МАГІСТЕРСЬКА РОБОТА

МР. ІПм - 73.00.00.000 ПЗ

Група ІПм-22-6

Гайдичук Андрій

2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Гайдичук Андрій Миколайович

(прізвище, ім'я, по батькові)

УДК 004.942

(індекс)

МАГІСТЕРСЬКА РОБОТА

Моделі, методи та алгоритми веб-скрапінгу засобами машинного навчання

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Гайдичук А.М.

(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Пасєка Микола Степанович, д.т.н., професор

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

В.о. завідувача кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Рецензент

проф. Лютак І.З.

(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
В.о. зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ 04 ” вересня 2023 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Гайдичуку Андрію Миколайовичу

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи “ Моделі, методи та алгоритми веб-скрапінгу засобами машинного навчання ”

керівник проекту (роботи) Пасєка Микола Степанович, д.т.н., професор

затверджені наказом закладу вищої освіти від “ 18 ” грудня 2023 р. № 738/7

2. Строк подання студентом проекту (роботи) 20 лютого 2024 р.

3. Вихідні дані до проекту (роботи) Теоретичні концепції та формальні моделі побудови та функціонування інформаційних технологій машинного навчання

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

1. Аналіз предметної області застосування технології веб-скрапінгу
2. Методи та алгоритми машинного навчання для аналізу текстів процесу веб-скрапінгу
3. Представлення архітектури системи та інтеграція інструментів для процесів веб-скрапінгу
4 Реалізація системи виконання процесу веб-скрапінгу для ресурсів електронної комерції

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

1. Схематичне зображення методів штучного інтелекту (рис. 1.1)
2. Схематичне зображення процесу веб-скрапінгу (рис. 1.2)
3. Етапи веб-скрапінгу (рис. 1.3)
4. Схема роботи веб-скрапера (рис. 1.4)
5. Класифікація інструментів веб-скрапінгу (рис. 1.5)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Нормоконтроль	доц., к.т.н. Вовк Р.Б.	
Перевірка на плагіат	доц., к.т.н. Вовк Р.Б.	

7. Дата видачі завдання 04 вересня 2023 р.

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури по темі магістерської роботи	01.10.2023	виконано
2	Аналіз предметної області застосування технології веб-скрапінгу	27.10.2023	виконано
3	Методи та алгоритми машинного навчання для аналізу текстів процесу веб-скрапінгу	15.11.2023	виконано
4	Представлення архітектури системи та інтеграція інструментів для процесів веб-скрапінгу	22.11.2023	виконано
5	Реалізація системи виконання процесу веб-скрапінгу для ресурсів електронної комерції	12.12.2023	виконано
6	Реалізація функціональності запропонованої технології	15.01.2024	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	20.02.2024	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 77 с., 30 рис., 47 джерел.

Тема: Моделі, методи та алгоритми веб-скрапінгу засобами машинного навчання

Об'єктом дослідження є процес веб-скрапінгу текстових даних з подальшою обробкою засобами машинного навчання.

Метою дослідження є дослідження моделей і методів та створення програмного інструментарію процесу веб-скрапінгу структурованих даних з веб-сторінок Інтернет магазинів ресурсів для подальшої класифікації отриманої інформації засобами машинного навчання.

Предмет дослідження – моделі, методи та засоби веб-скрапінгу та аналізу структурованих текстових даних.

Результати дослідження:

В роботі розроблено простий до імплементації алгоритм видобування структурованої інформації, який суміщує в собі процеси краулінгу та скрапінгу, таким чином спрощуючи інтеграційні та експлуатаційні процеси використання алгоритму

Висновок

Виконано проектування системи веб-скрепінгу для інтернет-магазину, що враховує як функціональні, так і технічні аспекти, що сприяє створенню ефективної та надійної інфраструктури для збору та обробки даних. Також було реалізовано інтерфейс користувача, а також безпосередньо систему веб-скрепінгу для збору даних Інтернет магазинів, що відповідає всім заявленим функціональним вимогам до системи.

ВЕБ-СКРАПІНГ, КРАУЛІНГ, ПОШУК ПОСИЛАНЬ, МАШИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, АНАЛІЗ ДАНИХ, ДОБУВАННЯ ДАНИХ, ІНТЕРНЕТ МАГАЗИН, ПАРСЕР

ABSTRACT

Master's thesis: 77 pages, 30 figures, 47 sources.

Topic: Models, methods and algorithms of web scraping by means of machine learning

The object of the research is the process of web scraping of text data with further processing by means of machine learning.

The purpose of the study is to research models and methods and create a software toolkit for the process of web scraping of structured data from web pages of Internet resource stores for further classification of the received information by means of machine learning.

The subject of research is models, methods and means of web scraping and analysis of structured text data.

Research results:

In the work, an easy-to-implement algorithm for extracting structured information was developed, which combines the processes of crawling and scraping, thus simplifying the integration and operational processes of using the algorithm

Conclusion

The design of a web scraping system for an online store was carried out, taking into account both functional and technical aspects, which contributes to the creation of an efficient and reliable infrastructure for data collection and processing. A user interface was also implemented, as well as a web scraping system for collecting data from online stores, which meets all the stated functional requirements for the system.

WEB SCRAPING, CRAWLING, LINK SEARCH, MACHINE LEARNING, NEURAL NETWORKS, DATA ANALYSIS, DATA MINING, ONLINE SHOPPING, PARSER

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

DOM – Document Object Model

GET-запит – запит, що виконується для отримання вмісту вказаного ресурсу

URL – Uniform Resource Locator

SEO – Search Engine Optimization

.htaccess – конфігураційний файл веб-сервера Apache, який дозволяє керувати роботою веб-сервера

CAPTCHA – completely automated public turing test to tell computers and humans apart

AI - Artificial intelligence

DB - Data base

EC2 - Elastic compute cloud

ORM - Object-relational mapping

RSS - Really simple syndication

S3 - Simple storage service

SQS - Simple queue service

ВСТУП

Актуальність теми.

Стрімкий зріст інтернет-технологій, а також здешевлення та покращення доступності як і користувацьких пристроїв з доступом у інтернет, так і серверних потужностей для створення власного контенту у мережі. Мільярди тільки проіндексованих пошуковими сервісами сторінок містять дані на будь-яку тематику, і як і об’єм, так і різноманіття цих даних з кожним роком тільки збільшується. Завдяки цьому інтернет став важливим, потужним та всеосяжним джерелом абсолютного різноманіття неструктурованих даних.

Дані, а особливо – видобута з даних істотна інформація є ключовими у прийнятті рішень, здійснення досліджень, тощо. Коректні, повні дані у достатньому об’ємі дають можливість розуміння досліджуваного предмету, явища, проблеми. Саме тому видобування даних з мережі Інтернет є розвиненою та поширеною практикою як у веденні бізнесу, так і у здійсненні досліджень.

Обсяг даних, які постійно розміщуються та передаються з допомогою Інтернет, збільшується з великою швидкістю. Глобальність та доступність Інтернет дозволяє отримувати та використовувати результати праці людей, що знаходяться у вільному доступі, і ділитися власними. Часто пошук в Інтернет передбачає не лише ознайомлення з тією чи іншою інформацією з метою отримання нових знань, а й порівняння, аналіз та узагальнення даних, які знаходяться на різних вебсайтах та сторінках. Наприклад, для порівняння та аналізу цін, перелікута порівнянняхарактеристик товарів, збору контактної інформації для залучення потенційних клієнтівтощо. Поряд з цим багато веб-сайтів не дозволяють користувачам напряму зберігати дані для особистого використання. Виходом із ситуації може бути монотонне ручне копіювання потрібної інформації у власні файли та форматування їх відповідно до потреб. Такий процес,

безумовно, є неефективним, оскільки вимагає багато часу.Багато великих вебсайтів, наприклад, Google, Twitter, Facebook, StackOverflow тощо, мають API, які дозволяють отримати доступ до їхніх даних у структурованому форматі. Це найкращий варіант, але є інші сайти, які не дозволяють користувачам отримувати доступ до великих обсягів даних у структурованій формі. Веб-скрапінг – це автоматизований процес отримання великого обсягу даних із вебсайтів таперетворення їх в структуровані дані [1, 2]. Програми, які здійснюють такий процес, називаються вебскраперами і здатні отримувати потрібний вміст HTML, працювати з JavaScript, фільтрувати отримані даніта виводити їх у формі готових баз даних, таблиць Excel, файлів CSV або окремих APIтощо, а також оминати встановлені сайтами обмеження.

Якісний веб-скрапінг є поширеною та популярною послугою, однак для отримання якісних масових даних з багатьох ресурсів і досі потрібно розробляти спеціалізовані екстрактори даних з конкретних ресурсів або закладати істотні фінансові витрати на розробку таких екстракторів підрядниками які на цьому спеціалізуються або закладати співставні витрати на придбання ліцензій промислових сервісів, які надають API для використання високотехнологічних екстракторів, побудованих на алгоритмах машинного зору та застосовують засекречені алгоритми очистки та перевірки видобутих даних.

Однак, для багатьох досліджень та бізнесів такий підхід є фінансово недосяжним, отже у простих аналітичних алгоритмів є місце для розвитку – як з точки зору дешевизни та простоти розробки та підтримки, так і у невибагливості до умов експлуатації (наявних обчислювальних потужностей. Дана робота присвячена поліпшенню простих алгоритмів та доведенню доцільності їх використання.

Процеси веб-скрапінгу залишаються дуже актуальними в сучасному світі з розвитком цифрової економіки та величезного обсягу даних, що

доступні онлайн. Ось кілька ключових причин, чому веб-скрапінг залишається актуальним:

- дані як конкурентна перевага: Багато компаній використовують дані з Інтернету для аналізу ринків, конкурентів, ціноутворення тощо. Веб-скрапінг дозволяє отримувати ці дані з різних джерел і використовувати їх для прийняття стратегічних рішень.

- моніторинг ринків та трендів: Здійснювати моніторинг ринків через автоматизовані засоби стає набагато ефективніше за ручне збирання і аналіз даних. Веб-скрапінг допомагає підтримувати компанії в курсі останніх трендів і змін на ринку.

- поповнення баз даних: Веб-скрапінг використовується для автоматичного оновлення та поповнення баз даних компаній, довідників, каталогів тощо.

- Аналіз настроїв і відгуків: Деякі компанії використовують веб-скрапінг для аналізу відгуків клієнтів або соціальних медіа для зрозуміння настроїв споживачів щодо їхніх продуктів або послуг.

- Фінансовий аналіз: У фінансовому секторі веб-скрапінг використовується для отримання фінансових даних, котирувань акцій, збору інформації про компанії тощо.

Хоча веб-скрапінг є потужним інструментом, важливо дотримуватися відповідних етичних та юридичних меж у процесі його використання, оскільки незаконне або неетичне скрапінг може призвести до юридичних проблем або негативного впливу на репутацію компанії.

Метою дослідження є дослідження моделей і методів та створення програмного інструментарію процесу веб-скрапінгу структурованих даних з веб-сторінок Інтернет магазинів ресурсів для подальшої класифікації отриманої інформації засобами машинного навчання.

Для досягнення поставленої мети потрібно **виконати наступні завдання:**

- провести огляд існуючих підходів та програмних аналогів у областях веб-скрапінгу даних з веб-ресурсів та оцінки їх якості;

- реалізувати алгоритми витягування, підготовки та класифікації даних;

- порівняти результати, отримані розробленим алгоритмом та результатами тренування алгоритмів машинного навчання на даних, видобутих ним з існуючим аналогом та результатами тренування на даних аналогу;

- виконати реалізацію системи здійснення процесу веб-скрапінгу для ресурсів електронної комерції.

Об'єктом дослідження є процес веб-скрапінгу текстових даних з подальшою обробкою засобами машинного навчання.

Предметом дослідження є моделі, методи та засоби веб-скрапінгу та аналізу структурованих текстових даних.

Наукова новизна одержаних результатів.

Було створено алгоритм у якому суміщено процеси пошуку посилань та видобування інформації, доведено доцільність використання простих алгоритмів для збору даних з ресурсів у мережі Інтернет з ціллю використання у тренуванні алгоритмів машинного навчання.

Практичне значення магістерської роботи полягає в розробці системи здійснення веб-скрапингу для збору даних Інтернет магазинів які були отримані дані про продукти з таких інтернет магазинів як Amazon та Walmart. Також з використанням вже підготовленого і натренованого AI було проаналізовано відгуки про товар.

Структура магістерської роботи. Робота складається зі вступу, трьох розділів та висновків. Загальний обсяг роботи становить 77 сторінок і містить 30 рисунків, список використаних джерел із 47 найменувань.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЗАСТОСУВАННЯ ТЕХНОЛОГІЇ ВЕБ-СКРАПІНГУ

1.1. Архітектура та особливості процесу веб-скрапінгу

Обробка інформації є ключовим етапом в багатьох дослідницьких проектах, аналітиці, маркетингових дослідженнях та інших сферах, де необхідно отримати об'єктивну та релевантну інформацію. Першим етапом є збір даних. Він має вирішальне значення для отримання якісних результатів та досягнення поставлених цілей. Для цього можуть використовуватися такі технології як веб-скрапінг (англ. web scraping) та API (англ. Application Programming Interface). Веб-скрапінг це технологія, яка полягає в автоматичному зборі даних з веб-сайтів. Він використовується для отриманням великих об'ємів даних з різних джерел в інтернеті, зокрема з соціальних мереж, новинних порталів, інтернет-магазинів та інших веб-ресурсів. Для збору даних використовуються спеціальні програми, які називаються скраперами або павуками.

Вони працюють за допомогою HTTP-запитів до веб-сторінок, зчитуючи HTML-код і видобуваючи з нього потрібну інформацію. Іноді скрапери використовують технології штучного інтелекту та навчання з підкріпленням, щоб автоматизувати процес вибору необхідної інформації. Іншим методом отримання даних з інтернету є технологія API (англ. Application Programming Interface) – інтерфейс програмування додатків, що дозволяє різним програмним системам взаємодіяти між собою. API надає стандартний спосіб для інтерактивної комунікації між різними програмними системами, дозволяючи їм обмінюватися даними, запитами та відповідями. Дану технологію можна уявити як міст між двома програмними системами. Вона надає засіб для передачі даних між ними, а також правила для того, як ці дані повинні бути передані. API може бути використаний для доступу до різних функцій та сервісів, таких як соціальні мережі, веб-служби, онлайн-магазини,

різноманітні сервіси мережевої інфраструктури та ін. API може мати різні форми та протоколи передачі даних. Найбільш поширеними формами API є REST (Representational State Transfer) та SOAP (Simple Object Access Protocol). REST API дозволяє отримувати доступ до ресурсів в мережі за допомогою HTTP-запитів GET, POST, PUT та DELETE. SOAP API передає дані у форматі XML, використовуючи HTTP-протокол. Збір даних є лише першим етапом, після отримання масиву інформації виконується обробка даних, аналіз, структуризація, кластеризація, класифікація та подальше зберігання. Найпоширенішими методами, що використовуються при обробці інформації в інтернет-просторі є штучний інтелект, а саме машинне навчання. На рис. 1.1 зображена структура методів штучного інтелекту.

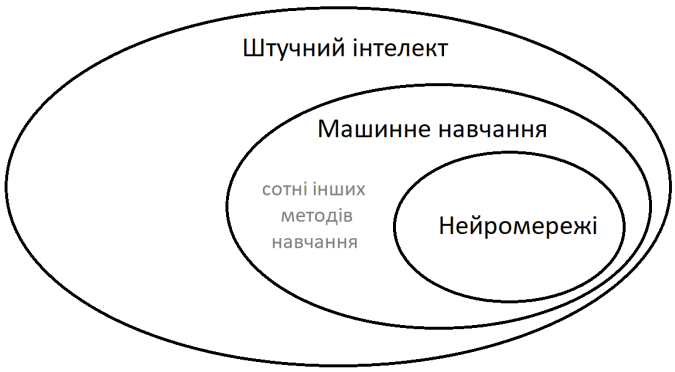

Рис. 1.1. Схематичне зображення методів штучного інтелекту

Веб-скрапінг – це автоматизований збір даних з різноманітних інтернет-ресурсів. Загальний принцип його роботи можна пояснити наступним чином: певний автоматизований код виконує GET-запити на сервер, отримує відповідь, аналізує HTML-документ, шукає потрібну інформацію та перетворює її в попередньо вибраний формат.

У свою чергу, веб-скрапер – це програма, спроектована для автоматичного збору даних з інтернету та подальшої їх обробки. Тобто, веб-скрапери дозволяють автоматизувати доступ до веб-сайтів, імітуючи при цьому поведінку людини. Ці інструменти дозволяють автоматично

отримувати нові чи оновлені дані та зберігати їх для подальшої обробки та використання [1].

Збирання даних у відкритих джерелах Інтернет використовується для різних бізнес-цілей, зокрема, для дослідження ринку та аналізу конкурентів, для заповнення стрічки новин, для створення навчальних наборів даних для моделей машинного навчання, для відстеження змін цін на веб-сайтах електронної комерції, з метою залучення потенційних клієнтів для формування списків адрес електронної пошти та номерів телефонів тощо.

Ручне видобування таких даних передбачає безпосереднє виділення та копіювання інформації з веб-сторінок або ж використання інструментів розробника веб-оглядача та роботу з вихідним кодом веб-сторінки. В такому випадку з допомогою буфера обміну потрібні дані вставляють у файл, зберігають та використовують далі відповідно до поставленого завдання для досягнення певної мети. Автоматизований процес вибірки потрібних даних з веб-сторінок передбачає використання готових інструментів – вебскраперів (рис.1.2), що, безумовно, є ефективнішим.

Рис. 1.2. Схематичне зображення процесу веб-скрапінгу

Як правило, такі програмні засоби працюють наступним чином [3]:

- інструмент збирання програмно надсилає HTTP-запити на сервери, на яких розміщені цільові веб-сторінки;

- сервери повертають вихідний код HTML для цільових сторінок;

-інструмент збирання аналізує HTML і витягує потрібні дані;

-витягнуті дані зберігаються дляподальшого аналізу або обробки.

В наш час веб-скрапінг використовують з багатьох причин: агресивна конкуренція, Інтернет-перегони, шахрайство, хакерські атаки та спам. Веб-скрапінг також використовують, щоб без особливих зусиль викрасти будь-які потрібні дані. Вони часто імітують звичайну поведінку користувачів, що ускладнює їх виявлення та блокування.

