизований і не вимагає ручних зусиль (наприклад, опис зразків навчальних сторінок B2C) у процесі вилучення. Нам потрібно лише дати системі адресу веб-сторінки сторінки продукту, і вона виявить схему бази даних.

2. АНАЛІЗ ІСНУЮЧИХ МОДЕЛЕЙ ТА РІШЕНЬ В СХЕМАХ ВИДОБУВАННЯ ДАНИХ

У цьому розділі ми представляємо відповідну роботу в літературі з вилучення веб-даних. У цій області було проведено ряд робіт. Тут ми представляємо найбільш значущі роботи, виконані в області вилучення даних. Дослідники використовували кілька методів аналізу даних для веб-даних для вилучення інформації, з роками вилучення веб-даних покращилося від ручного до автоматичного, використовуючи кілька методів аналізу даних. Ми класифікували існуючу роботу на основі використаної наукової техніки, і таким чином ми маємо підходи до вилучення веб-даних на основі порівняння, інтерпретації та граматики. Перша категорія систем створює свої обгортки на основі порівнянь між новою веб-сторінкою та шаблоном. Для порівняння веб-сторінок або блоків даних дослідники використовували методи зіставлення рядків, дерев і вирівнювання кількох послідовностей. У другій групі систем, на відміну від інших дослідників, автори запропонували використовувати візуальну інформацію на веб-сторінці замість базового html-коду. Для вилучення веб-даних використовувалася така інформація, як координати блоків даних на веб-сторінці та дерева Dom на основі інтерпретації. У третій категорії систем використовується граматичний підхід для побудови їх обгорток. Вони спостерігали за закономірностями та використовували регулярні вирази для ідентифікації та подальшого вилучення даних.

2.1 Граматичні підходи видобування даних

Системи вилучення даних, які виконували вилучення даних, виводячи граматику з html-коду веб-сторінки. Загальну структуру тегів, яку використовує веб-сторінка, можна розкрити за допомогою регулярних виразів, вирівнювання

рядків, які в подальшому використовуються для створення правил для вилучення.

IEPAD: вилучення інформації на основі виявлення шаблону

IEPAD — це напівкерована система навчання, яка витягує дані з Інтернету без використання деяких ручних методів, таких як маркування сторінок зразків. Він підкреслює використання повторюваних шаблонів, які трапляються на веб-сторінках, щоб представити цільові дані у вигляді структури, яка називається деревами PAT. Крім того, для осмислення всіх екземплярів записів використовується вирівнювання кількох послідовностей.

Основний внесок включає:

1) Ідентифікує цільовий запис даних без будь-якого втручання людини, досліджуючи той факт, що записи даних виникають у вигляді повторюваних шаблонів у веб-документі, а дані переміщуються на веб-сторінки з базових баз даних за допомогою загального шаблону.

2) Використовує структуру даних, що називається деревами PAT, двійковим суфіксним деревом, щоб виявити повторювані шаблони, оскільки вони фіксують лише точну відповідність для суфіксів.

3) Використовує множинне вирівнювання послідовностей за методом центральної зірки, яке починається з кожного повторення і закінчується перед початком наступного.

Архітектура системи складається з 3 модулів, перш за все це генератор правил вилучення, засіб перегляду шаблонів і модуль Extractor. Якщо перший модуль аналізує структуру тегів html і генерує правила, другий модуль відображає правила для вибору одного з них для вилучення, а третій модуль витягує всі записи даних, які відповідають вибраному правилу. Їх алгоритм пояснюється на робочому прикладі наступним чином

Приклад 1. Наприклад, розглянемо ці 2 рядки html-коду як вхідні дані для системи на першому етапі.

```
  <B>Конго</B>< I>242</I><BR>
<B>Єгипет</B><I>342</I><BR>$
```

HTML-сторінка надається як вхід до цього компонента, який потім перетворюється в рядок абстрактних уявлень, які називаються токенами, які ідентифікуються двійковим кодом. Тут маркери поділяються на два типи, маркери тегів і текстові маркери, де маркери тегів є тегами html, а текстові маркери є оригінальним текстом між тегами, позначеними Text(). Виявлення шаблонів на основі маркерів тегів, які в основному блокують теги рівня, за винятком текстових тегів, призведе до більш абстрактних шаблонів, які представляють всю веб-сторінку.

Приклад 2. Рядок маркера, згенерований перекладачем для наведеного вище прикладу, виглядає наступним чином

```
Html(<B>)Text(_)Html(</B>) Html(<I>)Text(_)Html(<BR>)
Html(<B>)Text(_)Html(</B>)Html(<I>)Text(_)Html(<BR>)$
```

Вони закодовані у двійкові рядки, як показано нижче

`Html (<B>) 000, Html (</B>) 001, Html (<I>) 011`, тоді шаблон буде 000001011$Двійковий шаблон, який є результатом попереднього кроку, надається як вхід для цього кроку. Дерево Patrica — це специфікація бінарного суфіксного дерева, де 0 йде ліворуч, а 1 — праворуч. Тут кожен внутрішній вузол є вказівкою того, який біт буде використаний для розгалуження. Кожна послідовність бітів, починаючи з кожного закодованого маркера, продовжуючи його до кінця рядка маркера. Кожна представлена позицією біта. Важливим зауваженням у дереві pat є те, що всі рядки суфіксів з однаковим префіксом будуть під одним і тим же піддеревом. Крім того, кожне ребро пов’язане з міткою віртуального краю, яка є сестринкою між цими двома вузлами. Ми можемо однозначно ідентифікувати сестринський рядок за його префіксом. Краї, позначені знаком $, вказують на кінець рядка, як показано на рисунку 2.1

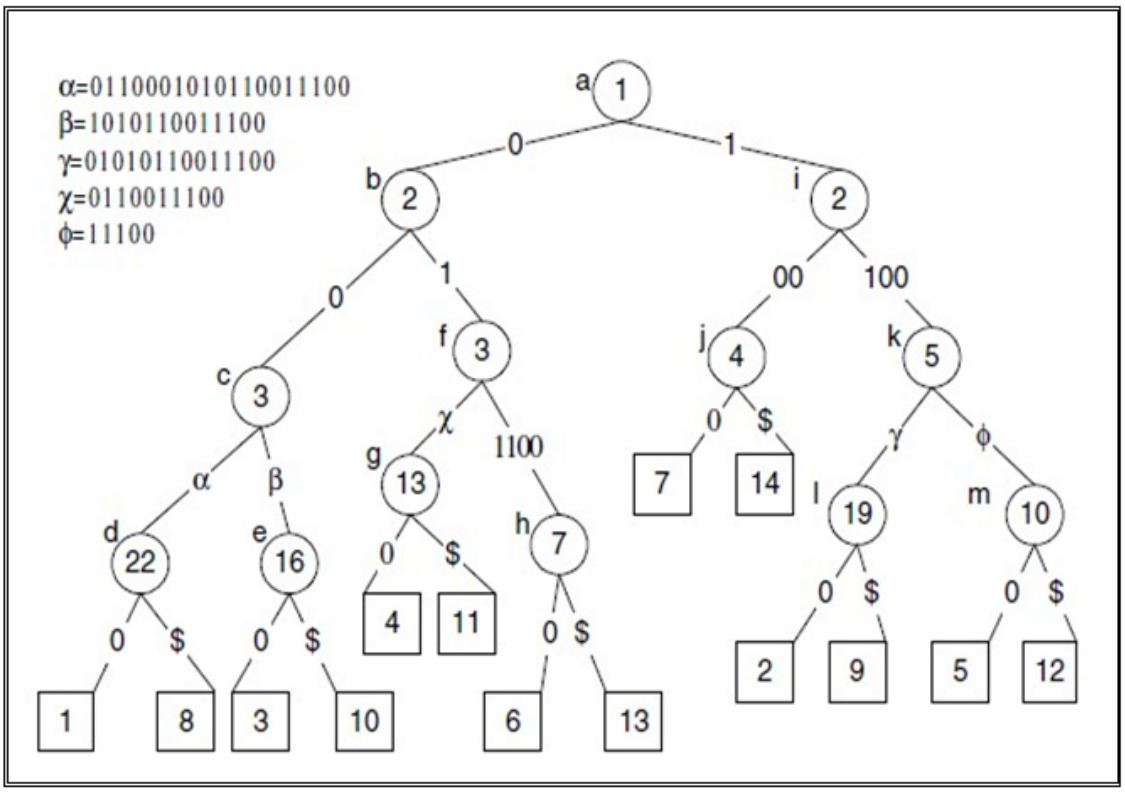


Рисунок 2.1 – Частина дерева рішень

Кожна мітка шляху внутрішнього вузла представляє повторювану послідовність вводів. Отже, щоб виявити повторювані шаблони, нам тепер потрібно лише спостерігати за мітками шляху. Таким чином, проблема ідентифікації записів даних звужується до отримання повторюваних шаблонів в IEPAD. Для мітки шляху у внутрішньому вузлі v, якщо його піддерево містить різні ліві символи, воно називається Left_Diverse. Виходячи з цього, пропонується лема про те, що максимальне повторення має бути лівим різноманітним. Кількість випадків, коли виникла будь-яка закономірність, можна легко дізнатися за кількістю зустрічей і опорними позиціями. Отже, користувач повинен визначити порогове значення або навіть значення за замовчуванням. Регулярність, компактність і охоплення визначаються як стандартне відхилення інтервалу між двома сусідніми входженнями, що розраховується як регулярність. У той час як компактність є мірою щільності максимального повтору, вона використовується для усунення максимальних

повторів, які розсіяні за межами даного зв’язку. Поряд з цим, покриття вимірює обсяг вмісту в максимальних повторах. IEPAD використовує множинне вирівнювання рядків для відповідності шаблону, він не дозволяє частковому чи невідповідному збігу розглядатися тільки рядки, які точно збігаються. Наприклад, припустимо, що «kjl» — це шаблон, виявлений для рядка маркерів, якщо у нас є кілька вирівнювань kjlmn, kjlm, kjlmo, шаблон вилучення можна узагальнити як «kjl[m\x]o[p\-]», де «–» позначає відсутні символи. Таким чином, задача трансформується, щоб знайти кратне вирівнювання k рядків. Алгоритм апроксимації використовується для знаходження центрального рядка Sc у K рядках. Як тільки центральний рядок знайдено, кожен рядок ітеративно вирівнюється з центральним рядком для побудови множинного вирівнювання, яке, у свою чергу, використовується для побудови шаблону вилучення.

Процес вилучення починається, коли користувач вибирає один або кілька шаблонів із засобу перегляду шаблонів. Екстрактор шукає в дереві PAT максимальну кількість зустрічей, щоб знайти цільові дані, але якщо веб сторінка не може бути виражена як дерево PAT, ніж звичайний шаблон

Алгоритм узгодження рядків зразка можна використовувати для вилучення. Їх результати Вирівнювання показує 97% точність успішного визначення цільових даних. Однак основні короткі висновки статті включають наступне:

1) Припущення, що декілька записів даних будуть однорідними, не вдається, оскільки записи даних на веб-сторінках можуть мати різні атрибути через відсутні значення.

2) Потрібен навчений користувач, який може виводити правила для вилучення фактичних записів даних.

3) Їхнє припущення про те, що цільові дані будуть під одним і тим же батьківським вузлом, не вдається для великої кількості веб-сторінок, оскільки немає необхідності в HTML5 після використання тегів блоку для розділів веб-сторінок.

4) Не пропонує жодної стратегії подолання шуму чи непотрібних даних.

5) Їх використання двійкових суфіксних дерев дає лише точні збіги, що є величезним недоліком через відсутні значення атрибутів у записах даних.

6) Не обговорювали гетерогенність веб-даних, таких як зображення, та процес їх вилучення.

STALKER: Ієрархічний підхід до індукції обгортки

Це контрольований підхід до вилучення даних, коли користувач позначає деякі зразки сторінок, з яких система генерує правила вилучення. Дані на веб-сторінках, які мають подібну структуру, можуть бути вилучені шляхом зіставлення цих правил. Він виконує ієрархічне вилучення даних за допомогою формалізму вбудованого каталогу.

Основні ідеї:

1) Введення концепції ієрархічного вилучення даних за допомогою вбудованого дерева каталогів, у якому листки є атрибутами, а внутрішній вузол є кортежами.

2) веб-вміст може бути витягнутий за допомогою деяких послідовних маркерів, які називаються орієнтирами, які дозволяють обгортку щоб знайти дані x з в області даних.

3) Для обробки відсутніх значень і множинних перестановок виконується багаторазове сканування. Сталкер запропонував представити веб-сторінку як типове вбудоване дерево каталогу, де кожен листовий вузол містить цільові дані, а внутрішні вузли утворюють список з K кортежів.

Знову елементами в k кортежів можуть бути листовий вузол l або інший список L. Кожне ребро дерева пов’язане з правилом вилучення, щоб витягти x з p.

Наприклад, розглянемо опис ресторану, наведений нижче

```
1:<p> Назва: <b> Яла </b><p> Кухня: тайська <p><i>
2:4000 Колфакс, Фенікс, AZ 85258 (602) 508-1570
3:</i> <br> <i>523 Vernon, Las Vegas, NV 89104 (702) 578-2293
```

```
4:</i> <br> <i>403 Pica, LA, CA 90007 (213) 798-0008</i>
```

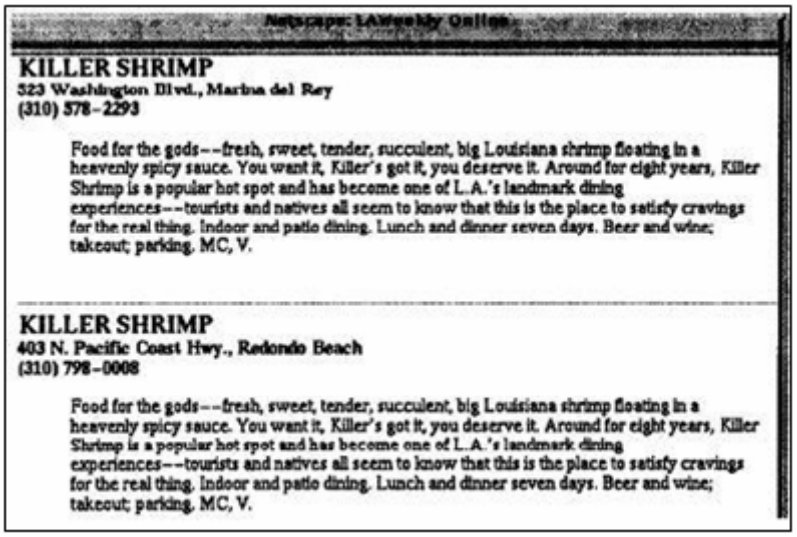


Рисунок 2.2 – Опис структури даних

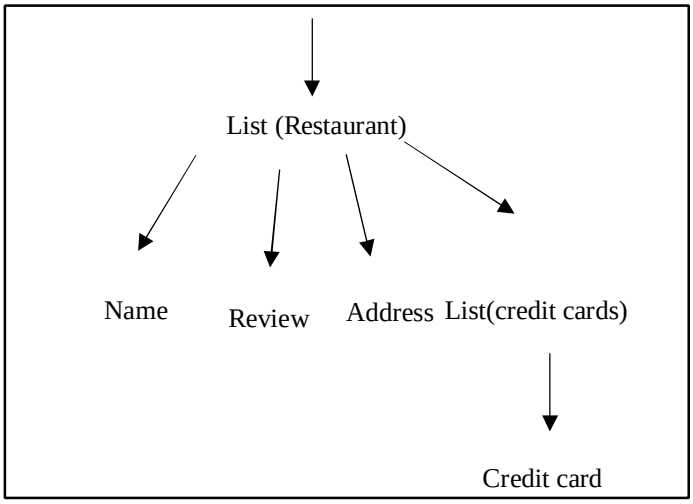


Рисунок 2.3 - Графічна структуризація

Дані витягуються за допомогою двох правил: правила початку та правила завершення. Для наведеного вище прикладу, щоб визначити назву ресторану, правилом початку є SkipTo(, а правилом кінця буде SkipTo() Таким чином, для кожного елемента даних асоціюється правило вилучення, і обгортка має генерувати його автоматично. Також можна використовувати правила SkipUntil() і NextLand mark ().

Правила вилучення також можуть бути виражені як орієнтирні автомати, де кожна транзакція позначена транзакцією. У процесі вилучення можна використовувати диз'юнктивні правила.

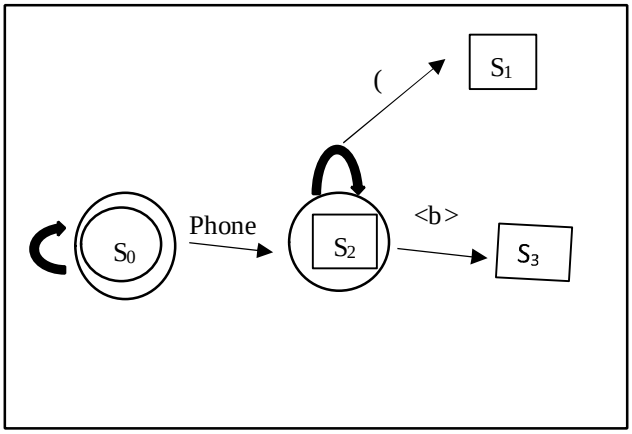


Рисунок 2.4 - SLG для початку області

Приклад: 2

```
El: 513 Pica, <b>Венеція</b>, телефон: I-<b>800/b>-555-1515
E2: 90 Colfax, <b> Palms </b>, телефон: (818) 508-22222
E3: 523 1st St., <b> LA </b>, телефон: l-<b>mk</b>-578-2293
Ред.: 403 Vernon, <b> Watts </b>, телефон: (310) 798-0008
```

Індуктивний алгоритм сталкера генерує правила вилучення для захоплення кожного елемента даних, беручи як вхідний набір послідовності маркерів префіксів. Користувач визначає початок x із зразкової сторінки за допомогою графічного інтерфейсу. Наприклад, вважайте, що користувач позначив коди регіонів, тоді послідовний алгоритм покриття починається з генерування лінійного l, оскільки він потім охоплює всі приклади та генерує диз'юнкти. У вищезгаданому прикладі алгоритм спочатку генерує SkipTo((), він відповідає E2 і E4, але не відповідає E1, E3. Тоді як у другій ітерації алгоритм розглядає лише нерозкриті приклади і генерує SkipTo(phone) і SkipTo(<b>). рис 1 представляє slg, який приймає обидва правила. Таким чином Stalker намагається створити slg, який приймає всі позитивні приклади і відкидає всі

негативні. Алгоритм містить функцію під назвою learnDisjuncts() генерує всіх кандидатів, багаторазово вибирає та уточнює найкращого кандидата доки вона не знайде ідеальний диз'юнкт, який приймає лише позитивні сторінки. Щоб знайти найкращий диз'юнкт, stalker шукає диз'юнкт, який приймає найбільшу кількість позитивних прикладів. Інша функція Refine() отримує чисті диз'юнкти(), роблячи свої орієнтири більш конкретними або шляхом додавання нових станів до автоматів. Далі йде GetTokens(), який видає маркери, які з’явилися принаймні один раз. Їхні експерименти доводять, що їх метод gy досягає 97% точності. Але недоліки статті: 1) Їхній алгоритм вимагає кількох проходів над вхідними даними для вилучення цільових даних. 2) Продуктивність алгоритму повністю залежить від наданих зразків навчальних сторінок, що робить систему залежною від навченого користувача. 3) Їхня техніка складна з наукової точки зору і вимагає багато часу, тому її не можна використовувати в програмах реального часу. 4) Підтримка обгортки є ще одним величезним недоліком цієї системи, щоразу, коли надходить нова структура презентації, обгортку потрібно оновлювати.

