Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Модель автоматизації збору даних з вебресурсів на основі генеративного штучного інтелекту

(тема)

Виконав:
здобувач 2 року навчання,
групи СПм-23-3
Ярослав КОПАЙЛО
(власне ім’я, прізвище)

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

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

Освітня програма
Системне програмування
(повна назва освітньої програми)

Керівник: доц. Тетяна ФІЛІМОНЧУК
(посада, власне ім’я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім’я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

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

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

Освітня програма Системне програмування
(повна назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Копайло Ярославу Руслановичу
(прізвище, ім’я, по батькові)

1. Тема роботи Модель автоматизації збору даних з вебресурсів на основі генеративного штучного інтелекту

затверджена наказом по університету від “ 21 ” квітня 2025 р. № 296 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 16 червня 2025 р.

3. Вхідні дані до роботи 1) генеративний штучний інтелект, 2) формат збирання даних, 3) модульна архітектура, 4) структура бази даних, 5) набір скраперів

4. Перелік питань, що потрібно опрацювати у роботі
1) Порівняльний аналіз моделей збору даних
2) Проектування моделі автоматизованого збору даних на базі генеративного ШІ
3) Аналіз технологій і середовища реалізації
4) Методика оцінювання ефективності автоматизованого збору даних
5) Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд-презентація – 16 слайдів

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

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Огляд вебскрапінг моделей	22.04.25-29.04.25	
2	Вибір та обґрунтування методики дослідження	30.04.25-05.05.25	
3	Вибір інструментальних засобів	06.05.25-09.05.25	
4	Розробка вебскрапінг моделі	10.05.25-21.05.25	
5	Проведення експериментів	22.05.25-02.06.25	
6	Оформлення матеріалів кваліфікаційної роботи	03.06.25-05.06.25	
7	Подання кваліфікаційної роботи керівникові та	06.06.25-09.06.25	
	її попередній захист		
8	Подання кваліфікаційної роботи на рецензу-	10.06.25-12.06.25	
	вання		

Дата видачі завдання " 21 " квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Тетяна ФІЛІМОНЧУК
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 82 с., 1 рисунок, 3 табл., 25 джерел.

ГЕНЕРАТИВНИЙ ШТУЧНИЙ ІНТЕЛЕКТ, ВЕБЗБИРАННЯ ДАНИХ, АВТОМАТИЗАЦІЯ, МОДУЛЬНА АРХІТЕКТУРА, БАЗА ДАНИХ, СКРАПЕРИ, ГЕНЕРАТИВНА МОДЕЛЬ, LARGE LANGUAGE MODEL.

Метою роботи є вдосконалення моделі автоматизованого вебскрапінгу через інтеграцію сучасних технологій штучного інтелекту та оптимізацію архітектури. Особливу увагу слід приділити адаптивності системи до змін вебресурсів, масштабованості обчислювальних процесів та підвищенню продуктивності за рахунок використання розподілених обчислень.

У ході виконання кваліфікаційної роботи було здійснено аналіз сучасних методів автоматизації збору даних із вебресурсів на основі генеративного штучного інтелекту, зокрема із застосуванням великих мовних моделей (LLM). Розглянуто переваги та недоліки традиційних вебскраперів, визначено основні критерії ефективності таких систем: адаптивність до змін структури вебсторінок, гнучкість, стабільність, продуктивність та зручність обслуговування. Було запропоновано розширену модель, яка передбачає багаторівневу взаємодію між окремими модулями. За результатами дослідження проведено порівняння запропонованої моделі з існуючими підходами та підтверджено, що інтеграція LLM забезпечує високу точність збору інформації, знижує потребу в ручному налаштуванні та покращує адаптивність до швидкозмінних вебсередовищ. Результати дослідження візуалізовано графічно та сформовано практичні рекомендації щодо використання запропонованої моделі в аналітичних проєктах, які працюють з великими обсягами вебданих.

ABSTRACT

Master’s thesis: 82 pages, 1 figure, 3 tables, 25 sources.

GENERATIVE ARTIFICIAL INTELLIGENCE, WEB DATA COLLECTION, AUTOMATION, MODULAR ARCHITECTURE, DATABASE, SCRAPERS, GENERATIVE MODEL, LARGE LANGUAGE MODEL.

The purpose of this work is to refine the automated web-scraping model through the integration of cutting-edge artificial intelligence technologies and architectural optimization. Special emphasis is placed on the system’s adaptability to changes in web resources, the scalability of computational processes, and enhanced performance through distributed computing.

In the course of this qualification study, we analyzed modern methods for automating data collection from web resources based on generative AI—specifically leveraging large language models (LLMs). We examined the advantages and drawbacks of traditional web scrapers and identified the key performance criteria for such systems: adaptability to changes in page structure, flexibility, stability, throughput, and ease of maintenance. An extended model was proposed, incorporating a multi-layer interaction scheme among individual modules. Comparing our proposed model with existing approaches demonstrated that integrating LLMs delivers high data-collection accuracy, reduces the need for manual configuration, and improves responsiveness to rapidly evolving web environments. The findings have been presented graphically, and practical recommendations have been formulated for applying the proposed model in analytical projects that handle large volumes of web data.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ADCM – модель автоматизації збору даних з вебресурсів (англ., Automation Data Collection Model)

AJAX – асинхронний JavaScript і XML (англ., Asynchronous JavaScript and XML)

AM – модуль аналізу сайту (англ., Analysis Module)

AnM – модуль аналітики (англ., Analytics Module)

API – інтерфейс програмування додатків (англ., Application Programming Interface)

BSON – бінарний JSON (англ., Binary JSON)

CNN – згорткова нейронна мережа (англ., Convolutional Neural Network)

DCM – модуль збирання даних (англ., Data Collection Module)

DeepDOM – модель машинного навчання для класифікації вузлів DOM

DOM – модель об’єкта документа (англ., Document Object Model)

DPSM – модуль обробки та збереження даних (англ., Data Processing & Storage Module)

ETL – процес вилучення, перетворення та завантаження (англ., Extract, Transform, Load)

HTML – мова розмітки гіпертексту (англ., HyperText Markup Language)

JSON – текстовий формат обміну даними (англ., JavaScript Object Notation)

LLM – великі мовні моделі (англ., Large Language Models)

LSTM – довга короткочасна пам’ять (англ., Long Short-Term Memory)

NLP – обробка природної мови (англ., Natural Language Processing)

NoSQL – нереляційні бази даних (англ., Not Only SQL)

RM – модуль генерації правил скрапера (англ., Rules Module)

SVM – метод опорних векторів (англ., Support Vector Machine)

XML – мова розмітки (англ., eXtensible Markup Language)

ВСТУП

Інформаційні технології розвиваються стрімкими темпами, впливаючи на всі сфери життя [1]. Однією з ключових сучасних технологій є вебскрапінг – процес автоматизованого збору даних із вебсайтів, який суттєво спрощує отримання, аналіз та використання інформації. Він широко застосовується у бізнесі, ринковій аналітиці, моніторингу конкурентів та інших сферах, дозволяючи автоматизувати процеси збору даних і мінімізувати витрати на ручну обробку.

Останні досягнення у сфері штучного інтелекту, зокрема великі мовні моделі (LLM – Large Language Models), значно розширили можливості автоматизації технологічних процесів. LLM демонструють здатність аналізувати текстові дані, генерувати програмний код, структурувати інформацію та формувати складні правила, що базуються на розпізнаванні різних патернів. Завдяки цьому вони можуть зробити вебскрапінг ще гнучкішим, адаптивним та більш ефективним у роботі з вебданими.

Автоматичний аналіз вебсторінок дозволяє не лише ефективно структурувати інформацію, а й визначати ключові закономірності у великих масивах даних. Застосування LLM у цьому процесі забезпечує розширені можливості оптимізації та створення більш ефективних алгоритмів збору інформації.

Для реалізації таких рішень однією з найкращих мов програмування є Python. Вона пропонує широкий набір бібліотек для вебскрапінгу, обробки даних та інтеграції з LLM. Простий синтаксис, гнучкість та активна спільнота розробників роблять її ідеальним вибором для масштабованих та продуктивних систем. Додатково, такі інструменти, як RabbitMQ (система черг повідомлень), Prometheus (система моніторингу) та Grafana Loki (система логування) допомагають забезпечити надійний обмін даними, підвищити гнучкість архітектури та спростити масштабування систем.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Сучасний стан і проблематика збору даних із вебресурсів

1.1.1 Актуальність теми

У сучасних умовах стрімкого розвитку інформаційних технологій та глобальної діджиталізації збір даних із вебресурсів набуває критичного значення для економічного, наукового та соціального розвитку суспільства. Зростання обсягів інформації, яка щодня генерується в Інтернеті, вимагає новітніх та ефективних методів її обробки та аналізу. Вебресурси стають головним джерелом даних для різних систем, починаючи від маркетингових досліджень та закінчуючи аналітичними платформами, що використовуються у наукових дослідженнях та прийнятті управлінських рішень.

Ключовою проблемою, що виникає в умовах сучасної інформаційної експансії, є недосконаліть традиційних підходів до збору даних [2]. Ручний моніторинг вебсторінок, який колись був достатньо ефективним для невеликих обсягів інформації, сьогодні виявляється недостатнім через швидке зростання даних та їхню розмаїтість. Ручне налаштування засобів збору даних не тільки вимагає значних трудових ресурсів, але й часто призводить до виникнення помилок при обробці інформації. Такий підхід не дозволяє оперативно реагувати на зміни у структурі вебресурсів, що є вирішальним фактором для підтримки актуальності та достовірності отриманих даних.

Крім того, існує проблема відсутності узгодженості засобів збору даних, які працюють із різними форматами та стандартами подання інформації. Розділення функцій між окремими системами створює додаткові труднощі при інтеграції даних, що отримуються з різних джерел. Це призводить до частих ситуацій, коли дані з одних ресурсів не сумісні з інформацією, зібраною з інших, що знижує загальну ефективність

аналітичних платформ та ускладнює прийняття обґрунтованих рішень.

У зв’язку з цим зростає попит на автоматизовані методи збору даних, що базуються на сучасних технологіях, зокрема генеративному штучному інтелекті. Використання алгоритмів, здатних адаптуватися до змін у вебсередовищі, дозволяє не лише прискорити процес збору даних, але й забезпечити високу точність та релевантність отриманої інформації. Автоматизація зменшує залежність від людського фактору, мінімізуючи ризик помилок, а також дозволяє масштабувати процес збору даних відповідно до зростаючих потреб бізнесу, наукової спільноти та аналітичних систем.

Отже, актуальність теми збору даних із вебресурсів зумовлена як технічними викликами, так і необхідністю вдосконалення існуючих методів обробки інформації. Нові підходи, що інтегрують автоматизацію [3] та сучасні технології штучного інтелекту, відкривають перспективи для створення систем, здатних ефективно вирішувати завдання збору, аналізу та інтеграції даних, що є ключовим фактором у забезпеченні конкурентоспроможності та інноваційності в сучасному інформаційному середовищі.

1.1.2 Основні виклики традиційного вебскрапінгу

Традиційні методи вебскрапінгу стикаються з низкою значних викликів, що обумовлені як динамічністю сучасних вебсайтів, так і різноманітністю форматів подання інформації. Сучасні вебресурси характеризуються високою гнучкістю у структурі контенту, що призводить до численних труднощів при автоматизованому зборі даних. У цьому контексті важливо розглянути ключові фактори, що ускладнюють процес збору даних за допомогою традиційних підходів, зокрема змінну структуру сайтів, розмаїття форматів даних та наявність захисних механізмів.

Перш за все, динамічна природа сучасних вебсторінок спричиняє

серйозні труднощі при розробці ефективних скраперів. Багато сайтів використовують адаптивний дизайн, що дозволяє їм змінювати макет та розташування елементів у залежності від пристроїв користувача чи поточного трафіку. Така змінність призводить до того, що наперед заздалегідь запрограмовані правила вилучення даних, побудовані на певній структурі DOM, швидко втрачають свою ефективність після невеликих змін на сайті. Навіть незначні корекції у верстці або додавання нових блоків інформації можуть спричинити помилки у роботі скрапера, що вимагає постійного ручного налаштування та підтримки.

Додатковою проблемою є різноманітність форматів подання інформації. Вебсайти можуть використовувати різні технології для відображення даних: HTML, AJAX, JavaScript, а також різні формати, такі як JSON чи XML, що вимагають специфічного підходу до вилучення інформації. Традиційні методи, засновані на аналізі статичного HTML-коду, часто не можуть впоратися з динамічними змінами, спричиненими використанням клієнтських скриптів або асинхронних запитів до серверів. Це обмежує можливості стандартних скраперів, оскільки для обробки таких сценаріїв необхідно розробляти додаткові алгоритми, що здатні враховувати змінність даних у реальному часі.

Наявність захисних механізмів на вебсайтах також стає серйозною перешкодою для традиційного вебскрапінгу. Багато сайтів впроваджують різноманітні засоби захисту, такі як CAPTCHA, блокування IP-адрес або використання технік запобігання ботам. Ці методи спрямовані на те, щоб ускладнити автоматизований доступ до ресурсів та захистити інформацію від несанкціонованого вилучення. У результаті, традиційні скраперські рішення часто змушені обходити подібні бар’єри, що значно ускладнює процес збору даних і може призводити до частих збоїв у роботі системи.

Крім того, ручне налаштування скраперів потребує значних людських ресурсів. Розробка та підтримка окремих рішень для кожного сайту займає багато часу, що не відповідає сучасним вимогам швидкого реагування на

змінні ринкові умови. Висока залежність від людського фактора підвищує ризик виникнення помилок, а також знижує ефективність систем, особливо коли обсяг даних продовжує зростати. Постійне втручання спеціалістів у процес підтримки скраперів стає дороговартісним та важким у масштабуванні, оскільки навіть незначні зміни на вебсайті можуть призвести до необхідності повної перебудови логіки вилучення даних.

Зважаючи на все вищезазначене, традиційний вебскрапінг стає все менш ефективним у контексті сучасного інформаційного простору. Для вирішення цих проблем необхідно впроваджувати новітні технології, що дозволяють автоматизувати процеси адаптації до змін структури вебсторінок, оптимізувати роботу з різними форматами даних та враховувати захисні механізми сайтів. Використання методів штучного інтелекту, зокрема генеративних мовних моделей, відкриває нові перспективи для розробки адаптивних скраперів, які здатні автоматично генерувати правила вилучення даних та оперативно реагувати на зміни у вебсередовищі.

Таким чином, основні виклики традиційного вебскрапінгу, пов’язані із змінною структурою сайтів, різноманітністю форматів подання інформації та впровадженням захисних механізмів, свідчать про необхідність переходу до більш інтелектуальних та автоматизованих підходів у зборі даних. Автоматизація цього процесу дозволяє не лише знизити витрати людських ресурсів, але й забезпечити високу точність, масштабованість та стабільність систем, що є ключовими умовами успішного функціонування аналітичних платформ у сучасних умовах стрімкого зростання інформаційних потоків.