Деякі автоматизовані інструменти також надають розширені функції, наприклад, здатність обробляти файли cookie або обходити Умови використання сайту, які забороняють або обмежують копіювання вмісту. Взагалі кажучи, для сканування веб-сторінок потрібні дві частини, а саме сканер і веб-скрапер. Сканер (краулер) – це алгоритм штучного інтелекту, який переглядає веб-сторінки для пошуку конкретних необхідних даних, переходячи за посиланнями в Інтернеті. Веб-скрапер призначений для отримання даних із веб-сайту. Конструкції веб-скраперів можуть дуже відрізнятися залежно від складності та обсягу проєкту, щоб вони могли швидко й точно отримувати дані.

Можливі сценарії використання інструментів веб-скрапінгу:

• Збір даних для маркетингових досліджень;

• Збір даних з інтернет-ресурсів, що мають інформаційний характер (новинні ресурси, електронні видання, газети та журнали);

• Створення баз даних на основі відкритих контактних даних (електронні адреси користувачів, номери телефонів);

• Збір даних з інтернет-магазинів для моніторингу цін у власних цілях;

Викрадення авторського контенту для подальшого використання (статті, зображення, відео);

• Збір інформації про спеціальні пропозиції та знижки;

• Збір інформації з порталів для продажу нерухомості;

• Збір інформації з порталів для пошуку роботи;

• Інтеграція даних з декількох джерел;

• Моніторинг сайтів для бронювання квитків, готелів;

• Моніторинг даних погоди;

• Збір урядових даних.

Якщо деякі з наведених сфер використання веб-скраперів не є критичними, то інші можуть завдати серйозних збитків компаніям, виходити за рамки етики, моралі та, в окремих випадках, навіть законів.

Веб-скрапінг складається з наступних основних, тісно зв'язаних етапів: аналіз веб-сайтів, сканування веб-сайтів та організація і структуризація даних (рис. 1.3).

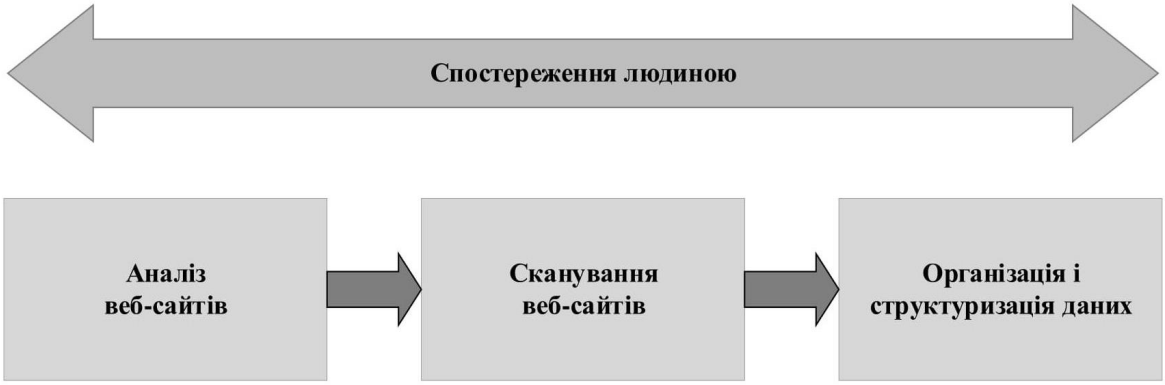

Рис. 1.3. Етапи веб-скрапінгу

Аналіз веб-сайту вимагає вивчення базової структури веб-сайту для розуміння того, як зберігаються, використовуються та опрацьовуються необхідні дані. Це вимагає базового розуміння архітектури мережі Інтернет, мови розмітки веб-сторінок (наприклад, HTML, CSS, XML, XBRL тощо) та базові розуміння баз даних (наприклад, MySQL). Сканування веб-сайтів передбачає розробку та запуск скриптів, які автоматично переглядають веб-сайт і отримують необхідні дані. Ці програми сканування (або скрипти) часто розробляються з використанням таких мов програмування, як R і Python. Це пов'язано з загальною популярністю цих мов у бібліотеках Data Science і доступності (наприклад, «rvest» пакет в мові програмування R або Beautiful Soup library в мові програмування Python) для автоматичного сканування і аналізу веб-даних. Після того, як необхідні дані будуть проаналізовані з

обраного ресурсу, їх необхідно очистити, попередньо обробити і організувати таким чином, щоб забезпечити подальший аналіз цих даних. Враховуючи обсяг даних, що використовується, програмний підхід може бути необхідним і для економії часу. Багато мов програмування, такі як R і Python, містять бібліотеки обробки природних мов (NLP) і функції обробки даних, корисні для очищення та організації даних. Зазвичай ці три елементи веб-скрапінгу не можуть бути повністю автоматизованими і часто вимагають принаймні деякого втручання людини та контролю. Схема роботи показана на рисунку 1.4.

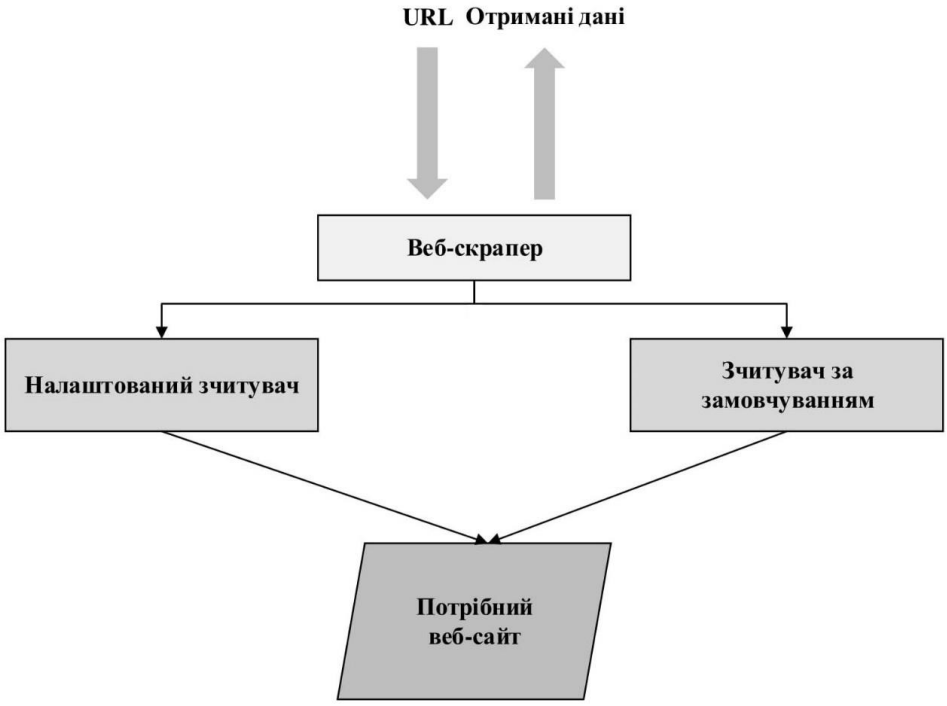

Рис. 1.4. Схема роботи веб-скрапера

Зазвичай веб-скрапери працюють за наступним алгоритмом:

1. Підготовка механізму отримання HTML-коду по запиту типу GET.
2. Аналіз DOM-структури потрібного інтернет-ресурсу.
3. Визначення вузлів з потрібною інформацією.
4. Налаштування обробника вузлів.
5. Виведення даних в нормалізованому вигляді (н-д, в форматі JSON).

На вході система отримує URL потрібної сторінки, а на виході віддає нормалізовані дані (наприклад, в форматі JSON). Отримавши URL, система визначає, якому зчитувачу потрібно направити сторінку на обробку. У випадку, якщо система знає архітектуру веб-ресурсу, URL отримує спеціально налаштований зчитувач, в іншому випадку – сторінку починає аналізувати зчитувач, який використовується за замовчуванням. Як правило, в таких випадках використовується найбільш стабільний зчитувач [3].

1.2. Огляд технік виконання веб-скрапінгу

Веб-скрапінг – це процес автоматичного збору даних або інформації в мережі Інтернет. Це сфера, що активно розвивається і досить часто вимагає певних удосконалень в автоматизованій обробці тексту, смисловому розумінні, сфері штучного інтелекту та взаємодіях типу «людина – комп'ютер». Сучасні рішення для веб-скрапінгу варіюються від спеціальних, що вимагають людських зусиль, до повністю автоматизованих систем, які можуть перетворювати цілі веб-сайти в структуровану інформацію з деякими обмеженнями.

Нижче наведено перелік можливих рішень, що можуть активно використовуватись для веб-скрапінгу:

1. Ручне копіювання та вставка інформації.

Іноді навіть найкраща автоматизована технологія збору інформації з веб-сторінок не може замінити аналіз зроблений людиною та ручне копіювання даних. В деяких випадках це може бути єдиним робочим та надійним рішенням, коли сайти цілеспрямовано створюють різні перепони для веб-роботів щоб запобігти автоматизованому доступу до ресурсів.

2. Відповідність текстового шаблону.

Простий та ефективний метод веб-скрапінгу для збору даних з веб-сторінок. Цей метод може використовувати команду grep в UNIX системах

або пошук за допомогою регулярних виразів таких мов програмування як Perl чи Python.

3. HTTP-програмування.

Статичні та динамічні веб-сторінки можуть бути отримані шляхом посилання HTTP-запитів до віддаленого веб-сервера за допомогою мережевого програмування.

4. Синтаксичний аналіз HTML.

Багато веб-сайтів мають великі кількості сторінок, що динамічно генеруються із базового структурованого джерела, подібного до бази даних. Дані однієї категорії зазвичай кодуються на подібні сторінки звичайним сценарієм або шаблоном.

Під час інтелектуального аналізу даних програма, яка виявляє такі шаблони в певному джерелі інформації, витягує її вміст і переводить її у реляційну форму, що називається обгорткою. Алгоритми генерації обгорток припускають, що вхідні сторінки відповідають загальному шаблону і що їх можна легко ідентифікувати за загальною схемою URL.

Більш того, деякі напівструктуровані мови запитів, такі як Xquery і HTQL, можуть бути використані для розбору HTML-сторінок і для отримання, перетворення і систематизації вмісту сторінки.

5. Аналіз DOM.

Вбудовуючи повноцінний веб-браузер, наприклад, Safari або браузер Google Chrome, програми можуть отримувати динамічний вміст, створений скриптами на клієнтській стороні. Ці елементи керування веб-браузером також аналізують вміст веб-сторінки та його структуру. На основі таких даних програми можуть отримувати частини сторінок.

6. Вертикальна агрегація.

Деякі компанії займаються розробкою вертикально-специфічних платформ для збору інформації в мережі інтернет. Ці платформи створюють і контролюють безліч «ботів» для таких цілей без «людини в циклі» (без

прямого залучення людини). Також такі платформи не потребують виконання робіт пов’язаних з конкретним сайтом.

Підготовка передбачає створення бази знань для використання у будь яких випадках, а потім така платформа автоматично створює ботів для аналізу інтернет ресурсів. Надійність та ефективність платформи вимірюється якістю інформації, яку вона отримує (як правило, кількістю полів) і її масштабованістю (наскільки швидко вона може масштабуватися до сотень або тисяч сайтів). Ця масштабованість в основному використовується для аналізу динамічних сайтів, які постійно збільшуються.

В основному звичайні агрегатори вважають такі сайти занадто складними і потребують докладання чималих зусиль для аналізу, обробки та збору інформації [4].

7. Розпізнавання семантичних анотацій.

Сторінки, на яких здійснюється сканування, можуть охоплювати метадані або семантичні розмітки і анотації, які можна використовувати для пошуку окремих фрагментів даних. Якщо анотації вбудовані на сторінках, як це робить Microformat, цю техніку можна розглядати як особливий випадок розбору DOM. В іншому випадку, анотації, організовані в семантичний шар, зберігаються і керуються окремо від веб-сторінок, тому веб-скрапери можуть отримати схему даних і інструкції з цього шару перед тим, як збирати дані зі сторінки.

8. Аналіз веб-сторінок за допомогою комп’ютерного зору.

Існують деякі методи, які використовують машинне навчання та комп’ютерне бачення, які намагаються ідентифікувати та витягувати інформацію з веб-сторінок, інтерпретуючи сторінки візуально, як це робить людина.

9. Xpath. XML Path Language, або Xpath – це мова запитів, що працює з XML-документами. Оскільки документи XML основані на представленні документів у вигляді дерев, Xpath можна використовувати для навігації по дереву, вибираючи вузли на основі різних параметрів. Xpath можна

використовувати спільно з аналізом DOM і витягувати всю веб-сторінку, а також публікувати її в потрібному місці.

10. Засоби Google документів.

Google таблиці можна використовувати як інструмент веб-скрапінгу, і він дуже популярний серед користувачів. З допомогою Google таблиць, веб-скрапер може використовувати функцію IMPORTXML(,), щоб витягувати дані з веб-сайтів. Цей метод є доволі ефективним в випадках, коли за допомогою методів веб-скрапінгу користувач хоче отримати конкретні дані чи шаблони з веб-сайту. Також цей метод використовують для перевірки чи захищений веб-сайт від веб-скрапінгу [5].

1.3. Інструменти веб-скрапінгу

Оскільки існують різні інструменти веб-скрапінгу та послуги, що пропонують компанії, дуже важко визначити які з них є більш ефективними та надійніші в порівнянні з іншими. Основуючись на тому, як вони працюють, нижче наведено класифікацію інструментів веб-скрапінгу, доступних на ринку (рис. 1.5).

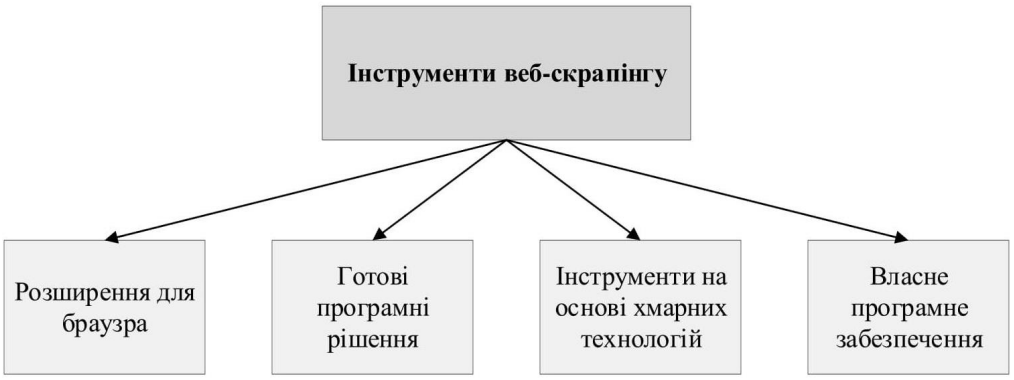

Рис. 1.5. Класифікація інструментів веб-скрапінгу

1. Розширення для браузера.

Розширення для браузера є прекрасним інструментом, щоб витягти невеликі частини даних. Веб-скрапінг за допомогою плагіна для браузера, а

не окремого програмного забезпечення, встановленого на ПК є найбільш ефективним рішенням для перегляду, аналізу та подальшої модифікації даних.

Розширення для браузера дозволяє встановити його і вибрати спосіб, за допомогою якого потрібно отримати дані з вибраного веб-сайту. Дані буде завантажено у вибраному форматі (наприклад CSV, JSON) або в будь-якому іншому зручному форматі. Розширення для браузера майже не вимагають зусиль для встановлення та налаштування. Такі розширення мають лише мінімальні налаштування для отримання даних з веб-сайтів.

Також, через свою простоту, такі рішення мають певні обмеження. Розширення для браузера може витягти дані лише з однієї сторінки за один раз. Для обробки великої кількості даних є інші типи інструментів для веб-скрапінгу.

2. Готові програмні рішення.

Оскільки попит на дані та інформацію зростає з кожним днем, IT компанії взялися за розробку повноцінного програмного забезпечення для веб-скрапінгу.

Як і будь-яке інше програмне забезпечення, програмне забезпечення для веб-скрапінгу передбачає його повноцінне встановлення на ПК. Більшість таких програм повністю сумісні з найбільш поширеними операційними системами (Windows, MacOS).

Таке програмне забезпечення має більш широкий спектр налаштувань, тому таке рішення може бути більш гнучким і ефективним. Для початку роботи достатньо лише підлаштувати веб-скрапер під свої потреби.

Повноцінні програми для веб-скрапінгу використовують найбільш популярні і зручні у подальшому використанні формати для зберігання зібраних даних, такі як JSON, CSV.

На відміну від розширень для браузерів, програми для ПК дозволяють одночасно сканувати декілька сторінок і можуть обробляти більшу кількість інформації. Вони розраховані на невеликі та середні сайти.

3. Інструменти на основі хмарних технологій

У порівнянні з іншими інструментами, веб-скрапінг на основі хмарних технологій вважається найбільш ефективним та надійним рішенням.

Використовуючи інструменти на основі хмарних технологій не потрібно встановлювати жодного програмного забезпечення. Зазвичай налаштування в таких системах дуже прості і водночас мають найбільше варіацій порівняно з двома попередніми типами веб-скраперів. Одразу після налаштування система починає роботу. Дані можна отримати через API у веб-браузері і завантажити у вибраному форматі.

Хмарні технології не обмежують веб-скрапінг певним об’ємом даних завдяки використанню декількох обчислювальних середовищах. Таким чином можна зробити висновок, що використання хмарних технологій для веб-скрапінгу є найбільш доцільним при роботі з великими об’ємами даних та складними веб-сайтами.

У порівнянні з іншими інструментами, які вимагають втручання людини на деяких етапах веб-скрапінгу, інструменти веб-скрапінгу з використанням хмарних технологій можуть полегшити процес підтримки веб-скрапінгу та зробити його повністю автоматизованим і безпроблемним.

4. Власне програмне забезпечення.

Потрібно взяти до уваги, що можна створити своє власне програмне забезпечення для веб-скрапінгу використовуючи майже будь яку мову програмування. Таке рішення має свої переваги. Наприклад, користувач може налаштувати програму повністю під себе і створити найкращі умови для збору інформації саме з тих ресурсів, які йому потрібні.