Езефі і Тітас запропонували структуру для орієнтованого на орієнтацію веб-схеми вилучення та їх інтеграцію як дворівневий процес видобутку. Вони прокладають шляхи для вилучення гетерогенних веб-даних за допомогою єдиного підходу з використанням NFA.

Основні внески Paper:

1) Пропонує архітектуру з 4 модулями для веб-вилучення та зберігання даних, що передбачає усунення шуму для входу в базу даних.

2) Ідентифікує блоки даних і записи даних шляхом визначення роздільника.

3) NFA для кожного типу вмісту генерується таким чином, щоб цільові дані можна було витягти з доменних дерев.

4) Пов’язує теги рівня аркуша з вмістом, оскільки він складається з важливої інформації про пов’язаний вміст.

Щоб ідентифікувати область даних і блок даних, вони створили дерево DOM, T представляє всю веб-сторінку, тоді регіони даних є піддеревами T. Також автор стверджує, що область даних і блок можуть бути з будь-якими тегами, такими як ,< table>,<tr>, і цей список неповний. Вони помічають, що області даних вбудовані між тегами і <table >, і вони використовували “{,}” для представлення даних в області даних. Вони визначають блок даних як кортеж, коли він розкладається на плоску структуру, це досягається схемою. Щоб ідентифікувати кортеж і блок даних, вони визначили 6 типів блоків вмісту в домені клієнта бізнес 2. Це блок даних продукту, блок даних списку, форма, текст, декоративний/синглтон, шум. Але метою є блок даних про продукт, який зазвичай має вигляд {< image >,{< title >, < number >, . . .,< brand >, < price >}} , деякі сторінки також можуть мати вигляд {< image >,< title >, < brand >, < price >}. Це та інформація, яку потрібно ідентифікувати та витягти. Щоб витягти типи кортежів вмісту, вони використовували NFA. NFA - це недетермінований кінцевий автомат, де кожна пара станів і nput є наступним станом. Щоб створити amNFA, який може витягувати цільові дані з усіх сторінок списку продуктів, автор спостерігає за вичерпним списком схем. Деякі з них такі:

```
Продукт (title:string, image:image-file,prodNum:string, brand:string, price:long);
Продукт (title:string, image:image-file,prodNum:string, price:long);
```

Таким чином, NFA має бути в змозі ідентифікувати будь-яку схему, витягнути кортежі даних і відправити їх до класу об’єктів вмісту продукту. Подібна ситуація зі списком, формою, текстом і вмістом зображення.

Автором запропоновано архітектуру з 6 модулів, які послідовно викликаються основним алгоритмом і завжди вихід попереднього модуля вводиться в наступний. Кожен модуль описується таким чином. Було

запропоновано міні-алгоритм сканера, який приймає вхідні дані як URL-адресу, він сканує нтернет, витягує сторінку та зберігає її в локальному каталозі. Він відхиляє коментарі з документа.

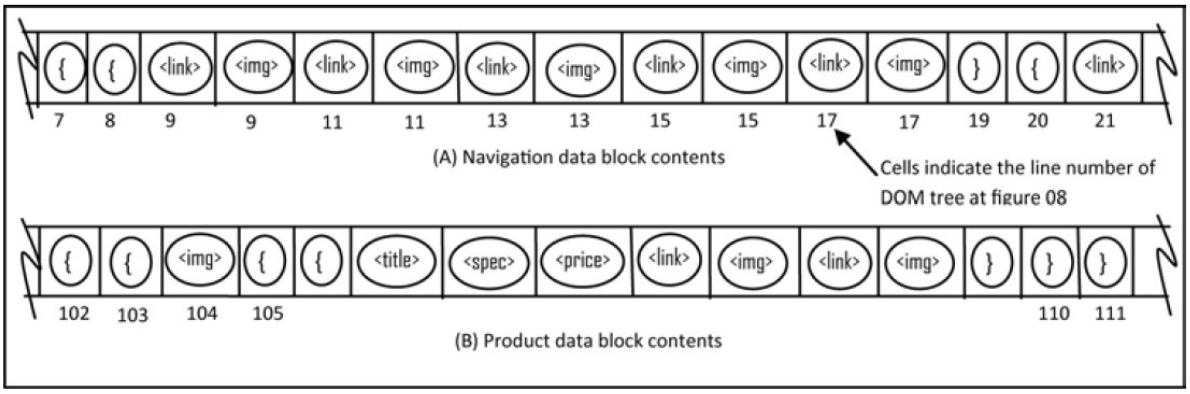


Рисунок 2.5 - Вміст блоку даних

Далі йде модуль очищення. Причина використання очищувача полягає в тому, що веб-дані зазвичай брудні. Очищення даних має бути виконано, щоб уникнути будь-яких відсутніх тегів або відповідних тегів, тому що дерево DOM побудовано на основі цих тегів, а також важливо мати якісні теги. Після очищення викликається модуль Etractor Content. Завдання цього модуля полягає в тому, щоб взяти чистий HTML-файл для побудови дерева DOM і призначити відповідний клас об'єктів класу та помістити об'єкти в список масиву вмісту. Кожен блок даних розділяється роздільником. об'єкт. Потім алгоритм викликає Mine contentObject.Identifytuple(), який витягує різні типи кортежів і поміщає їх у відповідні контейнери. Цей алгоритм генерує шаблон Seed NFA для блоків даних із позитивних сторінок. Він витягує об’єкти з усіх кортежів, зіставляючи їх з уточненими NFA і зберігаючи ідентичні кортежи (наприклад, кортежі об’єктів запису продукту, кортежи списку) у відповідний список TupleList. Виконується перевірка підрахунку, щоб перевірити, чи всі кортежі були вилучені. Стиснення кортежів використовується для узагальнення кортежів одного і того ж блоку даних. так що шаблон будь-якого кортежу, що містить шаблон різної довжини, може бути представлений в одній категорії. Щоб

створити таблицю бази даних, використовується підхід «зірка» зі сховища даних для формування таблиць фактів і розмірів зі списку масивів кортежів разом із назвою компанії. Вони досягли точності 100% і відкликання 96%. Основні короткі висновки статті включають 1) встановлення відповідності кожному кортежу в ArrayList з NFA послідовно є методом жорсткого підбору, який не може бути застосованим до великих веб-сайтів. Обслуговування обгортки: потрібно оновлювати щоразу, коли зустрічається нова структура.

RoundRunner: на шляху до автоматичного вилучення даних із великих веб-сайтів автоматично створює обгортку для вилучення даних зі сторінок html шляхом порівняння сторінок на основі їх схожості та відмінності. Даний набір зразків HTML-сторінок того самого класу знаходить вкладений тип вихідного набору даних і витягує набір вихідних даних, з якого були згенеровані сторінки.

Основний внесок:

1) Обгортки генеруються в автоматизований спосіб, не потрібна взаємодія з користувачем, а також будь-які навчальні приклади.

2) Здатний обробляти будь-які записи даних, як вкладені, так і плоскі, і не вимагає попереднього знання схеми.

3) Виявлення шаблонів базується на подібності та відмінності між двома веб-сторінками. Тут пропущені збіги використовуються для визначення відповідного вмісту.

4) Відповідно до статті генерується звичайна граматика, яка використовується для розбору сторінки, в цьому випадку виявлення такої граматики лише за позитивними сторінками не дає хороших результатів.

5) Запропоновано решітково-теоретичний підхід, який базується на тісній відповідності між вкладеними типами та регулярним виразом без об’єднання.

Генерація веб-сайтів — це процес кодування вихідних джерел даних у теги html, тепер вилучення інформації має бути процесом декодування. Оскільки вкладена структура веб-сторінок є ієрархічною, що представляє їх у регулярних

виразах без об'єднання, а пошук мінімального UFRE шляхом ітеративного обчислення найменших верхніх меж решітки RE для створення загальної обгортки є рішенням для пошуку схеми веб-сторінки. Тут теги html є вхідними рядками S1,S2,…..Sk.

Запропоновано алгоритм відповідності під назвою ACME, Align, Collapse under mismatch, and extract. Вхідними даними для цього алгоритму є список підготовлених HTML-токенів і дві веб-сторінки, які спочатку розглядаються як версія обгортки, а потім будуть перетворені в обгортку, яка витяги зі схожих сторінок.

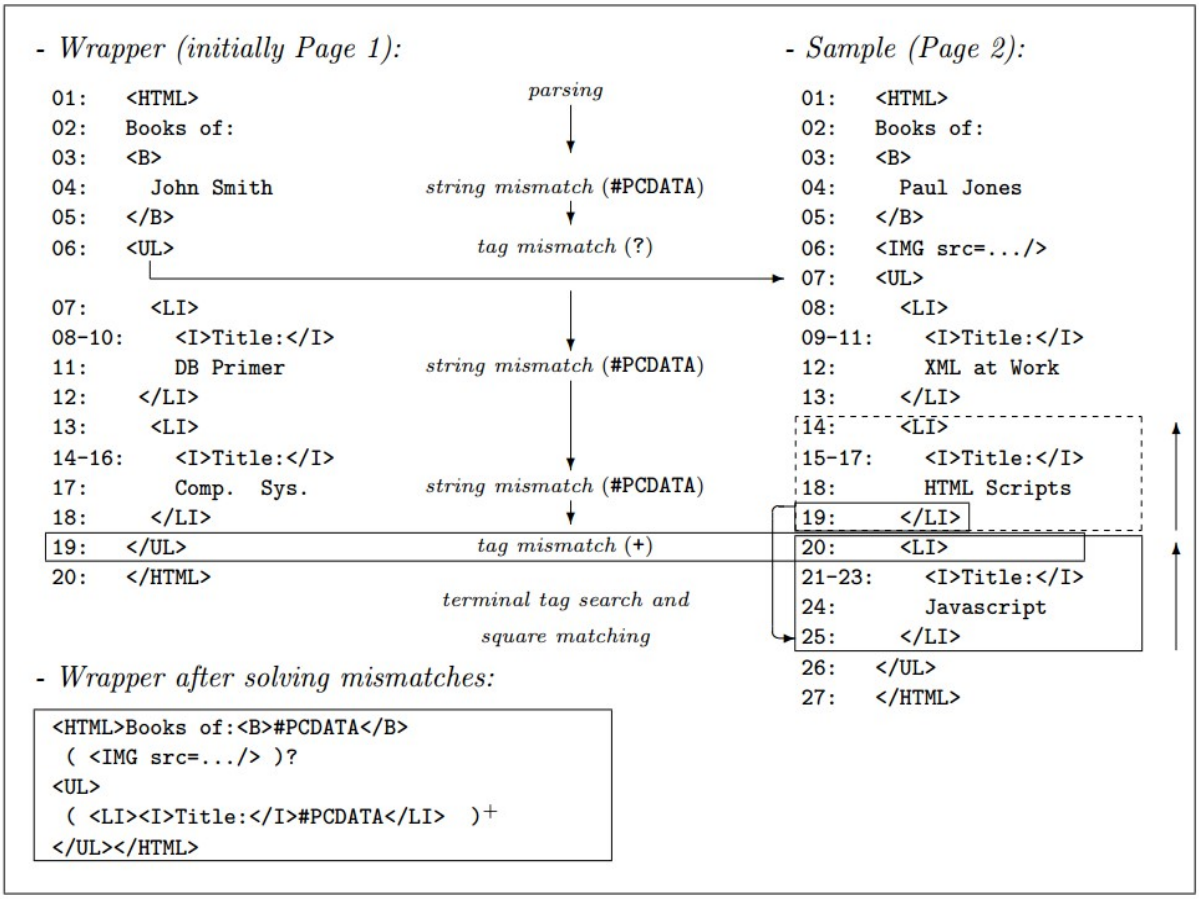


Рисунок 2.6 - Збіг двох книжкових веб-сторінок

Обгортка узагальнюється, коли виникає невідповідність. Згідно з цим документом, можуть мати місце дві невідповідності: Невідповідність рядків і невідповідність тегу. Невідповідність рядків — це не що інше, як різні значення атрибутів, тому їх можна узагальнити, замінивши #PCDATA. У згаданому

прикладі рядків «john smith», «Paul jones» є невідповідність рядків, яку можна подолати, замінивши їх на #PCDATA. Однак невідповідність тегів не так проста щоб обробляти як невідповідність рядків. У цьому випадку використовуються опціональні. Алгоритми намагаються з'ясувати повторюваний шаблон, якщо є невідповідність, він вважає цей атрибут необов'язковим. Наприклад, у прикладі невідповідність виникає на маркері 6 через наявність тегу <IMG > на сторінці 2, тоді алгоритми виконують перехресне зіставлення та розглядають тег <IMG> як обгортки.

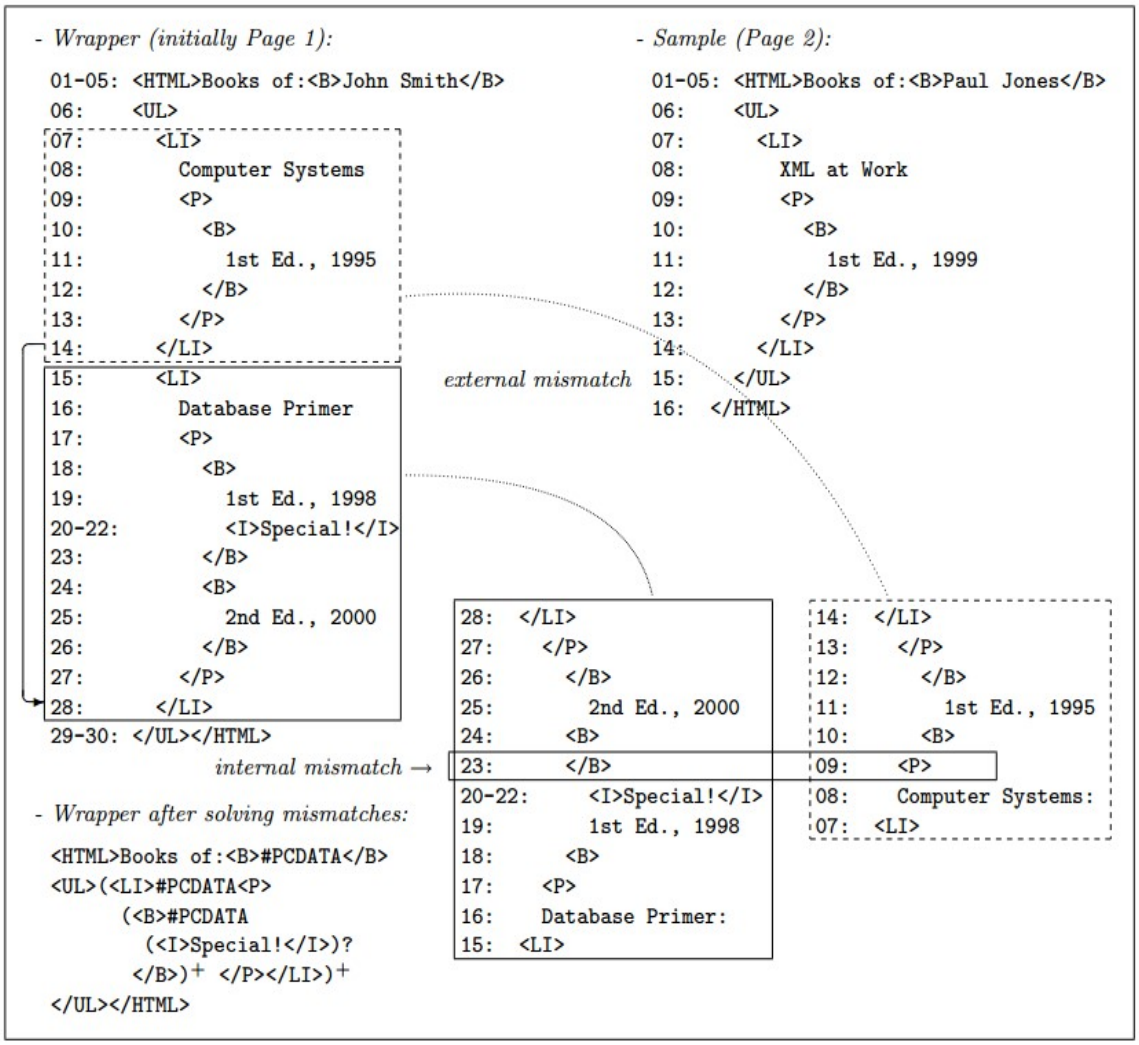


Рисунок 2.8 - Комплексна невідповідність

Алгоритм спочатку знаходить, де існує мінімальна потужність у зразку чи в обгортці, а потім знаходить термінальний тег. Там досліджується можливість

квадрата-кандидата форми <UI>……</LI> в обгортці, яка є помилкою, а наступна у зразку <LI>……..</LI> тут успішно. Таким чином, він робить висновок, що вибірка містить кандидати для лексем від 20 до 25. Тепер, коли алгоритм знайшов квадрат s, обгортку необхідно узагальнити як S+ шляхом безперервного пошуку в обгортці повторюваних зустрічей.

Але проблема виникає, коли ми намагаємося усунути невідповідність, і це призводить до більшої кількості внутрішніх невідповідностей. Варіант сторінок про автора показаний у прикладі 2, синтаксичний аналіз зупиняється на рядку 15 через невідповідність тегів, коли алгоритм намагається знайти маркери-кандидати, зіставляючи вгору від термінального рядка, це призводить до все більшої кількості невідповідностей. Цю ситуацію можна вирішити, розглядаючи кожну внутрішню невідповідність як іншу невідповідність тегів, намагаючись вирішити її так само, як і зовнішню невідповідність. У наведеному прикладі внутрішня невідповідність включає маркери 23, 9, після розв'язання вона веде до списку редакторів, а друга невідповідність розглядається як необов'язковий шаблон <I>Особливий!</I>.

Отже, з усіх наведених вище випадків алгоритм Match можна підсумувати таким чином: Вхідні дані для алгоритму є обгорткою w і зразковою сторінкою s вона намагається узагальнити обгортку, знаходячи невідповідності. Тому пошук рішення збігу (w,s) - це відвідування дерева I-АБО, насправді рішення полягає в пошуку рішення для кожної невідповідності, яка сталася в процесі аналізу дерева, і буде включати додавання вузла або пошук його . Недолік:

1) Позначення полів після вилучення вручну необхідне, оскільки вони мають анонімні імена (наприклад, a,b,c,d тощо),

2) Алгоритм має експоненційну тимчасову складність щодо довжини введення, тому він не може використовуватися в режимі реального часу.