1.1.3 Необхідність впровадження нових підходів

Сучасний розвиток технологій та експоненціальне зростання обсягів інформації, що генерується в Інтернеті, ставлять перед дослідниками та практиками задачу пошуку ефективних методів вилучення даних із вебресурсів. Традиційні підходи, які базуються на ручному налаштуванні

скраперів та фіксованих правилах вилучення, вже не відповідають вимогам часу, адже вони не можуть оперативно адаптуватися до змін у структурі сайтів, забезпечувати необхідну точність та масштабованість процесу. Саме тому виникає нагальна потреба у впровадженні нових, більш гнучких та інтелектуальних методів збору даних.

Однією з ключових проблем є статичність традиційних скраперів [4]. Вони побудовані на заздалегідь визначених алгоритмах вилучення інформації, що робить їх вразливими до будь-яких змін у верстці вебсторінок. Навіть незначні коригування на сайті можуть призвести до повного виходу з ладу системи збору даних, що вимагає постійного ручного втручання та корекції. Такий підхід є не лише трудомістким, але й економічно невигідним при масштабному застосуванні. Таким чином, традиційні методи демонструють свою обмеженість і не відповідають вимогам сучасного інформаційного середовища, де необхідно оперативно реагувати на зміни та адаптуватися до нових форматів даних.

У зв’язку з цим зростає попит на автоматизовані рішення [3], що базуються на використанні сучасних технологій штучного інтелекту. Інтелектуальні системи збору даних повинні мати можливість самостійно аналізувати структуру вебресурсів, виявляти закономірності та автоматично генерувати правила вилучення інформації, що забезпечує їхню адаптивність до змін. Завдяки таким рішенням можна значно знизити залежність від ручного налаштування та мінімізувати ризик виникнення помилок, що виникають у традиційних підходах.

Одним із перспективних напрямків у цій галузі є застосування генеративного штучного інтелекту (LLM – Large Language Models) [5]. Сучасні генеративні моделі, такі як GPT-4 та інші, здатні не лише генерувати текст, але й створювати програмний код та алгоритми для вилучення даних на основі аналізу заданих зразків. Використання LLM дозволяє автоматизувати процес створення XPath-запитів, що є основою вилучення інформації з HTML-документів. Такий підхід забезпечує гнучкість, адже

моделі можуть адаптуватися до нових типів контенту, аналізуючи його структуру в режимі реального часу та оптимізуючи правила вилучення без постійного людського втручання.

Інтеграція генеративного ШІ відкриває нові можливості для побудови систем збору даних, що поєднують ефективність автоматизації з високою адаптивністю. Це дозволяє не лише підвищити точність вилучення даних, але й забезпечити масштабованість систем, що є критично важливим при роботі з великими обсягами інформації. Автоматизовані рішення на основі LLM можуть здійснювати самоаналіз та корекцію своїх алгоритмів, що сприяє постійному вдосконаленню процесу збору даних та зниженню витрат на підтримку систем.

Отже, необхідність впровадження нових підходів обумовлена критичними недоліками традиційних методів вебскрапінгу. Застосування інтелектуальних технологій, зокрема генеративного штучного інтелекту, є перспективним шляхом для подолання існуючих викликів. Інтеграція LLM у систему збору даних дозволяє автоматизувати процес адаптації до змін у структурі вебресурсів, забезпечити високу точність вилучення інформації та знизити витрати на ручну підтримку. Це відкриває широкі перспективи для розвитку сучасних аналітичних платформ та сприяє впровадженню інноваційних рішень у сфері обробки вебданих.

1.2 Аналіз методів та інструментів вебскрапінгу

1.2.1 Історія та еволюція вебскрапінгу

Історія вебскрапінгу починається ще з перших днів появи вебтехнологій, коли Інтернет ставав основним джерелом інформації для широкого загалу. На початкових етапах розвитку Інтернету основним форматом представлення інформації була статична HTML-розмітка, що відкривало можливість для дослідників та програмістів створювати прості

скрипти для вилучення даних. Перші спроби отримання даних з HTML-сторінок орієнтувались на використанні базових засобів програмування, таких як Perl, PHP або Python, що дозволяли за допомогою регулярних виразів та простих парсерів вилучати інформацію зі статичних вебресурсів. Такі підходи, хоч і були досить примітивними, проте стали основою для подальшого розвитку технологій збору даних.

У ті часи вебскрапінг часто здійснювався шляхом написання спеціалізованих скриптів, які орієнтувалися на конкретні сторінки чи сайти. Програмісти розробляли власні алгоритми для аналізу HTML-коду, знаходження потрібних тегів та вилучення текстових даних. Цей процес вимагав значних зусиль та глибокого розуміння структури вебсторінок, адже кожен сайт мав свою унікальну розмітку. Такі підходи швидко демонстрували свою ефективність для невеликих обсягів інформації, але з ростом популярності Інтернету та збільшенням кількості ресурсів вони стали неефективними для масштабних задач.

Згодом з розвитком мов програмування та появою спеціалізованих бібліотек, з’явилися більш структуровані підходи до вирішення задач вебскрапінгу. Одним із важливих етапів еволюції стало впровадження бібліотек, що спрощували роботу з HTML-структурами. Наприклад, бібліотека BeautifulSoup для мови програмування Python дозволила значно полегшити процес парсингу HTML-документів. Використовуючи простий та інтуїтивно зрозумілий інтерфейс, вона дозволяла розробникам швидко створювати скрипти для вилучення інформації, обробляючи DOM-структуру сторінок. Такий підхід дав змогу не лише автоматизувати обробку даних, але й зробити її більш універсальною, оскільки BeautifulSoup могла працювати з різними версіями HTML та навіть коректно обробляти неформатований код.

Поступово розвиток вебскрапінгу супроводжувався появою більш потужних та масштабованих рішень. Одним із знакових етапів став розвиток фреймворків, таких як Scrapy. Цей фреймворк був створений для вирішення задач, пов’язаних із масовим збором даних з різноманітних вебресурсів.

Scrapy забезпечує не лише зручне створення парсерів, але й підтримку асинхронної обробки запитів, що дозволяє ефективно працювати з великими обсягами інформації. Завдяки модульній структурі та можливості інтеграції з іншими технологіями, Scrapy швидко завоював популярність серед розробників, що займалися вебскрапінгом для аналітичних, маркетингових та наукових задач.

Не менш важливим етапом у розвитку вебскрапінгу стало впровадження технологій для роботи з динамічним контентом. Сучасні вебсайти часто використовують JavaScript для динамічного завантаження даних, що ускладнює задачу традиційного парсингу статичного HTML. Саме тому з’явилися інструменти, такі як Selenium, які дозволяють автоматизувати роботу з браузерами та взаємодіяти з вебсторінками як справжній користувач. Використання Selenium відкриває можливості для обходу захисних механізмів, симуляції поведінки користувача та отримання контенту, який завантажується динамічно. Такий підхід дозволяє вирішити проблему доступу до інформації, що розміщується за допомогою JavaScript, і є надзвичайно важливим для збору даних з сучасних вебресурсів.

Поряд із зростанням функціональності та масштабованості сучасних фреймворків, еволюція вебскрапінгу [6] відзначається також переходом від одноразових скриптів до комплексних систем, що включають у себе елементи автоматичного тестування, моніторингу та обробки помилок. Сучасні рішення дозволяють інтегрувати вебскрапінг із іншими компонентами аналітичних платформ, забезпечуючи ефективний обмін даними між різними системами. Це відкриває нові перспективи для використання зібраної інформації у великих аналітичних проектах, дослідженнях ринку та розробці бізнес-стратегій.

Історична еволюція вебскрапінгу свідчить про поступовий перехід від простих ручних методів до складних автоматизованих систем. Початкові підходи, засновані на використанні регулярних виразів та простих скриптів, дали поштовх до розвитку спеціалізованих бібліотек та фреймворків, таких

як BeautifulSoup, Scrapy та Selenium. Ці інструменти не лише значно спростили процес вилучення даних, але й відкрили шлях до створення інтегрованих систем, здатних працювати з великими обсягами інформації та динамічним контентом.

Таким чином, історія вебскрапінгу є прикладом того, як технологічний розвиток стимулював появу нових методів та інструментів для збору даних з вебресурсів. Від початкових спроб отримання даних з HTML-сторінок до сучасних платформ, що забезпечують комплексну автоматизацію процесів збору, аналізу та обробки інформації, цей процес демонструє постійний рух до підвищення ефективності, гнучкості та адаптивності інструментів збору даних. В результаті, сучасні технології вебскрапінгу стають незамінним інструментом для бізнесу, науки та державного управління, сприяючи прийняттю більш обґрунтованих рішень та розробці інноваційних проектів на основі великих обсягів вебданих.

1.2.2 Основні бібліотеки та фреймворки

У сучасному процесі вебскрапінгу значну роль відіграють спеціалізовані бібліотеки та фреймворки, що дозволяють автоматизувати вилучення даних із вебресурсів з високою точністю та ефективністю. Серед найпопулярніших інструментів виділяють Scrapy [7], BeautifulSoup [8] та Selenium [9]. Кожен з цих інструментів має свої переваги, специфіку застосування та сфери оптимального використання, що забезпечує їх адаптивність до різноманітних задач з вилучення даних.

Перш за все, Scrapy є ефективним фреймворком для збирання вебданих, написаним мовою Python. Його основна перевага полягає у здатності виконувати масштабоване та високошвидкісне опрацювання вебресурсів завдяки асинхронному виконанню запитів. Scrapy створено з урахуванням потреб великих проєктів, де необхідно обробляти значні обсяги вебсторінок у короткі терміни. Завдяки модульній структурі, розробники

можуть легко розширювати функціональність системи, додаючи модулі для обробки помилок, збереження даних у базах або реалізації систем моніторингу. Серед типових сценаріїв використання Scrapy – масовий збір даних, аналіз великих обсягів інформації для маркетингових досліджень, аналітичних платформ та моніторингу контенту. Важливо зазначити, що Scrapy дає змогу не лише отримувати дані, а й здійснювати їх попередню обробку за допомогою спеціального ланцюга обробки (пайплайну), що робить його універсальним інструментом для комплексних задач.

Другим значущим інструментом є BeautifulSoup – бібліотека для парсингу HTML та XML-документів, яка також реалізована мовою Python. BeautifulSoup відома своєю простотою у використанні та інтуїтивно зрозумілим інтерфейсом, що дозволяє розробникам швидко знаходити та вилучати необхідні елементи з DOM-структури вебсторінок. Завдяки своїй гнучкості, BeautifulSoup є надзвичайно корисною у випадках, коли потрібно працювати з невеликою кількістю сторінок або коли структура даних не є надто складною. Бібліотека дозволяє ефективно обробляти навіть некоректно сформатований HTML, що часто зустрічається у старих або менш професійних вебресурсах. До основних сфер застосування BeautifulSoup належить робота з проєктами, де пріоритетом є швидкість розробки та легкість налаштування, а також в тих випадках, коли не потрібна масштабована обробка даних у реальному часі.

Ще одним важливим інструментом у арсеналі вебскрапінгу є Selenium – фреймворк для автоматизації дій у браузері, який дозволяє симулювати поведінку реального користувача. На відміну від Scrapy та BeautifulSoup, які працюють переважно з статичним HTML-кодом, Selenium здатний обробляти динамічний контент, який завантажується за допомогою JavaScript. Це робить його незамінним для роботи із сучасними вебсайтами, де значна частина інформації генерується в режимі реального часу. Selenium дозволяє виконувати кліки, вводити дані в форми, прокручувати сторінки та виконувати інші дії, що імітують поведінку людини, що відкриває широкі

можливості для обходу захисних механізмів сайтів. Завдяки цьому, Selenium використовується у проєктах, де необхідно отримати доступ до даних, що динамічно змінюються, наприклад, у випадку соціальних мереж, новинних порталів або онлайн-магазинів. Водночас, через високу ресурсомісткість, Selenium зазвичай застосовується для вузькоспеціалізованих задач або в тих випадках, коли інші інструменти не забезпечують необхідного рівня доступу до контенту.

Крім зазначених основних інструментів, існують також інші бібліотеки та фреймворки, які можуть доповнювати чи розширювати функціональність традиційних рішень. Наприклад, бібліотеки requests та lxml часто використовуються разом з BeautifulSoup для оптимізації запитів до вебсайтів та більш ефективного парсингу XML/HTML-документів. Requests забезпечує простий та зручний інтерфейс для виконання HTTP-запитів, що дозволяє отримувати вміст сторінок, а lxml – потужний інструмент для роботи з XML та HTML, який може обробляти великі обсяги даних з високою швидкістю. Комбінація цих бібліотек дозволяє створювати легкі та водночас ефективні рішення для вилучення даних, що добре підходять для менш складних проєктів.

У підсумку, вибір конкретного інструменту або фреймворку для вебскрапінгу залежить від характеру задачі, вимог до масштабованості та специфіки обробки даних. Scrapy є оптимальним рішенням для великих проєктів з масовим збором даних, де пріоритетом є швидкість та масштабованість. BeautifulSoup, зі свого боку, забезпечує швидку та зручну обробку HTML-коду, що робить його ідеальним для невеликих або середніх за обсягом проєктів, де важлива гнучкість та простота налаштування. Selenium залишається ключовим інструментом для роботи з динамічним контентом, коли необхідно взаємодіяти із сайтом подібно до реального користувача, що дозволяє ефективно отримувати дані, що завантажуються асинхронно.

Таким чином, сучасна екосистема інструментів для збирання вебданих

вирізняється широким вибором рішень, кожен з яких має свої переваги та сфери застосування. Інтеграція цих технологій дозволяє розробникам створювати гнучкі, адаптивні та високопродуктивні системи для збору даних, що є важливим фактором у прийнятті обґрунтованих рішень у бізнесі, науці та державному управлінні. Розвиток таких технологій стимулює подальше вдосконалення методів вилучення інформації, сприяючи автоматизації процесів та зниженню витрат людських ресурсів. В майбутньому, з огляду на швидкі темпи розвитку технологій, можна очікувати ще більшу інтеграцію інтелектуальних систем з вебскрапінгом, що відкриє нові перспективи для аналізу даних та розвитку аналітичних платформ.

1.2.3 Обмеження та недоліки стандартних підходів

Попри значні переваги сучасних бібліотек та фреймворків для вебскрапінгу, таких як Scrapy, BeautifulSoup та Selenium, ці інструменти мають певні обмеження, що впливають на їхню ефективність у реальних умовах. Основною проблемою є необхідність постійного ручного налаштування та доопрацювання скраперів, особливо в умовах швидких змін у структурі вебсторінок.

Перш за все, традиційні підходи до вилучення даних базуються на заздалегідь визначених правилах для аналізу DOM-структури сторінок. Проте вебсайти постійно оновлюють свій дизайн, змінюють розташування елементів, додають нові блоки або модифікують існуючі. Навіть невеликі корективи у верстці можуть призвести до того, що раніше створені XPath-запити або CSS-селектори перестають коректно визначати потрібні елементи. Це вимагає постійного перегляду та оновлення логіки вилучення, що збільшує час на підтримку системи та підвищує ризик виникнення помилок.