Відомі скрапінг-продукти:

- grabz.it – надає ряд послуг, таких як: знімки веб-сайту, архівація, побудова коротких прев’ю посилань, тощо. Також дає можливість будувати спеціалізовані скрапери, як на серверах користувача, так і в хмарному середовищі grabz за допомогою HTTP-запитів до API grabz;

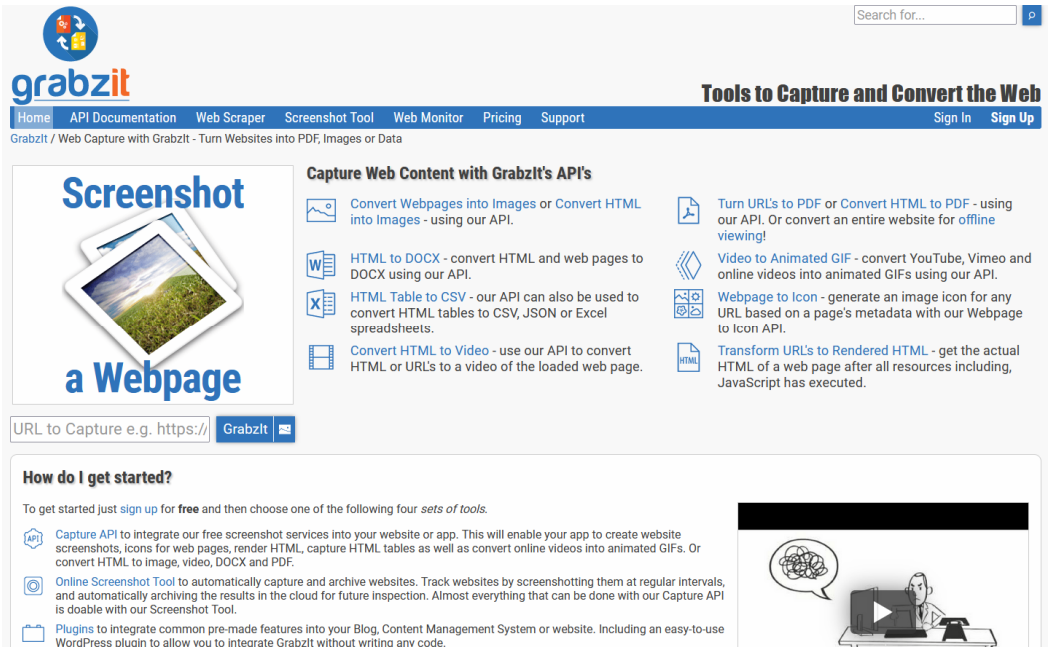

Рис. 1.6. Платформа для веб-скрапінгу grabz.it

- webscraper.io – інструмент для побудови спеціалізованих скраперів без коду, у інтерактивному інтерфейсі, дозволяє будувати власні структури даних засновуючись на потребах бізнесу. Має можливість застосування як у вигляді браузерного розширення, так і на власних серверах чи в хмарному середовищі, виконуючи HTTP-запити до API, яке надається сервісом webscraper;

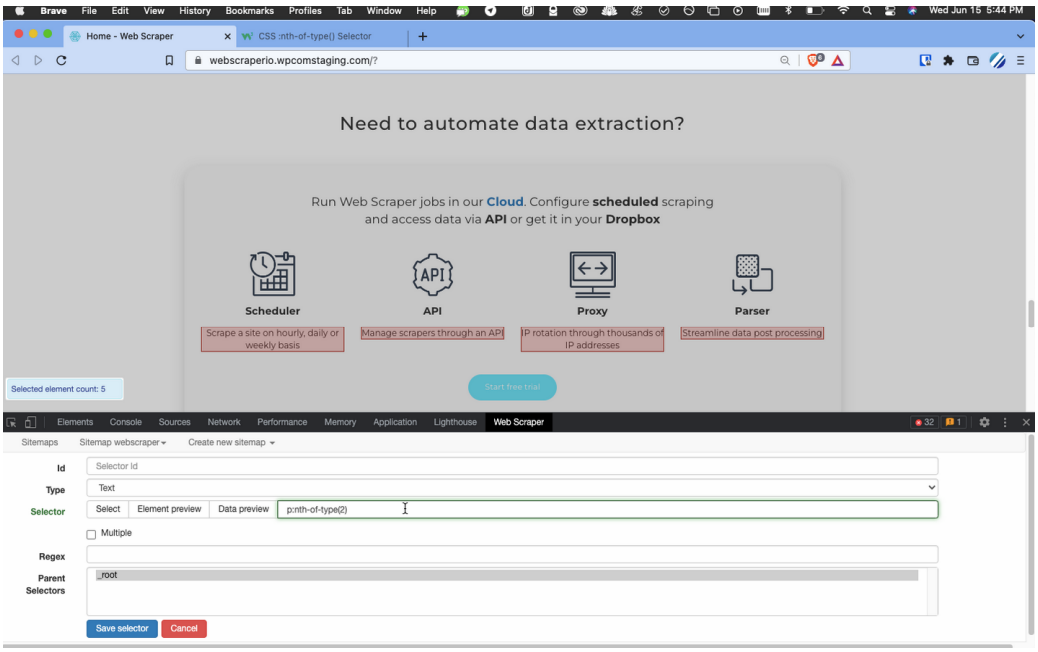

Рис. 1.7. Інструмент webscraper.io

- parsehub.com – аналог webscraper.io, спеціально пристосований для видобування великих об’ємів даних;

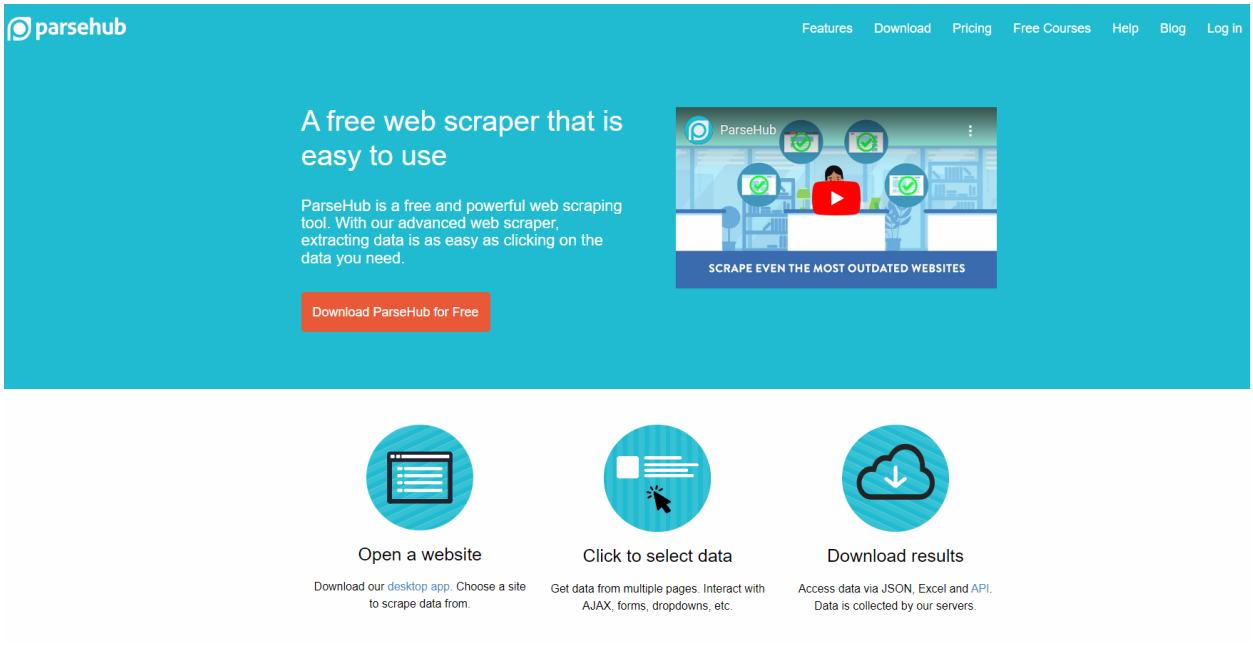

Рис. 1.8. Платформа parsehub

- diffbot.com – ресурсо- та областe-агностичний продукт, який застосовуючи алгоритми машинного навчання, будує графи знань з наданих наборів веб-сторінок та інструментарій для аналізу побудованих графів знань. Важливими особливостями роботи даного сервісує властивість мімікрувати під природньо-людську поведінку для обману ресурсів, які знаходяться під прицілом сервісу;

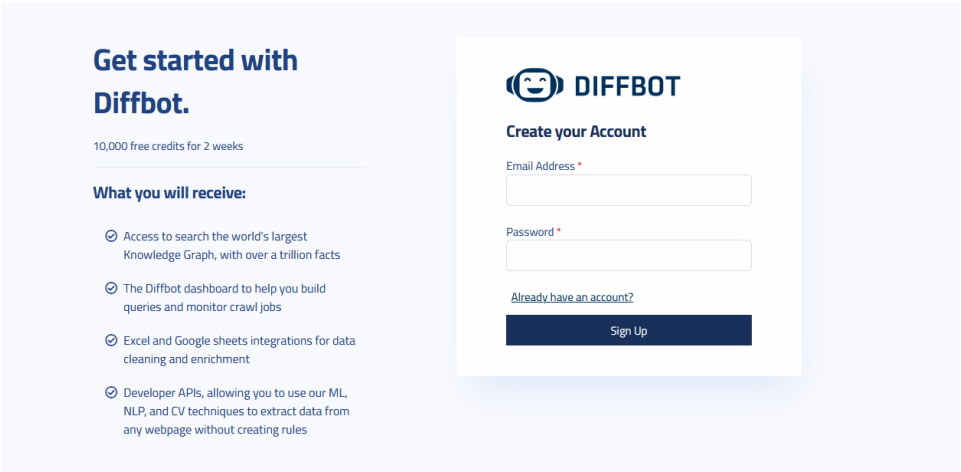

Рис. 1.9. Приклад реєстрації в diffbot.com

- scrapinghub.com – надає API для користування спеціальними ресурсо-агностичними, але специфічними для конкретної предметної області скраперами. Якість та чистота даних при збереженні ресурсо-агностичності хмарних скраперів, які надаються сервісом scrapinghub досягається за допомогою налаштовуваних аналітичних алгоритмів, які визначають структури аналізованої сторінки та багатокрокового автоматичного, автоматизованого та ручного процесів контролю якості даних.

1.4. Визначення, специфічні до сфери обробки природних мов на основі машинного навчання

Токенізація – перетворення тексту як послідовного набору символів на послідовний набір значущих одиниць тексту, при чому на один токен – значиму одиницю тексту – може припадати від невеликої частини одного слова до декількох слів (в залежності від характеристик аналізованої мови, контексту речення, тощо).

TF-IDF (term frequency-inverse document frequency) – статистичний показник у обчислювальній лінгвістиці та обробці природних мов. TF – частота вживання слова в конкретному документі. IDF – логарифм від загальної кількості документів, розділеної на кількість документів у яких трапляється вищезазначене слово, та позначає рівень уживаності слова в наявному корпусі документів. Таким чином, множення TF на IDF дає можливість оцінити важливість слова у документі відносно до важливості слова у корпусі, частиною якого цей документ є, таким чином даючи метрику релевантності терміну.

Шумові слова (стоп-слова) – слова, які не містять інформації і є зайвими для алгоритмів пошуку чи аналізу тексту. Вони поділяються на загальні (цифри, знаки, прийменники, слова які застосовуються занадто часто у масиві на якому здійснюється аналіз, тощо), та залежні (такі, які не містять інформації у конкретному прикладі аналізованого тексту).

Наївний байєсів класифікатор – класифікатор на основі теорії ймовірності. Для визначення ймовірності входження тестованого елемента до класу використовує теорему Байєса, базуючись на наївному припущенні незалежності параметрів елемента.

Перцептрон – вид найпростіших нейронних мереж, які складаються з трьох видів нейронів: рецептори, асоціативні нейрони та реагуючі нейрони. Рецептори є вхідними нейронами мережі, які збирають вхідні дані. Кожен рецептор в вхідному прошарку нейронної мережі з'єднаний з лише одним нейроном наступного шару, який складається з асоціативних нейронів. Асоціативні прошарки є прихованими прошарками нейронної мережі та відповідають за обробку вхідної інформації. Кожен нейрон асоціативного прошарку пов'язаний з кожним нейроном попереднього та наступного прошарків. Після одного або декількох асоціативних прошарків мережі знаходиться прошарок реагуючих нейронів, які є виходами нейронної мережі, та виводять інформацію назовні.

Пасивно-агресивний класифікатор – інтерфейсно подібний перцептрону класифікатор, пристосований для роботи з великими масивами потокових даних. Його особливістю є специфічний алгоритм навчання, який забезпечує швидке навчання та можливість навчання або додаткового навчання на потокових даних. Назва походить від подвійної поведінки під час навчання: Пасивна (якщо передбачення правильне – модель не треба змінювати, оскільки такі дані не є істотними причинами для зміни моделі), Агресивна (якщо передбачення хибне – модель коригується оскільки такі дані є істотними для її роботи).

Рекурентні нейронні мережі – клас алгоритмів машинного навчання, у якому дозволяється з'єднання глибших прошарків мережі з ближчими до поверхні, що формує орієнтований в часі граф та дозволяє штучній нейронній мережі мати внутрішній стан, та змінювати свою подальшу поведінку в залежності від попередніх вхідних даних.

LSTM – різновид рекурентних нейронних мереж, які містять LSTM-вузли. LSTM-вузол є рекурентним вузлом, у якому не застосовується функція активації для рекурентних складових вузла, що дозволяє нейронній мережі мати можливість запам’ятовувати дані як на короткий, так і на довгий проміжок часу. Такі вузли як правило складаються з чотирьох складових –- “клітина”, яка відповідає за пам’ять, і три клапани – вхідний, вихідний та стираючий пам’ять.

Word2Vec – підхід та комплекс нейронних мереж для вивчення словесних асоціацій з великих об’ємів даних. Word2Vec-мережі є простими двошаровими мережами, націленими на конструювання лінгвістичних контекстів слів. Ці мережі навчаються на величезних корпусах для створення багатовимірних (декілька сотень) векторних просторів, в яких кожному слову призначається певна точка в цьому просторі, а схожі за контекстом використання слова отримують геометрично близькі позиції.

Висновки до розділу

В даному розділі проведено розглянуті такі поняття як веб-скрапер та веб-скрапінг. Були розглянуті види програмного забезпечення, що реалізують автоматизований доступ до веб-ресурсів. Також було викладено і описано існуючі підходи у попередній підготовці та обробці текстових даних для використання у алгоритмах машинного навчання, розібрано типові існуючі підходи з класичного машинного навчання та нейронних мереж і вказано існуючі рішення в цій сфері, включно з описом їх специфік.

РОЗДІЛ 2. МЕТОДИ ТА АЛГОРИТМИ МАШИННОГО НАВЧАННЯ ДЛЯ АНАЛІЗУ ОТРИМАНИХ ТЕКСТІВ В ПРОЦЕСІ ВЕБ-СКРАПІНГУ

2.1. Алгоритм підготовки даних

Для забезпечення якості результатів тренування алгоритмів машинного навчання було використано алгоритм підготовки даних, задачею якого є створення та наповнення файлу навчальних даних з файлів результатів скрапінгу. Також алгоритм виконує первинну обробку тексту видобутих статей, приводячи їх до одноманітного регуляризованого формату. Останнім кроком трансформації є перевірка слів на входження в список шумових слів (російська та українська). Фільтрація шумових слів є важливим елементом ефективності алгоритмів класифікації та категоризації текстів живими мовами, та широко використовується у машинній обробці натуральних мов [10].

Нижче приведено опис послідовності дій, які виконуються під час підготовки даних трансформатором:

```
Відкрити файл з списком шумових слів
Створити Set з списку шумових слів
Для кожного файлу у списку файлів результатів скрапінгу:
    Відкрити файл
    Для кожного запису в файлі:
        Якщо запис не є статтею:
        Пропустити
    Присвоїти статті маркування 0 чи 1 відповідно до її джерела
    (ненадійне/надійне)
    Видалити джерело з статті
    Залишити в статті тільки буквенні символи та пробіли
    Видалити повторні пробіли та знаки переносу рядка
    Привести статтю до нижнього регістру
    Розбити статтю на окремі слова
```

```
Для кожного слова в статті:
        Якщо слово є в списку шумових слів:
            Видалити слово з статті
    Зібрати статтю з залишкових слів
    Зберегти результати обробки файлу в вихіний файл
Перемішати записи в вихідному файлі за допомогою варіації методу Фішера-Єйтса
```

Даний алгоритм було реалізовано на мові ECMAScript для платформи Node.js. Виділення та видалення небуквенних та інших символів було виконано за допомогою регулярних виразів. Реалізація алгоритму тасування Фішера-Єйтса була запозичена з бібліотеки реалізацій типових функцій lodash.

2.2. Алгоритм TF-IDF-векторизації

TF-IDF тексту у загальному сенсі є набором значень «важливостей» конкретних слів у контексті тексту відносно контексту корпусу документів.

Першою частиною TFIDF є частота слова (term frequency) у документі, яка обчислюється таким чином:

$$TF_i = \frac{n_i}{\sum_k n_k},$$

де TF_i є частотою слова t_i, n_i є кількістю входжень слова у документі, а в знаменнику вказується загальна кількість слів у документі.

Другою частиною TF-IDF є інверсна частота документу – інверс частоти входження слова у корпусі, яка обчислюється таким чином:

$$IDF_i = \log \frac{|D|}{|(t_i \in d_i)|},$$

Де IDF_i є інверсною частотою документу, $|D|$ є об'ємом корпусу, $|(t_i \in d_i)|$ є часткою корпусу, яка містить слово t_i. Варто зауважити, що основа логарифму у цій формулі не має значення доки вона є однаковою у межах дослідження. Зміна ж основи призводить до зміни ваги кожного слова на постійний множник.

Добутком частин TF та IDF є показник TF-IDF:

$$TFIDF_i = TF_i \cdot IDF_i.$$

Оскільки у цій роботі TF-IDF-метрика використовується не для оцінки статичного корпусу текстових документів, а як один з кроків у класифікації даних, які надходять динамічно – для досягнення прийнятної швидкодії результуючого програмного забезпечення формується лексично репрезентативний (відносно текстів, які будуть підлягати аналізу) корпус документів, і обчислюється IDF словника такого корпусу.