3) Це вимагає більше однієї навчальної сторінки та є підходом із напівнаглядом.

2.2 Підходи на основі процесів порівняння

Ці підходи знаходять спільні риси для ідентифікації записів даних, проводячи порівняння між фрагментами сторінки, розглядаючи веб-сторінку як дерево, і виконуючи відстань редагування дерева, відстань редагування рядка. Такі системи детально пояснюються наступним чином.

Тут розглядається проблема вилучення даних за допомогою техніки вирівнювання дерева. Він порівнює суміжні підрядки з початковими тегами, що мають той самий батьківський вузол, замість того, щоб порівнювати всі суфікси. Їхній основний внесок:

1) Використання підходу, заснованого на баченні, для побудови дерева Dom та ідентифікації записів даних.

2) Вкладена структура записів даних не втрачається завдяки методу вирівнювання дерева, який використовується в процесі вилучення.

3) Нерелевантні підрядки не порівнюються, щоб уникнути непотрібних обчислень.

Автор стверджує, що записи даних можуть не бути суміжними, а у вкладеній структурі вони також можуть починатися з середини будь-якого іншого запису. Також він стверджує, що дерева, засновані на інформації тегів, не завжди гарантують правильність, тому дерева тегів на основі інтерпретації є надійними. На основі цих спостережень він пропонує новий метод вирівнювання дерев на основі інтерпретації дерев.

Побудова дерева тегів HTML на основі інтерпретації передбачає, що кожен елемент HTML представляється у вигляді прямокутника. Дерево тегів будується на основі вкладених прямокутників шляхом знаходження 4 меж прямокутника та виявлення зв’язку утримування між прямокутниками. Замість негайного видобутку записів даних автор пропонує спочатку видобути області даних, а потім записи даних. Область даних знаходить, виконуючи відстань редагування рядка. Узагальнений вузол використовується для позначення

піддерева. На відміну від інших методів, автор каже, що записи даних можуть не бути суміжними, але узагальнені вузли суміжні. Записи даних ідентифікуються тим фактом, що відстань між записами даних в області даних менша за відстань між будь-якими прогалинами в записі даних. Може бути випадок, коли область даних може мати більше одного узагальненого вузла або навіть дві або більше областей даних з кількох узагальнених вузлів, такі випадки можуть бути оброблені алгоритмом.

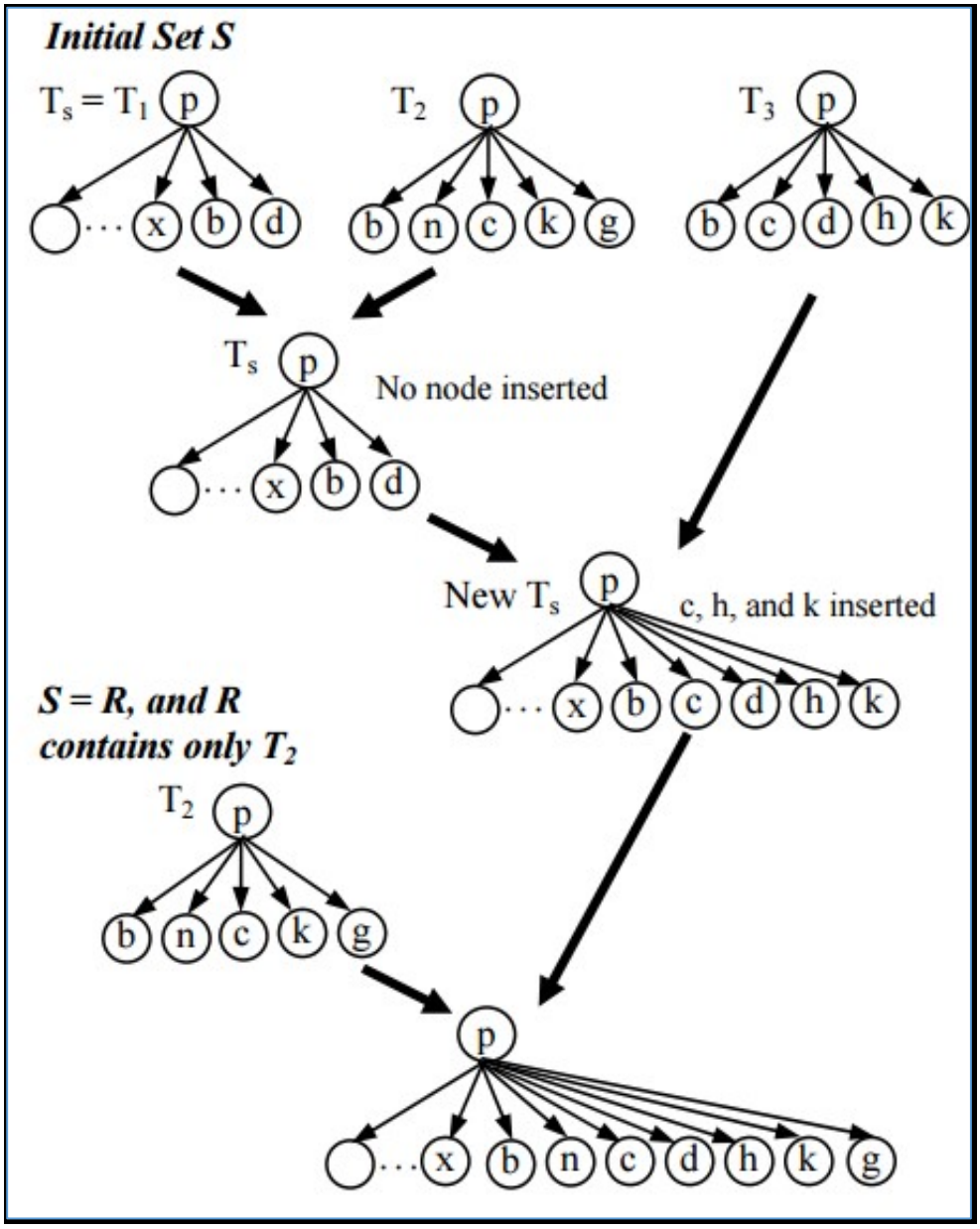


Рисунок 2.9 - Ітераційне вирівнювання дерева з двома ітераціями

Вилучення даних виконується шляхом створення кореневого дерева тегів для кожного запису даних, після того як усі записи даних ідентифікуються, піддерева всіх записів даних перебудовуються в одне піддерево. Потім дерева тегів усіх записів даних у кожній області даних вирівнюються за допомогою методу часткового вирівнювання на основі збігу дерева. У цьому підході автор вирівнює кілька дерев тегів шляхом прогресивного зростання насіннєвого дерева Ts, тут початкове дерево є деревом, яке має максимальну кількість дочірніх вузлів. Для кожного Ti алгоритм намагається знайти для кожного вузла в Ti відповідний вузол у Ts. Якщо збіг знайдено, між цими вузлами створюється зв'язок, якщо ні, то вузол вставляється в дерево зародження. У цьому випадку питання, де вставити новий вузол або набір невідповідних однорідних вузлів однозначно, є питанням. У такому випадку він додається до прапора R set у значення true, щоразу, коли нове вирівнювання знайдено ітеративно, R порівнюється для відповідності. Процедура пояснюється на прикладі нижче.

Є 3 дерева T1, T2, T3 і розглядають T1 як інтерпретації, оскільки воно має більше дочірніх вузлів. Алгоритм призначає Ts початковому дереву, яке зараз T1. Інші два дерева порівнюються з насіннєвим деревом для вирівнювання. Ts і T2 створюють один збіг, вузол b. Вузли n, c, k і g не відповідають Ts. Отже, він намагається вставити їх у Ts. Однак жоден з вузлів n, c, k і g в T2 не може бути вставлений в Ts, оскільки неможливо знайти унікальне розташування. Потім він вставляє T2 в R, який є списком дерев, які, можливо, потребують подальшої обробки. При узгодженні T3 з Ts всі незрівняні вузли c, h і k можуть бути вставлені в Ts. Таким чином, T3 не буде вставлено в R, встановлений «flag = true», щоб вказати, що знайдено якісь нові вирівнювання/збіги або деякі невідповідні вузли вставлено в Ts. Коли алгоритм зустрічає S= і flag="true", він розуміє, що в S більше немає дерев, але є деякі нові вирівнювання та вставки. Якщо якісь дані ще не оброблені, вони будуть розміщені в одному стовпці таблиці бази даних. Таким чином, усі записи даних витягуються та зберігаються в базі даних. Результати показали, що їхній підхід дає 99% відкликання та

точності. Короткий опис статті включає 1) Цей підхід не працює і не може бути розширений за допомогою сторінок, які мають один запис даних, наприклад сторінку з докладними відомостями. 2) Вкладені записи даних не можуть бути вилучені за допомогою цього підходу.

DEBYE: вилучення даних за прикладом

Контрольована система навчання, де користувач позначає приклади сторінок за структурою, вивчається зі сторінок за допомогою шаблонів, і з неї структури можна витягти шляхом порівняння нових записів даних з попередніми. Пропонується об'єктно-орієнтоване вилучення вмісту без втрати зв'язку між елементами даних. Спочатку він витягує атомні компоненти, а потім збирає їх в об’єкти. Використовує відповідність шаблону для визначення подібних маркерів для вилучення. У цій роботі кожна частина даних розглядається як об’єкт у згаданому прикладі 4 чотирьох авторів та деяких їхніх книг, кожна частина даних представлена об’єктом.

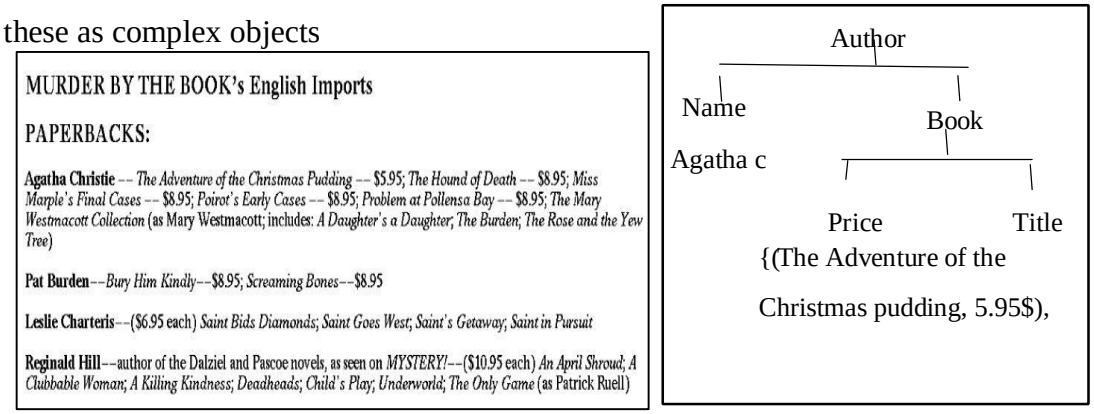


Рисунок 2.10 - Знімок веб-сторінки та ієрархічної структури для об’єкта

У Debye є 2 модулі The Graphical Interphase і The Extractor. Об'єкт позначається як пара O= <T, V> така, що V є домен (T), де V є типом O. Таким чином, O є екземпляром T. domain(T)={a1,a1,a3… .an}(n>=1). Випадки A-типів називаються атомами. Пара <T, ai> називається парою значень атрибутів. Користувач повинен виділити цільові дані з веб-сторінки за допомогою

графічного інтерфейсу. Ці цільові дані розглядаються як об’єкти. Такі зібрані об’єкти використовуються для створення oe_patterns в xml. Ці шаблони використовуються для вилучення даних з нових веб-сторінок. Початкова та кінцева позиції появи даних позначаються ipos, fpos відповідно. Шаблони пар значень атрибутів описуються як <T,ai>, локальний синтаксичний контекст пов’язаний з кожним avp, це маркери, оточені avp. Використовуючи цю контекстну інформацію, будуються шаблони avp. У наведеному прикладі для того, щоб ідентифікувати модель avp для типу ціна, це було б, як зазначено в (a), що значення має передбачатися Spre = «Смертельна брехня--$», зокрема, і суфіксом Ssuf = «;dead online», тому він строго витягує лише конкретні значення. Більш узагальнений шаблон позначений (b), він позначає, що значення ціни має префікс «$», а потім «;». а «*» вказує на будь-яке значення в цьому місці.

```
<TUPLE type="Author">
   <ATOM type="Name">
      <VALUE ipos="2058" fpos="2073">Agatha Christie</VALUE>
   </ATOM>
   <LIST type="Book">
      <TUPLE type="Book">
         <ATOM type="Title">
            <VALUE ipos="2084" fpos="2122">The Adventure of the Christmas Pudding</VALUE>
         </ATOM>
         <ATOM type="Price">
            <VALUE ipos="2131" fpos="2135">5.95</VALUE>
         </ATOM>
      </TUPLE>
      <TUPLE type="Book">
         <ATOM type="Title">
            <VALUE ipos="2257" fpos="2280">The Hound of Death</VALUE>
         </ATOM>
         <ATOM type="Price">
            <VALUE ipos="2289" fpos="2293">8.95</VALUE>
         </ATOM>
      </TUPLE>
   </LIST>
</TUPLE>
```

Рисунок 2.12 – Початкова структура коду

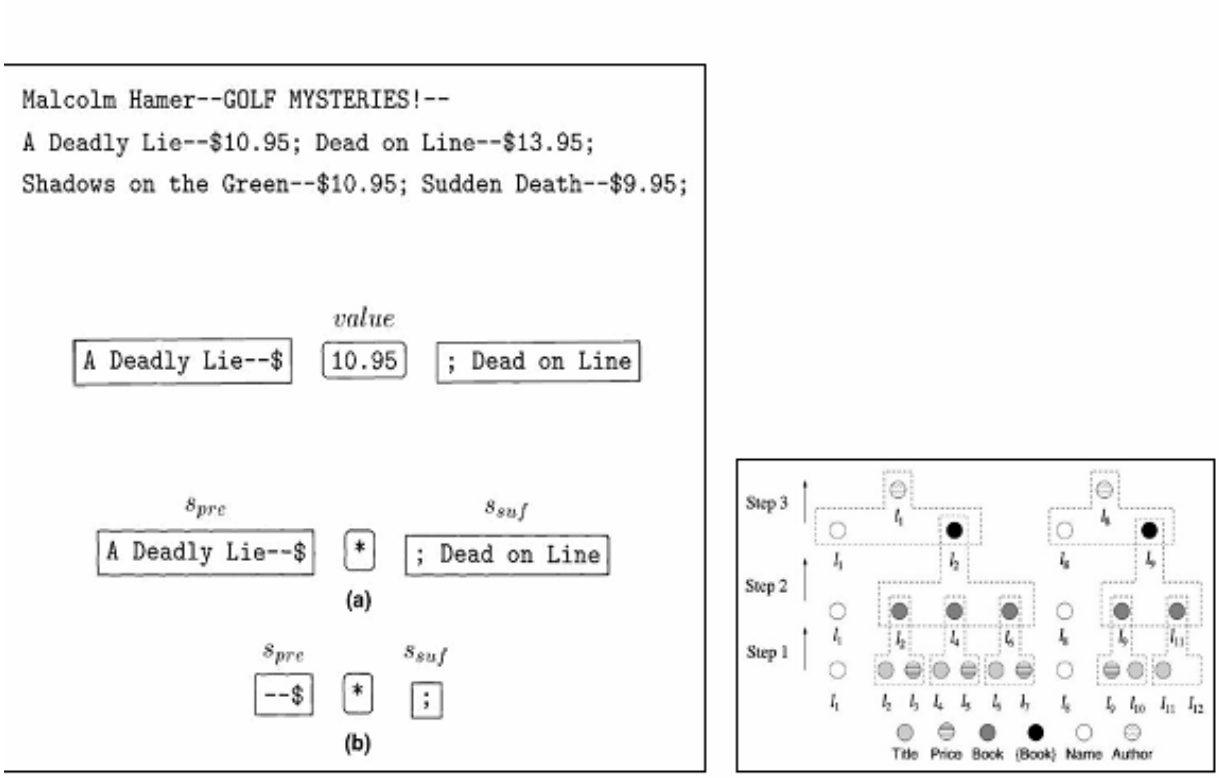


Рисунок 2.13 - Виконання знизу вгору

Шаблони ОЕ представляють структуру даних, які користувач повинен витягти, його внутрішні вузли утворюють інше дерево ОЕ або вузли типу (наприклад, ціна, книга), а листи є шаблонами avp. Існують дві стратегії вилучення: стратегія «знизу вгору» та «зверху вниз». Стратегія «зверху вниз» є прямою, шаблони ое збираються для кожного прикладу об’єкта і використовують ці шаблони для виявлення нових об’єктів. Ця процедура повторюється залежно від кількості наданих прикладів сторінок. Тут весь об’єкт розбивається на компоненти, що називається підходом зверху вниз. Цей підхід погано працює, якщо сторінка має змінну структуру.

У стратегії «знизу вгору» avp пріоритетно розпізнаються та витягуються. Для цього підходу об’єкти представлені у вигляді потрійного O = <T,V,L>, де t – тип об’єкта, v – значення, l – індекс. Останній ряд кіл позначає avps. Кожен avp пов'язаний з міткою li. Суміжні пари заголовок, ціна формуються як книжкові екземпляри. Значення в L9 розташовані в зворотному

порядку, а L11 не має цінової складової, такі випадки можна обробляти стратегією «знизу вгору». Список об'єктів, зібраних після першого кроку, вказується у 2-му рядку. Далі екземпляри книги групуються як список екземплярів книги. З прикладу l1 поєднується з l2, а l8 поєднується з l9. Таким чином, дані можна отримати з подібних сторінок.

Основні недоліки:

1) Точність вилучення падає низькою у разі відсутності атрибутів або атрибутів з кількома впорядкованими.

2) Для розширення кожного атомарного атрибута потрібне багаторазове сканування вхідної сторінки.

2.3. Процес видобування інформації з веб-сторінок

В розділі представлені методи вилучення даних із веб-сторінок у будь-якому форматі, якщо сторінка містить декілька записів, то пропонується методика на основі онтології, якщо це детальна сторінка, то відбувається порівняння з шаблоном.

Основні внески:

1) Подає веб-дані як об'єкти та витягування даних як процес вилучення значень атрибутів об'єкта.

2) Запропоновано механізм вилучення на основі онтології для сторінок зі списком і підхід на основі збігу дерева для сторінок деталей.

3) Правила вилучення виявляються на основі положень відмінностей у дереві dom. У своєму підході автор розглядає веб-сторінки як 2 типи:

1) Головна сторінка

2) Сторінка деталей.

Головна сторінка – це та, що містить список даних *target*.