Ще одна суттєва проблема – це залежність від експертизи розробників. Для того, щоб налаштувати скрапер для конкретного сайту, необхідно мати глибокі знання структури HTML та принципів роботи вебсторінок.

Розробнику доводиться не лише писати код, але й аналізувати складну та часто хаотичну структуру DOM, що може значно ускладнювати процес розробки. Така залежність від людського фактору створює вузьке місце в системі: навіть невеликі помилки у визначенні селекторів можуть призвести до значного зниження якості зібраних даних або до повного виходу скрапера з ладу.

Крім того, стандартні інструменти зазвичай розраховані на роботу з певними типами даних та не завжди здатні ефективно адаптуватися до нових форматів або технологій, що використовуються на сучасних вебресурсах. Наприклад, динамічні сторінки, де контент завантажується за допомогою AJAX-запитів або WebSocket, можуть бути непередбачуваними для традиційних бібліотек. У таких випадках використовуються більш ресурсоємні рішення, як-от Selenium, що потребує емуляції роботи браузера, що, у свою чергу, призводить до зростання витрат на обчислювальні ресурси та збільшення часу обробки запитів.

Ще одним недоліком є обмежена здатність стандартних підходів до обробки помилок. У разі виникнення непередбачених змін у структурі сайту або при скиданнях з’єднання з сервером, система може не встигати коректно відреагувати, що призводить до збоїв у зборі даних. Навіть якщо впроваджені механізми моніторингу та логування, усунення проблем зазвичай потребує втручання розробника для аналізу та виправлення помилок, що знову ж таки збільшує експлуатаційні витрати та час простою системи.

У підсумку, хоча сучасні бібліотеки та фреймворки для вебскрапінгу надають потужні можливості для автоматизації збору даних, їхня ефективність обмежується необхідністю ручного налаштування, залежністю від високої кваліфікації розробників та труднощами при адаптації до швидкозмінних умов вебсередовища. Це створює додаткове навантаження як на технічну команду, так і на фінансові ресурси підприємства, що застосовує ці технології. Саме тому виникає потреба у розробці більш гнучких та адаптивних систем, які б змогли автоматично реагувати на зміни у DOM-

структурі сторінок, мінімізувати ручне втручання та знизити витрати часу на підтримку. Сучасні дослідження зосереджені на інтеграції методів штучного інтелекту в процес вебскрапінгу, що відкриває можливість створення самонавчальних систем, здатних автоматично генерувати та коригувати правила вилучення даних залежно від поточного стану вебресурсу. Такі інтелектуальні системи зможуть зменшити залежність від людського фактору та забезпечити більш ефективну та стабільну роботу у довгостроковій перспективі.

Отже, незважаючи на успіхи стандартних рішень, їхні обмеження свідчать про необхідність впровадження нових підходів, що поєднують автоматизацію з високою адаптивністю до змін у вебсередовищі. Це є ключовим напрямком для подальшого розвитку технологій збору даних та створення інноваційних аналітичних платформ, здатних ефективно обробляти великі обсяги інформації в умовах постійних змін.

1.3 Концепція генеративних мовних моделей, їх роль у вебскрапінгу

1.3.1 Поняття генеративного штучного інтелекту

Генеративні мовні моделі (LLM) – це комплексні штучні нейронні мережі, розроблені для обробки, аналізу та генерації природної мови. Завдяки своїй високорівневій архітектурі та тренуванню на величезних обсягах текстових даних, LLM здатні не лише «розуміти» текст, а й генерувати нові фрагменти, що є логічним продовженням заданого контексту. Сучасні моделі, такі як GPT-3.5/4, Claude, LLaMA 2 та інші, демонструють високий рівень здатності створювати якісний текст, генерувати програмний код, складати інструкції або навіть писати книжки та літературні твори.

Основною особливістю LLM є їхня масштабованість та універсальність [10]. Вони побудовані на базі трансформерної архітектури,

яка дозволяє ефективно опрацьовувати послідовності даних, зокрема текст, завдяки механізму уваги (attention mechanism). Цей підхід дозволяє моделі фокусуватися на ключових словах та фразах у контексті, що є критично важливим для розуміння сенсу та побудови адекватних відповідей. Завдяки цьому LLM можуть виконувати різноманітні задачі: від перекладу текстів до створення нових статей, від генерації коду до розробки інструкцій з вирішення конкретних задач.

На практиці генеративний штучний інтелект застосовується у багатьох сферах. Наприклад, у сфері вебскрапінгу LLM можуть автоматично генерувати XPath-запити, аналізувати структуру вебсторінок та навіть самостійно коригувати правила вилучення даних, що значно знижує потребу у постійному ручному втручанні. Крім того, вони здатні оптимізувати процес вилучення інформації, адаптуючись до змін у DOM-структурі вебсайтів, що є суттєвим плюсом у порівнянні з традиційними методами. Такий підхід забезпечує більш точне вилучення даних та сприяє зниженню операційних витрат.

Одним із прикладів популярної генеративної моделі є GPT-3.5/4 від компанії OpenAI. Ця модель здобула визнання завдяки своїй здатності генерувати зв’язний, інформативний та креативний текст. GPT-3.5/4 демонструє високу точність у різних задачах – від відповіді на запитання до розробки програмного коду. Завдяки своїм масштабним параметрам (десятки мільярдів параметрів) модель має здатність аналізувати складні мовні конструкції, враховувати контекст і навіть виявляти емоційне забарвлення тексту.

Ще одним вагомим представником генеративних мовних моделей є Claude, розроблений компанією Anthropic. Ця модель орієнтована на дотримання принципів безпечної та етичної генерації тексту, з акцентом на уникнення небажаного контенту та відповідність високим стандартам якості. Claude здатний адаптуватися до різних контекстів, генеруючи не лише текст, а й структуровані дані, що може бути особливо корисним при інтеграції з

іншими системами збору та аналізу інформації.

Модель LLaMA 2, розроблена компанією Meta, є ще одним прикладом сучасної генеративної моделі. Її розробка спрямована на створення високопродуктивного рішення з відкритим кодом, що дозволяє дослідникам та розробникам гнучко адаптувати модель під специфічні завдання. Завдяки оптимізованій архітектурі LLaMA 2 може працювати з меншими витратами обчислювальних ресурсів, що робить її привабливим вибором для інтеграції в різноманітні системи, зокрема і в проекти вебскрапінгу.

У концепції генеративного штучного інтелекту велике значення має також здатність моделей до навчання на основі зразків. LLM можуть адаптуватися до нових даних і вдосконалювати свої алгоритми через процес перенавчання, що дозволяє їм постійно покращувати результати генерації. Це особливо важливо для застосування в динамічному середовищі вебскрапінгу, де зміни у структурі даних є нормою. За допомогою перенавчання та корекції моделей можна значно підвищити точність вилучення інформації, знизити кількість помилок та забезпечити стабільність роботи систем.

Генеративні мовні моделі не обмежуються лише створенням тексту. Вони також здатні генерувати програмний код, що є критичним у контексті автоматизації процесів збору даних. Наприклад, за допомогою GPT-4 можна створити скрипти для вилучення даних з вебресурсів, що автоматично адаптуються до нових вимог або змін у структурі сайтів. Такий підхід дозволяє значно зменшити затрати часу на ручне налаштування систем та забезпечує більш високий рівень автономності.

Отже, генеративний штучний інтелект і, зокрема, LLM представляють собою потужний інструмент, який кардинально змінює підходи до обробки інформації. Їхня здатність до генерації тексту, коду та інструкцій відкриває широкі можливості для автоматизації складних процесів у різних галузях, включаючи вебскрапінг. Популярні моделі, такі як GPT-3.5/4, Claude та LLaMA 2, демонструють високий рівень адаптивності та точності, що робить їх незамінними у сучасних системах аналізу даних.

Ключовим моментом є те, що генеративні мовні моделі здатні самостійно адаптуватися до змін у зовнішньому середовищі, що дозволяє автоматизувати процеси, які раніше вимагали значного людського втручання. Це є суттєвим кроком вперед у створенні інтелектуальних систем, здатних ефективно вирішувати задачі збору та обробки даних, що відповідають сучасним вимогам. Генеративний ШІ не тільки підвищує ефективність роботи існуючих рішень, але й відкриває нові горизонти для інтеграції інтелектуальних систем в різні сфери діяльності, що сприяє розвитку інноваційних підходів до обробки інформації.

Таким чином, поняття генеративного штучного інтелекту охоплює не лише здатність моделей створювати текст, але й їхню універсальність у вирішенні широкого спектру задач, від генерації коду до аналізу складних структур даних. Завдяки своїм потужним можливостям LLM стають ключовим інструментом для розробки сучасних систем вебскрапінгу, де вони сприяють автоматизації та оптимізації процесів збору інформації, забезпечуючи високу точність та адаптивність до швидкоплинних умов сучасного інформаційного простору.

1.3.2 Переваги та можливості LLM у процесі збору даних

Генеративні мовні моделі (LLM) відкривають нові перспективи у сфері автоматизації збору даних завдяки своїй здатності аналізувати складну структуру HTML та самостійно виявляти ключові патерни, необхідні для вилучення інформації. Завдяки своїм глибоким нейронним мережам, моделі здатні «розуміти» структуру вебсторінок, що дозволяє їм генерувати XPath-запити або інші селектори без необхідності ручного прописування правил.

Однією з ключових переваг LLM є їхня здатність адаптуватися до різних типів вебресурсів. В умовах, коли традиційні методи збору даних вимагають постійного оновлення правил через зміни у DOM-структурі, генеративні моделі можуть швидко виявляти зміни та коригувати свої

алгоритми вилучення. Це дозволяє системі автоматично реагувати на змінну структуру сторінок, визначаючи, які елементи є ключовими для вилучення даних. Завдяки цьому зменшується залежність від людського фактору, що традиційно обумовлює високі витрати часу та ресурсів на підтримку систем.

Ще одна суттєва перевага полягає в гнучкості процесу генерування правил. LLM здатні працювати в режимі напівавтоматичного визначення шаблонів для вилучення даних. Наприклад, модель може аналізувати HTML-код сторінки, розпізнавати структуру таблиць, списків чи інших елементів, і на основі цього формувати XPath-запити, які найбільш точно виділяють необхідну інформацію. Такий підхід дозволяє не лише прискорити процес розробки скраперів, але й забезпечити їхню здатність ефективно обробляти дані у різноманітних форматах.

Крім того, LLM можуть адаптувати свої алгоритми на основі зворотного зв’язку. Якщо, наприклад, згенерований XPath не забезпечує потрібного результату, система може самостійно проаналізувати помилки та скоригувати параметри запиту. Це створює умови для постійного вдосконалення процесу вилучення даних та забезпечує більшу стабільність роботи системи навіть при динамічний структурі сайту. Подібна адаптивність значно знижує витрати на ручне налаштування та коригування, що є важливим у великих проєктах з масовим збором даних.

Важливою характеристикою генеративних моделей є швидкість обробки інформації. Завдяки оптимізованим алгоритмам та паралельній обробці, LLM можуть миттєво аналізувати вхідні дані та генерувати правила вилучення практично в режимі реального часу. Це особливо актуально у випадках, коли необхідно оперативно реагувати на зміни на вебресурсах, наприклад, при моніторингу новинних порталів або соціальних мереж, де інформація оновлюється дуже часто.

Завдяки своїй універсальності, генеративні мовні моделі здатні працювати як з текстовими даними, так і з іншими форматами інформації, що містяться на вебсторінках. Вони можуть інтегруватися в більші системи

збору даних, де виконують роль «розумного» компонента, здатного попередньо аналізувати структуру сторінок та видавати рекомендації щодо оптимізації процесу вилучення. Таким чином, застосування LLM дозволяє не тільки автоматизувати процес збору даних, але й підвищити якість отриманої інформації, зменшивши кількість помилок і непотрібних даних.

У підсумку, переваги генеративних мовних моделей у вебскрапінгу полягають у високій адаптивності, гнучкості генерування правил та швидкості обробки інформації. Завдяки цим властивостям LLM є потужним інструментом для автоматизації збору даних, що дозволяє мінімізувати ручне втручання та оптимізувати процес вилучення даних у сучасних умовах швидкоплинного інформаційного середовища. Інтеграція таких моделей створює передумови для розробки нових, інтелектуальних систем збору даних, здатних ефективно працювати з різноманітними вебресурсами та забезпечувати високоточний аналіз інформації без необхідності постійної підтримки з боку розробників.

1.3.3 Огляд наукових та прикладних досліджень

У сучасній літературі з питань інтеграції штучного інтелекту з вебскрапінгом представлено численні приклади, що демонструють потенціал генеративних мовних моделей (LLM) [10] у підвищенні ефективності вилучення даних. Зокрема, ряд досліджень відзначає можливості LLM у автоматизації генерації XPath-запитів та адаптації до змін у DOM-структурі вебсайтів. Ці дослідження свідчать про успішну інтеграцію алгоритмів на базі GPT-4 для створення гнучких скраперів, що дозволяють знизити необхідність ручного налаштування правил вилучення інформації.

Крім того, прикладні проєкти демонструють, як використання LLM в поєднанні з подієво-орієнтованою архітектурою сприяє створенню більш адаптивних та масштабованих систем збору даних. У другому розділі пропонується модель, яка інтегрує можливості LLM для автоматичної

генерації правил вилучення даних із подієвою архітектурою, що забезпечує гнучку обробку вхідних даних та оперативну реакцію на зміни у вебсередовищі. Такий підхід дозволяє не лише автоматизувати процес збору інформації, але й підвищити якість даних за рахунок швидкого виявлення та корекції помилок.

Попри це, напрям інтеграції генеративного ШІ у вебскрапінгу досі залишається недостатньо дослідженим. Багато аспектів, пов’язаних із безпекою, етикою збору даних та оптимізацією алгоритмів автоматичного генерування правил, потребують додаткових наукових розробок. Сучасні дослідження свідчать, що застосування LLM може суттєво поліпшити процес вилучення даних, проте існують обмеження, пов’язані з високою складністю моделей та необхідністю їх постійного перенавчання під специфіку кожного окремого ресурсу. Таким чином, огляд наукових та прикладних досліджень демонструє перспективність використання генеративних мовних моделей у вебскрапінгу, а також вказує на необхідність подальших досліджень для оптимізації їх інтеграції з подієво-орієнтованими архітектурами. Цей напрям має значний потенціал для створення адаптивних систем збору даних, що здатні ефективно реагувати на зміни у структурі вебресурсів і забезпечувати високоякісний аналіз інформації.

1.4 Інструменти та середовище для реалізації автоматизованої системи

1.4.1 Використання Python та його бібліотек

Python здобув репутацію однієї з найоптимальніших мов програмування для реалізації систем вебскрапінгу та інтеграції технологій штучного інтелекту завдяки своїй простоті, гнучкості та величезній спільноті розробників. Основною перевагою Python є його зручність у написанні читабельного коду, що дозволяє швидко розробляти та модифікувати алгоритми, а також легко інтегрувати різні технології, зокрема інструменти для вебскрапінгу та генеративного ШІ.