Це відбувається за таким принциповим імперативним алгоритмом:

```
Для кожного слова в кожному документі корпусу:
    Якщо слово не входить в словник:
        Додати слово в словник та призначити йому індекс
Замінити слова в документах на відповідні індекс
Створити вектор входжень розміром, рівним словнику, заповнений 0
Послідовно для кожного індексу в кожному документі корпусу:
    За індексом збільшити на 1 значення в векторі входжень
Для кожного елементу вектору входжень:
    Обчислити дільник кількості документів корпусу на значення
елементу
    Зберегти в елемент логарифм значення вище.
    Зберегти словник та вектор IDF
```

Для обчислення TF-IDF та підготовки вхідної матриці алгоритму машинного навчання використовується такий алгоритм з використанням значень, передобчислених алгоритмом зазначеним вище:

```
Завантажити словник та вектор IDF
Створити масив-результат з елементами [індекс в словнику, лічильник входжень]
Для кожного слова в документі:
    Якщо індекс слова в словнику є в масиві результаті:
        Збільшити лічильник входжень слова на 1
    Інакше:
        Додати [індекс, 1] в масив
Для кожного елементу в масиві входження:
    Поділити лічильник входжень слова на кількість слів у масиві входження, домножити на значення IDF взяте по індексу слова в словнику
    Замінити лічильник входжень на значення, обчислене в попередньому кроці.
```

2.3. Представлення роботи алгоритму токенізації

Альтернативним методом підготовки сирого тексту є токенізація. У цьому методі ми навчаємо токенізатор найчастішим словам збираючи словник, та за допомогою цього словника обертаємо набір слів тексту на вектор індексів цих слів у словнику.

Внутрішній вигляд словника представлено таким чином:

```
[слово1,слово2,слово3,…]
```

є словником відповідності індексу слову,

```
{слово1:1,слово2:2,слово3:3,…}
```

Є словником відповідності слова індексу.

Словник наповнюється методом аналізу доступного корпусу тестів та обрізання до максимуму необхідних:

```
Створити словник входжень слів
Послідовно для кожного слова в кожному документі корпусу:
    Якщо слово є в словнику входжень:
        Збільшити кількість входжень на 1
    Інакше:
        Поставити кількість входжень слова як 1
Відсортувати слова за входженнями
Для кожного слова в n найвживаніших слів:
    Призначити слову індекс в словнику відповідності індексів словам
    Призначити індексу слово в словнику відповідності слів індексам
```

Оскільки даний метод використовуватиметься для підготовки даних для використання у нейронній мережі такий метод має видавати вектор сталого розміру, і, відповідно, стандартний метод токенізації тексту було модифіковано. В стадію підготовки токенізатора було додано етап обчислення середньої довжини тексту у корпусі як сталого розміру вхідного вектору нейронної мережі. Алгоритм підготовки тексту з участю токенізації та відповідно змін описаних вище виглядає таким чином:

```
Для кожного слова в документі:
    Якщо слово наявне в словнику:
        Замінити слово на індекс з словника
    Інакше:
        Замінити слово на 0
Якщо довжина токенізованого вектору документа менша за середню довжину документу корпусу:
```

```
    Розширити від початку вектор на кількість нулів, яка
відповідає різниці довжини документа та середньої довжини
документів корпусу
Інакше:
    Обрізати вектор до відповідності середній довжині документів
корпусу
```

Таким чином, за допомогою доповненої токенізації досягається рівномірний, передбачуваний та вирівняний вхідний вектор нейронної мережі.

2.4. Дослідження архітектури рекурентної нейтронної мережі Long Short-Term Memory

Архітектура мережі LSTM - Long Short-Term Memory (довга короткочасна пам’ять - ДКЧП) була винайдена у 1997 році. До винаходу цієї архітектури призвів аналіз проходу похибки в існуючих на той момент рекурентних нейронних мережах, який показав що довга пам’ять мережі була недоступна в тогочасних архітектурах, оскільки зворотно поширена похибка має властивість експоненційно зменшуватись або збільшуватись.

ДКЧП-прошарок створений з множини рекурентно зв’язаних блоків, так званих «блоків пам’яті», схожі на диференційовані версії оперативної пам’ятті сучасного комп’ютеру. Кожен такий блок складається з одного або декількох рекурентно поєднаних блоків пам’яті та трьох елементів множення – воріт входу, виходу та забуття. Мережа може працювати з клітинами тільки через ворота, які є аналогами операцій запису, читання та скидання пам’яті. Таким чином, вхід до клітини мережі множиться на активацію вхідних воріт, вихід клітини в мережу множиться на активацію, а попередні значення клітини множаться на значення воріт забуття.

На рис. 2.1 зображено ДКЧП-блок з однієї клітини. Внутрішній стан клітини зберігається рекурентним з’єднанням вагою 1. Три ворота збирають

активації ззовні та зсередини блоку, та керують клітиною за допомогою елементів множення (синіх кіл). Вхідні та вихідні ворота масштабують вхід та вихід клітини, а ворота забуття масштабують внутрішній стан, наприклад скидаючи його в 0 (провокуючи скидання внутрішнього стану). Стискаючі функції *h* та g застосовуються в означених місцях.

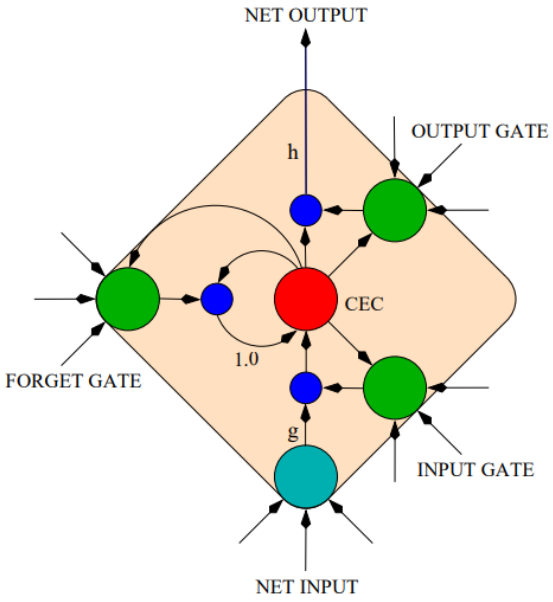

Рис. 2.1. LSTM - блок з однією клітиною

Відмінність ДКЧП-моделей від звичайних рекурентних моделей яскраво прослідковується у математичному вираженні цих нейронних мереж.

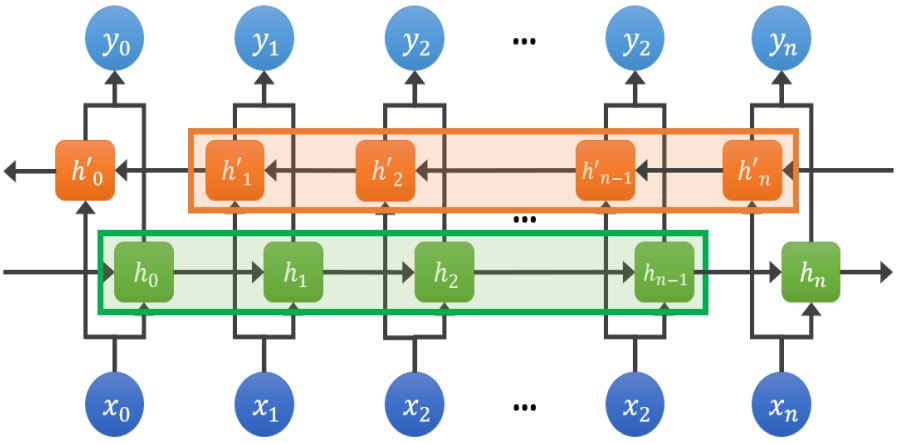

Рис. 2.2. Розгорнений у часі нейрон рекурентного прошарку нейронної мережі

Нехай $\{x_1,\ldots,x_n\}$ є ембедінгом токенізованого тексту. Тоді результатом роботи рекурентної нейронної мережі є вихідний вектор y_t для кожного токена x_t. Результат обчислюється повторенням наступних рівнянь від t = 1 до t = n:

$$h_t = H(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$
$$y_t = W_{hy}h_t b_y,$$

де h_t є прихованою послідовністю векторів, W є матриці вагів (наприклад, W_{xh} є матрицею вагів, які поєднують вхідний та прихований прошарки), b позначає вектори упередженості та H є функцією активації прихованого прошарку. За допомогою векторів h рекурентні моделі і досягають ефекту пам'ятті та впливу попередніх контекстів на поточні. Через проблему "зникаючого градієнта" стандартні рекурентні нейронні мережі не можуть використовувати всю вхідну історію.

Описані вище принципи роботи нейронної мережі довгої короткострокової пам'ятті (клітини пам'ятті та ворота роботи з нею) застосовані до рекурентної нейронної мережі трансформують її в алгоритм, який може бути описаний таким чином:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)$$
$$c_t = f_t c_{t-1} + i_t \sigma \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o)$$
$$h_t = o_t \tanh c_t$$

Де σ є логістичною сігмоїд-функцією, *i*, *f*, *o*, та *c* є векторами вхідних воріт, воріт забуття, вихідних воріт та активації клітини. *b* є вивченими векторами упередженості.

Однією з проблем як звичайних рекурентних нейронних мереж, так і мереж довгої короткочасної пам’яті є урахування лише попередніх контекстів.

Живим мовам властиві двусторонні зв’язки у реченнях, який виражається, наприклад, у смисловому впливі слів на початку речення на слова в середині, так і вплив слів у кінці речення на слова в початку, тощо. Такі смислові впливи є невід’ємною частиною органічності походження багатьох мов.

Для роботи з такого плану проблемами нейронним мережам, відповідно, необхідне знання не тільки про попередній контекст, а і про майбутній. У контексті нейронних мереж довгої короткочасної пам'ятті для розв'язання цієї проблеми використовуються двусторонні ДКЧП-мережі.

Така архітектура нейронної мережи має можливість розв’язання як попереднього контексту, так і наступного контексту конкретного слова.

Токенізовані тексти мають вигляд цілочислених векторів, у яких кожному слову тексту відповідає унікальне число. Оскільки багато слів можуть мати різні сенси в різних контекстах, або навпаки – декілька слів можуть мати однаковий сенс осмисленим є наявність тренованого прошарку нейронної мережі, який би за вивченим словником виконував трансформацію слів у вектори, які характеризують ці слова відповідно її навченому досвіду. Для цього використовуються Embedding-шари.

Ембедінг-шар трансформує вхідний вектор токенізованого тексту на матрицю, де кожному слову відповідає вектор у n-мірному просторі, який допомагає подальшим прошаркам нейронної мережі характеризувати це слово. По суті ембедінг-шар вивчає lookup-матрицю, в якій кожному слову в словнику відповідає вищезазначений вектор, та під час роботи мережі простим матричним множенням виконує трансформацію.

Таким чином, нейронна мережа з використанням ембедінг шару може наблизитись до розуміння не просто комбінацій слів, а комбінацій сенсів слів

(звісно, у тому вигляді, який доступній нейронній мережі виходячи з наявних тренувальних даних та її архітектури).

Оскільки двустороння мережа довгої короткотривалої пам’ятті видає багатовимірний вектор у якості результату, а очікуваний від нейронної мережі у випадку даної магістерської роботи результат є відношенням або невідношенням до конкретного класу – багатовимірний результат ДКЧП-шару має бути приведений до одновимірного результату у проміжку [0,1]. Інакше кажучи, після обробки даних ДКЧП-шаром має відбутись прийняття остаточного рішення. Це виконується за допомогою щільного повнозв’язного прошарку нейронної мережі.

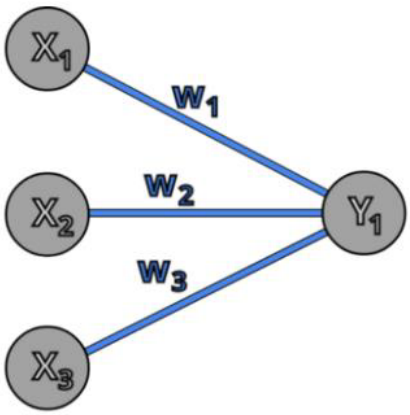

Рис. 2.3. Щільний прошарок нейронної мережі

Щільний повнозв’язний прошарок є насправді стандартною класичною нейронною мережею, вихід якої розраховується за формулою:

$$y_i = H(W_{i-1}y_{i-1}),$$

де y_i є вихідним значенням нейрона, H є функцією активації, x_i є вхідним значенням нейрона попереднього прошарку, W_i є вагою нейрона попереднього прошарку.

2.5. Методи оцінки точності моделей

Моделі, побудовані за допомогою алгоритмів машинного навчання, повинні мати можливість давати точні прогнози, щоб мати реальну цінність для розв’язання прикладних задач та застосування в різних дослідженнях.

Хоча навчання моделі є ключовим кроком, проте те, як модель узагальнює дані та шукає взаємозв’язки, є не менш важливим аспектом, який слід враховувати при використанні машинного навчання. Необхідно знати, чи насправді дана модель адекватно працює, і, отже, чи можна довіряти її прогнозам. Необхідно виключити випадок, коли модель просто запам'ятовує дані, які були подані на вхід, і в цьому випадку не може робити якісних прогнозів, використовуючи нові дані.

Вищезазначені питання вирішуються за допомогою методів оцінки точності та ефективності моделей машинного навчання, які є невід’ємною складовою будь-якого дослідження та роботи з обробкою даних. Методи оцінки точності моделі необхідні для встановлення загальної ефективності та адекватності моделей при використанні нових даних.

Методи оцінки точності моделі поділяють на 2 типи:

- метод витримки (hold-out);
- перехресна перевірка.

Обидва методи використовують набір тестового набору даних для оцінки ефективності моделі. Не рекомендується використовувати дані, які були використані для побудови моделі. Це пов’язано з тим, що в такому випадку виникне проблема, що була зазначена раніше: перенавчання.

2.5.1 Метод витримки (hold-out)

Метод витримки (hold-out) для оцінки точності моделі представляє собою механізм розподілу набору даних на навчальний і тестовий набір. Ілюстрацію цього методу наведено на рисунку 2.4.

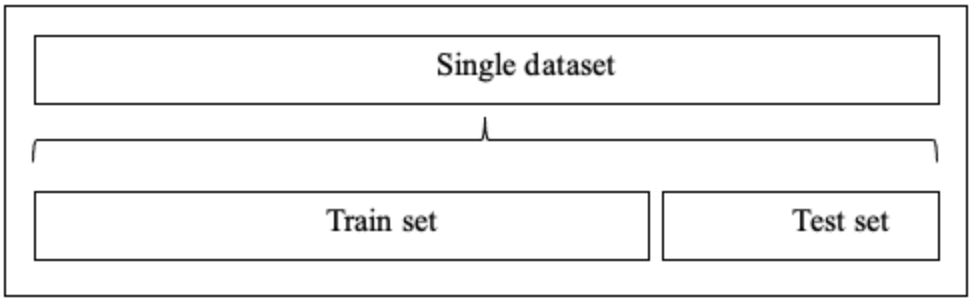

Рис. 2.4. Ілюстрація розподілу набору даних

На рисунку 2.4 можна побачити, що набір даних (single dataset) розділений на дві частини. Більша частина даних виділена для навчання моделі – train set. Решта даних буде використовуватися для тестування моделі – test set. Відсоток розподілу визначається в залежності від обсягу даних, доступних для дослідження. Зазвичай набір даних для навчання може складати від 60% до 80%, у свою чергу від 20% до 40% набору даних використовується для тестування моделі.

Метою оцінки витримки є тестування моделі на даних, відмінних від тих, на основі яких відбувалось навчання. Це забезпечує неупереджену оцінку навчальних показників.

Цей підхід має переваги у вигляді високої швидкості, простоти та гнучкості. Проте при використанні цього методу варто пам’ятати, що відмінності у навчальному та тестовому наборах даних можуть призвести до певних відмінностей й в оцінці точності.

2.5.2 Перехресна перевірка

Перехресна перевірка - це методика оцінки прогнозних моделей шляхом розділення вихідної вибірки на навчальний набір для навчання моделі та набір тестів для її оцінки.

Найчастіше використовують саме метод k-перехресної перевірки. При k-кратній перехресній перевірці вихідна вибірка випадковим чином розподіляється на k зразків рівного розміру. K – ціле число, параметр, який вказується на початку роботи дослідником, зазвичай k беруть від 5 до 10.

З k зразків один зразок зберігається як дані для тестування моделі, а решта k-1 зразків використовуються як навчальні дані. Потім процес перехресної перевірки повторюється k разів. Потім результати можуть бути усереднені (або об’єднані будь-яким іншим способом), щоб отримати єдину оцінку.

Наприклад, при виконанні п’ятикратної перехресної перевірки дані спочатку поділяються на 5 частин приблизно однакового розміру. Навчання моделі відбувається послідовно, у даному випадку разів підряд. На першому кроці модель тренується з використанням першого зразку даних як тестового набору, а решта зразків використовується як навчальні набори. Це повторюється для кожного з 5 отриманих наборів даних. Далі оцінка точності усереднюється, щоб отримати загальну оцінку моделі. Ілюстрація цього прикладу наведена на рисунку 2.5.

Рис. 2.5. Ілюстрація процесу виконання п’ятикратної перехресної перевірки

Як бачимо, кожна зразок виконує роль тестового набору рівно один раз і виконує роль навчального набору k-1 рази (у прикладі з п’ятикратною перехресною перевіркою – 4 рази). Це суттєво зменшує дисперсію, оскільки

більшість даних використовується в тестовому наборі. Зміна навчальних та тестових наборів також сприяє підвищенню ефективності цього методу.

Перехресна перевірка, як правило, є найкращим методом, оскільки вона дає моделі можливість тренуватися на декількох наборах навчальних та тестових даних. Завдяки цьому вдається отримати більш чітку оцінку того, наскільки адекватно модель буде працювати на нових, ще не використаних даних. З іншого боку, метод hold-out залежить лише від одного розподілення даних на навчальний та тестовий набір. Через це метод hold-out є залежним від того, як саме дані розподілені на навчальний та тестовий набір.