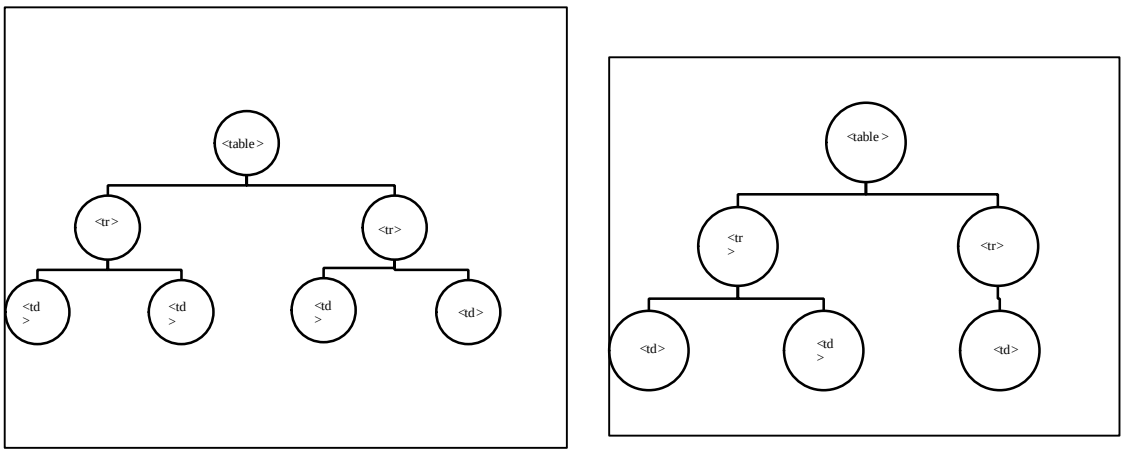


Рисунок 2.14 - Дерево DOM з піддеревами

Наприклад, список продуктів у середовищі електронної комерції. Коли користувач натискає певний продукт, вся інформація з'являється на сторінці з деталями. Більшість із запропонованих методів розглядають веб-сторінку або як список, або як сторінку з детальною інформацією, але це не так. Веб-сторінка сканується як дерево DOM, а вузли класифікуються за їх вагою, виконуючи операцію скорочення на вузлах з вага 0. Тепер, коли веб-сторінка змодельована як дом-дерево, пошук записів даних є пошуком у ширину. Знаходить повторювані подібні піддерева, порівнюючи подібність тексту вузлів за відстанню редагування рядка. У дереві текстові вузли маркуються, розглядаючи html, спеціальні символи та сполучники як роздільники. Для визначення значень атрибутів з головної сторінки значення атрибутів отримуються з онтології. Вони можуть бути атомарними або неатомарними, іноді навіть обидва. Однак ми повинні зіставити токени зі значеннями атрибутів, які отримані за допомогою онтології. Застосування онтології дозволяє нам виявити структуру записів даних, потужність витягнутих значень атрибутів і, можливо, зв'язки між ними. Точність вилучення може бути покращена за допомогою оніологічного вилучення, яке може вказувати мітки атрибутів, регулярний вираз або ключові слова. Але певні значення даних не можна витягти з домену, наприклад, запис даних, який потрібно витягти, ім'я студента має нескінченний домен. Щоб

визначити значення атрибутів зі сторінок деталей, вилучення даних зі сторінки з докладними відомостями базується на тому, що веб-сторінка є шаблоном, створеним за допомогою вбудованих значень із базових баз даних. Таким чином, вони представляють веб-сторінку як сторінку HTML зі змінними. Ці змінні в процесі оцінки шаблону об'єднані з моделлю даних, що має атрибути продуктів.

Вважається, що розбіжність між веб-сторінками виникає в 3-х місцях:

1) Різниця між структурою дерева DOM і структурою піддерев на веб-сторінці. Нижче наведено зразки домінних дерев, які представляють подібну структуру, але піддереви не схожі, оскільки їх кількість дочірніх дерев відрізняється.

2) Різниця між атрибутами або значеннями HTML. Тому що не кожен запис даних матиме однакові атрибути.

3) Відмінності в текстових вузлах, що представляють різні значення для вузлів.

Визначаючи відмінності, автор пропонує вилучення, оскільки спочатку виконується отримання вихідного HTML-коду з 2 сторінок деталей, а потім застосовується алгоритм diff для отримання всіх різних місць. Далі слід знайти найближчий елемент HTML, який містить інше місце в піддереві. Якщо інше місце знаходиться в дочірньому вузлі, зрозуміло, що сторінки мають однакову структуру, за винятком варіацій значень, тому оголошуйте весь наступний вузол як кандидата для вилучення. Але якщо різниця полягає в атрибуті HTML, просто проігноруйте її. Якщо це інший випадок, то веб-сторінки не мають подібної структури, тому передають їх до іншого процесу вилучення. Таким чином, дані можна витягти з детальних сторінок і сторінок зі списком, визначивши їх відмінності.

Серед недоліків:

1) Онтологія – це в основному ручний процес і дорогий, часто несумісний.

2) Певні атрибути можуть мати нескінченний домен. Наприклад, назви продуктів.

2.4. Підходи на основі інтерпретації

Lixto — це інтерактивний інструмент для вилучення даних, створений на основі мови Elog. Elog використовує журнал даних, як-от логічний синтаксис і семантика. Який перекладає відповідні рядки html у XML. Система LIXTO надає інтерфейс користувача для вибору даних, які цікавлять користувача. Вибрані таким чином дані перекладаються в xml. Користувачеві не потрібно мати уявлення щодо основної мови ELOG. Elog — це мова програмування обгортки, вона генерує обгортки для вилучення даних, вводячи дані, вибрані користувачем із веб-сторінки. Це набір правил, подібних до журналу даних, що містять особливі умови вилучення в їх тілах. Після створення обгортки її можна автоматично витягти з сторінки, яка постійно змінюється. Система lixto пов'язує вибраний користувачем екземпляр із узагальненим шляхом до дерева в дереві розбору HTML, де ідентифікує подібні правила. Наприклад, користувач вибирає шаблон <ITEM>, а потім підшаблон <PRICE>, це зв'язок підшаблона виражає, що екземпляр <PRICE> має відбутися у шаблоні <ITEM>. Потім система шукає такі закономірності. Користувач також може додавати умови, такі як після/до/не після/діапазон тощо. Встановлення таких фільтрів може отримати ідеально бажану інформацію. Створення шаблону виконується користувачем, який явно надає шаблони своїм батькам, може вказати будь-які необхідні атрибути, а потім уже виділяє всі об'єкти у наданому прикладі. Наприклад, якщо користувач вибирає <TALBE> як шаблон, LIXTO шукає наявність цього тегу в усіх документах, а також виділяє теги <TR>. Створюється правило для створення правила для вилучення всіх цільових елементів даних. Шаблони, створені разом з ієрархічними зв'язками, перекладаються в XML, використовуючи їх імена як теги <XML> за замовчуванням. Механізми мови вилучення такі, предикати Head

визначаються в Elog як запис(S, X), де S вказує батьківський шаблон, наприклад, <TABLE> у нашому прикладі. X шукає піделементи, які кваліфікуються як S. Таким чином визначені предикати заголовка — це шаблони, які Elog визначає та витягує дані відповідно до веб-сторінки. Lixto пропонує два механізми вилучення даних - вилучення рядків і вилучення дерева. У механізмі вилучення дерева елементи ідентифікуються за їх стежками дерева. Прикладом шляху дерева може бути *table.*.tr, тут * вказує, що можна назвати підстановочним знаком, це вказує, що може статися будь-що. Для правильного вилучення лише цільових даних можна застосувати умови атрибутів. Умова атрибута — це трійка, яка визначає необхідне ім'я, значення та відповідний регулярний вираз. Другий метод вилучення заснований на струнах. При розборі листові вузли дерева вказують на жала. Для їх захоплення використовуються регулярні вирази. Атоми Elog відповідають спеціальним предикатам із чітко визначеною семантикою наступним чином. У Elog функція, що відображає дане джерело S на набір елементів, що відповідають epd, розглядається як відношення subelem(S, epd, X). Subelem(s, epd, x) оцінюється як істина, якщо s — це область дерева, epd — це визначення шляху елемента, а x — область дерева, що міститься в s, де корінь x відповідає epd. Предикати визначення вилучення вказують набір екземплярів Extraction, предикати умови контексту вказують, що якесь інше піддерево або текст повинні (не) з'являтися після/перед бажаним шаблоном вилучення. Наприклад, на сторінці з кількома таблицями кінцева таблиця може бути ідентифікована зовнішньою умовою, яка вказує, що після потрібної таблиці не з'являється жодна таблиця. Предикати внутрішньої умови — це семантичні умови, такі як isCountry(x) або Date(x). Іншим класом предикатів є предикати зразків, які вказують на шаблони та їх зв'язок із їхніми батьками. Стандартне правило в Elog таке, як зазначено тут New(S, X) ← Par (-, S), Ex(S, X), Co(S, X, . . .)[a, b], де S є батьківська змінна екземпляра, X є змінною екземпляра шаблону, Ex (S, X) є атомом визначення виділення, а необов'язкові Co(S, X) є додатковими умовами. Набір таких правил утворює

шаблон. Усі правила у шаблоні використовують ту саму інформацію, як-от ім’я шаблону, його батьківський елемент тощо. Такі шаблони утворюють програму вилучення, яку можна застосувати до подібних сторінок. Наприклад, обгортка для наведеного вище прикладу, кожен елемент зберігається у власній таблиці, витягнутій за допомогою <record>, усі такі шаблони визначаються в одному записі. Атрибут price використовує атрибут концепції, а саме isCurrency, який відповідає рядку, наприклад $, а шаблон ставок використовує посилання на ціну. Останні два візерунки - це струни.

Пропонується метод, який називається ієрархічним умовним маркуванням. Новий графічний підхід до взаємної оптимізації розпізнавання записів та іменування атрибутів.

Основний внесок включає:

1) Цей підхід витягує дані зі списків веб-сторінок, а також веб-сторінок з детальною інформацією.

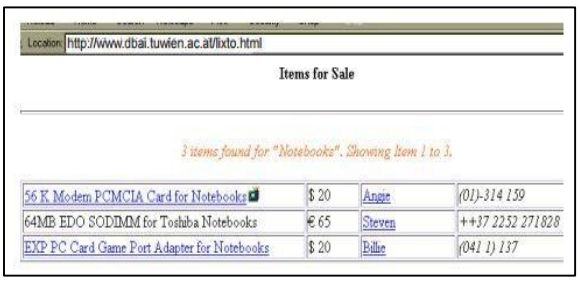
2) Використовує механізм побудови дерева на основі інтерпретації.

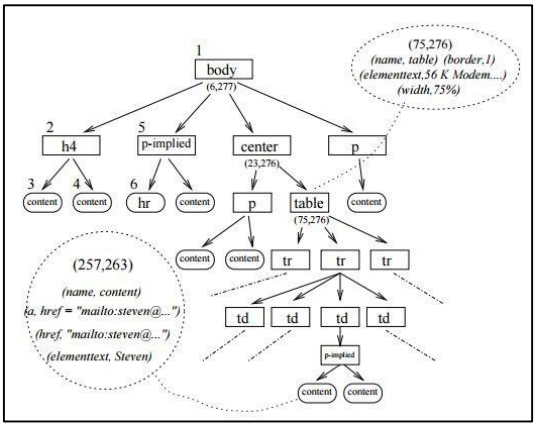
3) Виявлення проблемного запису даних розглядається як призначення міток запису даних блоку в дереві інтерпретації. Позначення атрибутів розглядається як призначення міток листовим блокам, і обидві завдання можна виконувати одночасно.

4) Ефективно включає всі корисні функції та їх значення.

Веб-сторінки представлені у вигляді дерев інтерпретації за допомогою таких функцій макета сторінки, як шрифт, колір, роздільники розміру – це горизонтальні та вертикальні лінії, які візуально не перетинають жодного вузла. Блок на веб-сторінці представлений у вигляді вузла в дереві інтерпретації. Умовні випадкові поля — це випадкові поля Маркова. Якщо x, y — 2 випадкові величини, то буде позначено, а y — змінні над відповідними мітками. Тут y може бути лінійним ланцюгом.

G= (V, E) where x=label over observation





:

Рисунок 2.9 - Дерево розбору HTML для сторінки

Тепер проблему вилучення веб-даних можна звузити як знаходження максимальної умовної ймовірності у над x, де x — ознаки всіх блоків, y — можливі призначення міток.



Рисунок 2.10. – Відображення структури даних

На основі ієрархічного представлення даних можна побудувати модель, позначаючи внутрішні вузли прямокутниками, а листові вузли — еліпси. З кожною міткою пов’язана випадкова величина Y. На основі графіка моделі будується дерево з'єднань шляхом відображення еліпсів, вузлів кліки та прямокутників як роздільників. Потім алгоритм вибирає довільний вузол кліки як кореневий вузол.

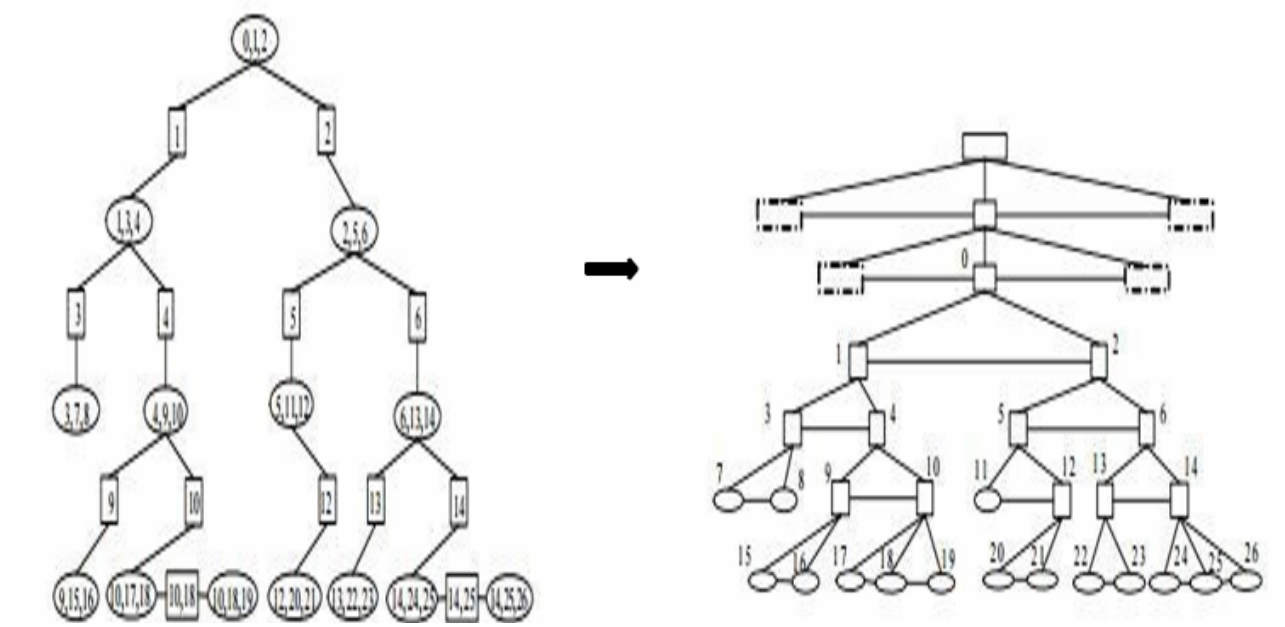


Рисунок 2.11 – Приклад виконуваної трансформації

На рисунку 2.11 показано дерево з'єднань вищезгаданого прикладу. Значення в кожному внутрішньому вузлі — це значення цього конкретного вузла разом із його дочірніми вузлами. Таким чином, значення в корені дорівнює {0,1,2}. За цим слідують дві фази. Використовуючи модель HCRF для веб-витягування, необхідно визначити простори міток з наступним розділенням між змінними на листкових вузлах і змінних у внутрішніх вузлах, тому що нам потрібно витягти значення атрибута вузла laf, де потрібно ідентифікувати a для внутрішнього вузла. чи є це запис даних чи ні. Необхідно чітко визначити простір лівої мітки та внутрішній простір міток для кожної своєї мети. Наприклад, простір етикетки аркуша для вилучення даних про продукти може бути {Назва продукту, ідентифікатор продукту, ціна, опис тощо}.

Однак внутрішній простір міток має 2 розділи, об’єкт _type незалежний і dobject tpe dependent. Існують певні мітки, такі як заголовок сторінки, панель навігації, хвіст сторінки тощо, які посилаються на різні частини веб-сторінки. Такі об'єкти не залежать від типу_об'єкта. Проміжні мітки, такі як дані reocrs, залежать від Object_type. Функції елементів і блоків у HCRFS є для кожного елемента, алгоритм витягує як інтерпретації, так і

функції вмісту, всю інформацію можна отримати з функцій дерева інтерпретації. Відстань до дерева двох блоків визначається як відстань редагування відповідних піддерев. Щоб використати подібність між блоками, можна також обчислити форму dtype distamce. Вони також вважають, що повторювана інформація на веб-сторінках не дає корисної інформації, як приклад вони цитують, що кнопка «додати в кошик» і такі типи зустрічаються неодноразово в кожну сторінку, які не є корисними.

3. ПОБУДОВА ПРОГРАМНОЇ СИСТЕМИ ВИДОБУВАННЯ ДАНИХ У WEB ТА ЇЇ ОЦІНКА

Визначення проблеми.

Враховуючи будь-який веб-сайт бізнес-клієнта, W домену d складається з кількох сторінок зі списком продуктів, які генеруються загальним шаблоном T із базових баз даних із

Схема S з кортежами α. Ідентифікація S на веб-сторінці, витягуючи α для зберігання в базі даних і далі в сховищі даних для порівняльного аналізу.

Як обговорювалося вище, тут пропонується об’єктно-орієнтована модель даних для вилучення та аналізу неоднорідного веб-вмісту. Вони надали основу для вилучення веб-даних на веб-сторінці B2C і алгоритм (званий WOM) для вилучення веб-об’єктів. Незважаючи на те, що він досяг своєї мети, він був дуже складним, і їх обгортку потрібно оновити, якщо він зіткнеться з новою структурою. Ми вивчили їхню роботу та запропонували дворівневий процес майнінгу для виявлення знань. Ця дипломна робота розробляє архітектуру (ми називаємо її WOM_S) для аналізу веб-контенту та структури за допомогою об’єктно-орієнтованої моделі. Він розробляє, розширює та модифікує необхідні алгоритми для системи WOM_S. Він також виявляє схему бази даних і дає рекомендації щодо інтеграції даних на другому етапі.

3.1 Вирішені проблеми видобування даних

1. У системі WOM вони розділили вхідну веб-сторінку на 3 зони: верхній колонтитул, тіло і нижній колонтитул за допомогою покажчиків серії 1 і серії 2, створених за їхнім спостереженням, що набір із п'яти або більше однотипних тегів <a> позначає початок зони тіла, позначеної серією 1, тоді як останній набір із 5 або більше тегів <a> вказує на кінець зони тіла, записаної вказівником серії 2. Це було зроблено для того, щоб усунути шум і звузити

пошук цільових даних, які є даними продукту, до зони тіла. Це припущення може не завжди працювати, як очікувалося, оскільки кожен веб-сайт має свій власний спосіб дизайну та презентації. Ми усунули весь цей процес, визначивши повторювані теги рівня блоку.