Одним із ключових факторів, що обґрунтовує вибір Python, є величезна кількість доступних бібліотек, які охоплюють майже всі аспекти розробки програмного забезпечення. Для задач вебскрапінгу тут широко використовуються бібліотеки, такі як BeautifulSoup, Scrapy, та Selenium. BeautifulSoup забезпечує ефективний парсинг HTML та XML документів, дозволяючи швидко ідентифікувати та вилучати потрібні елементи з DOM-структури вебсторінок. Scrapy, як повнофункціональний фреймворк, дозволяє будувати масштабовані системи збору даних з підтримкою асинхронного виконання запитів, що особливо корисно при роботі з великими обсягами інформації. Selenium, зі свого боку, забезпечує автоматизацію взаємодії з браузерами, що є незамінним при роботі з динамічними сторінками, де контент завантажується за допомогою JavaScript.

Крім того, Python володіє потужними бібліотеками для роботи зі штучним інтелектом та машинним навчанням. Серед них – TensorFlow, PyTorch, scikit-learn та інші, які дозволяють розробляти моделі для аналізу даних, генерації тексту та інтеграції з LLM. Завдяки таким інструментам розробники можуть створювати системи, що поєднують автоматизований збір даних із подальшою аналітикою, прогнозуванням та прийняттям рішень на основі отриманої інформації. Важливо зазначити, що інтеграція LLM, таких як GPT-3.5/4, Claude чи LLaMA 2, в екосистему Python забезпечується за допомогою спеціалізованих пакетів, які спрощують роботу з API цих моделей. Це дозволяє безперешкодно впроваджувати передові технології штучного інтелекту у системи вебскрапінгу.

Ще одним важливим аспектом є активна спільнота користувачів Python, яка забезпечує постійну підтримку, оновлення бібліотек та доступ до численних навчальних ресурсів. Це дозволяє полегшити опанування необхідних інструментів для початківців, а досвідченим розробникам – ефективно вирішувати складні задачі. Велика кількість відкритих проєктів на GitHub та інших платформах також сприяє обміну досвідом та сприяє

швидкому впровадженню інноваційних рішень у сфері вебскрапінгу та ШІ.

У контексті інтеграції з LLM особливу увагу варто приділити використанню ключових пакетних менеджерів, таких як pip та conda, які забезпечують швидке встановлення, оновлення та управління залежностями проєктів. Conda, зокрема, дозволяє створювати ізольовані середовища, що гарантує сумісність різних версій бібліотек, що є важливим для стабільної роботи комплексних систем. Крім того, такі фреймворки, як Flask або Django, надають можливості для розробки вебінтерфейсів, що може бути корисним при створенні систем моніторингу або адміністрування процесів збору даних.

Завдяки своїй універсальності, Python стає ідеальним вибором для розробки комплексних систем, що об’єднують вебскрапінг із передовими технологіями штучного інтелекту. Використання Python дозволяє швидко прототипувати нові ідеї, інтегрувати різноманітні технології та забезпечувати високу гнучкість у розробці. Це дає можливість створювати адаптивні та масштабовані рішення, здатні ефективно працювати в умовах постійних змін у вебсередовищі.

Таким чином, поєднання широкого спектру бібліотек для вебскрапінгу, потужних інструментів для роботи з ШІ, активної спільноти розробників та зручних засобів управління пакетами робить Python оптимальним середовищем для розробки автоматизованих систем збору даних. Це забезпечує високий рівень адаптивності, ефективності та надійності, що є критично важливим для реалізації сучасних аналітичних рішень на базі генеративного штучного інтелекту.

1.4.2 Роль систем RabbitMQ, Prometheus, Grafana Loki

У сучасних автоматизованих системах збору даних критично важливим є забезпечення надійної та масштабованої взаємодії між різними компонентами. Для цього часто використовуються системи, що реалізують подієво-орієнтовану архітектуру, зокрема RabbitMQ, а також інструменти для

моніторингу та логування, як-от Prometheus та Grafana Loki.

RabbitMQ [11] є потужним брокером повідомлень, який дозволяє забезпечити асинхронну взаємодію між різними модулями системи. Основна його функція полягає у передачі повідомлень через черги, завдяки чому компоненти, відповідальні за вилучення, обробку та зберігання даних, можуть працювати незалежно один від одного. Це означає, що під час обробки великої кількості запитів або при виникненні тимчасових збоїв у певному компоненті, система не припиняє свою роботу, оскільки повідомлення зберігаються в черзі до моменту їхньої обробки. Асинхронні черги, реалізовані за допомогою RabbitMQ, дозволяють значно підвищити масштабованість системи, оскільки вони забезпечують рівномірний розподіл навантаження та дозволяють адаптувати обчислювальні ресурси до змін у потоці даних. Таким чином, RabbitMQ є ключовим елементом для побудови гнучкої та надійної системи збору даних, що здатна оперативно реагувати на високий обсяг інформації.

Разом із забезпеченням асинхронної обробки повідомлень важливу роль відіграють системи моніторингу та логування, які дозволяють відстежувати стан роботи всіх компонентів системи в режимі реального часу. Prometheus – це потужна система моніторингу з відкритим вихідним кодом, яка спеціалізується на зборі та збереженні метрик з різних джерел. За допомогою Prometheus [12] можна збирати дані про продуктивність системи, такі як завантаження процесора, використання пам’яті, час відгуку запитів та інші критичні параметри. Застосування цієї системи дозволяє оперативно виявляти аномалії у роботі вебскрапінгових модулів, що може сигналізувати про виникнення проблем, наприклад, при зміні DOM-структури або у випадку збою в мережній взаємодії. Завдяки високій гнучкості налаштувань Prometheus, адміністратори можуть створювати власні правила оповіщення, що дозволяє швидко реагувати на непередбачені ситуації та забезпечувати безперебійну роботу системи.

Важливим інструментом є Grafana Loki [13], який спеціалізується на

централізованому збиранні та аналізі логів. Він інтегрується з Prometheus та Grafana, що дозволяє створити єдину платформу для відстеження як метрик, так і логів у системі. Завдяки Loki, можна швидко знаходити помилки та інциденти, аналізувати лог-файли з усіх компонентів системи, що використовують RabbitMQ для обміну повідомленнями. Це полегшує процес налагодження та оптимізації системи, адже завдяки централізованому логуванню адміністраторам не потрібно шукати інформацію в різних джерелах. Відображення логів у зручних візуальних панелях Grafana дозволяє швидко інтерпретувати дані та приймати оперативні рішення щодо вирішення технічних проблем.

Ключовою перевагою використання RabbitMQ, Prometheus та Grafana Loki є їхня взаємодія, яка забезпечує повну видимість стану системи. Інтеграція цих інструментів дозволяє не лише контролювати потік повідомлень та відстежувати продуктивність, але й оперативно реагувати на будь-які збої у роботі компонентів. Наприклад, у випадку перевантаження однієї з черг RabbitMQ, Prometheus може відразу сповістити адміністратора через налаштовані оповіщення, а Grafana Loki допоможе аналізувати лог-файли для виявлення причин виникнення проблеми. Це забезпечує високу надійність та ефективність роботи всієї системи збору даних, що є критично важливим у умовах високої динаміки інформаційних потоків.

Отже, використання RabbitMQ для реалізації подієво-орієнтованої взаємодії, у поєднанні з системами моніторингу Prometheus та централізованого логування Grafana Loki, створює потужну основу для побудови адаптивних та масштабованих систем збору даних. Таке комплексне рішення забезпечує не лише стабільність роботи системи при високих навантаженнях, але й дозволяє оперативно виявляти та вирішувати проблеми, мінімізуючи втрати часу та ресурсів. Це є особливо важливим для сучасних проєктів, де швидкість обробки інформації та можливість адаптації до змін у вебсередовищі визначають конкурентоспроможність та ефективність бізнес-процесів.

1.4.3 Обґрунтування вибору бази даних

У сучасних системах збору та обробки даних важливим етапом є вибір бази даних, що забезпечує ефективне зберігання напівструктурованого або динамічного контенту. Традиційні реляційні системи управління базами даних (СУБД) спроєктовані для роботи зі структурованими даними, де всі записи мають чітко визначену схему. Проте, у випадку з вебскрапінгом, дані, як правило, мають напівструктурований характер, що змінюється в залежності від контенту конкретного сайту. Це створює низку проблем при використанні реляційних баз даних.

По-перше, реляційні СУБД вимагають попереднього визначення схеми таблиць, що може бути складною задачею для даних, які не мають фіксованої структури. Кожна зміна у форматі або структурі отриманих даних часто вимагає модифікації схеми, що супроводжується значними витратами часу та ресурсів. Такий підхід стає неефективним у разі, коли необхідно швидко адаптуватися до змін у вебресурсах, де структура може змінюватися навіть на рівні окремих елементів. Крім того, у реляційних базах даних зберігання напівструктурованого контенту часто вимагає використання додаткових таблиць або полів, що ускладнює процес запиту та аналізу даних.

У цьому контексті бази даних типу NoSQL, зокрема MongoDB, набувають особливої популярності. MongoDB [14] є документно-орієнтованою базою даних, яка зберігає інформацію у вигляді документів у форматі BSON (бінарний JSON). Завдяки цьому підхід до зберігання даних є набагато більш гнучким, оскільки документи можуть містити різні поля та структури без необхідності попереднього визначення схеми. Це значно спрощує адаптацію системи до різноманітних форматів даних, що надходять із вебресурсів, і дозволяє ефективно зберігати як структуровані, так і напівструктуровані дані.

Однією з основних переваг MongoDB є масштабованість. Система підтримує горизонтальне масштабування за допомогою шардінгу, що

дозволяє розподіляти дані між кількома серверами. Це особливо важливо для великих проєктів, де обсяг даних може швидко зростати, а потреба у високій продуктивності системи стає критичною. Завдяки можливості горизонтального масштабування MongoDB здатна обробляти великі потоки даних, що надходять у режимі реального часу, забезпечуючи при цьому стабільну продуктивність.

Крім того, MongoDB надає зручні засоби для роботи з даними через гнучкий інтерфейс формування запитів, який підтримує як прості, так і складні агрегаційні операції. Це дозволяє здійснювати аналіз даних без необхідності додаткових ETL-процесів для приведення інформації до єдиного формату. Вбудовані можливості для реплікації та автоматичного резервного копіювання сприяють підвищенню надійності системи та забезпечують безперебійну роботу у разі відмов окремих вузлів.

Отже, вибір бази даних є ключовим фактором у розробці автоматизованих систем збору даних. Реляційні СУБД, незважаючи на свою ефективність при обробці чітко структурованих даних, не є оптимальними для зберігання напівструктурованої інформації, яка характерна для вебскрапінгу. У цьому контексті MongoDB надає більшу гнучкість, масштабованість та зручність у роботі, що дозволяє адаптувати систему до постійних змін у структурі даних та забезпечувати ефективне зберігання та аналіз великого обсягу інформації. Таким чином, застосування NoSQL баз даних, зокрема MongoDB, є оптимальним рішенням для сучасних систем збору та обробки динамічного контенту.

1.5 Аналіз вимог до моделі автоматизованого збору даних

1.5.1 Функціональні вимоги

Система автоматизованого збору даних повинна задовольняти низку функціональних вимог, які охоплюють весь процес: від виявлення релевантних вебсторінок до автоматичної генерації XPath-запитів,

збереження вилученої інформації та формування детальних звітів. Основна мета – забезпечити ефективну, точну та адаптивну обробку великих обсягів даних, що надходять із вебресурсів, а також оперативну реакцію на зміни у структурі сайтів.

По-перше, система має здійснювати автоматичне виявлення вебсторінок, які містять потрібну інформацію. Це включає використання алгоритмів для обходу сайтів, аналізу їх структури та визначення релевантних сторінок за допомогою попередньо визначених критеріїв. Важливо, щоб система могла працювати у масштабованому режимі, охоплюючи численні джерела даних без втрати продуктивності.

По-друге, одним із ключових аспектів є автоматична генерація XPath-запитів. Система повинна аналізувати DOM-структуру вебсторінок і на основі визначених патернів створювати правила вилучення даних, що дозволяють точно визначати розташування елементів. Цей процес має бути максимально автоматизованим, щоб мінімізувати необхідність ручного налаштування та корекції у випадку змін у верстці сайтів. Автоматична адаптація до нових структур забезпечує підвищення точності вилучення інформації.

Далі, система має забезпечувати ефективне зберігання отриманих даних у відповідній базі даних. Беручи до уваги специфіку напівструктурованих даних, необхідно використовувати рішення, що дозволяють зберігати різноманітні формати даних (наприклад, NoSQL [15] бази даних, як-от MongoDB). Такий підхід гарантує гнучкість при зміні структури даних та полегшує їх подальшу обробку та аналіз.

Формування звітів є наступним важливим етапом функціональних вимог. Система повинна не лише зберігати дані, але й надавати можливість їх аналітичної обробки, генерувати візуалізації, таблиці та діаграми для подальшого прийняття рішень. Функціонал формування звітів має бути інтегрованим із системами моніторингу, що дозволяє оперативно реагувати на зміни у даних та виявляти потенційні проблеми.

Щодо вимог до швидкості збору даних, система повинна забезпечувати високий рівень продуктивності при паралельній обробці численних запитів. Асинхронна обробка повідомлень та використання черг повідомлень (наприклад, через RabbitMQ) дозволяють розподілити навантаження між різними компонентами, що критично важливо для систем з високим обсягом інформації. Швидкість збору визначає здатність системи оперативно реагувати на нові дані та змінні умови роботи вебресурсів.

Точність вилучення даних – ще один ключовий аспект. Система має мінімізувати кількість помилок при визначенні XPath-запитів та вилученні інформації, забезпечуючи високу якість зібраних даних. Використання сучасних алгоритмів обробки тексту, генеративних мовних моделей та механізмів автоматичного коригування правил сприяє досягненню високої точності, що зменшує потребу у ручному втручанні.

Останнім, але не менш важливим, є забезпечення адаптивності системи до змін у вебресурсах. Зміни у DOM-структурі, впровадження нових технологій на сайтах або застосування захисних механізмів можуть суттєво впливати на процес збору даних. Тому система повинна бути спроможною в режимі реального часу виявляти такі зміни та коригувати свої алгоритми, гарантуючи безперервність роботи та мінімізацію втрат даних.

Отже, функціональні вимоги до моделі автоматизованого збору даних охоплюють весь спектр задач: виявлення вебсторінок, автоматичну генерацію XPath-запитів, ефективне зберігання даних та формування звітів, при цьому система має демонструвати високу швидкість, точність та адаптивність. Забезпечення таких вимог є критичним для створення сучасних аналітичних платформ, здатних працювати в умовах постійно змінюваного інформаційного середовища.

1.5.2 Нефункціональні вимоги

Нефункціональні вимоги до моделі автоматизованого збору даних визначають критерії, які повинні забезпечити надійність, масштабованість,

адаптивність та зручність підтримки системи. Крім того, слід враховувати питання безпеки даних та дотримання етичних норм, що дозволяє уникнути порушення правил використання вебресурсів та законів про захист даних.