Метод hold-out доречніше використовувати у декількох випадках. А саме: якщо на вході набір даних великого обсягу або якщо необхідно зменшити час роботи.

При цьому для роботи методу перехресної перевірки потрібно більше обчислювальної потужності та часу порівняно з методом hold-out, оскільки перехресна перевірка декілька разів розподіляє вихідний набір даних на навчальний та тестовий.

Висновки до розділу

В даному розділі описано математичні засади та алгоритми, використані або розроблені, описано відповідні специфіки, тощо. Досліджено алгоритм обробки та підготовки даних для застосування у машинному навчанні. Викладено та математично пояснено TF-IDF метрику тексту та показано псевдокод процесу трансформації тексту у розріджений TF-IDF вектор для споживання алгоритмами машинного навчання. Також представлено алгоритм навчання токенізатора словнику на корпусі текстів та показує процес трансформації тексту у розріджений словарний вектор.

Досліджено математичні принципи функціонування рекурентних нейронних мереж як загального класу та, зокрема, мереж Довгої Короткочасної Пам’ятті. Також було обгрунтовано необхідність застосування

саме ДКЧП-мереж, пояснено відмінність двусторонніх ДКЧП-мереж від звичайних та важливість їх використання у обробці натуральних мов. У кінці розділу описано математичні засади та необхідність застосування Ембедінгів та щільних мереж при роботі з рекурентними та ДКЧП-мережами.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ ВИКОНАННЯ ПРОЦЕСУ ВЕБ-СКРАПІНГУ ДЛЯ РЕСУРСІВ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

3.1. Представлення архітектури системи та інтеграція інструментів для процесів веб-скрапінгу

Для архітектури системи інтернет магазину рекомендується використовувати мікросервісну модель, де кожна функціональна частина реалізується окремим мікросервісом. Це дозволяє підтримувати гнучкість та масштабованість системи.

У системі повинна бути використана відповідна база даних для зберігання різноманітної інформації, такої як дані про товари, замовлення, користувачів. Для ефективного управління базою даних можна розглядати різні системи, такі як MySQL, PostgreSQL або NoSQL рішення в залежності від потреб.

Щодо інтеграції, важливо забезпечити взаємодію із зовнішніми сервісами, такими як платіжні шлюзи, системи доставки та інші інструменти, які допоможуть оптимізувати роботу магазину та полегшити процеси для клієнтів.

Враховуючи ці принципи, система буде здатна ефективно обробляти та зберігати дані, забезпечуючи стабільну та масштабовану архітектуру.

Головною концепцією системи є модульність, що дозволить отримати наступні переваги:

– легший розвиток. Розподілення на фронтенд та бекенд дозволяє розділити функціональність системи на менші, незалежні модулі. Це спрощує розробку та підтримку, оскільки кожен модуль може розроблятися та підтримуватися окремо;

– покращена масштабованість. Модульність дозволяє додавати нові ресурси та функціональність лише там, де це потрібно. Це полегшує

масштабування системи, оскільки стає можливо масштабувати окремі найбільше навантажені модулі;

– забезпечення стабільності та надійності. Якщо один з модулів виходить з ладу, інші модулі можуть продовжувати працювати. Це допомагає запобігти відмовам у системі та забезпечує вищу стабільність та надійність;

– зменшення залежностей. Модульна архітектура зменшує залежності між різними частинами системи, що полегшує розробку та підтримку, а також сприяє швидшому відновленню після відмов;

– легше тестування та налагодження. Модульність дозволяє проводити тестування кожного модуля окремо, що спрощує виявлення та виправлення помилок;

– забезпечення безпеки. Ви можете керувати доступом до різних частин системи окремо, що дозволяє підвищити безпеку даних та системи в цілому.

Загалом, модульність та розподілення на фронтенд та бекенд сприяють створенню більш ефективних, стабільних та масштабованих систем для збору даних.

Для системи збору даних можна виділити наступні модулі:

– Scheduler Server – відповідальний за запуск збору даних за розкладом;

– Scrapers API Servers – група серверів, відповідальна за доступ до даних, а також роботу з даними в рамках процесу збору;

– Scrapers Servers – група серверів, відповідальна за збір даних з інтернет магазинів;

– Proxy Servers – група серверів, відповідальна за зміну фізичної локації запита на збір даних;

– Saver Servers – група серверів, відповідальна за попередню обробку та збереження інформації за допомогою Scrapers API Servers;

– Workers Servers – група серверів, відповідальна за виконання довгострокових задач, наприклад операції порівняння зображень за допомогою AI;

– DB – група серверів, відповідальна за довгострокове збереження даних.

Взаємодія між зазначеними модулями наглядно представлена на рисунку 3.1.

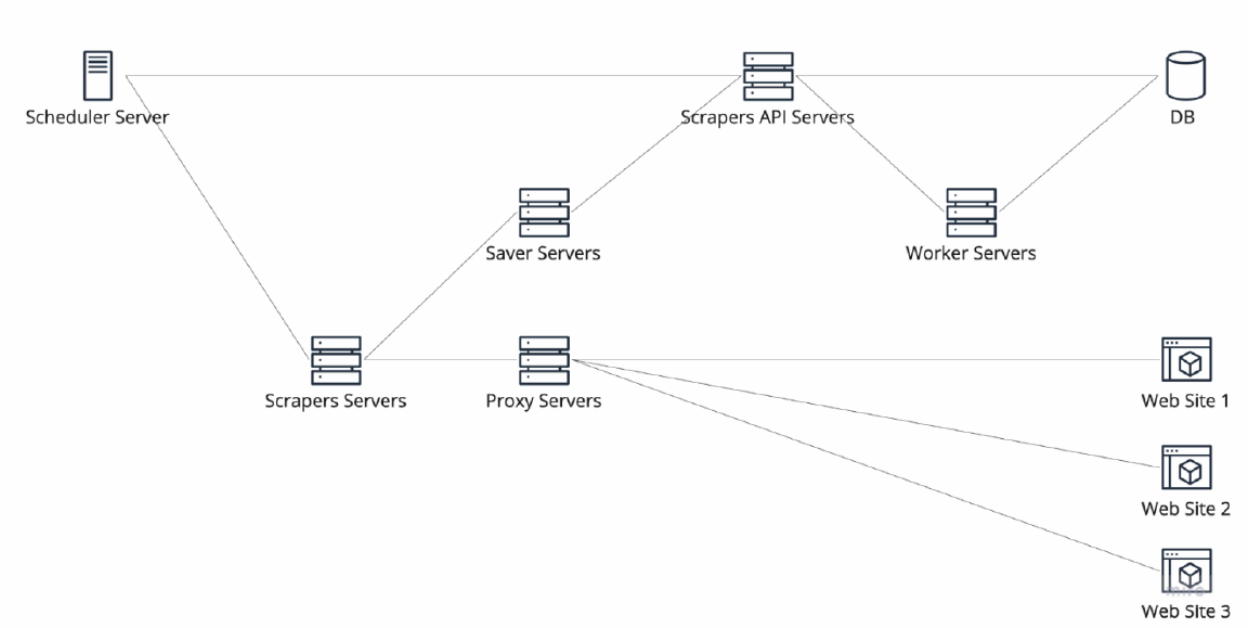

Рис. 3.1. Архітектурна схема розподілення на модулі системи веб-скрапінгу

Для UI системи веб-скрапінгу можна виділити наступні модулі:

– WAF – firewall для підвищення рівня захисту додатку від зовнішніх атак;

– UI Servers – група серверів, відповідальна за програмний інтерфейс, з яким буде працювати користувач системи;

– Worker Servers – група серверів, відповідальна за виконання довгострокових задач, наприклад підготовки імпорту або експорту даних;

– DB – група серверів, відповідальна за довгострокове збереження даних;

– Storage – група серверів, відповідальна за довгострокове збереження об’єктів, наприклад зображень або звітів.

Зв'язок між зазначеними модулями наглядно представлений на рисунку 3.2.

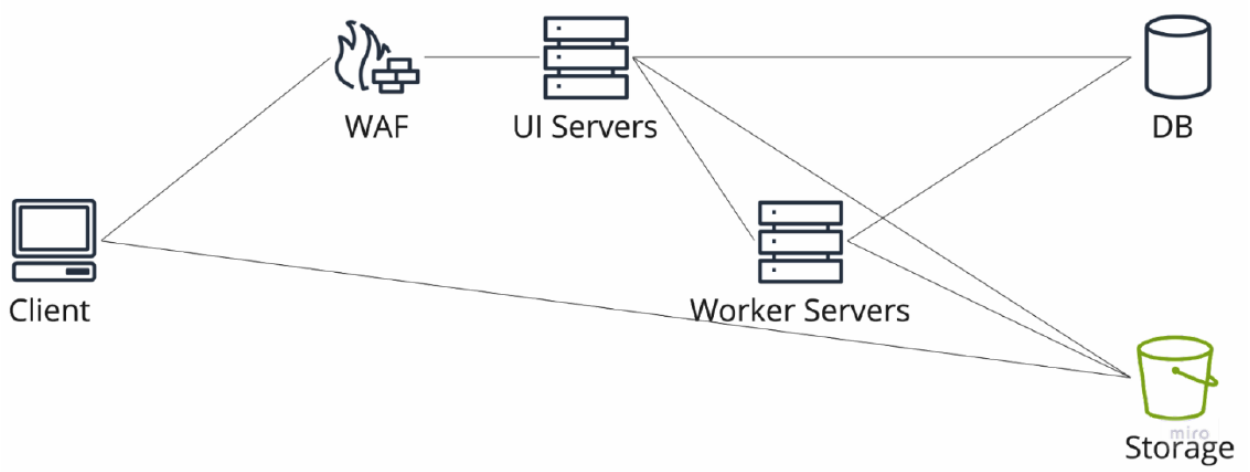

Рис. 3.2. Архітектурна схема розподілення на модулі UI

Інтеграція інструментів для веб-скрапінгу з інтернет магазинів може вимагати використання сервісів проксі та рішень для обходу систем капчі. Сервіси проксі дозволяють отримувати дані з інтернет магазинів, обмежуючи блокування IP-адрес користувача. Використання сервісів для розв'язування капчі, у свою чергу, надають можливість автоматизовано розв'язувати капчі, що можуть виникнути під час збору даних, забезпечуючи безперервну роботу інтегрованої системи збору даних.

В залежності від обсягів даних, збір може вимагати значних системних ресурсів для підтримки процесу збору та збереження інформації.

3.2. Архітектурне рішення щодо розгортання системи

Для розгортання системи потрібно вибрати хостинг, який надає хмарні послуги. Збір даних може потребувати великих обсягів ресурсів, а також можливості детальної конфігурації, це потрібно враховувати при виборі.

Хостинг від AWS, Microsoft Azure та Google Cloud Platform – це хмарні послуги, які надають інфраструктуру для розгортання та управління веб-додатками і сервісами. AWS пропонує широкий спектр рішень та гнучкі опції, Azure відзначається інтеграцією з продуктами Microsoft та розширеними можливостями для корпоративних систем, а Google Cloud

Platform відомий своєю швидкістю, інструментами для штучного інтелекту та аналітики даних. Кожна з цих платформ пропонує великий перелік сервісів та функціональні можливості для задоволення потреб розробників та підприємств у хмарному середовищі.

Деплоймент на AWS може бути найбільш вигідним з кількох причин:

– широкий спектр послуг. AWS пропонує величезний вибір послуг, які можуть використовуватися для різноманітних потреб, включаючи обчислення, зберігання, використання бази даних, машинне навчання, аналітику та інше. Це дозволяє легко масштабувати та налаштовувати інфраструктуру під конкретні вимоги;

– доступність. AWS має дата-центри по всьому світу, що дозволяє розгортати додатки територіально ближче до аудиторії. Це забезпечує менший час відгуку та поліпшує продуктивність для користувачів у різних регіонах;

– масштабованість та гнучкість. AWS дозволяє легко масштабувати кількість серверів, як в більшу, так і в меншу сторону, в залежності від потреб додатку. Також він надає можливість використання служб автоматичного масштабування, щоб забезпечити ефективне використання ресурсів;

– безпека. AWS забезпечує високий рівень безпеки та відповідність стандартам. Також є можливість використовувати різні інструменти для керування доступом, шифрування та моніторингу, щоб захистити дані;

– інновації та нові технології. AWS постійно впроваджує нові сервіси та технології, які дуже легко використовувати у власному додатку, що може полегшити подальшу підтримку додатків;

– спільнота та підтримка. AWS має велику та активну спільноту розробників. Це означає, що можна швидко знайти відповіді на свої питання, а також скористатися різними ресурсами та матеріалами для розвитку проекту.

Вибір хостингу надає можливість оновити і деталізувати архітектуру системи з використанням визначених сервісів і ресурсів.

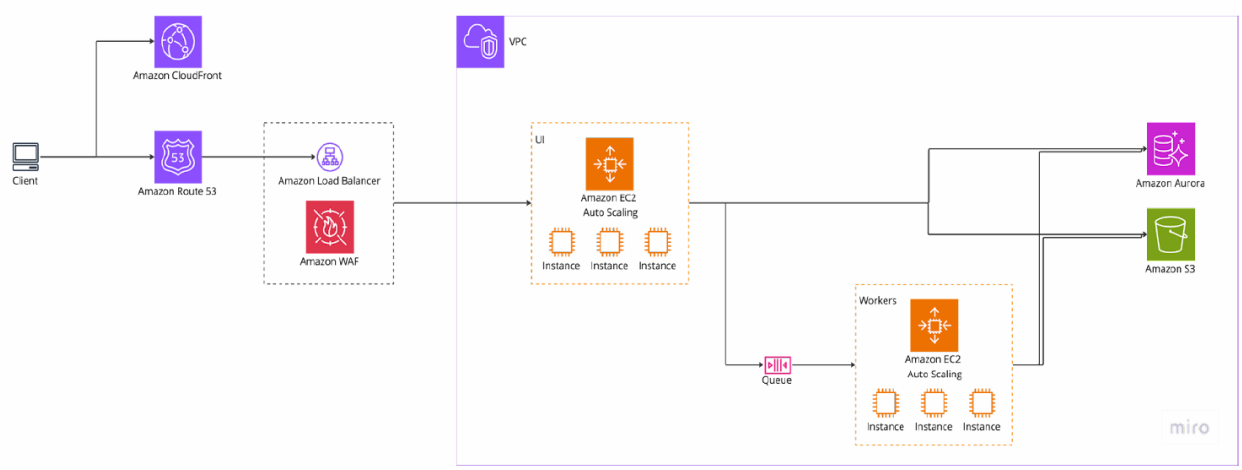

Рис. 3.3. Архітектурна схема розподілення на модулі UI на AWS

Результатом оновлення архітектури на рисунку 3.1, з використанням рекомендованих позначень для сервісів від AWS – є рисунок 3.3, на якому було введено наступні сервіси:

– Amazon CloudFront – це служба CDN (Content Delivery Network), яка дозволяє розповсюджувати контент, такий як зображення, відео та інші статичні файли, по серверах, що розташовані по всьому світу, для поліпшення швидкості завантаження та забезпечення низької затримки для кінцевих користувачів;

– Amazon Route 53 – це служба управління доменними іменами та системи назв (DNS), яка надає можливість реєстрації доменів, управління записами DNS та забезпечення високої доступності та надійності для додатків;

– Amazon Load Balancer – це сервіс балансування навантаження, який автоматично розподіляє вхідний трафік між різними екземплярами Amazon EC2 або іншими ресурсами, забезпечуючи високу доступність та оптимізацію використання ресурсів;

– Amazon WAF – це служба захисту веб-додатків, яка дозволяє захищати веб-додатки від вразливостей та атак, використовуючи налаштування правил та фільтрів, щоб контролювати та мінімізувати потенційні загрози;

– Amazon EC2 Auto Scaling – це сервіс автоматичного масштабування, який дозволяє автоматично змінювати кількість екземплярів Amazon EC2, щоб забезпечити оптимальну продуктивність та ефективне використання ресурсів в залежності від обсягу роботи;

– Amazon Aurora – це служба керування базами даних, яка надає високозабезпечений та масштабований сервіс для MySQL та PostgreSQL. Вона забезпечує високу доступність, надійність та автоматичне резервне копіювання;

– Amazon S3 – це служба зберігання об'єктів, яка надає масштабоване та безпечне зберігання для об'єктів, таких як файли, зображення та дані. Вона дозволяє зберігати та отримувати дані з будь-якого місця у мережі Інтернет.

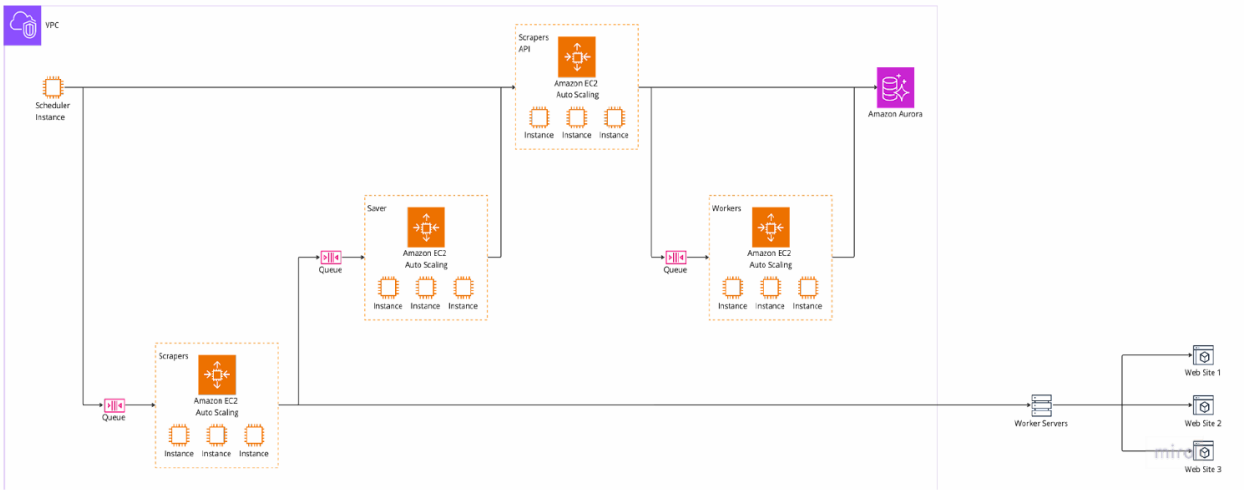

Рис. 3.4. Архітектурна схема розподілення на модулі системи збору даних на AWS

Результатом оновлення архітектури на рисунку 3.2, з використанням рекомендованих позначень для сервісів від AWS – є рисунок 3.4, на якому

можна побачити додаткове використання черг для комунікації між модулями. Черги на AWS реалізовані за допомогою Amazon SQS. Amazon SQS – це служба керування чергами повідомлень, яка дозволяє легко розподіляти роботу між різними компонентами додатка або системи. Вона надає простий та масштабований спосіб обміну повідомлень між різними частинами архітектури, а також забезпечує високу доступність та надійність.