2. Вони класифікували дані на веб-сторінці на 6 типів об'єктів вмісту: продукт, список, форма, текст, шум і одиночний. Вони виявили конструкції для кожного і побудували

NFA, щоб ідентифікувати їх, це займає дуже багато часу і проходить під наглядом, оскільки їхній NFA продукту має бути навчений, перш ніж ідентифікувати нові дані. Ми подолали цей виснажливий процес за допомогою нашого частого алгоритму пошуку структур у розділі

3. У своїй роботі використовували NFA для ідентифікації подібних кортежів, однак веб-дані дуже сприйнятливі до змін, тому з часом багато нових структур записів даних про продукти можуть розвиватися, і їх NFA має оновити нову структуру кожного разу.

3.2 Представлення проблемної області

Для конкретного домену веб-сайтів B2C ми вибрали для копіювання всі веб-сторінки з багатими даними, тобто сторінку зі списком продуктів.). Зі загальноприйнятої структури веб-сторінки B2C, веб-сторінка списку продуктів зазвичай є сторінкою з багатими даними. Ми помітили, що сторінка зі списком продуктів зазвичай містить короткий список усіх або окремих видів продуктів. На веб-сайті B2C є набір сторінок зі списком продуктів. Ми визначаємо сторінку списку продуктів таким чином:

Визначення 1: якщо "w" - це веб-сайт B2C, а "p" - це веб-сторінка в "w", така що w є nj =1 pj , то сторінка pj є w, де j≥1, є сторінкою списку продуктів, якщо ' pj' містить набір кортежів τ типу α, де α ≥ 1, що мають відмінний тип екземпляра.

Визначення 2: блок даних містить конкретну інформацію про конкретний продукт, таку як назва продукту, ціна, марка, розмір тощо.

Якщо 'p' є веб-сторінкою, вона складається з різних наборів кортежів 't' типу α, де кожен α є набір подібних списків об'єктів і кожен об'єкт має скінченний набір атрибутів r, тоді кожен об'єкт з відповідними йому атрибути утворює блок даних.

Приклад: на рисунку показана веб-сторінка p, вона складається з різних наборів кортежів типу product, advertisements, navigation, header тощо, і кожен тип має подібний список об’єктів, наприклад, тип продукту має 8 об’єктів з набором атрибутів. як ціна, колір, марка, жорсткий диск тощо,

Ці блоки даних повинні зберігатися в базі даних як кортеж даних. У дереві HTML Dom блок даних є піддеревом дерева Dom, що має той самий або інший батьківський вузол. Теги, які використовуються для формування блоку даних, це ,<table>,<tr>,<td> тощо, відповідно до консорціуму w3.

Визначення 3: усі блоки даних на веб-сторінці разом утворюють області даних, і веб-сторінка може мати кінцевий набір областей даних залежно від типу даних, які вона представляє.

Якщо «p» є веб-сторінкою, вона складається з набору областей даних типу β, де кожен β є набір блоків даних і є кінцевими.

Приклад: на малюнку, поданому p, є веб-сторінка, яка складається з набору областей даних типу product, navigation, header, advertisements тощо, де кожна має набір блоків даних. Область даних продукту утворена 8 блоками даних, подібними за загальними атрибутами.

Після вилучення області даних у реляційну базу даних формується таблиця бази даних продукту з інформацією про кожен продукт у вигляді кортежу даних. У дереві HTML Dom область даних являє собою групу подібних блоків даних з однаковим батьківським вузлом і є рідними.

Оскільки такі веб-сторінки мають кілька областей даних з блоками даних, немає простого способу вибрати нашу цільову інформацію, якою є дані про

продукт. Крім того, цей набір ключової інформації подібний майже для всіх сторінок списку продуктів, але їх схеми можуть відрізнятися. Спостерігаючи за різними веб-сайтами B2C, ми запропонували нашу систему автоматичного вилучення веб-даних на основі цих спостережень.

Спостереження 1. Усі веб-сайти B2C представляють і стилюють свій вміст за допомогою зовнішнього CSS. Якщо всі блоки даних про продукт належать до однієї категорії, що відображаються на сторінці, мають схожий стиль, групуючи їх за допомогою тегу і призначаючи однакові стилі з файлу CSS за допомогою атрибута class.

Ми скористалися цим спостереженням, щоб розпізнати регіони даних про продукт без виконання трудомістких обчислень і класифікацій на сторінці, що робить ціле завдання непридатним.

Однак тепер, коли ми розпізнали всі області даних, однією з інших складних завдань є визначити серед них область даних продукту.

Спостереження 2. У представленні веб-сторінки в дереві Dom найбільш повторюваним набором тегів є теги блоку даних продукту. тобто базовий вихідний HTML-код веб-сторінки з багатим продуктом складається з тегів, які представляють усі дані, які відображаються, і в них ми помітили, що теги, які використовуються для представлення блоків даних про продукт, схожі один на одного, за винятком зміна змісту.

Таким чином, визначення набору безперервних тегів як структури по всій веб-сторінці звужує наш пошук області даних про продукт.

Спостереження 3: Структура блоків даних продукту – це найбільш повторюваний набір тегів html з максимальною довжиною по всій веб-сторінці.

Таким чином, наше останнє спостереження привело нас до вказівки на область даних продукту з усіма блоками продукту. Однак ми не припускаємо, що всі блоки даних є безперервними, тому ми шукаємо лише блоки, які використовують те саме ім'я класу, але ми не порівнюємо всі теги один з одним, як це робили більшість існуючих систем.

3.3 Пропонована архітектура та алгоритм видобування даних

Ми розробили архітектуру для автоматичного вилучення записів даних про продукти з веб-сайтів B2C, ми звертаємося до неї WOM_Simple, як показано на рисунку 3.1 нижче:

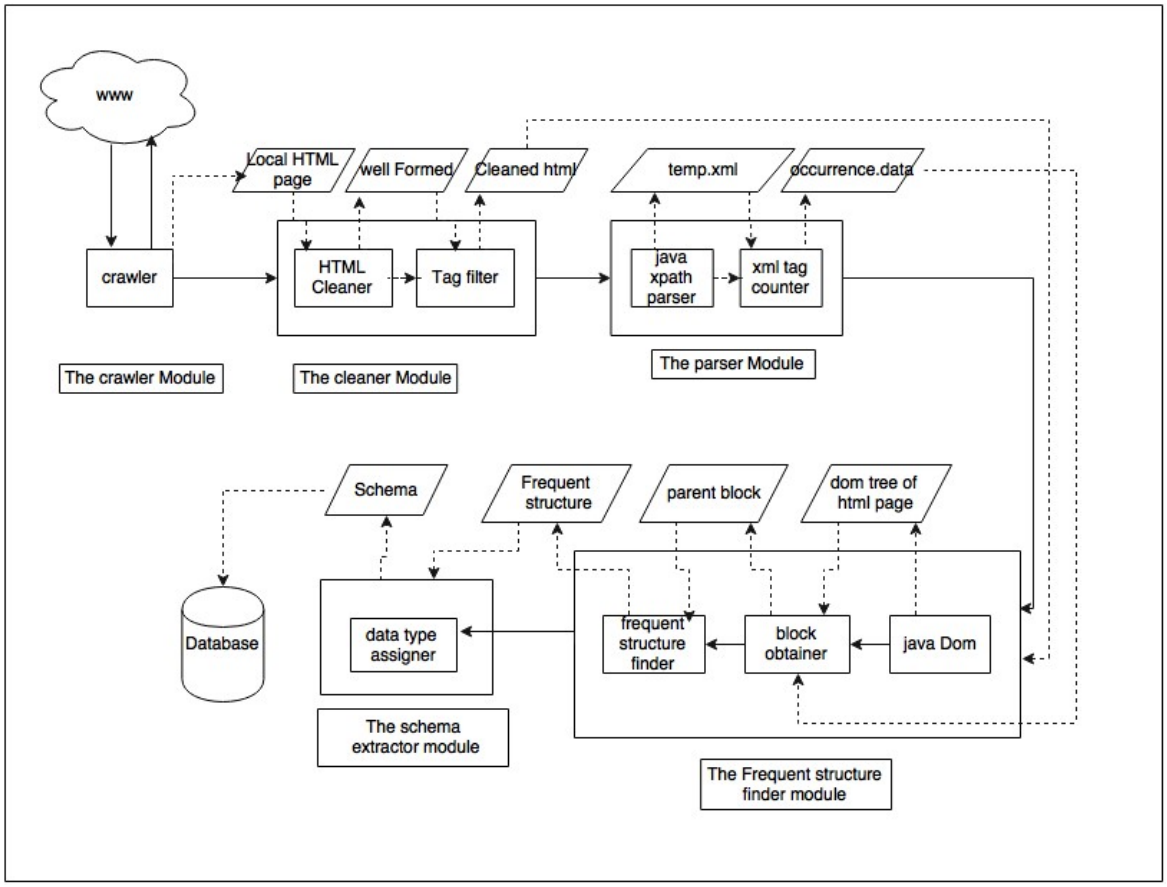


Рисунок 3.1 - Архітектура WOM_S

Архітектура на рисунку 3.1 має 5 модулів:

1) Модуль Crawler

2) Модуль Cleaner

3) Модуль Parser

4) Модуль Frequent Structure Miner

5) Модуль Schema Extractor.

Усі ці модулі послідовно викликаються нашим основним алгоритмом WOM_S, як показано на рисунку 3.2.

Алгоритм WOM_S()
Вхідні дані: веб-адреси двох або більше сторінок зі списком продуктів (Weburls).
Вихідні дані: набір схем таблиць (S) для кожного веб-документа.
Інше: набір файлів HTML (WDHTMLFile) веб-документів,
WDHTMLFile# (набір чистих файлів HTML), TempXMLFile, файл tagoccur Початок
(A) для кожної веб-адреси (Weburls)
Почніть
(1) WDHTMLFile = Викликати Crawler() для вилучення HTML веб-сторінки (Weburls) у локальний
каталог.
(2) WDHTMLFile_ = Викликати HTMLCLEANER-2.2, щоб очистити HTML-код
(3) Викличте Parser(), щоб спочатку створити тимчасовий XML-файл WDHTMLFile_ як Temp XML-файл, а потім використовувати регулярні вирази у Temp XML-файлі, щоб підсумувати кількість зустрічей усіх тегів у файлі tagoccur.
(4) Викличте засіб пошуку Frequent Structure(), щоб спочатку створити дерево DOM
DocumentBuilderFactory() за допомогою чистого html-файлу WDHTMLFile_. Потім він використовує файл tagoccur з деревом DOM, щоб ідентифікувати теги блоку та його атрибут класу, який має найчастішу подібну повторювану послідовність тегів.
(5) Викличте Schema Extractor(), який використовує структуру часто і дерево DOM, щоб пов'язати всі отримані дані з типом даних і створити знайдену схему таблиці в базі даних і вставити в неї дані.
Кінець

Рисунок 3.2 - Основний алгоритм WOM_S

Робота алгоритму та всіх модулів буде обговорена та пояснена нижче.

Модуль сканера: Наш модуль сканера відповідає за вилучення вихідного HTML-коду даної веб-сторінки продукту зі всесвітньої мережі (WWW). Цей модуль приймає в якості вхідних даних адресу веб-сторінки (URL), за допомогою якої він витягує та зберігає вихідний код HTML у файлі в каталозі локального комп'ютера як uncleaned.html. Наш модуль сканера схожий на

модуль WOM. Вони (система WOM) запропонували міні-алгоритм сканера, який сканує WWW, щоб знайти цільову веб-сторінку, транслювати весь веб-документ, включаючи теги, тексти та вміст зображень і потім він створює дзеркало оригінального веб-документа в каталозі локального комп'ютера. Він скидає коментарі з html-документа.

```
      Algorithm Crawler()
      Вхідні дані: веб-адреса сторінки зі списком продуктів
(Weburl).
      Вихід: Html-сторінка вхідної URL-адреси як
uncleaned.html у локальному каталозі Begin
      (1) URL-адреса = нова URL-адреса(inputUrl) //Створити
об'єкт URL-адреси та призначити вхідну URL-адресу
      (2) URLConnection urlConnection = url.openConnection()
//Створити об'єкт з'єднання url
      (3) BufferedReader bufferedReader = новий
BufferedReader(новий
      InputStreamReader(urlConnection.getInputStream()))
      //Загортаємо URL-з'єднання в буферизований зчитувач
      (4) у той час як ((рядок = bufferedReader.readLine())
!= нуль)
      {
      content.append(рядок + "\n")
      }
      //Читаємо URL-з'єднання за допомогою буферизованого
зчитувача
      (5) BufferedWriter writer = new BufferedWriter(new
FileWriter(uncleaned.html) Якщо записувач не має значення
NULL
      Writer.write (вміст)
      //Запис вмісту у файл uncleaned.html за допомогою
bufferedwriter
      (1)   (6) Зберегти файл uncleaned.html.
```

Рисунок 3.3 - Обхідний алгоритм

На рисунку 3.3 показано зразок веб-сторінки для введення для нашого сканера, який є веб-сторінкою з багатим продуктом із веб-сайту bestbuy.ca за URL-адресою www……../laptops.html, яка є вхідною інформацією для нашого модуля сканера. Наш алгоритм створює об’єкт URL і призначає йому вхідну URL-адресу, а потім створює об’єкт URLConnection для встановлення з’єднання

з сервером за допомогою методу OpenConnection(). Коли сервер відповідає на з'єднання зі сторінкою html, наш алгоритм загортає його в буферизований зчитувач і зчитує весь вміст, отриманий від сервера, у змінну і, нарешті, записує його у файл під назвою uncleaned.html, зберігаючи його в локальному каталозі. На цьому етапі ми маємо наш вихідний код html для обробки. На рисуннку показано вихідний HTML-код вхідної веб-сторінки, починаючи з тегу <html> і закінчуючи </html> між усім вмістом веб-сторінки у відповідних тегах.