Надійність системи є першочерговою задачею, оскільки від неї залежить безперервність збору даних та їх точність. Модель має бути спроможною працювати у цілодобовому режимі, забезпечуючи стабільну обробку запитів навіть при виникненні непередбачуваних збоїв. Для цього використовуються механізми автоматичного відновлення роботи, реплікації даних та резервного копіювання, що дозволяють мінімізувати час простою системи.

Масштабованість – ключова вимога для систем, які повинні обробляти великі обсяги даних. Система має бути здатною до горизонтального масштабування, що дозволяє додавати нові вузли без значного переривання роботи. Такий підхід забезпечує можливість адаптації до зростаючих потоків даних і знижує ризик перевантаження окремих компонентів. Використання асинхронних черг повідомлень та розподілених обчислювальних ресурсів сприяє більш ефективному управлінню навантаженням.

Адаптивність системи передбачає здатність оперативно реагувати на зміни у структурі вебресурсів. Система має автоматично виявляти відхилення у форматі даних або зміну DOM-структури та коригувати алгоритми вилучення, що мінімізує потребу в ручному втручанні. Завдяки цьому досягається висока точність збору даних навіть у динамічному інформаційному середовищі.

Зручність підтримки є ще одним важливим аспектом, оскільки вона впливає на швидкість виправлення помилок та адаптацію системи до нових вимог. Документована архітектура, модульність рішень та використання відкритих стандартів дозволяють знизити витрати на експлуатацію системи та спростити процес її модернізації.

Щодо захищеності, система повинна гарантувати конфіденційність та цілісність даних, не допускаючи несанкціонованого доступу. Вона має

відповідати вимогам законодавства про захист персональних даних та дотримуватися етичних норм у зборі інформації з вебресурсів. Впровадження шифрування, системи аутентифікації та аудиту дій забезпечує відповідність стандартам безпеки. Таким чином, нефункціональні вимоги до моделі автоматизованого збору даних охоплюють критично важливі аспекти: забезпечення надійності, масштабованості, адаптивності та зручності підтримки, а також гарантування високого рівня захищеності та етичної відповідності. Дотримання цих вимог є фундаментальним для створення ефективної та безпечної системи збору та обробки даних у сучасних умовах стрімкого розвитку інформаційного простору.

1.5.3 Взаємозв’язок вимог та обраних технологій

В сучасних системах автоматизованого збору даних ключовим аспектом є забезпечення інтеграції функціональних та нефункціональних вимог з технологічним стеком, що використовується для реалізації рішення. Генеративний штучний інтелект (Generative AI) відіграє тут вирішальну роль, оскільки його здатність автоматично генерувати правила вилучення даних значно спрощує процес збору інформації. Завдяки можливостям LLM (Large Language Models) система може аналізувати структуру вебсторінок, визначати ключові патерни в HTML-коді та створювати XPath-запити без необхідності постійного ручного втручання, що відповідає вимогам до високої точності та швидкості збору даних.

Подієво-орієнтована архітектура, яку реалізовано за допомогою RabbitMQ, дозволяє ефективно організувати обмін інформацією між різними компонентами системи. Використання асинхронних черг повідомлень забезпечує розподіл навантаження та дозволяє обробляти великі обсяги даних у режимі реального часу. Така архітектура не лише сприяє високій масштабованості, але й забезпечує адаптивність системи до змін у вебресурсах, оскільки кожен компонент може працювати незалежно,

реагуючи на повідомлення про події. Це означає, що при виникненні проблем чи змінах у структурі сторінок система може миттєво адаптуватися, мінімізуючи вплив на загальну продуктивність.

Використання Python та його бібліотек безпосередньо відповідає як функціональним, так і нефункціональним вимогам, оскільки ця мова програмування забезпечує простоту інтеграції, швидкість розробки та високу гнучкість. Бібліотеки, такі як Scrapy, BeautifulSoup та Selenium дозволяють створювати ефективні механізми вилучення даних, а потужні фреймворки для роботи з AI, такі як TensorFlow та PyTorch, забезпечують підтримку генеративних моделей. Застосування таких інструментів дозволяє системі автоматично генерувати правила вилучення, реагувати на зміни в DOM-структурі та швидко адаптуватися до нових типів контенту.

Крім того, використання пакетних менеджерів, таких як pip та conda, сприяє швидкому розгортанню та оновленню програмних компонентів, що є важливим для забезпечення надійності та зручності підтримки системи. Така екосистемна інтеграція дозволяє оперативно вирішувати проблеми, пов’язані з оновленням бібліотек або змін у вимогах до даних, що надходять із вебресурсів.

Застосування RabbitMQ в ролі брокера повідомлень забезпечує не тільки асинхронну обробку даних, але й реалізує подієво-орієнтовану взаємодію між компонентами. Це дозволяє системі масштабуватися горизонтально, додаючи нові вузли без зупинки роботи, а також ефективно управляти потоками даних. Автоматичне чергування повідомлень допомагає уникнути затримок у обробці запитів, що в умовах високої навантаженості є критично важливим.

Інтеграція систем моніторингу та логування, таких як Prometheus та Grafana Loki, сприяє постійному контролю за роботою системи, своєчасному виявленню збоїв і швидкому реагуванню на них. Цей аспект відповідає вимогам щодо надійності та безпеки, оскільки забезпечує прозорість операційних процесів та дозволяє адміністратору оперативно вживати

заходів для відновлення нормальної роботи.

Отже, взаємозв’язок вимог та обраних технологій у моделі автоматизованого збору даних реалізується шляхом інтеграції Generative AI, що забезпечує автоматизацію процесу вилучення даних, та подієво-орієнтованої архітектури, яка гарантує масштабованість та адаптивність системи. Використання Python та його екосистеми бібліотек дає змогу створити гнучке рішення, яке відповідає як функціональним, так і нефункціональним вимогам, забезпечуючи високий рівень продуктивності, точності та надійності. Цей комплексний підхід дозволяє не тільки оптимізувати процес збору даних, але й ефективно реагувати на зміни у вебсередовищі, що є критично важливим для сучасних аналітичних платформ.

2 МОДЕЛЬ АВТОМАТИЗАЦІЇ ЗБОРУ ДАНИХ З ВЕБРЕСУРСІВ НА ОСНОВІ ГЕНЕРАТИВНОГО ШТУЧНОГО ІНТЕЛЕКТУ

2.1 Аналіз останніх досліджень та публікацій

У межах дослідження автоматизованих методів збору та аналізу вебданих варто звернути увагу на низку наукових робіт та статей, що висвітлюють сучасні підходи до вебскрапінгу та обробки інформації. Зокрема, вивчаються інструменти на базі мови програмування Python, такі як Scrapy, BeautifulSoup та Selenium, які дозволяють ефективно отримувати, структурувати та аналізувати дані. Окрему увагу приділено ролі штучного інтелекту, включаючи NLP (Natural Language Processing) та машинному навчанню у вдосконаленні процесів обробки текстових та структурованих даних. Також розглядаються технічні аспекти розробки масштабованих архітектур для збору даних, включаючи інтеграцію алгоритмів розпізнавання інформації та оптимізацію витрат обчислювальних ресурсів, що є критично важливими для побудови ефективних рішень у динамічному інформаційному середовищі.

У роботі [16] досліджено сучасні підходи до розробки та впровадження ефективних методів вебскрапінгу для автоматизованого збору та обробки даних з використанням мови програмування Python. Автори акцентують увагу на застосуванні бібліотек BeautifulSoup, Selenium та Scrapy, що забезпечують високу швидкість та точність отримання інформації з різноманітних вебджерел, зокрема вторинних ринків. Особливий наголос зроблено на автоматизації процесу за допомогою створення скриптів та використання планувальників завдань, таких як cron, що дозволяє безперервно оновлювати бази даних без втручання людини. Окремо розглянуто методи обробки та очищення зібраних даних, включаючи фільтрацію непотрібної інформації, дублікатів та шумів, що сприяє підвищенню якості даних. Практичне застосування результатів дослідження

демонструє їхню цінність для аналізу ринку, оцінки попиту та прогнозування якості, підкреслюючи важливість запропонованих методів.

У роботі [17] досліджено застосування методу HTML DOM для вебскрапінгу з метою автоматизованого збору наукових статей із Google Scholar. Автори акцентують увагу на важливості створення інструментів для колективного аналізу наукових публікацій установ, оскільки Google Scholar не надає вбудованих функцій для такої агрегації. У дослідженні описано процес ідентифікації специфічних елементів DOM-структури вебсторінок Google Scholar та розробку скриптів на PHP для ефективного вилучення даних. Зібрана інформація зберігається в базі даних MySQL, що дозволяє формувати звіти про профілі дослідників, їхні афіліації, кількість цитувань та списки публікацій. У ході експериментів було успішно зібрано значний обсяг даних про дослідників та їхні публікації, що демонструє ефективність запропонованого підходу. Результати підкреслюють практичну цінність методу для оцінки наукової діяльності та можуть бути використані для подальшого аналізу продуктивності дослідників та установ.

У роботі [18] досліджено синергію використання мови програмування Python, API та автоматизації для ефективного вебскрапінгу. Автори акцентують увагу на застосуванні бібліотек BeautifulSoup та Scrapy, які спрощують парсинг та вилучення даних з HTML-текстів. Використання API підвищує ефективність процесу, надаючи прямий доступ до структурованих даних та оптимізуючи конвеєр вилучення інформації. Автоматизація процесів дозволяє масштабувати операції скрапінгу, що сприяє швидкому та систематичному збору даних з різноманітних джерел. Окрему увагу приділено етичним аспектам вебскрапінгу, зокрема дотриманню політик та умов використання вебсайтів. У статті розглянуто найкращі практики та етичні норми, а також наведено реальні приклади застосування вебскрапінгу в академічних дослідженнях, що підкреслює його потенціал для відповідального та сталого використання в наукових цілях.

У роботі [19] висвітлюються принципи вебскрапінгу та автоматизації,

які спрямовані на створення системи збору та публікації технічних новин на вебсайті. Автори аналізують застосування інструментів, таких як BeautifulSoup, Selenium та Scrapy, для автоматичного збору, обробки та збереження даних у структурованих форматах. Окремо розглядається розробка інтегрованої системи, що включає логін та реєстрацію користувачів, а також шаблонну сторінку для відображення отриманих новин, що значно покращує доступність актуальної інформації. Запропонована методологія автоматизації процесу демонструє, як можна ефективно оптимізувати збір даних та оновлення контенту, що є важливим інструментом для конкурентної переваги у динамічному середовищі IT-індустрії.

У роботі [20] розглядається роль штучного інтелекту у вдосконаленні процесу вебскрапінгу. Авторка детально аналізує сучасні AI-методи, включаючи машинне навчання, обробку природної мови (NLP – Natural Language Processing) та комп'ютерний зір, які допомагають підвищити точність, продуктивність та масштабованість збору даних з вебресурсів. Окремо досліджуються проблеми традиційного вебскрапінгу, зокрема, обробка динамічного контенту, механізми захисту сайтів від скрапінгу та обробка неструктурованої інформації. Загальний висновок підкреслює значний потенціал AI у підвищенні ефективності збору та аналізу вебданих.

У роботі [21] аналізуються передові методи вебскрапінгу, інтегровані з штучним інтелектом, що дозволяють значно підвищити точність та ефективність збору інформації з динамічних вебресурсів. Автори детально розглядають застосування NLP для вилучення ключових даних, використання моделей LSTM (Long Short-Term Memory) та трансформерів для аналізу текстових послідовностей, а також CNN (Convolutional Neural Network) для розпізнавання структурних шаблонів у HTML-коді. Додатково розглянуто алгоритми машинного навчання, такі як SVM (Support Vector Machine) для класифікації контенту, що допомагають адаптуватися до змін вебструктур та обходити механізми захисту від скрапінгу.

У роботі [22] представлено всебічний огляд сучасних методів вебскрапінгу та їх застосувань у різних галузях. Автори аналізують основні компоненти вебскрапінгу, включаючи дизайн вебскрейперів та обговорюють різні підходи до збору даних з вебсайтів. Особлива увага приділяється порівнянню методів вебскрапінгу, таких як парсинг HTML, використання API та автоматизація браузерів, з акцентом на їх ефективність та придатність для різних типів вебресурсів. Крім того, розглядаються технології, що підтримують процес вебскрапінгу, включаючи мови програмування та фреймворки. Результати дослідження надають корисні рекомендації для науковців та практиків щодо вибору та впровадження оптимальних методів вебскрапінгу для ефективного вилучення та аналізу онлайн-даних.

Проаналізувавши дослідження та публікації, згадані вище, модель автоматизованої системи вебскрапінгу можна представити як комплексну схему, що охоплює технічні, функціональні та аналітичні аспекти. У середовищі Python така модель може бути описана як набір компонентів (2.1):

$$ADCM = \{DCM, DPSM, AnM, SCI\}, \quad (2.1)$$

де DCM (data collection module) – модуль збору даних;

DPSM (data processing and storage module) – модуль обробки та зберігання даних;

AnM (analytical module) – модуль аналітики;

SCI (system configuration interface) – інтерфейс налаштування системи.

Ця модель забезпечує базову взаємодію між користувачем та системою збору даних, проте для вирішення більш складних завдань вона не може бути використана і потребує розширення. Наприклад, простота базової моделі не враховує специфічних нюансів інтеграції AI-алгоритмів із традиційними методами обробки даних, що може впливати на стабільність роботи системи та обмежувати можливості її масштабування.

Розробка моделі вебскрапінгу є складним, багатоетапним процесом, що розпочинається з формування плану та охоплює послідовні фази аналізу, проєктування, програмної реалізації й тестування. На кожному з цих етапів ключову роль відіграє архітектура моделі, оскільки саме вона визначає логіку реалізації функціональних компонентів, встановлює часові рамки розробки та формує орієнтовну структуру витрат. Особливої актуальності набуває застосування динамічної архітектури, яка забезпечує гнучкість та масштабованість рішення. Це, своєю чергою, суттєво підвищує продуктивність та адаптивність моделі, що є критично важливим у контексті інтеграції модулів вебскрапінгу з сучасними технологіями штучного інтелекту.

На основі аналізу існуючих підходів до автоматизованого збору даних із вебресурсів було виявлено, що чимало робіт зосереджується або на ручному створенні правил для створення скраперів, або на інтерактивних мовних агентах. Такий поділ викликає обмеження у масштабуванні, адже традиційні «обгортки» вимагають постійного ручного доопрацювання, а агентно-орієнтовані рішення можуть бути надмірно ресурсомісткими й не завжди забезпечують повторне використання створених рішень.