3.3. Аналіз та вибір інструментів для процесу веб-скрапінгу

Активний розвиток електронної комерції, веб-сайтів та штучного інтелекту дали потужний поштовх розвитку інструментів та бібліотек для збору даних.

Інтернет магазини використовують різноманітні інструменти для збору даних та аналізу їхнього функціонування. Одним із ключових інструментів є аналітичні платформи, такі як Google Analytics, Adobe Analytics та Yandex Metrica. Ці інструменти надають глибокий аналіз трафіку, поведінки користувачів та результативності маркетингових кампаній.

Трекінг конверсії – ще одна важлива частина збору даних. Інструменти, такі як Hotjar та Crazy Egg, дозволяють відстежувати взаємодію користувачів із сторінками, аналізувати кліки та прокручування сторінок. Оптимізація конверсії може бути проведена за допомогою інструментів типу Optimizely та VWO.

Для збору даних про взаємодію користувачів із контентом інтернет магазини використовують системи управління контентом, а також інтеграції з соціальними мережами. Такі дані можуть включати відгуки, рейтинги товарів та коментарі користувачів.

Додатково, інтернет магазини можуть застосовувати інструменти машинного навчання для аналізу купівельної поведінки та прогнозування тенденцій ринку. Це допомагає бізнесу приймати обґрунтовані рішення щодо асортименту товарів та стратегій маркетингу.

Використання всіх цих інструментів дозволяє інтернет магазинам отримувати повний обсяг даних, необхідних для ефективного управління та вдосконалення свого функціонування.

До інструментів для збору даних також можна віднести системи управління взаємодією з клієнтами (CRM), які відстежують та аналізують інформацію про клієнтів, їхні покупки та інші важливі дані. Це дозволяє персоналізувати обслуговування та підходити до клієнтів індивідуально.

У сфері маркетингу використовуються інструменти автоматизації, такі як Mailchimp, SendinBlue чи HubSpot, для збору даних про ефективність електронних розсилок, взаємодії з рекламними кампаніями та конверсії.

Інтернет магазини також можуть використовувати інструменти аналізу соціальних мереж, які надають важливу інформацію про реакції користувачів на продукти та бренд через платформи, такі як Facebook Insights чи Twitter Analytics.

Загалом, використання різноманітних інструментів для збору та аналізу даних допомагає інтернет магазинам приймати обґрунтовані рішення, вдосконалювати свою діяльність та надавати користувачам персоналізований та ефективний досвід покупок.

Для збору даних інтернет магазинів найбільш універсальним методом є веб-скрапінг, при цьому вже є готові системи збору:

– Apify – це хмарна платформа для веб-скрапінгу та автоматизації веб-діяльності. Вона надає інструменти для створення веб-скраперів, розгортання їх в хмарі та керування видобутком даних;

– Octoparse – це візуальний веб-скрапер, який дозволяє користувачам створювати скрапери без програмування. Він має інтуїтивний інтерфейс та підтримує автоматичний збір даних з інтернет магазинів;

– Import.io – це інтернет-сервіс для збору даних з веб-сторінок. Він надає можливість створювати API для веб-сайтів, щоб легко добувати дані для подальшого аналізу;

– WebHarvy – це програма для парсингу вмісту веб-сторінок. Вона дозволяє користувачам витягувати дані з інтернет магазинів та зберігати їх у різних форматах, таких як CSV або Excel.

Перелічені вище системи вже є готовими програмними продуктами для надання послуг, при цьому вони або надають тільки можливості збору даних, без можливості аналітики, або мають обмежений функціонал.

Під час веб-скрапінгу важливо розуміти HTML-структуру веб-сторінок та взаємодію з DOM-деревом. Важливо звертати увагу на стійкість коду до можливих змін у HTML-структурі веб-сайту, оскільки вони можуть виникнути з часом. Регулярна перевірка та оновлення коду в разі потреби допоможуть уникнути проблем зі скрапінгом.

Перед тим як розпочати веб-скрапінг, рекомендується перевірити файл "robots.txt" веб-сайту для визначення обмежень щодо скрапінгу. Використання API від власників веб-сайтів може бути ефективнішим та менш конфліктогенним шляхом отримання даних.

Важливо враховувати, що при веб-скрапінгу потрібно уникати надмірного навантаження на сервери веб-сайтів та дотримуватися політик їх використання. Власники веб-сайтів можуть встановлювати обмеження та блокувати доступ для веб-скрапінгу, тому важливо дотримуватися правил і обмежень, вказаних на веб-сайті. Автоматизація процесу дозволяє здійснювати регулярне оновлення даних.

В цілому, веб-скрапінг є корисним інструментом для отримання даних, але використовувати його слід відповідально та з урахуванням етичних та юридичних аспектів.

Для реалізації нової системи веб-скрапінгу слід розглянути самі популярні інструменти, які в комбінації дозволять досягти мети проекту, це: Scrapy та Puppeteer.

Scrapy – це потужний фреймворк для веб-скрапінгу та видобування даних з веб-сайтів. Він написаний на мові програмування Python і створений з урахуванням потреб веб-розробників і дослідників даних. Scrapy надає ряд

переваг, що роблять його зручним і ефективним інструментом для збору і аналізу інформації з Інтернету. Ось деякі з них:

– асинхронність. Scrapy підтримує асинхронний підхід, що дозволяє швидко і ефективно збирати дані з різних джерел одночасно. Це допомагає зменшити час, потрібний для скрапінгу великих обсягів інформації;

– вбудовані засоби вилучення. Scrapy має потужні інструменти для вилучення даних із сторінок веб-сайтів. Ви можете використовувати CSS-селектори, XPath або регулярні вирази для точного визначення та вилучення необхідних даних;

– обробка вхідних даних. Scrapy дозволяє обробляти дані в реальному часі, перетворюючи їх у зручний для подальшого аналізу формат, наприклад, у JSON, CSV або базу даних;

– плагіни та розширення. Scrapy підтримує різні плагіни та розширення, що спрощують роботу з різними типами даних і джерелами, а також дозволяє легко налаштовувати і розширювати функціональність фреймворку;

– розподілені запити. Scrapy може розподіляти запити до веб-серверів для зменшення навантаження на ці сервери і збільшення швидкості скрапінгу.

Scrapy дозволяє автоматизувати процес збору даних і отримувати оновлення з веб-сайтів, що регулярно публікують нову інформацію. Фреймворк є надійним і широко використовується у галузі аналізу даних, розвідки і моніторингу веб-сайтів.

Puppeteer – це високорівнева Node.js бібліотека, яка надає інтерфейс для керування та автоматизації веб-переглядачів, таких як Google Chrome або Chromium. Ця бібліотека використовується для веб-скрапінгу, тестування веб-сайтів, автоматизації веб-додатків і багатьох інших завдань. Ключові особливості Puppeteer:

– контроль браузера. Puppeteer дозволяє створювати новий екземпляр веб-переглядача і керувати ним з коду, наприклад можна надсилати команди

для відкриття сторінок, навігації, взаємодії зі сторінкою та багатьма іншими операціями;

– захоплення вмісту. Puppeteer може використовуватись для отримання HTML-коду сторінки, зображень, стилів, JavaScript-файлів та іншого вмісту, що завантажується в браузері;

– робота з формами і кліками. Puppeteer надає зручний інтерфейс для взаємодії з формами, відправки запитів, кліків на посиланнях і багатьох інших подій на веб-сторінці;

– скріншоти і PDF-звіти. Puppeteer дозволяє створювати скріншоти сторінок та генерувати PDF-звіти з вмісту сторінок за допомогою;

– простота встановлення та використання. Puppeteer легко встановити через npm, і він має дружній API, який дозволяє швидко створювати скрапери та автоматизовані завдання;

– підтримка різних платформ. Puppeteer підтримує як Chrome, так і Chromium, що надає можливість вибрати, який браузер використовувати в залежності від потреби;

– асинхронність і чекери. Puppeteer ідеально підходить для асинхронної роботи та чекерів, що дозволяє ефективно очікувати завантаження сторінок та реагувати на події в браузері.

Загалом, Puppeteer є потужним і зручним інструментом для веб-скрапінгу і автоматизації завдань, пов'язаних з веб-переглядачами, завдяки чому він корисний для розробників і дослідників даних.

Важливо пам'ятати, що веб-скрапінг має бути використаний лише в межах закону та відповідно до політик конкретного веб-сайту. Слід зосереджуватися на забезпеченні конфіденційності та безпеки отриманих даних. Якщо отримані дані містять особисту інформацію, важливо дотримуватися відповідних стандартів і правил, щоб не порушити приватність користувачів та власників інформації.

Для покращення ефективності веб-скрапінгу, важливо використовувати асинхронні запити та оптимізуйте код. Це допоможе зменшити час

виконання та збільшити швидкість отримання даних. Також треба наголосити на необхідності регулярного моніторингу та аналізу результатів веб-скрапінгу, щоб переконатися, що вони відповідають очікуванням та не потребують коригувань.

Враховуючи всі ці аспекти, веб-скрапінг може бути ефективним інструментом для отримання даних з Інтернету з урахуванням етичних, юридичних та технічних аспектів.

3.4. Розробка інтерфейсу користувача

Розробка інтерфейсу користувача для системи веб-скрапінгу є важливим етапом в створенні функціонального та ефективного інструменту. Простота та логічність взаємодії з системою дозволяють користувачам швидко вивчити та використовувати всі її функції.

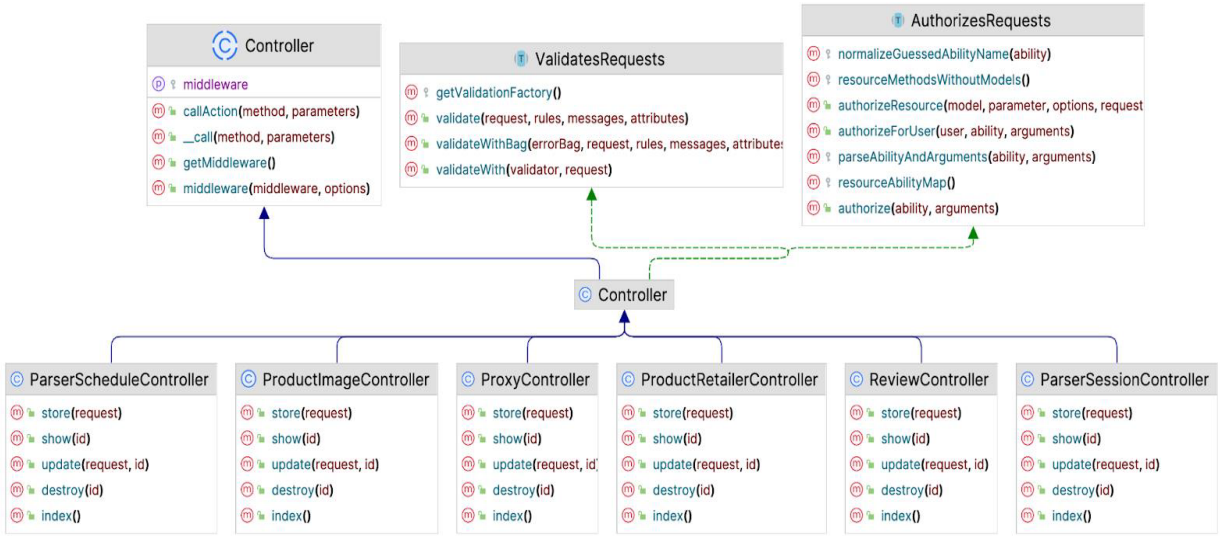

Рис. 3.5. Структура класів

При розробці інтерфейсу важливо враховувати різноманітні завдання, які можуть виконувати користувачі. Інтерфейс повинен бути адаптивним до різних завдань, починаючи від введення даних і закінчуючи їхнім аналізом. Інтуїтивне розміщення елементів та проста навігація сприяють зручності

використання системи. Структура класів необхідних для забезпечення функціонування системи представлена на рисунку 3.5.

Використання фреймворку Laravel, шаблонізатора Blade, а також бібліотеки з відкритим доступом AdminLTE дозволяють значно підвищити швидкість розробки за рахунок використання готових компонентів інтерфейсу і навіть цілих сторінок Прикладом готової сторінки – є сторінка логіну, для реалізації функціонування якої треба написати тільки бекенд частину.

Сторінка логіну представлена на рисунку 3.6, та представляє собою форму з полями для заповнення. Вона забезпечує можливість користувачам увійти в систему. В поточному додатку можливість реєстрації відсутня, тому для першого входу потрібно згенерувати користувача з правами admin, який вже зможе створити інших користувачів.

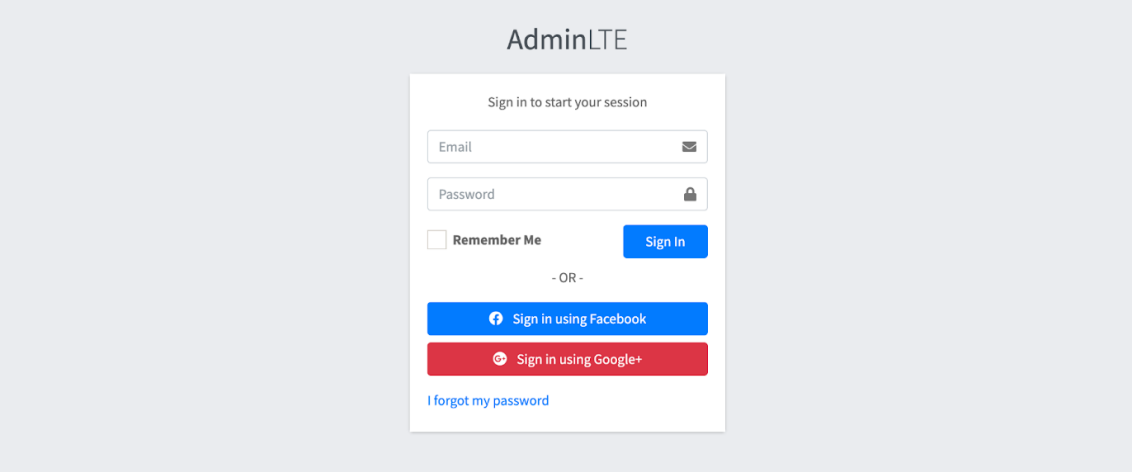

Рис. 3.6. Сторінка логіну

Після успішної авторизації користувач бачить сторінку Dashboard (рисунок 3.7), на якій має змогу ознайомитись з деякою статистикою стосовно роботі системи.

Завдяки AdminLTE і вбудованим інструментам візуалізації, таким як Chart.js, є можливість розширити статистику і додати нові типи графіків.

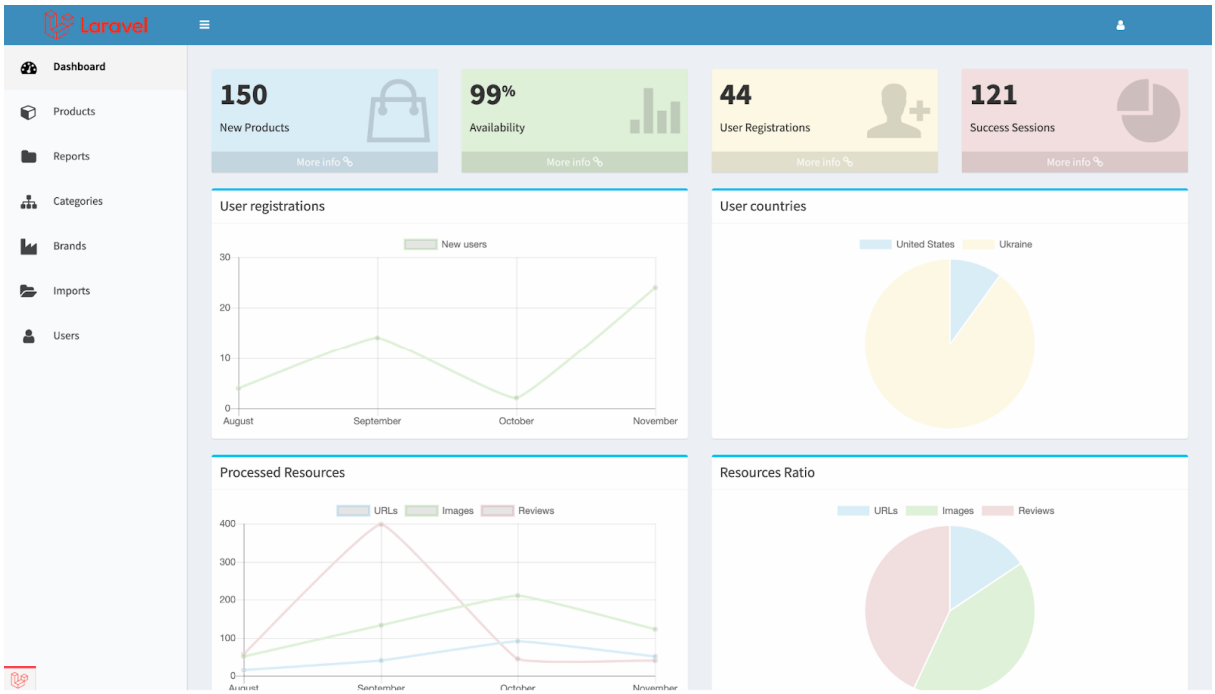

Рис. 3.7. Сторінка Dashboard

Сторінка створення бренда (рисунок 3.8) дозволяє створювати і налаштовувати бренди продуктів, а зручний редактор з функцією попереднього перегляду дозволить завантажити відповідне зображення бренду.