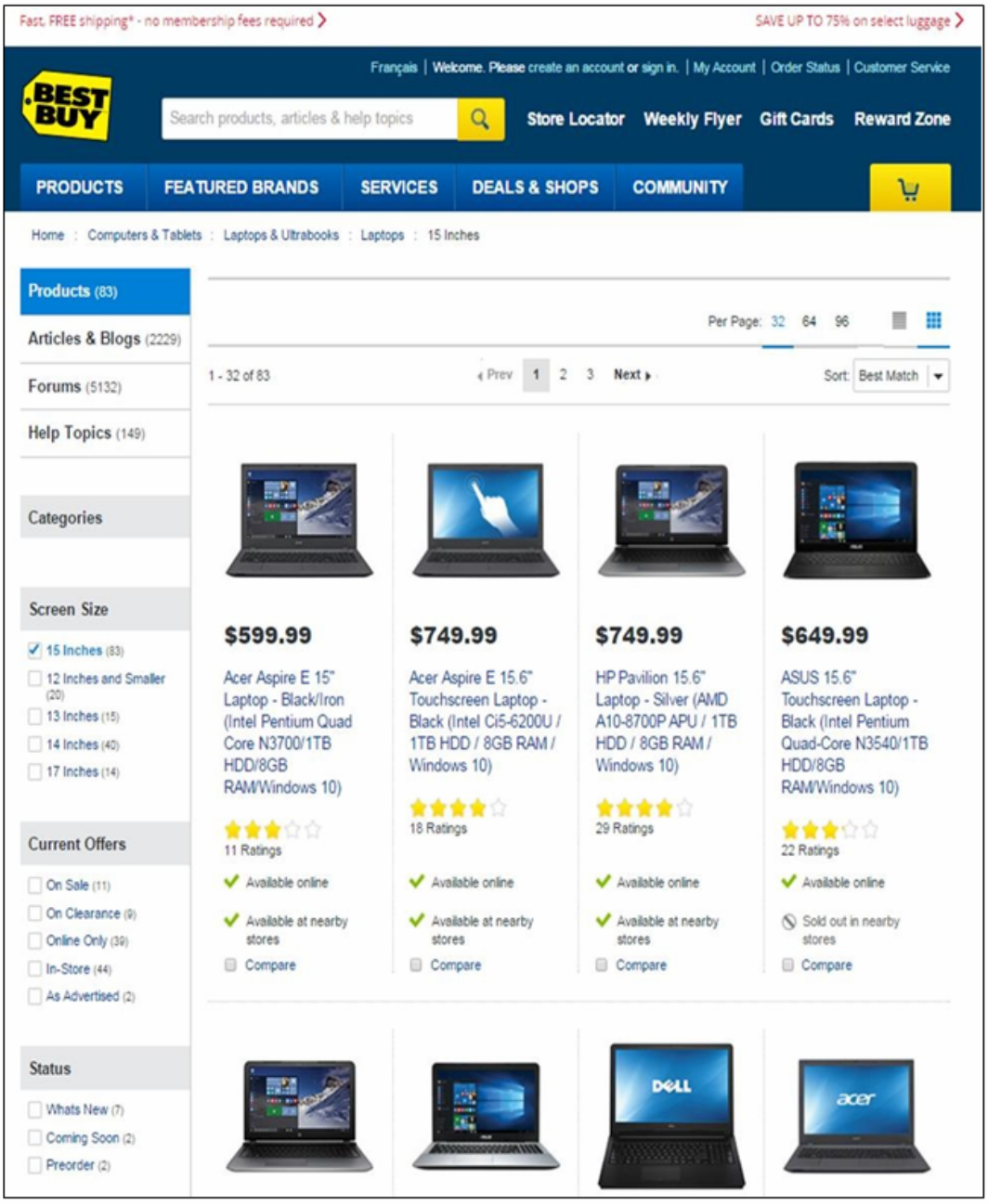


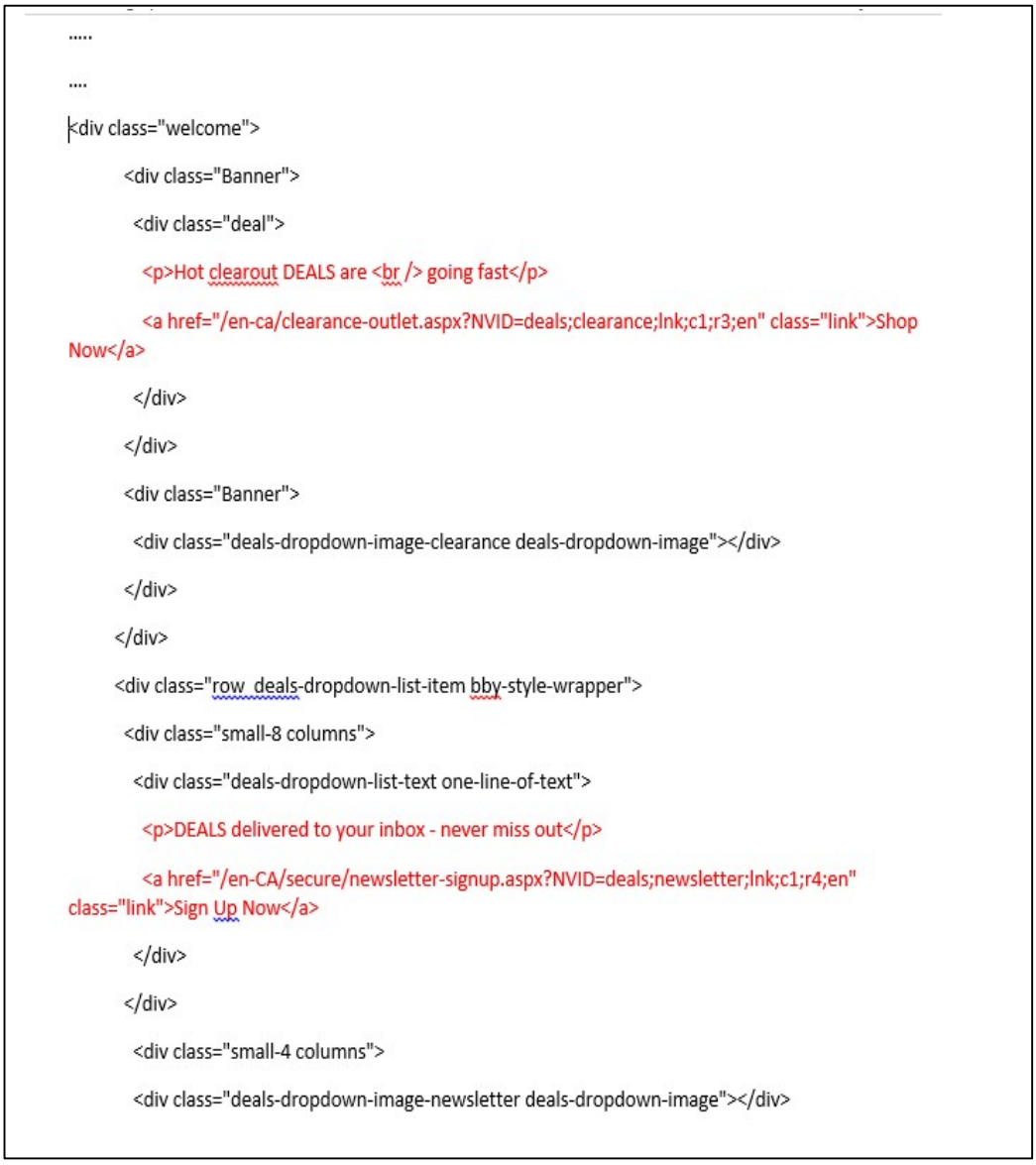
Рисунок 3.4 – Представлена структура товарів

```
      <html>
       <head>
                      <title>bestbuy</title>
       </head>
       <body>
       ……
       ……..
       <li class="top-lvl menu-item parent-item parent-
services"><a href="javascript:void(0)" class="linktop link-
services">SERVICES</a>
             <ul class="sub-nav mega-menu menu-list">
                 <li class="menu-item">
                     <div class="services-content">
                         <ul class="column-4">
                             <li class="menu-item first">
                                 <a class="icon icon-gs"
href="/en-
      CA/GeekSquad.aspx?NVID=Services;Geek%20Squad;im;en">Geek
Squad</a>
                                 <p>From computer glitches to
smartphone mishaps, Geek Squad can save the day. Our Agents
are ready and waiting to set up, support, protect and
repair.</p>
                                 <a href="/en-CA/category/geek-
      squad/22042a.aspx?NVID=Services;Geek%20Squad;im;en">Learn
more &gt;</a>                  <div class="prod-saving">
……..
       ……..
       …….
       ……
       ……..
       </body>
      </html>
```

Рисунок 3.5 – Вигляд Uncleaned.html

HTML не є структурованою мовою, теги не закриті або відсутні, неправильне впорядкування дуже поширені у вихідному коді html веб-сторінок. Неможливо побудувати дерево Dom з неправильно відформатованого html-коду, тому необроблений вихідний код html у локальному каталозі як uncleaned.html необхідно очистити перед будь-якою подальшою обробкою. Ідея очищення вихідного коду була успадкована від WOM, однак для цієї мети ми використали набагато просунутий очищувач. JTidy — це синтаксичний аналізатор HTML із

відкритим кодом у Java, який використовується для очищення деформованого HTML, щоб вставити відсутні теги, змінити порядок тегів, атрибутів або тексту, видалити коментарі та перетворити його на добре сформований Xhtml.



```
.....
....
<div class="welcome">
        <div class="Banner">
         <div class="deal">
          <p>Hot clearout DEALS are <br /> going fast</p>
          <a href="/en-ca/clearance-outlet.aspx?NVID=deals;clearance;lnk;c1;r3;en" class="link">Shop
Now</a>
         </div>
        </div>
        <div class="Banner">
         <div class="deals-dropdown-image-clearance deals-dropdown-image"></div>
        </div>
       </div>
       <div class="row  deals-dropdown-list-item bby-style-wrapper">
        <div class="small-8 columns">
         <div class="deals-dropdown-list-text one-line-of-text">
          <p>DEALS delivered to your inbox - never miss out</p>
          <a href="/en-CA/secure/newsletter-signup.aspx?NVID=deals;newsletter;lnk;c1;r4;en"
class="link">Sign Up Now</a>
         </div>
        </div>
         <div class="small-4 columns">
         <div class="deals-dropdown-image-newsletter deals-dropdown-image"></div>
```

Рисунок 3.6 - Очищення файлу HTML

Вхідним для модуля очищення є файл uncleaned.html, він обробляється JTidy відповідно до встановлених властивостей, а очищений файл Xhtml буде збережено в локальному каталозі як cleaned.xhtml. На малюнку виділено очищений фрагмент файлу Xhtml, який був згенерований після очищення вихідного html-файлу.

Необхідно перетворити наш чистий html-файл із кроку 2 у тимчасовий файл XML, який містить лише теги рівня блоку, щоб ми могли оцінити теги, щоб отримати їх кількість зустрічей. Модуль синтаксичного аналізатора приймає на вхід очищений html-файл з попереднього модуля. Тимчасовий файл xml (наприклад, малюнок) створюється шляхом запису всіх тегів рівня блоку, таких як , <table>, <tr>, , <li>, <ul> тощо, з вхідних даних html сторінка. У запущеному прикладі такі теги, як , і т.д. з файлу cleaned.xhtml записуються у тимчасовий файл xml. Під час створення тимчасового файлу xml всі атрибути в цих тегах рівня блоку, як-от «id», «src» тощо, були виключені, і збережено лише атрибут class, наприклад , у робочому прикладі оскільки атрибут class містить інформацію про те, який саме клас CSS використовується цими тегами, що дійсно допомагає нам знайти блок, який часто зустрічається з тим самим стилем. На малюнку показано тимчасовий файл xml для зразка сторінки введення з попередніх модулів, де всі теги блоку були записані лише з атрибутами класу. Алгоритм тепер викликає метод search_string(), який використовує розбір регулярних виразів Java, щоб перевірити наявність кожного тегу з відповідним атрибутом класу у всьому тимчасовому файлі xml. Таким чином, підсумок знайдених повторюваних тегів (тих, що зустрічаються принаймні 3 рази, оскільки рідко зустрічаються веб-сайти B2C з переліком менше 3 продуктів) разом із кількістю зустрічей зберігаються у файлі появи тегів. У запущеному прикладі зустрічався 1 раз, тег >div class= “banner”> зустрічався 2 рази у файлі temp.xml, а потім інші повторювані теги. На малюнку показано файл появи тегу для запущеного прикладу з усіма тегами блоку та їх входженнями.

```
        аналізатор алгоритму()
      {
      Вхідний очищений html-файл
      Вивести тимчасовий файл xml з тегами блоку
          Файл даних про зустріч із кількістю зустрічей
тегів
      Почніть
      1) Arraylist occurnodelist[ ]
      2) Для кожного рядка очищеного html-файлу
      3) рядок IF є тегом блоку
      4) Запишіть рядок у тимчасовий файл xml
      5) Кінець ЯКЩО
      6) Кінець для
      7) Для кожного рядка тимчасового xml
      8) Викликати функцію searchString(), передаючи
тимчасовий файл xml та рядок як параметр
      9) він повертає кількість зустрічей цього рядка
      10) Якщо occurnodelist[ ] не містить рядка
      11) Додати рядок до списку вузлів[ ]
      12) Запишіть рядок і кількість його входжень у файл
occurrence.data
      13) Кінець для
      Кінець
```

Рисунок 3.7 - Синтаксичний аналізатор алгоритму

```
       Алгоритм пошуку рядка()
      Вхідні дані: рядок із тимчасового файлу xml
      Вихід: кількість входжень даного рядка
      Почніть
      1) Створіть об'єкт для регулярного виразу Java
matcher matcher()
      2) Створіть об'єкт відповідника
      3) Для кожного рядка тимчасового xml
      4) Якщо рядка немає в списку фрагментів[]
      5) За допомогою об'єкта збігу перевірте, чи поточний
рядок у тимчасовому xml
      Відповідає рядку введення
      6) Підрахунок приросту
      7) Кінець Якщо
      8) Підрахунок повернення;
      9) Кінець
```

Рисунок 3.8 - Рядок_пошуку алгоритму

```
       <divclass=“welcome">
      <divclass=“bannar">
      <divclass="supporting-info">
      <divclass=“product”>
      <divclass=“deal”>
      <divclass= “prodwrap”>
      <divclass=“prod-im”>
<divclass=“p-img”> ……..
      ……..
      ……..
      <divclass=“review”>
```

Рисунок 3.9 - Тимчасовий файл Xml

```
<divclass=“product">1
  <divclass=“bannar">2
<divclass="supporting-info">4
<divclass=“welcome”>1
  <divclass=“deal”>4
<divclass= “prodwrap”>12
  <divclass=“prod-im”>12
<divclass=“p-img”>12
    ……..
……..

……..
    ……..
<divclass=“review”>9
```

Рисунок 3.10 - Файл даних про випадки

Модуль частого пошуку структури

Мета модуля частого пошуку структури — знайти найбільш часто повторювані блоки з більшою кількістю дочірніх елементів на веб-сторінці, які мають схожий стиль. Ці функції представлені у вихідному коді html як пошук часто повторюваних тегів блоків з однаковою назвою класу у всьому дереві

Dom, які є нашими блоками даних продукту. Список вузлів struct_list використовується для зберігання блоків продукту-кандидата, nodelength містить кількість повторень блоку продукту-кандидата, no of childs містить кількість дочірніх елементів для блоку продукту, і спочатку обидва значення дорівнюють нулю. NodeList data_tuples містить блок Final Product. Спочатку дерево Dom будується з файлу cleaned.xhtml, згенерованого в другому модулі. Цей Dom є представленням дерева всього вихідного коду html, обговорюваного в розділі, і модель Dom дерева використовується для представлення алгоритму у зрозумілій формі на рисунку 3.11 Іншим вхідним значенням для алгоритму fs_finder є файл тегів із синтаксичного аналізатора. Файл появи тегів містить усі теги блоку з назвою класу та кількістю повторень.

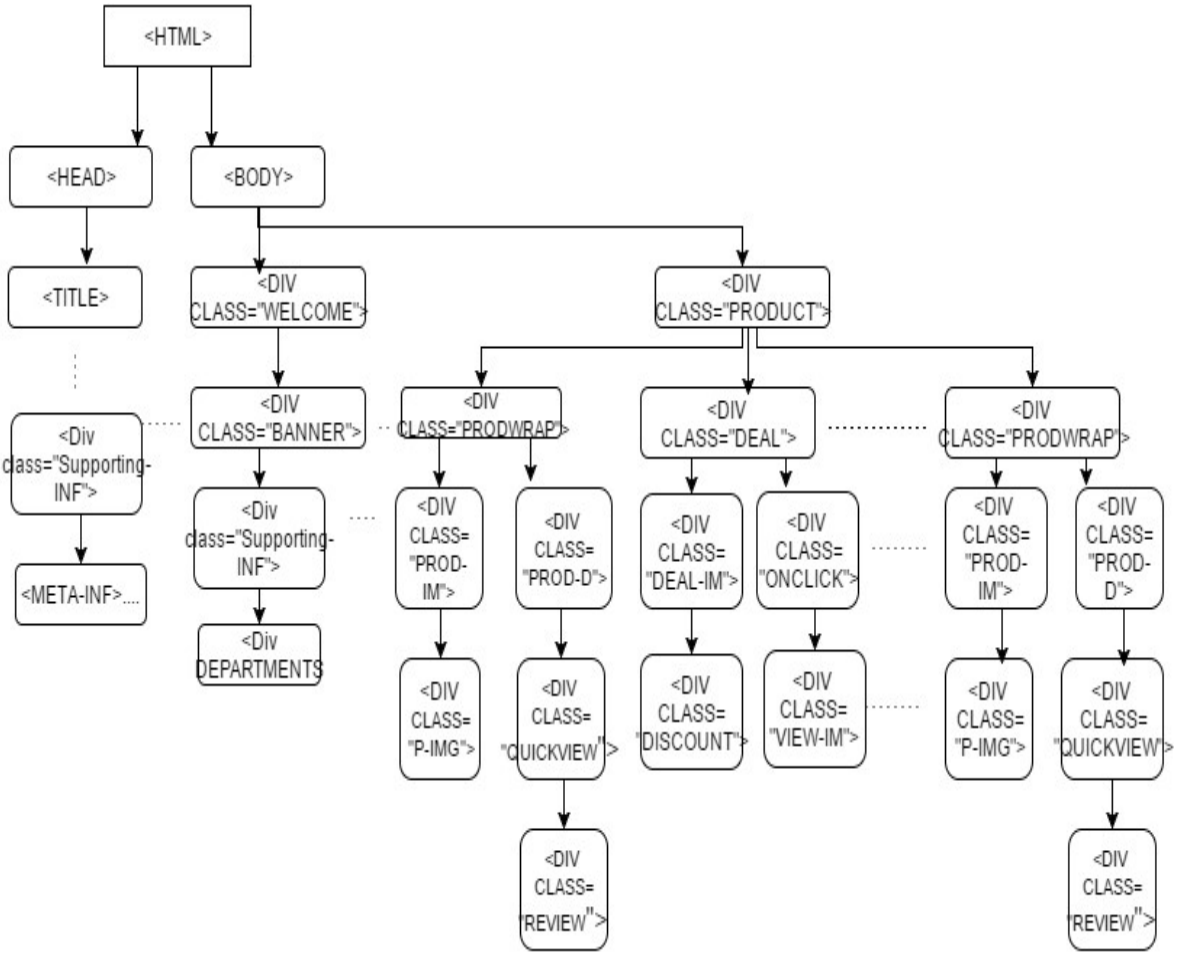


Рисунок 3.11 – Вигляд DOM-дерева

Для кожного тегу у файлі тегу, який має кількість зустрічей принаймні 3 (припускаючи, що веб-сторінка списку продуктів містить щонайменше 3 продукти), генерується динамічний вираз Xpath, а дерево Dom запитується, щоб знайти входження цього тегу. У поточному прикладі тег <divclass=“product"> опущено, оскільки його кількість менше 3, так само тег . Тег <div class=” supportinginfo”> зустрічався 4 рази, оскільки він задовольняє умову, що вираз Xpath генерується як path = "//div[@class='supporting-info']", і кожне входження цього тегу по всьому дереву Dom знайдено та присвоєно списку вузлів (рис. 42) для знайдіть кількість зустрічей за допомогою методу getlength() і кількість дочірніх вузлів за допомогою методу getchildnodes(), при цьому вони призначені для nodelength =4 і no_of_childs= 1 відповідно. Зверніть увагу, що тег має відбулося 4 рази по всьому дереву Dom, оскільки ми не можемо показати все дерево, лише 1 його поява видно у фрагменті, рисунок 3.12

<div class= ″ supporting-info″>	<div class= ″ supporting-info″>	<div class= ″ supporting-info″>	<div class= ″ supporting-info″>

Рисунок 3.12 - Список вузлів struct_List

Наступний тег <divclass=“deal”> також обробляється таким же чином, генеруючи шлях виразу Xpath = "//div[@class='deal']", кожне входження зберігається в NodeList, а значення nodelength дорівнює 4, а кількість дочірніх — 4 (як показано на малюнку), ці значення більші за попередні, тому data_tuples оновлюється до . Наступний тег <divclass= “prodwrap”> має довжину вузла = 12 і кількість дочірніх = 5 (малюнок), тому data_tuples тепер оновлено до . Наступним є <divclass=“prod-im”> з

nodelength=12, кількість дочірніх елементів = 1 наша умова, що блок продукту повинен мати максимальну кількість дочірніх елементів і повторюватися максимальну кількість разів, не буде задоволено цим тегом, оскільки хоча кількість довжина вузлів дорівнює кількості дочірніх елементів, тому цей тег не розглядається, а значення не оновлюються. Той самий процес триває до кінця файлу відтворення тегу, таким чином знаходить найбільш часто повторюваний блок з більшою кількістю дочірніх елементів і знайдений блок продукту в списку даних NodeList. У робочому прикладі блок продукту, виявлений алгоритмом, є , який має 5 дочірніх елементів і 12 випадків.