2.2 Обґрунтування складових моделі автоматизації збору даних

Для вирішення цієї проблеми запропоновано інтегрувати великі мовні моделі (LLM – Large Language Models) в окремі програмні модулі для прискорення процесів автоматичного створення правил збору даних, суттєвого підвищення рівня автоматизації та покращення автономності роботи всієї моделі. LLM – це генеративні моделі штучного інтелекту, які навчаються на великому обсязі текстових даних та здатні генерувати текст, програмний код чи правила вилучення інформації на основі вхідних запитів (промптів). Завдяки здатності аналізувати контекст та виявляти складні патерни, LLM демонструють високу точність у створенні алгоритмів та

правил для автоматичного вилучення інформації. Якість згенерованого коду та правил дозволяє суттєво знизити кількість помилок, підвищити стабільність роботи та ефективність обробки вебданих. Обрання оптимальної LLM моделі забезпечує максимальну продуктивність та автономність системи, що робить інтеграцію таких моделей перспективним та ефективним рішенням для автоматизації процесів збору даних.

У зв’язку з цим запропоновано нову модель [23], яка поєднує генеративні можливості штучного інтелекту (ШІ) з чітко структурованими модулями для вилучення та зберігання даних, що забезпечує гнучкість, адаптивність та масштабованість. Формально запропонована модель може бути описана за допомогою кортежу (2.2):

$$ADCM = \{CE, AM, RM, DCM, DPSM, AnM, SCI\}, \quad (2.2)$$

де CE (control element) – керуючий елемент;

AM (analysis module) – модуль аналізу сайту;

RM (rules module) – модуль генерації правил скрапера.

Перш за все, слід зазначити, що модулі, що входили до моделі (2.1), зазнали суттєвих змін. Наприклад, модуль аналітики, який у попередній версії забезпечував лише перегляд інформації, тепер розширено таким чином, що дозволяє налаштовувати Python-скрипти для обробки даних та генерувати графічні представлення зібраної інформації, що значно підвищує аналітичний потенціал системи. Аналогічно, модуль збору даних (DCM) оптимізовано для більш тісної інтеграції з іншими компонентами, що забезпечує швидший зворотний зв’язок через асинхронну взаємодію, а модуль зберігання та обробки даних (DPSM) отримав розширені функції нормалізації та аналізу, що дозволяє працювати з більш складними структурами інформації.

По-друге, до моделі було додано нові модулі, яких не було у початковій версії, що значно розширює її функціональність. Зокрема, додано

модуль керуючого елемента (CE), який виступає центральною ланкою координації між всіма іншими модулями та забезпечує більш ефективний розподіл завдань. Також впроваджено модуль аналізу сайту (AM), завдання якого полягає у попередньому аналізі вебсторінок з метою ідентифікації релевантного контенту, що дозволяє уникнути обробки технічних або повторюваних за структурою сторінок. Нарешті, інтегровано модуль генерації правил (RM), який за допомогою LLM автоматично створює та оптимізує XPath-запити для вилучення даних, що сприяє підвищенню точності збору інформації. Нові модулі дозволяють моделі не лише оперативно адаптуватися до змін у структурі вебресурсів, забезпечуючи більшу гнучкість, ефективність та масштабованість системи порівняно з першою версією, але й значно підвищують рівень автоматизації всієї моделі, впроваджуючи автоматичне оновлення правил збору даних, моніторинг ефективності збору та автоматичного коригування процесів вилучення інформації, що мінімізує потребу у ручному втручанні та забезпечує безперервну роботу моделі.

У порівнянні з традиційними рішеннями, запропонована модель дозволяє організувати подієво-орієнтовану систему, яка забезпечує автоматичний аналіз сайтів та виявлення контентних структур на них, автоматизоване вилучення та структурування вебданих, асинхронну взаємодію між модулями через чергу повідомлень, динамічну генерацію правил збору контенту на основі LLM, масштабовану обробку HTML-структур та ефективне управління даними у NoSQL-базах даних. Усі модулі використовують мову Python як основну мову програмування.

Керуючий центр (CE) є центральним компонентом модульної архітектури системи, який забезпечує координацію між усіма функціональними елементами, управління процесами та контроль виконання операцій. Його основна мета – забезпечити стабільність, масштабованість та адаптивність роботи системи, гарантуючи узгоджену взаємодію між всіма елементами.

СЕ складається з кількох основних частин, кожна з яких виконує використовує критично важливий функціонал для роботи всієї інфраструктури. Керуючий елемент будується на множині складових (2.3):

$$CE = \{MPS, CM, MS, LS, MS, CS\}, \quad (2.3)$$

де MPS (message processing service) – сервіс обробки повідомлень;

CM (control mechanism) – механізм управління;

MS (mechanism of synchronization) – механізм синхронізації компонентів;

LS (login service) – сервіс логування;

MS (monitoring service) – сервіс моніторингу системи;

CS (configuration service) – сервіс конфігурації.

Сервіс обробки повідомлень (MPS) є ключовим елементом комунікаційної інфраструктури системи. Він забезпечує асинхронний обмін даними між компонентами за допомогою механізму черг повідомлень RabbitMQ. Використовуючи підхід подієво-орієнтованої архітектури, система підписується на події, такі як «виявлено новий сайт», «завершено збір даних» або «помилка при отриманні контенту», та ініціює відповідні реакції без необхідності прямого виклику.

Механізм управлінням (CM) контролює виконання бізнес-логіки та розподіляє завдання між компонентами відповідно до заданих сценаріїв роботи. Він також відповідає за обробку виняткових ситуацій, що виникають під час роботи та реалізує механізми автоматичного перезапуску процесів у разі збоїв. Координація виконання завдань відбувається через систему черг повідомлень, що дозволяє мінімізувати затримки й оптимізувати використання ресурсів. Цей механізм контролюється інтерфейсом налаштування системи (SCI), який дозволяє задавати правила виконання операцій, налаштовувати пріоритети та визначати параметри масштабування.

Механізм синхронізації компонентів (MS) запобігає конфліктам між

сервісами та контролює узгодженість операцій. Він визначає пріоритетність виконання завдань, здійснює балансування навантаження та підтримує цілісність даних між компонентами під час їхньої взаємодії. Завдяки цьому забезпечується коректне виконання паралельних процесів і стабільність функціонування системи.

Сервіс логування (LS) відповідає за збір, зберігання та аналіз історії подій у системі. Він інтегрований із централізованою системою логування Grafana Loki, що дозволяє ефективно аналізувати лог-файли, знаходити проблемні місця в роботі системи та забезпечувати її стабільну роботу. Логування включає не лише успішні операції, а й помилки та попередження, що дозволяє швидко реагувати на збої та покращувати продуктивність системи.

Сервіс моніторингу системи (MS) забезпечує контроль стану компонентів моделі та дозволяє виявляти аномалії, що можуть вплинути на її продуктивність. Використовуючи систему збору метрик Prometheus, він відстежує навантаження на сервери, швидкість обробки запитів та доступність сервісів. Також генерує звіти про стабільність роботи системи та дозволяє своєчасно реагувати на можливі проблеми.

Сервіс конфігурації (CS) забезпечує управління параметрами системи, включаючи визначення списку вебресурсів для збору даних, налаштування шаблонів парсингу та обмеження для кожного джерела. Конфігураційні файли зберігаються у форматі YAML, що забезпечує їхню читабельність і зручність редагування. Завдяки інтерфейсу налаштування системи (SCI) користувач може легко змінювати параметри конфігурації, додавати нові вебресурси до списку для збору даних та налаштовувати кількість одночасно працюючих скраперів.

Керуючий центр виступає ключовою ланкою між усіма компонентами системи, забезпечуючи їхню безперебійну та узгоджену роботу. Завдяки подієво-орієнтованій архітектурі та розподіленій обробці повідомлень він дозволяє легко масштабувати систему, швидко адаптувати її до змін у

структурі вебресурсів та гарантувати стабільність її функціонування навіть у високонавантажених середовищах.

Модуль аналізу сайту (AM) є ключовим елементом процесу автоматизованого збору даних, що забезпечує проходження по всіх сторінках вебресурсу, ідентифікацію контентних сторінок та відправку цієї інформації для подальшої обробки. Його основна функція – визначення, які сторінки містять корисний контент, а які є технічними або інформаційними. Структурно він складається з наступних основних компонентів (2.4):

$$AM = \{SC, FM, ClS, RTM\}, \tag{2.4}$$

де SC (site crawler) – механізм обходу сайту;

FM (filtration mechanism) – механізм фільтрації унікальних сторінок;

ClS (classification service) – сервіс класифікації сторінок;

RTM (results transfer mechanism) – механізм передачі результатів.

Механізм обходу сайту (SC) забезпечує ітеративне сканування вебресурсу, переходячи за посиланнями та формуючи карту сторінок. Для цього використовується бібліотека Scrapy, яка дозволяє виконувати швидкий обхід у багатопотоковому режимі. Важливим аспектом є визначення структури сайту та формування списку всіх доступних URL для подальшої обробки.

Механізм фільтрації унікальних сторінок (FM) працює на основі аналізу структури URL та DOM-дерева (Document Object Model), запобігаючи дублюванню аналізу однакових сторінок. Він визначає, які сторінки містять пагінацію або динамічні елементи, та уникає їх повторного розгляду. Це дозволяє оптимізувати обчислювальні ресурси та зменшити кількість запитів до LLM [24].

Сервіс класифікації сторінок (ClS) використовує LLM для аналізу вмісту сторінки та визначення її типу. В процесі аналізу застосовуються заздалегідь підготовлені prompts (інструкції для мовної моделі), що

дозволяють моделі оцінювати, чи є сторінка контентною або вона містить лише технічну інформацію. На основі цього визначення формується список сторінок, які можуть бути використані для подальшого збору даних.

Механізм передачі результатів (RTM) забезпечує комунікацію з керуючим елементом (CE), надсилаючи інформацію про виявлені контентні сторінки. Після цього CE передає ці дані в модуль обробки та збереження даних (DPSM), де зберігається інформація про релевантні сторінки конкретного вебсайту для подальшого вилучення даних та генерації правил збору інформації.

Модуль аналізу сайту (AM) є критично важливою частиною системи, оскільки саме він визначає, з яких сторінок можна отримати корисні дані. Використовуючи ефективний механізм обходу, класифікацію через LLM та систему фільтрації унікальних сторінок, він забезпечує швидке та точне визначення структурованого контенту, необхідного для подальшої автоматизованої обробки.

Модуль створення правил скрапера (RM) відповідає за генерацію, тестування та адаптацію правил для автоматичного вилучення інформації з вебресурсів. Він отримує структуру вебсторінок та генерує правила на основі аналізу їхнього DOM-дерева, використовуючи інтеграцію з великими мовними моделями (LLM). RM забезпечує масштабованість та повторне використання правил, а також адаптацію до змін у структурі вебсайтів (2.5):

$$RM = \{RCM, GAM, RVS, RURS\}, \quad (2.5)$$

де RCM (rule creation mechanism) – механізм створення правил;

GAM (generation analysis mechanism) – механізм аналізу генерації;

RVS (rule validation service) – сервіс валідації правил;

RURS (rule update and reset service) – сервіс оновлення та переналаштування правил.

На першому етапі механізму створення правил (RCM) модуль отримує

вхідні дані про структуру вебсторінок та використовує LLM для створення первинних правил скрапінгу. Генеровані правила визначають, як знаходити необхідні дані на сторінках, використовуючи XPath (XML Path Language – мова запитів для навігації по XML/HTML-документах). У цьому процесі застосовуються адаптивні алгоритми, що дозволяють створювати як статичні, так і динамічні XPath-запити для вилучення інформації. Використання генеративного штучного інтелекту значно прискорює процес створення XPath-запитів порівняно з ручною генерацією, а також мінімізує вплив людського фактору на точність отриманих результатів. У ході дослідження було протестовано кілька моделей LLM (GPT-3.5 Turbo, GPT-4, LLaMA 2, Claude 3), щоб визначити найкраще поєднання ціни та ефективності. Оптимальною за результатами тестів виявилася модель GPT-4, яка продемонструвала високу точність генерації правил та швидке навчання на мінімальній кількості промптів. Щоб забезпечити точність та гнучкість вилучення даних, використовуються спеціальні правила (промпти) для LLM. Вони не лише налаштовують LLM для генерації XPath, але й оцінюють їхню ефективність, порівнюючи із вже отриманими шаблонами. Окремі промпти налаштовані на оптимізацію згенерованих шляхів, виправлення помилок та узгодження із загальними шаблонами структури сайту. Особливістю роботи з LLM є здатність моделі не лише автоматично генерувати XPath, а й самостійно ідентифікувати патерни та залежності у DOM-структурі сторінок, що забезпечує високу адаптивність до змін структури вебсайтів.

Процес генерації XPath заснований на підході багатоступеневої побудови правил: спочатку створюються базові запити, після чого система ітеративно аналізує HTML-структуру та оптимізує їх під конкретний сайт. Якщо згенеровані XPath-запити виявляються неефективними або повертають некоректні дані, система автоматично застосовує метод step-back generation (покрокового відкату) для повернення до попереднього стану генерації та коригування параметрів. Це дозволяє покроково тестувати кожен етап побудови XPath-запитів, виявляти вузькі місця у логіці генерації та

адаптувати правила до змін у структурі вебсторінки. Такий підхід дозволяє мінімізувати випадкові помилки та підвищити стабільність роботи скрапера навіть при зміні структури вебсторінок. Завдяки інтеграції LLM, система може швидко визначати, на якому саме етапі генерації XPath виникла помилка, а також автоматично пропонувати варіанти виправлення на основі попереднього досвіду.

Механізм аналізу генерації (GAM) – наступний інструментарій, де здійснюється перевірка узгодженості згенерованих правил із реальними вебсторінками. Використовуючи набір тестових сторінок, система перевіряє правильність знаходження контенту, визначаючи, чи виділяються потрібні елементи. Якщо виявлено некоректні елементи, система коригує правила перед подальшим використанням. Тут важливу роль також відіграє генеративний штучний інтелект, який дозволяє автоматизовано аналізувати результати тестування та класифікувати помилки за типами, що спрощує їх подальше виправлення.

Після успішного аналізу виконується валідація правил за допомогою відповідного сервісу (RVS), де здійснюється тестування правил на ширшому наборі сторінок сайту. Це забезпечує генерацію універсальних правил, які можна застосовувати до всіх подібних сторінок. Додатково виконується перевірка на стабільність та стійкість правил до незначних змін у HTML-структурі.

Завершальний етап – виклик сервісу оновлення та переналаштування правил (RURS). Якщо під час реального скрапінгу виникають помилки або структура сторінки змінилася, система автоматично передає правила на доопрацювання. Оновлені правила перевіряються за тим самим алгоритмом і у разі успішного проходження тестів зберігаються у модулі збереження даних (DPSM) для подальшого використання.

Механізм створення правил є критично важливим компонентом системи збору даних, оскільки дозволяє автоматизувати адаптацію скрапінгових правил та забезпечує високу ефективність вилучення даних

навіть у динамічних вебсередовищах.