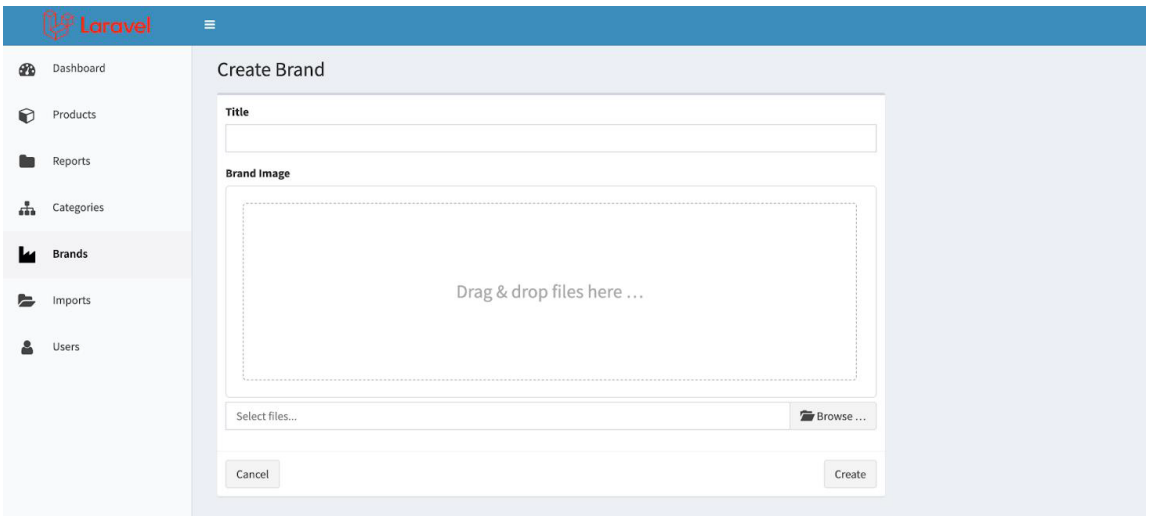

Рис. 3.8. Сторінка створення бренду

Для забезпечення функціонування UI була розроблена структура БД, що представлена на рисунку 3.9

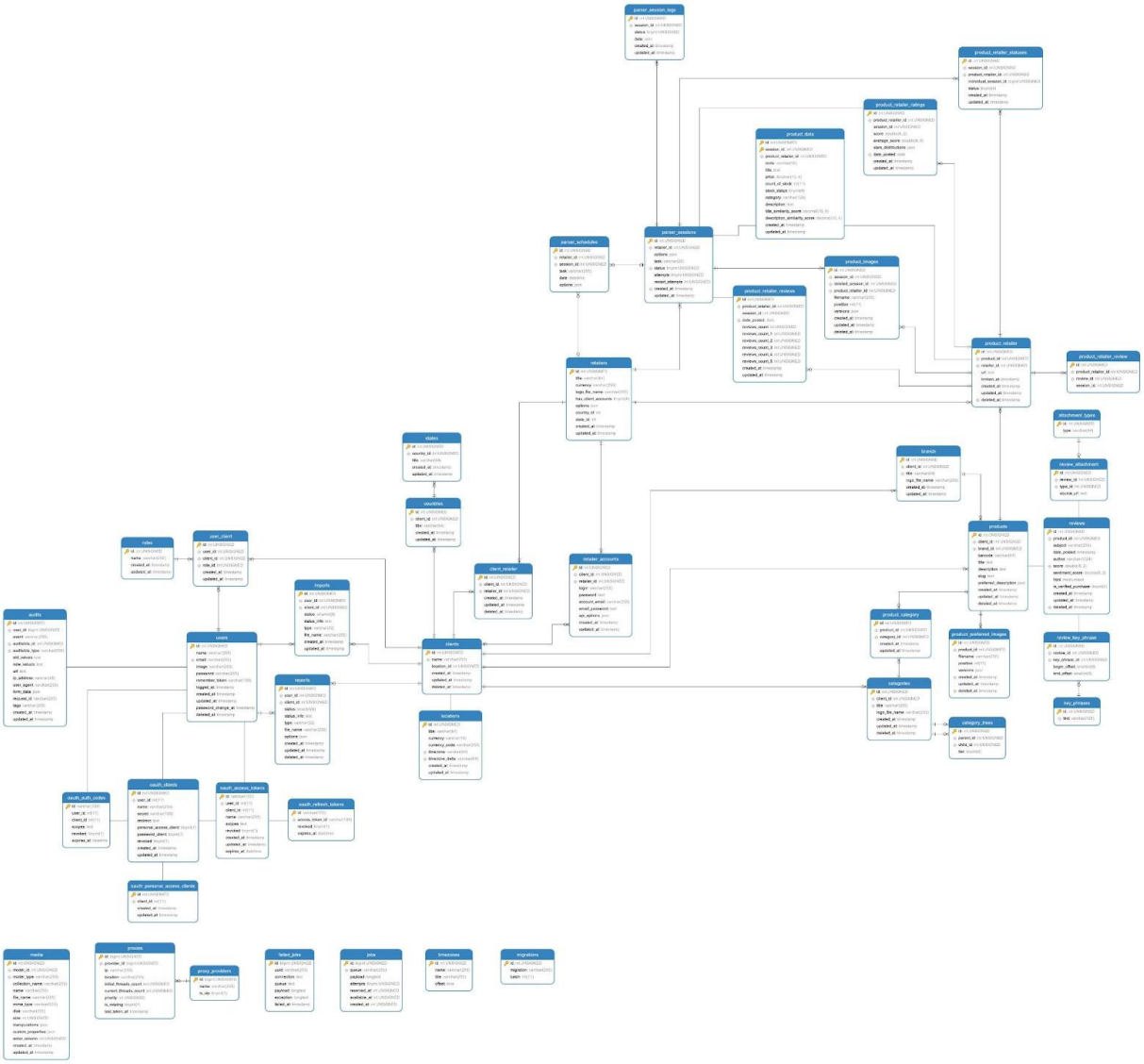

Рис. 3.9. Структура бази даних

3.5. Реалізація функціональності для процесу веб-скрапінгу

Технологія веб-скрапінгу даних з інтернет магазина в першу чергу залежить від технологій використаних для створення інтернет магазину, а також від методів захисту контенту. Складність збору даних може дуже сильно відрізнятись і бути як простим HTTPS запитом без використання додаткових сервісів, так і складним процесом з емуляцією веб браузера і дій користувача для отримання необхідної інформації.

Для реалізації збору даних з популярного інтернет магазина Walmart підійде фреймворк Scrapy, реалізований на мові Python. Процес веб-скрапінгу з інтернет магазина Walmart можна поділити на частини:

– отримання веб сторінки шляхом виконання HTTP запиту;

– пошук, вибірка і підготовка необхідних даних шляхом обробки відповіді запита;

– збереження отриманих даних в БД або передача на відповідний API.

Крок отримання веб сторінки шляхом виконання HTTP запиту не має додаткових складностей, крім необхідності використання проксі серверів, які допоможуть уникнути блокування IP-адреси сервера, з якого виконуються запити. Блокування IP-адреси призведе до повної неможливості подальшої роботи з веб сторінкою.

Пошук даних на веб сторінці виконується за допомогою вбудованих інструментів в фреймворк Scrapy, які дозволяють взаємодіяти з DOM структурою веб сайту та шукати необхідні елементи за допомогою селекторів. На рисунку 3.10 можна побачити приклад веб сторінки інтернет магазина Walmart з частиною DOM структури, відповідальної за вміст назви товару. Після визначення ключових позицій, які потрібно зібрати, з'являється можливість визначення необхідних DOM селекторів а також їх ієрархії.

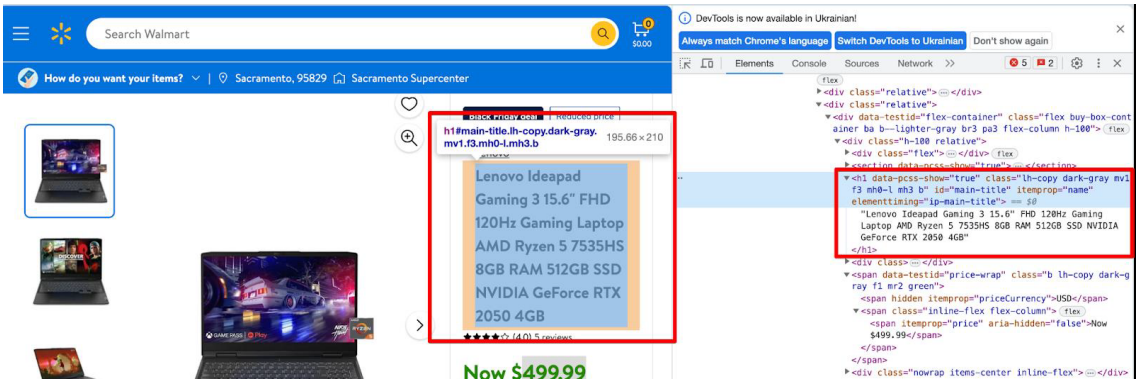

Рис. 3.10. Приклад сторінки і DOM структури продукту Інтернет магазину Walmart

Сучасні веб-сайти часто використовують JavaScript для візуалізації даних, що дозволяє надати веб-сторінкам інтерактивності, але це може стати проблемою для процесу збору даних. При використанні Scrapy – виконується тільки HTTP запит, що означає що будь які JavaScript-и не будуть виконані, це може потребувати додаткових HTTP запитів або додаткової обробки результатів першого запиту.

В випадку з Walmart – деяка інформація може бути отримана з JavaScript, який завантажується разом зі сторінкою продукту. Часто це може бути скрипт з id=”__NEXT_DATA__”, який містить в собі JSON з інформацією про продукт. Приклад такого скрипту з додатковою інформацією про продукт візуально представлений на рисунку 3.11.

Рис. 3.11. Додаткова інформація про продукт на сторінці Walmart

За допомогою обробки DOM структури сторінки, а також JSON інформації зі скрипта id=”__NEXT_DATA__”, з’являється можливість отримати всю необхідну інформацію про товар. Обробка представляю собою парсинг скрипту та його розбір на окремі частини. Приклад коду такої обробки представлено на рисунку 3.12.

```python
def parse_item(self, response):
    product = ProductItem()
    product['id'] = response.meta.get('product_id')
    product['url'] = response.meta['product_url']
    product['name'] = response.css('h1[itemprop="name"]::text').get().strip()
    product['price'] = response.css('span[itemprop="price"]::text').get().strip()
    product['currency'] = response.css('span[itemprop="priceCurrency"]::text').get().strip()

    match = re.compile(r'/(\d+)\?').search(product['url'])
    if match:
        product['code'] = match.group(1)

    xpath = '''
        //div[@data-testid="vertical-carousel-container"]
        //div[@data-testid="media-thumbnail" and not(.//*[@data-testid="ui-video-play-button-testid"])]//img/@src
    '''
    images = response.xpath(xpath).getall()
    product['images'] = [i.split('?')[0] for i in images]

    additional_data = json.loads(response.css('#__NEXT_DATA__::text').get())

    product['short_description'] = additional_data['props']['pageProps']['initialData']['data']['idml']['shortDescription']
    product['long_description'] = additional_data['props']['pageProps']['initialData']['data']['idml']['longDescription']

    availability = additional_data['props']['pageProps']['initialData']['data']['product']['availabilityStatus']
    product['stock'] = int(availability == "IN_STOCK")

    yield product
```

Рис. 3.12. Приклад коду для веб-скрапінгу інформації про продукт з Інтернет магазина Walmart за допомогою фреймворка Scrapy

Інтернет магазин Amazon має інші способи захисту сторінки, ніж Walmart, також багато контенту формується за допомогою JavaScript, що робить неможливим отримати всю інформацію про продукт завдяки HTTP запиту. Підвищена складність збору інформації вимагає використання додаткових інструментів, таких як відкрита бібліотека Puppeteer, реалізована на Node.js.

Процес збору даних з інтернет магазина Amazon можна поділити на частини:

– отримання веб сторінки шляхом емуляції браузера;

– пошук, вибірка і підготовка необхідних даних шляхом обробки відповіді запиту;

– збереження отриманих даних в БД або передача на відповідний API.

Головною відмінністю від скрапінгу Walmart є перший крок, в якому потрібно виконати емуляцію браузера і вже потім збирати інформацію

завдяки вбудованим інструментам в Puppeteer для роботи з DOM. Приклади структури і коди наведені на рисунках 3.13 і 3.14.

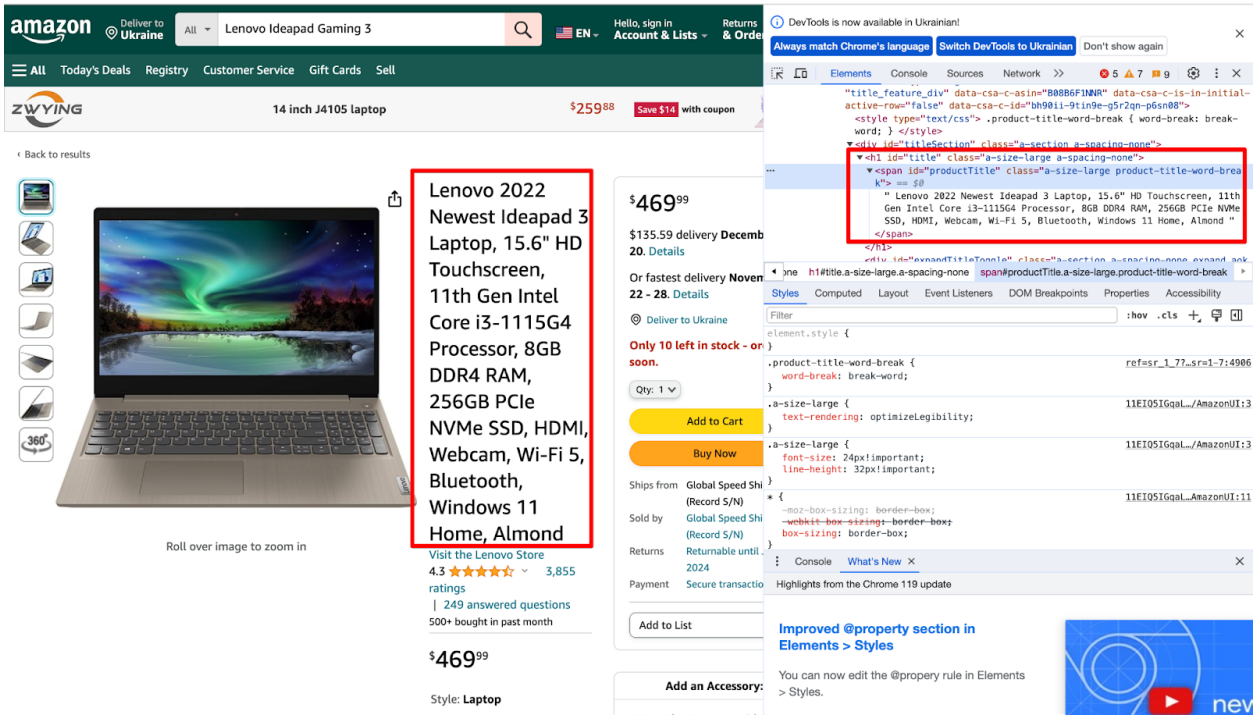

Рис. 3.14. Приклад сторінки і DOM структури продукту Інтернет магазина Amazon

```
async parseTitle(_response) : Promise<any>  {
    return this.page.evaluate(
        () : ...  => {
            let title : Element  = document.querySelector( selectors: 'span#productTitle');

            if (title !== null && typeof title.innerText === 'string') {
                return title.innerText.trim();
            }

            return '';
        }
    );
}
```

Рис. 3.15. Приклад коду для збору інформації про заголовок продукту з інтернет магазина Amazon за допомогою Puppeteer

Окрім головної інформації про продукт є можливість збирати відгуки покупців про товар (рисунок 3.16). Така інформація може допомогти

виробникам продуктів покращити товар, а продавцям коригувати ціну і плани продажів.

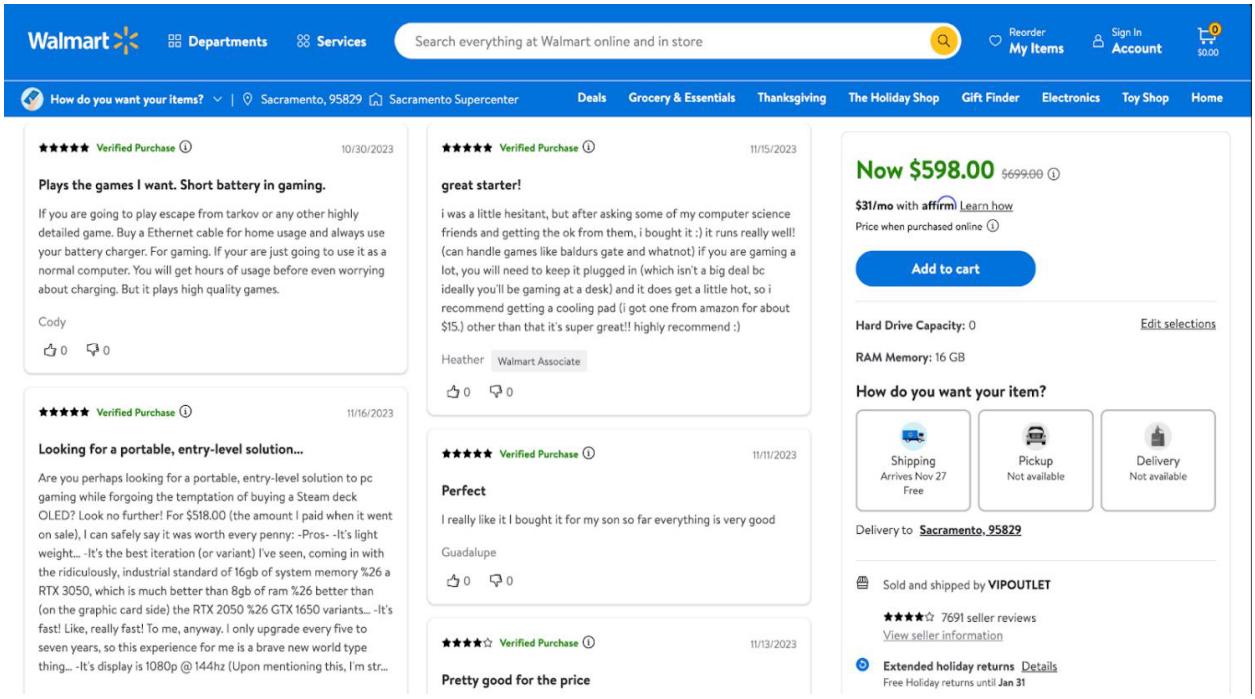

Рис. 3.16. Відгуки товару Інтернет магазина Walmart

Зібрані відгуки можуть принести більше користі, якщо вони будуть проаналізовані, відсортовані і розбиті на ключові слова завдяки Amazon Comprehend. Amazon Comprehend – це служба обробки природної мови (NLP), яка використовує машинне навчання для аналізу текстів (рисунок 3.17).