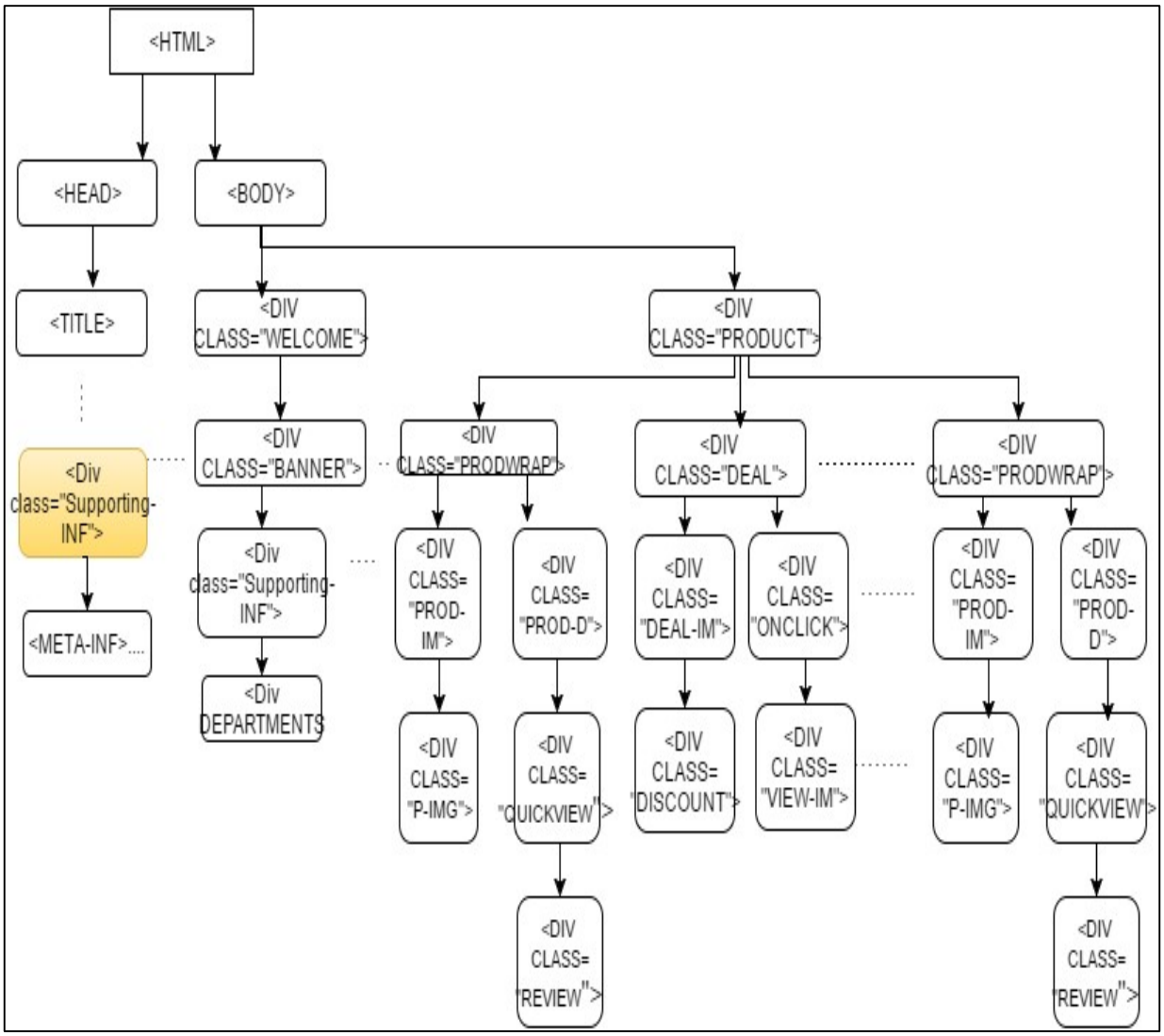


Рисунок 3.13 - Дерево Dom

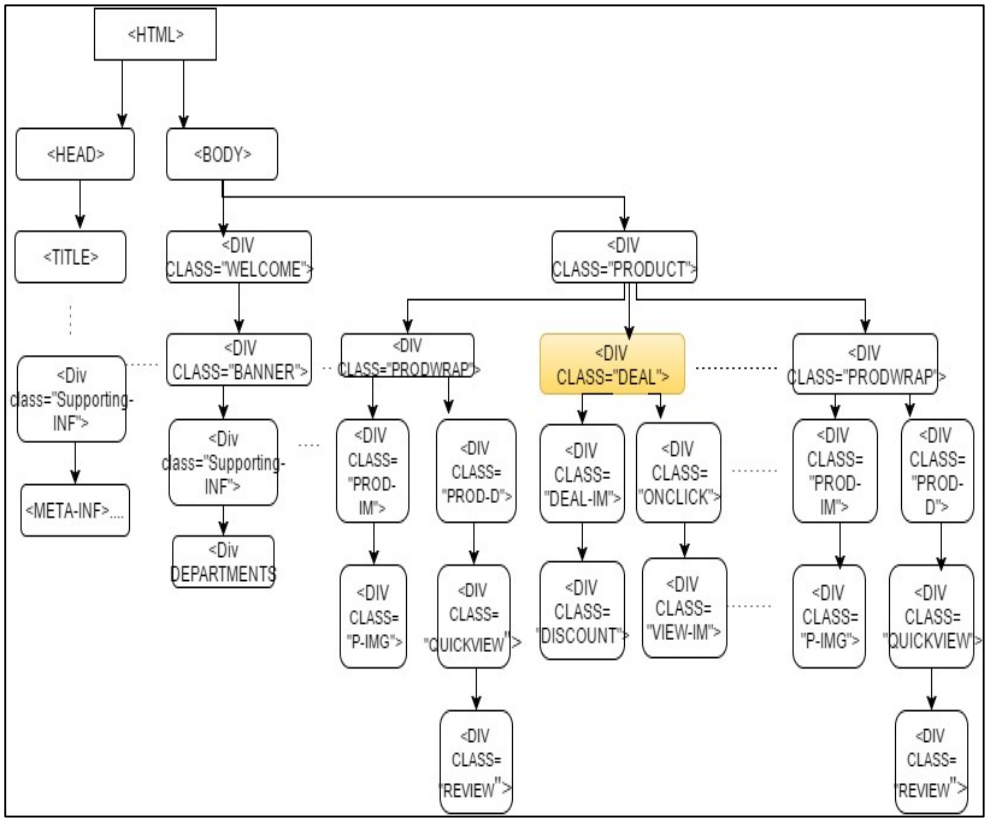
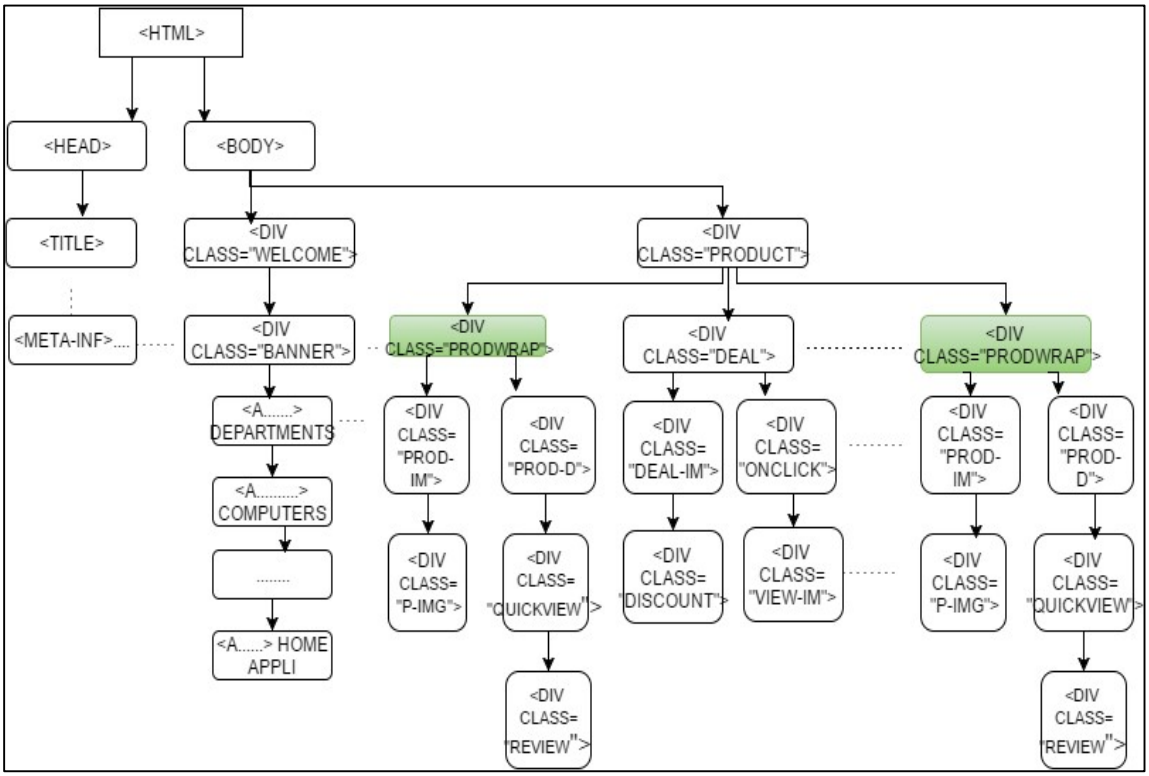


Рисунок 3.14. – Класова структура реалізації



>

Рисунок 3.15 - DOM - структура <div class= "prodwrap"

Використовуючи батьківський тег, його повний блок витягується з дерева Dom, у нашому запущеному прикладі тег всі його дочірні елементи витягуються та зберігаються в NodeList для виявлення схеми блоку продукту.

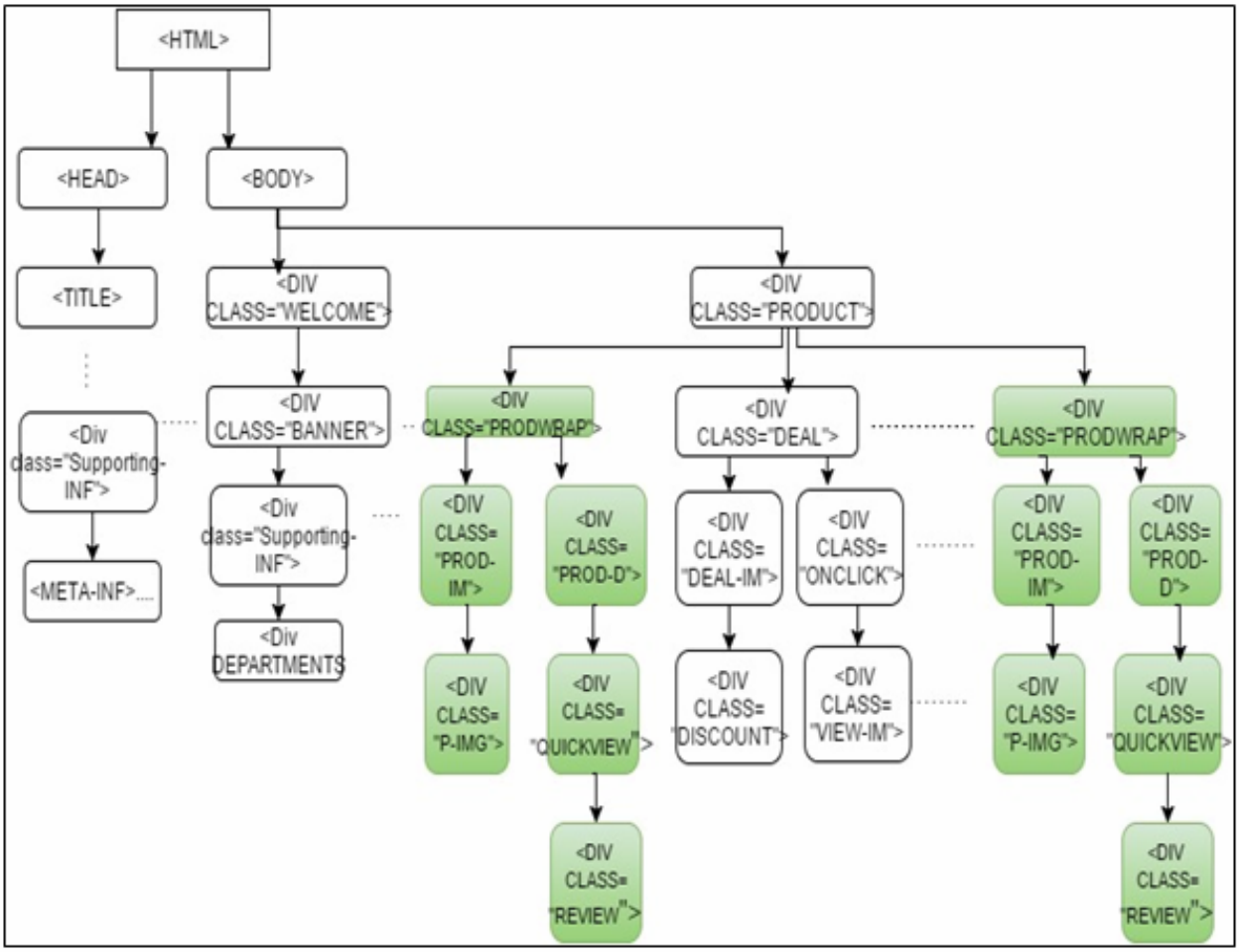


Рисунок 3.16 - Тегова структура рішення

Для батьківського тегу продукту всі його дочірні елементи витягуються в глибині першого пошуку, як структура дерева Dom, представлена на рисунку 3.17, і зберігаються в NodeList.

Алгоритм FSfinder()
Вхідні дані: файл даних про появу (відбувається), очищений файл html (очистити)
Вихід: Часта структура списку вузлів (FRS) = null (для назв блоків продуктів), кортежи даних (для значень блоків продукту) інше: Domtree: файл
FINAL nodelength = 0, кількість дочірніх елементів = 0, кінцеві дочірні елементи = 0 xmlnodetag: рядок (для повного вузла блоку), xpathExpression expr: рядок, шлях: рядок попередній блок продукту: ціле число
Структура тегів списку вузлів, список структур, кортеж даних, блок продукту xmlnodetag IDCount = 0, довжина вузлів: ціле Пакети: Java DOM, Java Xpath, Java transformer begin
1. Domtree = Виклик DocumentBuilderFactory(clean)
2. Створіть об'єкт xpath для xpathfactory(), щоб використовувати його методи для отримання всіх вузлів, які відповідають виразу шляху з дерева Dom.
3. для кожного тегу xml у файлі зустрічаються тегів,
3.1 Отриманий xml зберігається в xmlnodetag
3.2 шлях = “//div[@class=’“+xmlnodetag+”’]”; (Атрибут класу в xmlnodetag призначається змінній динамічно в кожному циклі)
3.3 xpathExpression expr =xpath.compile(path)
3.4 struct list= xpath.evaluate(Domtree, XPathConstants.NODESET)
// Це отримує всі вузли, які відповідають виразу шляху, використовуючи об'єкт xpath для цього xml
3.5 довжина вузла = структура list.getlength()
3.6 no of childs= struct list(0).getchildnodes().getlength()
3.7 Якщо (довжина вузла >= FINAL nodelelongth and no-of-child > final childs)
3.7.1 блок продукту = xmlnodetag
3.7.2 FINAL nodelength=довжина вузла
3.7.3 кінцеві дочірні вузли = кількість дочірних вузлів
3.7.4 кортеж даних = кінець списку структур для
4. якщо у блоку продукту є дочірній
5. для кожного дочірнього вузла в блоці продукту Почніть
5.1 FRS = FRS додати дочірній вузол до частої структури
5.2 Прочитати наступний вузол
6. Повернути кінець кортежу даних

Рисунок 3.17 - Алгоритм fs_finder

```
      < divclass = “prodwrap’’ >

              <h4 class="prodTitle">
                        <a
id="ctl00_CP_ctl00_C3_ctl00_LT_SL_ctl01_ctl00_CalltoAction"

      href="http://www.....................................
..............  ">Asus 5’6
         laptop</a>
               < divclass = “prodImage’’ >

                      < divclass = “prod – image’’ >

          <img src=“bestbuy/1043.img”> < /div > < /div >
      < divclass = “prodDetails’’ >
          <h4 class=“prod-Ram”>500GB</h4>
           <h4 class=“prod-HDD”>4GB</h4>
            <h4 class=“operat”>Windows 10</h4> < /div >
                < divclass = “ProdPrice’’ >
            <h1>$678”</h1> < /div >
        < /div > < divclass = “clear’’ > . . . < /div > < /div >
```

Рисунок 3.18 - Часта структура

Модуль Schema Extractor

Цей модуль виконує завдання вилучення як схеми таблиці, так і кортежів продукту з частої структури (послідовності тегів). Він перетворює схему таблиці бази даних, отриману за допомогою

Алгоритм FSFinder до схеми сховища даних шляхом додавання атрибута інтеграції під назвою «storeid» та атрибута історії, який називається «time». Ці два атрибути дозволять інтегрувати кортежі з різних веб-сайтів з різними мітками часу для порівняльних та історичних запитів. Кортеж добутків P — це

добуток з атрибутами a1, a2, a3, . . ., an. Цей модуль приймає як вхідні дані як шаблон частої структури, так і список вузлів data_tuples, який містить усі блоки даних продукту, які потрібно витягти та зберегти в базі даних. Екстрактор схем призначає відповідні типи даних на основі тегів типовому шаблону структури для схеми таблиці. Якщо це тег < img >, буде призначено зображення типу даних, якщо це теги < p >, < span >, < li >, < a >, буде призначено тип даних «рядок» тощо. , схема сховища даних виявлена на веб-сторінці. Це можна використовувати для створення та оновлення таблиці бази даних і подальшого створення сховища даних, коли до схеми таблиці додаються інтегративний атрибут «storeid» та історичний атрибут «time». Наявність схеми сховища даних дозволяє проводити порівняльний аналіз та історичні запити.

```
      Algorithm schemaFinder()
      Input: Frequent Structure
      Output: Data Base table with product
records
      Begin
      1)    String Web_Schema
      2)    FOR Each tag IN frequent structure
      3)    IF tag = img
      4)    Assign data type blob and add this to
web schema
      5)    ELSE IF tag = text
      6)    Assign data type varchar and add this
to web schema
      7)    END IF
      8)    END FOR
      9)    Create Table web_Schema
      End
      1)    For Each product block in data_tuples
      2)    For Each Tag in product block
      3)    If tag = img
      4)    Content += Retrieve src || data-url
      5)    Else if tag = text
      6)    Content += retrived text data
      7)    End if
      8)    End For
      9)    Insert content into database table
      10)   End For
```

Рисунок 3.19 - Екстрактор схем

Після визначення бази даних або схеми сховища фактичні екземпляри кортежів даних витягуються зі списку вузлів data_tuples, який містить усі блоки продукту з їхніми батьківськими вузлами. З дерева Dom кожен батьківський вузол перевіряється, чи є у нього дочірні вузли, і вміст оновлюється в базі даних/сховище.

Store id	Time	Column_1	Column_2	Column_3	Column_4	Column_5	Column_6	Column_7	Column_8
bestbuy_1	31-052016 11:00	Asus 5’6 laptop	bestbuy/1043.img	I5 processor	4GB	Looks good	$678	500GB	3
…	….	….	….	…	…	….	…	….	….

Рисунок 3.20 - Згенерована таблиця бази даних

Ми виконали реалізацію наших алгоритмів і представили їх у найбільш надійній, масштабованій та узагальненій формі. Потрібні додаткові вдосконалення наших алгоритмів нашої системи, щоб зробити їх достатньо надійними для роботи зі складними комерційними веб-сайтами та їхніми структурами. Наш модуль сканера має бути автоматизований, щоб позитивні веб-сторінки можна було виявити без участі людини. Наша система використовує JTidy для очищення необроблених HTML-сторінок, але розробка нашого власного цілеспрямованого очищувача для видалення шуму та очищення веб-сторінки може зменшити перевантаження наступних модулів з точки зору порівнянь та IO. Проте ми детально опишемо нашу попередню обробку та обмеження в наступному розділі. На даний момент ми протестували нашу систему на веб-сторінках у режимі реального часу з різних веб-сайтів із різними структурами та доменами, щоб продемонструвати нашу надійність та зручність використання.

3.4. Емпіричні оцінки пропонованого рішення

Ми перевірили нашу систему на 50 веб-сторінках у режимі реального часу, кожна з яких належить до іншого веб-сайту з різними продуктами та має різну структуру. Записи даних про продукти з кожного веб-сайту належать до різних доменів, таких як електроніка, музика, обладнання, подорожі, одяг, аксесуари тощо. Наша система реалізована мовою програмування Java. Потім ми запускаємо нашу систему в 32-розрядній операційній системі Windows 10, 8 ГБ оперативної пам'яті з процесором Intel core i5 2,60 ГГц Lenovo ThinkPad Machine. Ми використали стандартну точність і відкликання, щоб оцінити нашу систему як для кількості записів даних, які потрібно отримати, так і для їх кількості стовпців у схемі. Точність вимірюється як середнє у відсотках від кількості правильних блоків продукту, отриманих системою, до кількості вилучених блоків продукту, а відкликання вимірюється як середнє у відсотках від кількості правильних блоків продукту, отриманих системою, до фактичної кількості. блоки даних про продукт на веб-сторінці.