Модуль збирання даних (DCM) забезпечує автоматизований та масштабований процес вилучення інформації з вебресурсів відповідно до згенерованих правил. Його основна мета – обробка отриманих правил, вилучення даних та їх передача для подальшої обробки. DCM взаємодіє з іншими модулями системи, отримуючи правила через керуючий елемент (CE), виконуючи збір даних та передаючи їх у модуль обробки та збереження даних (DPSM). У разі виникнення помилок під час вилучення контенту, DCM повідомляє про це CE, що дозволяє оновити правила у модулі генерації правил (RM). Модуль зберігання даних складається із наступних компонентів (2.6):

$$DCM = \{REM, MM, DGS, DTM\}, \quad (2.6)$$

де REM (rule enforcement mechanism) – механізм виконання правил;

MM (multithreading mechanism) – механізм багатопотокової обробки;

DGS (data gathering service) – сервіс збирання даних;

DTM (data transmission mechanism) – механізм передачі даних.

Механізм виконання правил (REM) є початковим етапом, на якому модуль отримує збережені в DPSM правила збору контенту та застосовує їх до цільових вебсторінок. Використання XPath-селекторів та інших параметрів дозволяє точно ідентифікувати та вилучати необхідні дані з кожної вебсторінки. У разі невідповідності результатів система передає звіт у CE, який ініціює коригування правил через RM.

Механізм багатопотокової обробки (MM) забезпечує паралельний збір даних із вебсторінок, що значно прискорює процес скрапінгу. Завдяки використанню Scrapy та асинхронного підходу, система може обробляти десятки запитів одночасно, динамічно регулюючи швидкість обходу для запобігання блокуванню серверів.

Сервіс збирання даних (DGS) відповідає за вилучення та

структурування отриманої інформації. Витягнута інформація може містити текст, зображення, метадані та інші об'єкти вебсторінок. Дані зберігаються у проміжному форматі (JSON), після чого передаються на подальшу обробку.

Передача даних здійснюється через RabbitMQ, що забезпечує ефективний обмін інформацією між DCM та DPSM. У разі виявлення помилок або змін у структурі сайту DCM генерує відповідний івент та надсилає його до CE, який перенаправляє його в RM для оновлення правил.

Таким чином, DCM відіграє важливу роль у процесі автоматизованого збору даних, забезпечуючи точність, адаптивність та ефективну інтеграцію з іншими модулями системи. Він виконує функції обходу, вилучення та передачі інформації, використовуючи заздалегідь визначені правила для отримання релевантного контенту.

Модуль обробки та збереження даних (DPSM) є центральним компонентом системи, що відповідає за нормалізацію, структурування та збереження отриманої інформації. Він забезпечує підтримку всіх даних, що використовуються у процесі збору та аналізу вебконтенту, включаючи правила вилучення, унікальні сторінки, метаінформацію вебресурсів та сам контент. DPSM виконує критично важливу роль у підтримці цілісності даних та забезпечує швидкий доступ до необхідної інформації для інших модулів системи. Основною базою даних для DPSM є MongoDB, оскільки вона забезпечує гнучкість, масштабованість та ефективну обробку неструктурованих даних. Використання NoSQL-бази виправдане через динамічний характер збереження інформації, зокрема правил вилучення, структур вебсторінок та контентних даних. MongoDB дозволяє зберігати документи у форматі JSON, що спрощує роботу із вкладеними структурами та забезпечує швидкий доступ до потрібних елементів. Взаємодія з базою здійснюється через бібліотеку PyMongo, яка надає інструменти для CRUD-операцій (create, read, update and delete), налаштування індексів та оптимізації запитів, що дозволяє підвищити продуктивність системи. DPSM складається з наступних компонентів (2.7):

$$DPSM = \{DClM, NM, TS, SCM, AdM, CM, IS\}, \quad (2.7)$$

де DClM (data cleansing mechanism) – механізм очищення даних;

NM (normalization mechanism) – механізм нормалізації форматів;

TS (transfer service) – сервіс передачі даних;

SCM (schema creation mechanism) – механізм створення схеми;

AdM (adaptation mechanism) – механізм адаптації структури;

CM (control mechanism) – механізм контролю даних;

IS (integration service) – сервіс інтеграції даних.

Механізм очищення даних (DClM) забезпечує видалення зайвих символів, небажаних HTML-тегів, дублікатів та некоректних записів. Він використовує алгоритми фільтрації, регулярні вирази та автоматичне виправлення помилок у структурі даних, що підвищує їх якість перед збереженням.

Механізм нормалізації форматів (NM) приводить дані до єдиного узгодженого вигляду, очищує текст від зайвих пробілів, змінює регістри символів, форматує числові значення та дати у стандартний вигляд (ISO), забезпечуючи узгодженість у всій системі.

Сервіс передачі даних (TS) здійснює комунікацію з іншими модулями через керуючий елемент (CE). DPSM отримує структуровані дані, обробляє їх та надсилає у відповідні сховища або передає в інші модулі системи для подальшого аналізу та використання.

Механізм створення схеми (SCM) визначає структуру для NoSQL-бази MongoDB, автоматично формуючи схему збереження та забезпечуючи її відповідність вимогам системи. При створенні схеми враховуються типи отриманих даних, їхня ієрархія та можливість динамічного розширення, що забезпечує масштабованість системи без необхідності ручного оновлення.

Механізм адаптації структури (AdM) оновлює схему бази при виявленні нових атрибутів у вхідних даних, що дозволяє підтримувати гнучкість та розширюваність системи, адаптуючи її до зміни контенту.

Механізм контролю даних (CM) відповідає за перевірку цілісності та валідність інформації, запобігаючи дублюванню або втраті даних, перевіряє коректність форматів, відповідність заданій структурі та відсіює помилкові або неповні записи.

Сервіс інтеграції даних (IS) забезпечує можливість швидких агрегатних запитів, обробки великих обсягів інформації та синхронізації даних із зовнішніми системами, н дозволяє аналізувати інформацію у реальному часі, забезпечуючи її доступність для аналітичних модулів.

Таким чином, DPSM виконує повний цикл обробки, нормалізації та збереження даних, забезпечуючи їхню чистоту, узгодженість та ефективну інтеграцію в загальну систему збору та аналізу інформації.

Аналітичний модуль (AnM) відповідає за генерацію аналітичних звітів у форматі PDF, використовуючи дані з DPSM. Він інтегрує LLM для автоматичного формування аналітичних скриптів та візуалізації даних через бібліотеки Python. Основна задача модуля аналітики – отримання необхідної інформації, її обробка та подальша візуалізація у вигляді графіків, таблиць та узагальнень, які зберігаються у PDF-документах. Користувач взаємодіє з модулем через інтерфейс налаштування системи (SCI), де користувач задає параметри звіту у вигляді текстового опису аналітики, яку він хоче отримати. Після цього LLM аналізує запит, генерує відповідний Python-скрипт для обробки необхідних даних і формує аналітичний документ у форматі PDF. Модуль аналітики складається з наступних компонентів (2.8):

$$AnM = \{DQM, SM, DVS, GS\}, \tag{2.8}$$

де DQM (data query mechanism) – механізм запиту даних;

SM (script mechanism) – механізм створення скриптів;

DVS (data visualization service) – сервіс візуалізації даних;

GS (generation service) – сервіс генерації PDF.

Механізм запиту даних (DQM) ініціює отримання інформації з DPSM

через керуючий елемент (CE). Спочатку він запитує структуру доступних даних для обраного вебресурсу, після чого надсилає запит на отримання конкретних значень для подальшої обробки, що дозволяє модулю формувати коректні запити на аналітику.

Механізм створення скриптів (SM) використовує LLM для формування Python-скриптів, які виконують аналіз отриманих даних. Спеціалізований системний промпт дозволяє LLM адаптувати код до конкретного запиту, оптимізуючи його під вимоги аналітичного процесу. Автоматично згенерований скрипт виконується в ізольованому середовищі, забезпечуючи безпеку та коректність аналізу.

Сервіс візуалізації даних (DVS) відповідає за обробку результатів аналізу та побудову графіків, діаграм та таблиць. Для цього використовується бібліотека Matplotlib та інші інструменти візуалізації Python. Отримані візуальні представлення формуються відповідно до запиту користувача.

Сервіс генерації PDF (GS) створює фінальний звіт у PDF-форматі, включаючи текстові узагальнення, отримані графіки та таблиці. Весь контент структурується за допомогою спеціалізованого системного промпту, який адаптує розміщення елементів відповідно до логіки звіту. Генерація документа відбувається автоматично після завершення аналізу.

Таким чином, модуль аналітики забезпечує повний цикл: від отримання даних до їхньої обробки та формування PDF-звітів. Його інтеграція з LLM дозволяє динамічно адаптувати запити та автоматизувати створення аналітичних матеріалів, що значно підвищує ефективність роботи системи.

Інтерфейс налаштування системи (SCI) є центральним компонентом для управління та контролю всієї моделі. Це консольний інтерфейс, що забезпечує взаємодію користувача із системою через набір команд. Основна функція SCI – управління обробкою сайтів, налаштування параметрів збору даних, контроль ресурсів та генерація аналітичних звітів на основі отриманої інформації. SCI складається з таких компонентів (2.9):

$$SCI = \{PCM, CM, RGM\}, \quad (2.9)$$

де PCM (processing control mechanism) – механізм керування обробкою;

CM (configuration mechanism) – механізм конфігурації обробників;

RGM (report generation mechanism) – механізм генерації звітів.

Механізм керування обробкою (PCM) відповідає за додавання нових сайтів у процес збору даних. Користувач може передавати списки сайтів, задавати параметри скрапінгу та визначати правила обробки для кожного ресурсу. Запити на обробку надходять на керуючий елемент (CE), який координує їх виконання між іншими модулями.

Механізм конфігурації обробників (CM) дозволяє задавати кількість активних процесів збору даних, встановлювати ліміти на використання ресурсів та керувати параметрами виконання завдань. Це забезпечує гнучкість та оптимізацію системи під різні навантаження.

Механізм генерації звітів (RGM) ініціює створення аналітичних документів на основі отриманих даних. Він взаємодіє з модулем аналітики, передаючи запити на створення PDF-звітів. Користувач задає параметри аналітики у текстовому форматі, після чого система автоматично формує відповідний звіт.

Таким чином, інтерфейс налаштування системи забезпечує гнучке управління всією моделлю, дозволяючи користувачам налаштовувати обробку сайтів, контролювати ресурси та ініціювати створення аналітичних матеріалів у зручному консольному форматі.

3 ЕКСПЕРИМЕНТАЛЬНА ОЦІНКА ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ МОДЕЛІ АВТОМАТИЗОВАНОГО ВЕБСКРАПІНГУ ДАНИХ

3.1 Методологія та критерії оцінювання вебскрапінг-моделей

Для забезпечення об’єктивної оцінки запропонованої моделі автоматизованого вебскрапінгу даних на базі генеративних мовних моделей (LLM), необхідно визначити чітку методологію та критерії оцінювання. Основною метою експерименту є перевірка точності, стійкості та економічної доцільності розробленої архітектури. Для цього планується провести порівняльний аналіз із кількома іншими поширеними підходами.

Першим етапом експериментальної оцінки є підготовка тестового датасету. Він включає не менше 15 вебсайтів, які репрезентують різноманітні типи ресурсів: новинні портали, інтернет-магазини, форуми та інші інформаційні платформи. Кожен із обраних сайтів характеризується різною складністю DOM-структури, що дозволяє перевірити адаптивність та універсальність моделі в умовах різнорідних шаблонів подання інформації. Для підвищення репрезентативності результатів відібрані ресурси відрізняються за частотою оновлення контенту, обсягами даних та технічними особливостями, включаючи використання AJAX та динамічних JavaScript-компонентів. Важливим елементом підбору ресурсів [25] є також наявність різноманітних захисних механізмів від автоматизованих запитів, таких як CAPTCHA та IP-фільтрація, що дозволяє перевірити здатність моделі долати такі перешкоди.

Наступним етапом визначено набір метрик для оцінювання ефективності моделей. Основні показники включають Precision (точність), Recall (повноту) та F1-score для оцінки якості автоматичної генерації XPath-запитів. Precision визначає частку правильно вилучених елементів від загальної кількості зібраних даних, тоді як Recall характеризує частку правильно вилучених елементів від усіх релевантних даних. Комплексна

метрика F1-score дозволяє врахувати баланс між точністю та повнотою, забезпечуючи інтегральну оцінку ефективності моделі.

Крім того, важливими критеріями є середній час налаштування, який вимірюється в годинах роботи розробника, та середній час збору даних зі сторінки, виражений у секундах. Ці метрики дозволяють оцінити витрати людських та обчислювальних ресурсів, що є критично важливими для практичного застосування автоматизованих систем у промислових масштабах. Додатково до цього проводиться аналіз споживання обчислювальних ресурсів, таких як використання пам'яті CPU/GPU, а також економічна складова: вартість обчислень, виражена в доларах США на мільйон оброблених токенів.

Для порівняльного аналізу обрано три альтернативні підходи. Перший – класичний метод вебскрапінгу на основі фреймворку Scrapy із ручним налаштуванням XPath-запитів. Цей підхід широко застосовується на практиці, але характеризується високою залежністю від ручного налаштування та значними витратами часу на підтримку. Він вимагає регулярного оновлення правил парсингу через часті зміни в структурі вебсторінок, що призводить до додаткових витрат ресурсів на постійну підтримку та моніторинг.

Другий підхід, який включений до порівняльного аналізу, – використання моделі машинного навчання DeepDOM для класифікації вузлів DOM. Цей метод базується на попередньому навчанню моделі на великій кількості зразків, що дозволяє автоматично ідентифікувати необхідні елементи на вебсторінках, проте потребує значних обчислювальних ресурсів для підтримки високої точності. Важливо відзначити, що для ефективного використання DeepDOM необхідна постійна актуалізація навчальних даних, що додає додаткових витрат на періодичне перенавчання моделі.

Третій підхід – це використання шаблонного вебскрапера, що базується на регулярних виразах. Цей метод дозволяє швидко налаштовувати правила вилучення даних для ресурсів з відносно стабільною структурою, проте має

низьку адаптивність до змін структури вебсторінок і є малоефективним для роботи з динамічним контентом. Шаблонний підхід показує хороші результати в умовах незмінних структур, проте суттєво програє в ситуаціях, коли вебресурси регулярно оновлюються або змінюють формат представлення інформації.

Запропонована LLM-орієнтована модель порівнюється з зазначеними підходами за всіма визначеними критеріями. Такий комплексний підхід дозволить зробити висновки щодо реальних переваг та недоліків запропонованого рішення, що є критично важливим для подальшої оптимізації архітектури та визначення потенційних сфер його ефективного застосування. Зокрема, особлива увага приділяється оцінці здатності моделі адаптуватися до змін DOM-структури, мінімізувати ручне втручання, а також оптимізувати витрати на підтримку та обчислювальні ресурси. У рамках експерименту також планується врахувати потенційні обмеження та недоліки запропонованого підходу, такі як складність інтеграції з існуючими системами та залежність від доступності ресурсів генеративних моделей. Таким чином, комплексний підхід до оцінювання дозволяє створити повну картину ефективності запропонованої LLM-орієнтованої моделі автоматизованого вебскрапінгу.