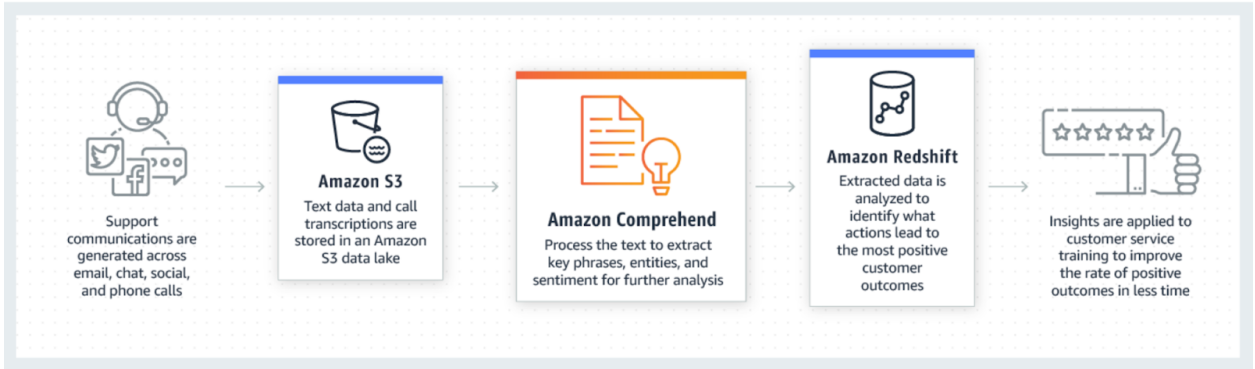

Рис. 3.17. Приклад взаємодії з Amazon Comprehend

Робота в Amazon Comprehend може проходити як з вже натренованими моделями, які пропонує AWS, так і з новими моделями, навчання яких буде проходити на ресурсах AWS, на основі даних завантажених користувачем.

Прикладом роботи з Amazon Comprehend може стати аналіз одного за відгуків користувача про товар на моделі, яку тренував AWS. Параметри запиту наступні:

{
"Text": "Are you perhaps looking for a portable, entry-level solution to pc gaming while forgoing the temptation of buying a Steam deck OLED? Look no further! For $518.00 (the amount I paid when it went on sale), I can safely say it was worth every penny: -Pros- -It's light weight... -It's the best iteration (or variant) I've seen, coming in with the ridiculously, industrial standard of 16gb of system memory %26 a RTX 3050, which is much better than 8gb of ram %26 better than (on the graphic card side) the RTX 2050 %26 GTX 1650 variants... -It's fast! Like, really fast! To me, anyway. I only upgrade every five to seven years, so this experience for me is a brave new world type thing... -It's display is 1080p @ 144hz (Upon mentioning this, I'm stressing upon the hertz to be the benefit here rather than the resolution...) -Cons- -The overall construction feels cheap... Although I don't mind it since I paid a little over $500.00 for it. -I recommend not playing unplug, for the battery life isn't that good %26 you lose fps... -The GPU (3050) inside is only worth 4gb of vram, which isn't great. -The touchpad is... meh... But I don't mind it... Overall, I'm very much pleased with it. Back in the day, if you wanted to play any AAA game at a reasonable setting, you'd have to shell out at least a grand. This review (and the positivity thereof) is really centered about getting this sucker at the price I did or even lower. Any higher %26 I'd say it isn't worth it. When I first saw this deal, it was better %26 exclusive for Walmart plus members %26 it sold out quick %26 for a good reason. I lucked

```
out with the second batch which saw a slight price increase. On
the whole, I'm happy.",
"LanguageCode": "en"
}
```

Аналіз настрою (Sentiment Analysis) – це сервіс, який використовує машинне навчання для визначення тону тексту, а саме чи відбувається у тексті позитивний, негативний чи нейтральний відгук. AWS Comprehend може виявляти емоції та ступінь впевненості у висловлюваннях. Для використання аналізу настрою в AWS Comprehend, ви можете відправити текстові дані на обробку за допомогою API-виклику або використовувати консоль AWS Comprehend.

Процес аналізу настрою включає в себе наступні етапи:

– введення тексту – передача тексту для аналізу до сервісу Comprehend;

– аналіз настрою. Comprehend використовує алгоритми машинного навчання для визначення того, наскільки текст виражає позитивний, негативний або нейтральний настрій. Також, він може надати інформацію про емоції, такі як радість, сум, гнів тощо;

– вивід – результат аналізу повертається у вигляді структурованої відповіді, де вказано виявлений настрій та, можливо, емоції.

Нижче наведено приклад аналізу відгуку:

```
{
     "Sentiment": {
     "Sentiment": "POSITIVE",
     "SentimentScore": {
     "Positive": 0.8915513157844543,
     "Negative": 0.0081268809735775,
     "Neutral": 0.006918027065694332,
     "Mixed": 0.09340381622314453
}
     }
}
```

Розбиття на ключові фрази (Keyphrase Extraction) – Comprehend може робити визначення та виділення найбільш важливих та семантично значущих фраз у тексті у межах процесу розбиття на ключові фрази. Ця функція допомагає розуміти головні теми або поняття, які взято за ключові, у тексті. Для відгуків про товар таке розбиття може виділити ключові слова або фрази, які найкраще описують основні враження чи коментарі користувачів. Це може допомогти у визначенні загальної тенденції або головних ідей, які виражені у тексті.

Процес розбиття на ключові фрази схожий з процесом аналізу настрою, окрім другого етапу, де замість аналізу настрою йде розбиття на ключові слова.

Результат розбиття на ключові фрази може містити сотні елементів. У прикладі нижче наведена частина результату обробки відгуку:


```
{
"KeyPhrases": [
     {
     "Score": 0.6280012726783752,
     "Text": "pc gaming",
     "BeginOffset": 64,
     "EndOffset": 73
},
{
     "Score": 0.9999241828918457,
     "Text": "the temptation",
     "BeginOffset": 89,
     "EndOffset": 103
},
{
     "Score": 0.9939930438995361,
     "Text": "a Steam deck OLED",
     "BeginOffset": 114,
     "EndOffset": 131
},
```

```
{
     "Score": 0.9998616576194763,
     "Text": "the amount",
     "BeginOffset": 163,
"EndOffset": 173
}
     ]
}
```


Висновки до розділу

Отже, в цьому розділі представлено опис функціональних та технічних вимог дозволив досягти розбиття системи на модулі, які будуть відповідати поставленим вимогам і зможуть забезпечити ефективне функціонування системи, а також взаємодію з інструментами для збору даних.

Все це допомогло визначитись з критеріями для розгортання системи а також вибором хостингу. Використання Amazon дозволяє отримати гнучку, здатну масштабуватись систему з великим рівнем надійності та захищеності, а також надає можливість використовувати переваги великої кількості доступних сервісів, які пропонує Amazon і які часто сумісні між собою.

Загалом, проектування системи веб-скрепінгу для інтернет-магазину враховує як функціональні, так і технічні аспекти, що сприяє створенню ефективної та надійної інфраструктури для збору та обробки даних.

Також було реалізовано інтерфейс користувача, а також безпосередньо систему веб-скрепінгу для збору даних Інтернет магазинів, що відповідає всім заявленим функціональним вимогам до системи.

Розроблена структура БД дозволяє зберігати зібрані дані, а також містить конфігурації системи, необхідні для її роботи. Завдяки розробленому UI є можливість взаємодії з системою, при цьому ця взаємодія може бути як з мобільних пристроїв, планшетів, так і з персональних комп’ютерів, а завдяки

скраперам є можливість отримувати свіжі дані з таких інтернет магазинів як Walmart та Amazon.

Використання таких інструментів як Amazon Comprehend надає потужні можливості для подальшої обробки і аналізу зібраних даних, наприклад аналіз настрою в відгуках можна використати для моніторингу задоволеності клієнтів від товару або підвищення рівня продажів завдяки маркетинговим кампаніям.

ВИСНОВКИ

В магістерській роботі досліджено моделі, методи та алгоритми веб-скрапінгу засобами машинного навчання.

В рамках даної роботи був проведений аналіз існуючих методів видобування та аналізу даних з ресурсів у мережі інтернет. Також було розглянуто методи класифікації текстів. Розроблений алгоритм було імплементовано та порівняно з наявним у індустрії програмним забезпеченням за допомогою тренування алгоритмів машинного навчання на зібраних даних, та застосовано результати натренованих алгоритмів для розробки програмного додатку.

Було проаналізовано існуючі підходи та програмні засоби і сервіси, вже присутні на ринку видобування структурованої інформації. Було розроблено простий до імплементації алгоритм видобування структурованої інформації, який суміщує в собі процеси краулінгу та скрапінгу, таким чином спрощуючи інтеграційні та експлуатаційні процеси використання алгоритму, а також окреслено загальний алгоритм підготовки текстових даних до аналізу методами машинного навчання.

Було оглянуто та викладено математичні засади деяких алгоритмів векторизації даних та відповідних методів машинного навчання, які застосовуються для обробки та класифікації текстових даних.

Окреслені алгоритми підготовки, векторизації та класифікації. Результати реалізації вищезазначених алгоритмів порівняно: алгоритм видобування було порівняно з існуючим на ринку найближчим аналогом за результатами тренування алгоритмів машинного навчання на видобутих даних.

На прикладі розроблених алгоритмів машинного навчання було доведено, що існуючі засоби машинного навчання здатні до повноцінної роботи та класифікації текстів на масиві даних, який включає в себе різну інформацію, а також що результати тренування класичних алгоритмів

машинного навчання є співставними з більш складними у розробці та тренуванні нейронними мережами довгої короткочасної пам'ятті.

В результаті розроблено систему веб-скрапінгу для збору збору даних Інтернет магазинів. Розроблена система відповідає всім заявленим функціональним та технічним вимогам а також має такі переваги як:

– зручний, сучасний та адаптивний UI;

– можливість швидкого доступу до важливої статистики;

– високий рівень захисту;

– високий рівень доступності;

– можливості для розширення функціональності, як шляхом додавання нового функціоналу так і нових інтернет магазинів.

Розроблена система може використовуватись в Інтернет комерції для забезпечення точною та актуальною інформацією про стан ринку та продуктів у інтернет-магазинах, а також як інструмент підвищення продажу конкретних брендів, груп товарів і окремих продуктів.

На прикладі розробленого програмного додатку веб-скрапінгу було доведено, що прості у розробці та підтримці загальні алгоритми видобування структурованої інформації є конкурентноспроможними з існуючими високотехнологічними системами за рахунок простоти імплементації, підтримки та цінових характеристик.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Web Scraping Tools to Extract Online Data [Електронний ресурс]. – 2019 – Режим доступу до ресурсу: https://www.hongkiat.com/blog/web-scraping-tools/
2. Introduction to Web Scraping - GeeksforGeeks.GeeksforGeeks. URL: https://www.geeksforgeeks.org/introduction-to-web-scraping/?ref=rp.
3. Що таке веб-скрейпінг і як він пов’язаний з проксі. Enterprise data gathering infrastructure | ASTROPROXY. URL: https://astroproxy.com/ua/blog/shho-take-veb-skreiping-i-yak-vin-povyazanii-z-proksi.
4. Web Scraping. Techopedia. URL: https://www.techopedia.com/definition/5212/web-scraping
5. What is Web Scraping and How to Use It? – GeeksforGeeks.Geeksfor Geeks. URL: https://www.geeksforgeeks.org/what-is-web-scraping-and-how-to-use-it/
6. ScrapingBot • Web Scraping API -Extract HTML content.Scraping-bot.io. URL:https://www.scraping-bot.io/
7. Web Scraping Tool & Free Web Crawlers | Octoparse.Web Scraping Tool & Free Web Crawlers | Octoparse. URL:https://www.octoparse.com/
8. A Comparative Study on Web Scraping [Електронний ресурс]. – 2015 – Режим доступу до ресурсу: http://ir.kdu.ac.lk/bitstream/handle/345059.pdf
9. Olston, C. and Najork, M. Web crawling. 2010. – Foundations and Trends in Information Retrieval 3:175-246.
10. Web Content Mining Techniques: A Survey [Електронний ресурс]. – 2012 – Режим доступу до ресурсу: https://research.ijcaonline.org/volume47/number11/pxc3880266.pdf
11. What are the Different Scraping Techniques [Електронний ресурс]. – 2017 – Режим доступу до ресурсу: https://www.shieldsquare.com/what-are-the-different-scraping-techniques/

12. Types of Web Scraping Tools [Електронний ресурс]. – 2017 – Режим доступу до ресурсу: https://medium.com/prowebscraper/types-of-web-scraping-tools-940f824622fb
13. The dangers of web scraping [Електронний ресурс]. – 2016 – Режим доступу до ресурсу: https://www.information-age.com/dangers-web-scraping-123461971/
14. Legality of Web Scraping and Crawling [Електронний ресурс]. – 2017 – Режим доступу до ресурсу: https://urlzs.com/3n4QC
15. Krotov, Vlad & Silva, Leiser. (2018). Legality and Ethics of Web Scraping.
16. Protection Against Web Scraping [Електронний ресурс]. – 2017 – Режим доступу до ресурсу: https://blog.jscrambler.com/protect-your-site-against-web-scraping/
17. P. H. Cording (2011). Algorithms for Web Scraping.
18. P YesuRaju and P KiranSree (2013). A language independent web data extraction using vision based page segmentation algorithm.
19. Zehuan Cai, Jin Liu, Lamei Xu, Chunyong Yin, Jin Wang (2017). A Vision Recognition Based Method for Web Data Extraction. Advanced Science and Technology Letters Vol.143 (AST 2017), pp.193-198
20. Jiawei Zhang and Bowen Dong and Philip S. Yu (2019). FAKEDETECTOR: Effective Fake News Detection with Deep Diffusive Neural Network
21. Jayashree M Kudari, Varsha V, Monica BG, Archana R (2020). Fake News Detection using Passive Aggressive and TF-IDF Vectorizer. International Research Journal of Engineering and Technology (IRJET) Volume: 07 Issue: 09 | Sep 2020. e-ISSN: 2395-0056 p-ISSN: 2395-0072.
22. Silva, C.. (2003). The importance of stop word removal on recall values in text categorization. 3. 1661 - 1666 vol.3. 10.1109/IJCNN.2003.1223656.
23. Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, Yoram Singer (2006). Online Passive-Aggressive Algorithms. Journal of Machine Learning Research 7 (2006) 551–585

24. S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. Neural Computation, 9(8):1735–1780, 1997.
25. Basaldella, Marco & Antolli, Elisa & Serra, Giuseppe & Tasso, Carlo. (2018). Bidirectional LSTM Recurrent Neural Network for Keyphrase Extraction. 10.1007/978-3-319-73165-0_18.
26. Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J.: Gradient flow in recurrent nets: the difficulty of learning long-term dependencies (2001)
27. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional lstm and other neural network architectures. Neural Networks 18(5), 602–610 (2005)
28. Rui Pereira, Marco Couto, Francisco Ribeiro, Rui Rua, Jácome Cunha, João Paulo Fernandes, and João Saraiva. 2017. Energy efficiency across programming languages: how do energy, time, and memory relate? In Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering (SLE 2017). Association for Computing Machinery, New York, NY, USA, 256–267.
29. Lei, Kai & Ma, Yining & Tan, Zhi. (2014). Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js. 661-668. 10.1109/CSE.2014.142.
30. Офіційна документація Laravel. URL: https://laravel.com/.
31. Офіційна документація AdminLTE. URL: https://adminlte.io/.
32. M.Stauffer. Laravel: Up and Running. URL: https://laravelupandrunning.com/.
33. Adel F. Architecture of complex web applications: With examples in Laravel (PHP). Independently published, 2019.
34. Офіційна документація Scrapy. URL: https://scrapy.org/.
35. Офіційна документація Node.js. URL: https://nodejs.org/en/docs/.
36. Офіційна документація Puppeteer. URL: https://pptr.dev/.
37. Redmond Paul. Lumen Programming Guide: Writing PHP Microservices, REST and Web Service APIs. Apress, 2016.

38. Pettit T., Cosentino S. The MySQL Workshop: A practical guide to working with data and managing databases with MySQL.Birmingham: Packt Publishing, 2022.
39. Офіційна документація AWS. URL: https://aws.amazon.com/.
40. R. Kalpana and K. L. Bansal, “A comparative study of data mining tools,” International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, 2014 — [Текст]
41. S. S. Prasad, M. Sonali, and S. Sonali, “Border security up gradation using data mining,” International Journal of Soft Computing and Engineering, vol. 4, March 2014
42. G. Faryral, B. H. Wasi, and Q. Usman, “Terrorist group prediction using data classification,” presented at the International Conferences of Artificial Intelligence and Pattern Recognition, Malaysia, 2014
43. Ghadi, Abderrahim. (2018). Application of Data Mining Classification Algorithms for Breast Cancer Diagnosis
44. R Umamaheswaran, R. Radha, ALLA. Preethi — [Текст] — in international research journal of engineering and technology (irjet). — Crime Rate Prediction using KNN — VOLUME: 07, ISSUE: 05, MAY 2020 — pp. 3065–3071
45. V. Pednekar, T. Mahale, P. Gadhave, A. Gore — [Текст] — in International Journal on Recent and Innovation Trends in Computing and Communication — Crime Rate Prediction using KNN — VOLUME: 6, ISSUE: 1, ISSN: 2321-8169 — pp. 124-127
46. Dryer, A. J. and Stockton, J. Internet ’Data Scraping’: A Primer for Counseling Clients. 2013. – New York Law Journal 15:1-3.
47. Mitchell, R. Web Scraping with Python: Collecting Data from the Modern Web. S. : O’Reilly Media, 2015.