3.5. Результати імітаційного експерименту

Метою нашого експерименту є вимірювання продуктивності системи WOM_S для вилучення записів даних. Таблиця 1 показує широкомасштабні експериментальні результати як показник продуктивності нашої системи WOM_S. Ми взяли одну сторінку для кожного веб-сайту для експерименту, і цифри в стовпці «Записи фактичних даних» показують кількість блоків продуктів на веб-сторінці, включаючи кількість стовпців у схемі для кожної веб-сторінки. Інший стовпець «вилучена кількість записів даних» показує кількість записів, які система ідентифікувала як записи даних про продукт, і кількість стовпців даних, які вона витягла. Загальна сума рядка вказує, що коли було 1489 записів даних, система витягнула всі їх точно, і вона не пропустила жодних

цільових даних і не видалила жодного шуму. Хоча алгоритм також показав хороші результати у вилученні схем із 342 стовпців загалом, він витягнув усі з них, однак він витягнув 16 невідповідних стовпців для 14 схем веб-сторінок, які не мають великого значення в базі даних продукту.

Ми помітили, що причиною цих невідповідних стовпців є шум, який був вбудований в блок продукту на веб-сторінці, тут шум означає текстові дані в самому блоці продукту, які не належать як атрибут об'єкту продукту. Наприклад, прапорець під кожним продуктом, що дозволяє користувачам порівнювати інші продукти, має значення текстового тегу, як «порівняння», «вибрати порівняння» або «швидкий перегляд» тощо, що є бітом шуму, оскільки він знаходиться всередині блоку даних продукту. у нас немає жодних методів, щоб видалити цей шум. Наприклад, як показано на рисунку 3.21.

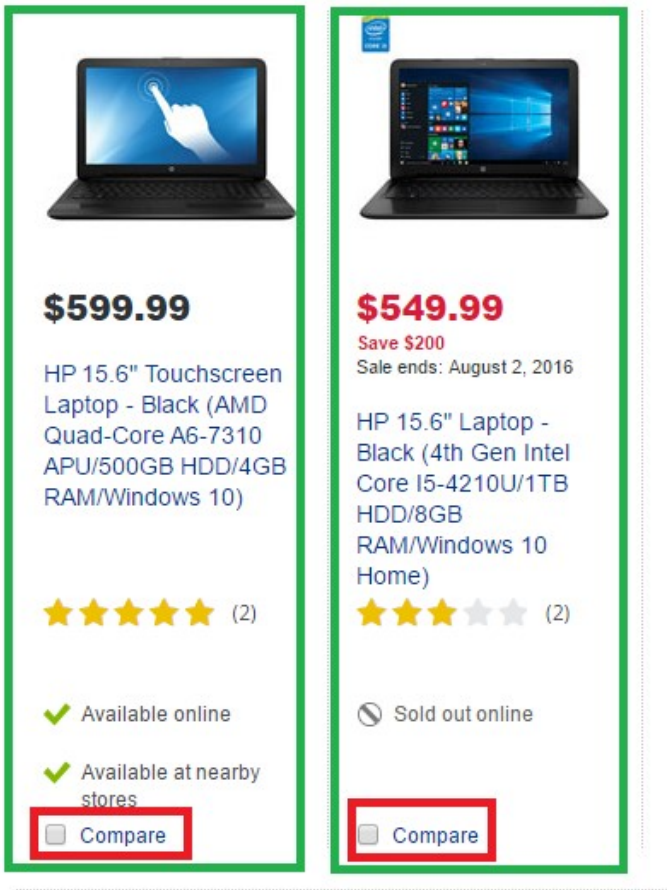


Рисунок 3.21 - Вилучений шум

3.6. Порівняння пропонованого алгоритму з WOM

Наша система працює краще, ніж WOM з точки зору масштабованості та розширюваності, які є двома ключовими аспектами веб-витягувачів даних. Масштабованість дуже важлива для системи вилучення даних, оскільки всесвітня павутина є величезною, і видобуток цільових даних із неї має бути можливим за посильну кількість часу, а також точний.

1. WOM послідовно порівнює кожен NFA з об'єктом контенту в масиві об'єктів контенту, і він має 6 типів NFA для однієї веб-сторінки, їх система дуже складна і займає більшу кількість часу в обчисленні порівняно з WOM_S. Також WOM виконує непотрібні обчислення, оскільки він витягує не тільки кортежи продуктів, але й інші 5 об’єктів типу вмісту, які взагалі не вибираються нашою системою. Ми перевірили продуктивність обох систем на великому веб-сайті електронної комерції (www.bestbuy.ca), який містить понад 16 категорій товарів (дитячі, електроніка, одяг тощо), кожна з яких має ще 15 підкатегорій (електроніка -> ноутбуки, мобільні телефони тощо) із в середньому 80 продуктами на кожній веб-сторінці є близько 20 000 товарів. Нашій системі знадобилося 838 мілісекунд, щоб ідентифікувати, витягти та створити таблицю сховища даних для веб-сторінки з 32 продуктами, тоді як WOM потребував 1276 мілісекунд для оновлення вже створеної бази даних. У наступному випадку перевірити чи це твердження є вірним.

Наявність більшої кількості продуктів на одній веб-сторінці збільшує час роботи. Ми протестували обидві системи на веб-сторінці з 96 продуктами. Результати показали лише незначне збільшення часу виконання в нашій системі, тоді як час виконання WOM збільшився майже вдвічі. Нарешті, щоб перевірити масштабованість, ми запустили обидві системи на всьому великому веб-сайті (www.bestbuy.ca з 20 000 продуктами), що доводить, що наша система має високу масштабованість, ніж WOM (7,66 хвилин), оскільки вона займає майже половину часу.4.00 хв.

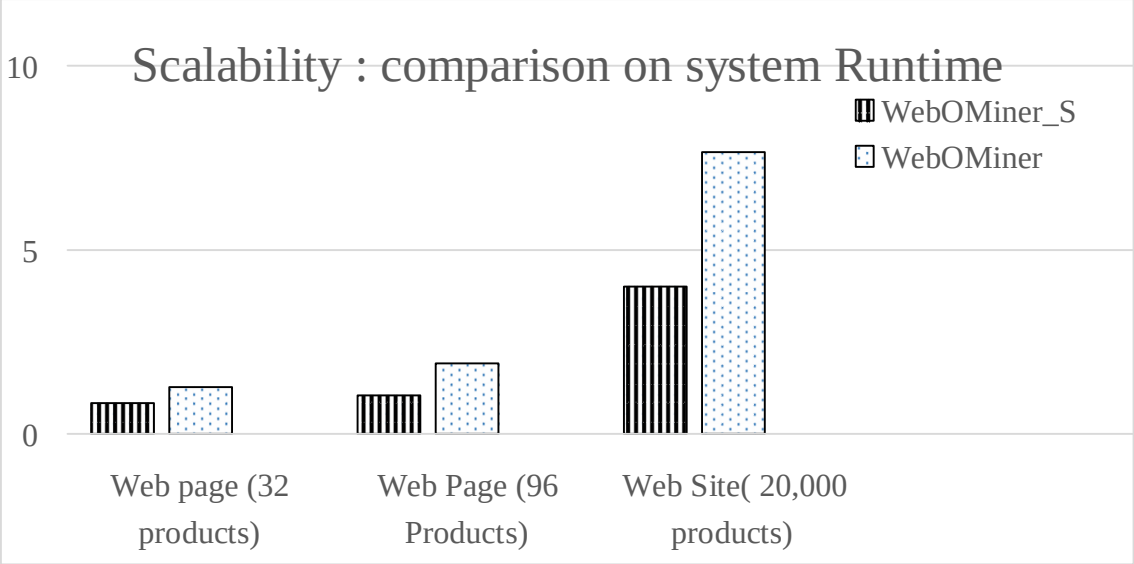


Рисунок 3.22 - Порівняння часу виконання системи

2. Залежність у процесі вилучення. WOM навчає свої NFA ідентифікувати кортежі продуктів, таким чином створюючи залежність між раніше отриманими даними продукту та новими, що робить його непридатним до різноманітних доменів, але запропонована система є повністю незалежною та динамічною в вилучення даних.

3. З точки зору продуктивності та коректності система досягла 100% точності та відкликання при ідентифікації блоків даних продукту. З іншого боку, наша виявлена схема має 95,55% точність і 100% відкликання при виявленні схеми веб-сторінок, де, оскільки WOM не може отримати схему автоматично, а їх схема попередньо визначена і однакова для кожного продукту.

Таблиці продуктів, що генеруються динамічно, не мають відповідних міток стовпців або конкретного впорядкування стовпців даних, а для кожної веб-сторінки на основі їхньої частої структури виявляється їхня схема.

Це є обмеженням, оскільки для інтеграції даних про продукти з кількох таблиць (bestbuy, Costco) необхідні конкретні стовпці (наприклад, ціна від bestbuy і ціна від Costco) не можуть бути автоматично виявлені для інтеграції, оскільки система не знатиме, який стовпець є ціною. . Щоб подолати цю

проблему, назви стовпців потрібно дати належним чином. Цю проблему можна вирішити за допомогою деяких спеціальних функцій, таких як:

1) Якщо всі елементи даних у стовпці містять знак «$», назвіть стовпець «ціна» та призначте типу даних ціле число.

2) Якщо довжина даних у стовпці перевищує 18, назвіть це «опис».

Якщо дані містять одну або дві цифри без знаку $, назвіть їх як «рейтинги».

ВИСНОВКИ

В дипломній роботі виконали реалізацію наших алгоритмів і представили їх у найбільш надійній, масштабованій та узагальненій формі. Потрібні додаткові вдосконалення наших алгоритмів нашої системи, щоб зробити їх достатньо надійними для роботи зі складними комерційними веб-сайтами та їхніми структурами. Наш модуль сканера має бути автоматизований, щоб веб-сторінки можна було виявити без участі людини.

Пропонована система використовує JTidy для очищення необроблених HTML-сторінок, але розробка власного цілеспрямованого очищувача для видалення шуму та очищення веб-сторінки може зменшити перевантаження наступних модулів з точки зору порівнянь та IO. На даний момент ми протестували нашу систему на веб-сторінках у режимі реального часу з різних веб-сайтів із різними структурами та доменами, щоб продемонструвати нашу надійність та зручність використання. Ми перевірили нашу систему на веб-сторінках у режимі реального часу, кожна з яких належить до іншого веб-сайту з різними продуктами та має різну структуру. Записи даних про продукти з кожного веб-сайту належать до різних доменів. Наша система реалізована мовою програмування Java.

Ми використали стандартну точність і відкликання, щоб оцінити нашу систему як для кількості записів даних, які потрібно отримати, так і для їх кількості стовпців у схемі. Точність вимірюється як середнє у відсотках від кількості правильних блоків продукту, отриманих системою, до кількості вилучених блоків продукту, а відкликання вимірюється як середнє у відсотках від кількості правильних блоків продукту, отриманих системою, до фактичної кількості блоків даних про продукт на веб-сторінці.

Реалізована система працює краще з точки зору масштабованості та розширюваності, які є двома ключовими аспектами веб-витягувачів даних.

СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА

[1] S. Abiteboul. Querying semi-structured data. In F. N. Afrati and P. Kolaitis, editors, Database Theory ICDT '97, 6th International Conference, Delphi, Greece, January 8-10, 1997, Proceedings, volume 1186 of Lecture Notes in Computer Science, pages 1-18. Springer, 1997.

[2] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Wiener. The lorel query language for semistructured data. Int. J. on Digital Libraries, 1(1):68-88, 1997.

[3] H. Ahonen, O. Heinonen, M. Klemettinen, and A. Verkamo. Applying data mining techniques for descriptive phrase extraction in digital document collections. In Advances in Digital Libraries (ADL'98), Santa Barbara, California, USA, April 1998, 1998.

[4] H. Ahonen, O. Heinonen, M. Klemettinen, and A. Verkamo. Finding co-occurring text phrases by combining sequence and frequent set discovery. In R. Feldman, editor, Proceedings of 16th International Joint Conference on Artificial Intelligence IJCAI-99 Workshop on Te~ Mining: Foundations, Techniques and Applications, pages 1-9, 1999.

[5] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study: Final report. In Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop, 1998, 1998.

[6] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In Proceedings of the glst annual international ACM SIGIR conference on Research and development in information retrieval August 2~ - gS, 1998, pages 37-45, Melbourne Australia, 1998.

[7] D. E. Appelt and D. Israel. Introduction to information extraction technology. In Proceedings of 16th International Joint Conference on Artificial Intelligence IJCAI-99, Tutorial, 1999.

[8] G. O. Arocena and A. O. Mendelzon. Weboqh Restructuring documents, databases, and webs. Theory and Practice of Object Systems, 5(3):127-141, 1999. SIGKDD Explorations. Volume 2, Issue 1 - page 10

[9] P. Atzeni and G. Mecca. Cut & paste. In Proceedings of the Sixteenth A CM SIGA CT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 1P-1~, 1997, Tucson, Arizona, pages 144-153. ACM Press, 1997.

[10] R. Baeza-Yates and e. Berthier Ribeiro-Neto. Modern Information Retrieval. Addison-Wesley Longman Publishing Company, 1999.

[11] M. Balabanovi'c and Y. Shoham. Fab: Contentbased, collaborative recommendation. Communications of the ACM, 40(3):66-70, 1997.

[12] A. Biichner, M. Baumgarten, S. Anand, M. Mulvenna, and J. Hughes. Navigation pattern discovery from internet data. In Proceedings of the WEBKDD '99 Workshop on Web Usage Analysis and User Profiling, August 15, 1999, San Diego, CA, USA, 1999.

[13] K. Bharat and M. 1~. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval August PJ - ~8, 1998, pages 104-111, Melbourne Australia, 1998.

[14] D. Billsus and M. Pazzani. A hybrid user model for news story classification. In Proceedings of the Seventh International Conference on User Modeling (UM '99), Banff, Canada, 1999.

[15] J. Borges and M. Levene. Mining association rules in hypertext databases. In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), August 27-31, 1998, New York City, New York, USA, 1998.

[16] 3. Borges and M. Levene. Data mining of user navigation patterns. In Proceedings of the WBBKDD'99 Workshop on Web Usage Analysis and User Profiling, August 15, 1999, San Diego, CA, USA, pages 31-36, 1999.

[17] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. In Seventh International World Wide Web Conference, Brisbane, Australia, 1998.

_[18] P. Buneman. Semistructured data. In Proceedings of the Sixteenth A CM SIGA CT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 1~-14, 1997, Tucson, Arizona, pages 117-121. ACM Press, 1997.

[19] P. Buneman, S. B. Davidson, G. G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In H. V. Jagadish and I. S. Mumick, editors, Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June J-6, 1996, pages 505-516. ACM Press, 1996.

[20] J. Carbonell, M. Craven, S. Fienberg, T. Mitchell, and Y. Yang. Report on the conald workshop on learning from text and the web. In CONALD Workshop on Learning from Text and the Web, June, 1998, 1998.

[21] J. Caxbonell, Y. Yang, and W. Cohen. Special issue of machine learning on information retrieval introduction. Machine Learning, 39:99-101, 2000.

[22] C. Cardie. Empirical methods in information extraction. AI Magazine, 18(4):65-79, 1997.

[23] S. Chakrabarti. Data mining for hypertext: A tutorial survey. ACM SIGKDD Explorations, 1(2):1-11, 2000.

[24] S. Chakrabarti, B. Dora, D. Gibson, J. Kleinberg, S. Kumar, P. P~ghavan, S. tL~jagopalan, and A. Tomkins. Mining the link structure of the world wide web. IEEE Computer, 32(8):60-67, 1999.

[25] S. Chakrabarti, B. Dora, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In L. M. Haas and A. Tiwary, editors, SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-~, 1998, Seattle, Washington, USA, pages 307-318. ACM Press, 1998.

[26] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The tsimmis project: Integration of heterogeneous information sources. In Proceedings of the lOth Meeting of the Information Processing Society of Japan, pages 7-18, 1994.

[27] W.W. Cohen. Learning to classify english text with ilp methods. In Advances in Inductive Logic Programming (Ed. L. De Raedt), IOS Press, 1995.

[28] W.W. Cohen. Some practical observations on integration of web information. In A CM SIGMOD Workshop on The Web and Databases (WebDB'99), pages 55-60, Philadelphia, Pennsylvania, USA, 1999.

[29] W. W. Cohen. What can we learn from the web? In Proceedings of the Sixteenth International Conference on Machine Learning (ICML '99), pages 515-521, 1999.

[30] R. Cooley, B. Mobasher, and J. Srivastava. Web mining: Information and pattern discovery on the world wide web. In Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97), 1997.

[31] R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining world wide web browsing patterns. Knowledge and Information Systems, 1(1), 1999.

[32] R. W. Cooley. Web Usage Mining: Discovery and Application of Interesting Patterns from Web data. PhD thesis, Dept. of Computer Science, University of Minnesota, May 2000.

[33] J. Cowie and W. Lehnert. Information extraction. Communications of the ACM, 39(1):80-91, 1996.

[34] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, mad S. Slattery. Learning to extract symbolic knowledge from the world wide web. In Proceedings of the Fifteenth National Conference on Artificial Intellligence (AAAI98), pages 509-516, 1998. SIGKDD Explorations. Volume 2, Issue 1 - page II

[35] F. Crimmins, A. Smeaton, T. Dkaki, and J. Mothe. T~trafusion: Information discovery on the internet. IEEE Intelligent Systems, 14(4):55-62, 1999.

[36] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. Journal of the American Society for Information Science, 41(6):391-407, 1990.

[37] L. Dehaspe and L. de Raedt. Mining association rules in multiple relations. In Proceedings of the 7th International Workshop on Inductive Logic Programming, volume 1297 of Lecture Notes in Computer Science, pages 125-132, Prague, Czech Republic, 1997. Springer.

[38] J. A. Delgado. Agent-Based Information Filtering and Recommender System On the Internet. PhD thesis, Dept. of Intelligence Computer Science, Nagoya Institute of Technology, March 2000.

[39] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In Proceedings of the 1998 A CM 7th international conference on Information and knowledge management, pages 148-155, Washington United States, 1998.

[40] J. S.. T. Eliassi-Rad. Intelligent agents for web-based tasks: An advice-taking approach. In Working Notes of the AAAI/ICML-98 Workshop on Learning for Text Categorization, Madison, WI, pages 588-589, 1999.

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи бакалавра: "Імплементація схем видобування даних в ВЕБ-просторі"

Обсяг пояснювальної записки: 83 аркуші

Дата закінчення дипломної роботи 20 червня 2022 р.

Підпис студента _____