3.2 Результати експериментів і порівняльний аналіз

Для отримання комплексного уявлення про ефективність запропонованої LLM-орієнтованої моделі автоматизованого вебскрапінгу було проведено серію експериментів, які дозволили оцінити її роботу за різними сценаріями. Зібрані результати дозволяють здійснити ретельний порівняльний аналіз з використанням статистичних методів оцінювання значущості отриманих даних. У межах проведених експериментів увага була зосереджена на комплексному розгляді різних аспектів роботи вебскрапінг-моделей, щоб отримати повну картину їхньої функціональності, гнучкості та

економічної ефективності.

Експериментальна база складалася із 15 різноманітних вебресурсів, серед яких були новинні портали, великі інтернет-магазини, спеціалізовані форуми та інформаційні агрегатори. Кожен ресурс обирався з урахуванням складності DOM-структури, частоти оновлення інформації, типу контенту, а також ступеня захисту від автоматизованого парсингу (наявність CAPTCHA, блокування IP-адрес). Це дозволило максимально реалістично оцінити продуктивність і адаптивність моделей у практичних умовах.

Для точного порівняльного аналізу було обрано набір ключових метрик, включаючи Precision, Recall та F1-score, які характеризують точність і повноту автоматичного вилучення інформації за допомогою згенерованих XPath-запитів. Також до уваги бралися витрати часу на початкове налаштування системи, час збору даних зі сторінок та необхідність повторних налаштувань у разі зміни структури вебресурсів.

На першому етапі експерименту було проведено узагальнення результатів за ключовими показниками (таблиця 3.1).

Таблиця 3.1. – Узагальнені результати порівняння вебскрапінг-моделей за якістю збору даних

Підхід	Precision	Recall	F1-score	Час налаштування (год)	Час збору (сек)
LLM-модель	0.9	0.92	0.91	1.5	0.42
Scrapy (ручні XPath)	0.86	0.61	0.71	7.8	0.38
DeepDOM	0.84	0.77	0.8	4.7	0.58
Template-based (регулярні вирази)	0.81	0.58	0.67	3.8	0.28

Для підтвердження статистичної значущості отриманих результатів використовувалися статистичні тести (t-тест та тест Вілкоксона). В результаті аналізу було отримано інтервали довіри на рівні 95% для всіх метрик ±0.02-0.04, а p-значення не перевищували 0.05, що підтверджує високу достовірність проведеного експерименту.

Окрему увагу було приділено аналізу трьох ключових сценаріїв, кожен з яких дозволяє виявити конкретні переваги та недоліки оцінюваних підходів.

Перший сценарій (Cold-start) досліджував роботу моделей з новими, невідомими ресурсами. Основною метрикою була Recall, що показує повноту збору інформації з вебсторінок (рисунок 3.1):

- LLM-модель: 0.92;
- Scrapy (ручні XPath): 0.61;
- DeepDOM: 0.76;
- Template-based: 0.57.

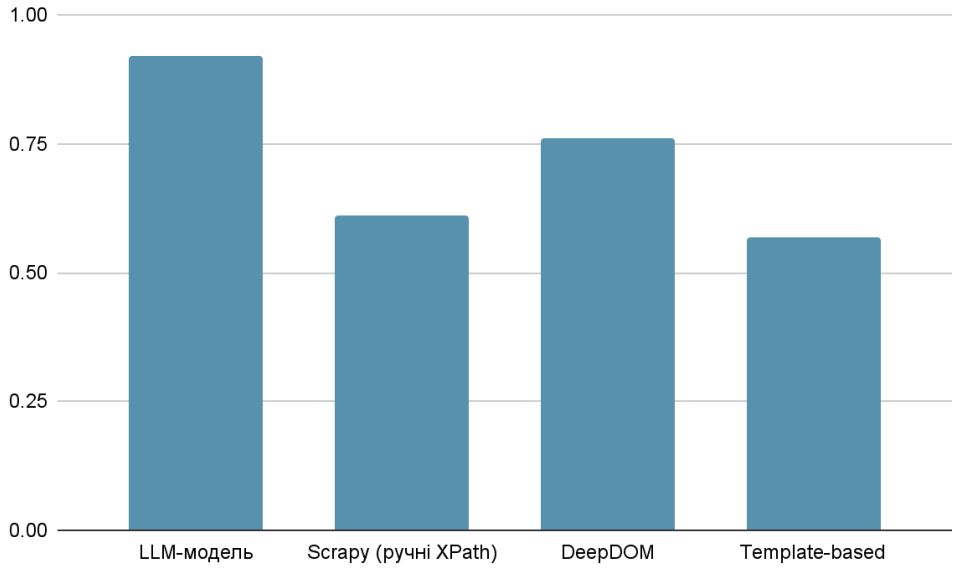

Рисунок 3.1 – Графік Cold-start Recall

Графік чітко демонструє значну перевагу LLM-моделі за показником Recall у порівнянні з іншими методами. Причиною цього є здатність LLM-моделі аналізувати та адаптуватись до нових DOM-структур без попередньо

визначених правил, на відміну від класичних підходів, що вимагають точних і ручних налаштувань.

Другий сценарій розглядав здатність моделей до адаптації при динамічних змінах DOM-структури вебсайтів. Вимірювався відсоток успішних повторних зборів без необхідності додаткових налаштувань (таблиця 3.2).

Таблиця 3.2 – Відсоток успішних повторних зборів ри

Підхід	Відсоток успіху (%)
LLM-модель	89
Scrapy (ручні XPath)	46
DeepDOM	71
Template-based (регулярні вирази)	35

Високий результат LLM-моделі пояснюється її можливістю швидко переосмислювати зміни DOM-структури через генерацію нових XPath-запитів на основі аналізу контексту. В той час як традиційні моделі змушені чекати ручного оновлення правил.

Третій сценарій досліджував витрати часу, необхідні для підтримки актуальності налаштувань моделей при регулярних змінах DOM. Результати представлені у вигляді порівняльної таблиці 3.3.

Таблиця 3.3 – Середні витрати часу на оновлення налаштувань моделей

Підхід	Час на внесення змін (години)
LLM-модель	0.9
Scrapy (ручні XPath)	6.2
DeepDOM	2.8
Template-based (регулярні вирази)	1.9

Менший час налаштування LLM-моделі пояснюється її автоматизованою здатністю швидко переорієнтовуватися на нову структуру сторінок без значних людських ресурсів.

Однак традиційні методи, такі як Scrapy та Template-based, залишаються вигідними для ресурсів зі стабільною структурою через меншу кількість операцій, що здійснюються під час обробки, і простоту логіки, яка не вимагає частих оновлень.

Таким чином, проведений аналіз демонструє суттєві переваги LLM-орієнтованої моделі вебскрапінгу, визначає її найефективніші сценарії використання і закладає основу для подальших досліджень, оптимізації та практичного застосування.

3.3 Конкурентні переваги та обмеження запропонованої моделі

Запропонована модель автоматизованого вебскрапінгу, побудована на основі генеративних мовних моделей (LLM), демонструє ряд значних конкурентних переваг у порівнянні з традиційними підходами, які використовуються для збору даних з вебресурсів. Водночас, як і будь-яке технологічне рішення, вона має певні обмеження, які слід враховувати під час її застосування на практиці.

Однією з основних переваг цієї моделі є автоматична генерація XPath-запитів, що дозволяє скоротити час початкового налаштування на 80% порівняно з класичними ручними методами. Це зменшує навантаження на технічних фахівців та дозволяє швидко розпочати процес збору інформації без значних витрат на підготовку та налаштування. Швидкий старт стає особливо цінним у проєктах з обмеженим бюджетом та жорсткими дедлайнами.

Ще однією значущою перевагою є здатність моделі до самоадаптації під час змін DOM-структури сторінок. Завдяки використанню генеративних моделей, які здатні контекстно аналізувати зміни, модель автоматично

оновлює XPath-запити. Це суттєво скорочує витрати на підтримку та оперативно усуває необхідність додаткового ручного втручання.

Важливим аспектом запропонованої моделі є її модульна архітектура. Вона дозволяє легко вмикати або вимикати окремі підсистеми відповідно до конкретних потреб проєкту. Це значно підвищує гнучкість і універсальність системи, дозволяючи адаптувати її до різних завдань та умов використання.

Горизонтальна масштабованість є наступною важливою перевагою. Архітектура моделі підтримує швидке розгортання додаткових інстансів для збільшення обсягів оброблюваних даних, що робить її оптимальною для використання у проєктах з великими обсягами інформації та високими навантаженнями.

З точки зору економічної ефективності, використання моделі дозволяє суттєво знизити витрати на DevOps-операції у довгостроковій перспективі. Це пов’язано з мінімізацією ручних втручань та скороченням часу на моніторинг і коригування системи. Згідно з результатами практичних експериментів, після 6 місяців використання моделі прогнозується ROI понад 150%, а витрати на підтримку мають скоротитися приблизно на 45%.

Однак, незважаючи на численні переваги, модель має певні обмеження. Першим суттєвим обмеженням є висока початкова вартість GPT-інференсу, особливо при обробці великих обсягів даних. Це може створювати суттєві фінансові перепони для невеликих проєктів або стартапів з обмеженим бюджетом.

Іншим важливим аспектом є потреба у ефективному кешуванні промптів, що використовуються генеративними моделями. Без належної організації кешу система може повторно виконувати ті ж самі дороговартісні обчислення, що негативно впливає на загальну продуктивність і збільшує операційні витрати.

Також суттєвою проблемою є ризики виникнення LLM-галюцинацій – ситуацій, коли генеративна модель створює некоректні або неіснуючі XPath-запити. Це може призвести до збору неповної або невірної інформації, що

негативно впливає на якість даних та потребує додаткового моніторингу і перевірок.

Для подолання зазначених обмежень розроблено ряд практичних рекомендацій. Однією з таких рекомендацій є застосування кешування результатів генерації XPath-запитів (Result-XPath), що дозволяє уникнути зайвих обчислень та значно підвищити продуктивність системи, зменшуючи витрати на повторні виклики моделі.

Також рекомендується комбінувати використання генеративної моделі з rule-based підходом для простих та стабільних сторінок, що мають статичну структуру. Це дозволяє ефективно зменшити витрати на генеративні інференси і забезпечити стабільність збору даних для відповідних типів ресурсів.

Ще однією ефективною стратегією є використання LLM-дистиляції для зниження вартості генеративних моделей. Цей метод дозволяє навчати менш ресурсомісткі моделі, що повторюють поведінку більш потужних і дорогих генеративних моделей, але при цьому потребують значно менших витрат на обчислення.

Таким чином, запропонована модель автоматизованого вебскрапінгу на базі генеративних мовних моделей має значні конкурентні переваги, що дозволяють ефективно вирішувати завдання збору інформації у різних умовах і масштабах. Водночас, знання та врахування її обмежень дозволяє ефективно управляти ризиками та оптимізувати процес збору даних, забезпечуючи високий рівень продуктивності та економічну ефективність у довгостроковій перспективі.

ВИСНОВКИ

У роботі було проведено комплексний аналіз сучасних підходів до автоматизованого збору даних із вебресурсів та експериментальна оцінка підтвердили ефективність запропонованої модульної моделі на основі LLM. Автоматична генерація правил вилучення інформації з використанням великих мовних моделей зменшила середній час розробки скрапера для одного сайту з 8-12 годин до 20-30 хвилин, що забезпечило більш ніж 16-кратне прискорення процесу збору даних без втрати їх точності та повноти.

Керуючий центр моделі гарантує асинхронну взаємодію між компонентами та підтримку масштабованості, а модульна структура дозволяє миттєво адаптувати правила вилучення до змін у DOM-структурі вебсторінок. Якість збору даних суттєво поліпшується завдяки здатності моделі автоматично коригувати алгоритми аналізу в реальному часі.

Інтеграція систем моніторингу (RabbitMQ, Prometheus, Grafana Loki) забезпечує автоматичне відстеження стану компонентів та своєчасне виявлення аномалій у роботі моделі.

Запропонована модель є перспективною платформою для подальшого розвитку рішень із автоматизованого вебскрапінгу: вона поєднує високий рівень адаптивності, продуктивності та надійності, що робить її оптимальним рішенням для широкого спектра прикладних задач – від маркетингової аналітики до наукових досліджень і бізнес-моніторингу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Zohaib M. (2023). A Responsive Framework for Research Portals Data using Semantic Web Technology. URL: https://arxiv.org/abs/2306.11642.

2. Brown M., Gruen A., Maldoff G., Messing S., Sanderson Z., Zimmer M. (2024). Web Scraping for Research: Legal, Ethical, Institutional, and Scientific Considerations. URL: https://arxiv.org/abs/2410.23432.

3. Huang W., Gu Z., Peng C., Li Z., Liang J., Xiao Y., Wen L., Chen Z. (2024). AutoScraper: A Progressive Understanding Web Agent for Web Scraper Generation. URL: https://arxiv.org/abs/2404.12753.

4. Xu Z., Liu Z., Yan Y., Liu Z., Yu G., Xiong C. (2024). Cleaner Pretraining Corpus Curation with Neural Web Scraping [online]. URL: https://arxiv.org/abs/2402.14652.

5. Велика мовна модель (LLM). Електронний посібник. URL: https://uk.shaip.com/blog/a-guide-large-language-model-llm.

6. Foerderer J. (2023). Should We Trust Web-Scraped Data? URL: https://arxiv.org/abs/2308.02231.

7. Official website Scrapy. URL: https://www.scrapy.org.

8. Beautiful Soup Documentation. URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc.

9. Official website Selenium. https://www.selenium.dev.

10. Мошура О., Деркач Т., Дмитренко А., Клочко Л., Лоза В. Особливості та можливості застосування llm у сфері розробки програмного забезпечення // Системи управління, навігації та зв'язку. Збірник наукових праць. Полтава: ПНТУ, 2024. № 4. С. 109-113.

11. RabbitMQ. RabbitMQ Documentation. RabbitMQ: Messaging that just works. URL: https://www.rabbitmq.com/documentation.html

12. Prometheus. Prometheus Documentation. Prometheus: Monitoring system & time series database. URL: https://prometheus.io/docs/introduction/overview

13. Grafana Loki. Grafana Loki Documentation. URL: https://grafana.com/docs/loki/latest/

14. MongoDB. MongoDB Manual. URL: https://docs.mongodb.com/manual/

15. NoSQL. “NoSQL.” Wikipedia, The Free Encyclopedia. URL: https://en.wikipedia.org/wiki/NoSQL

16. Січкарюк Р.К., Корніловська Н.В., Лур’є І.А., Вороненко М.О. Розробка та впровадження ефективних методів вебскрапінгу для автоматизованого збору і обробки даних з використанням Python. X Міжнародна наукова-практична конференція “Інформатика. Культура. Техніка”. Одеса: Національний університет «Одеська політехніка», Інститут комп'ютерних систем. 2024. Том 1, №1. С. 62-68.

17. Lotfi C., Srinivasan S., Ertz M., Latrous I. Web Scraping Techniques and Applications: A Literature Review. SCRS Conference Proceedings on Intelligent Systems, 2021. C. 381-394

18. Pandey K., Tale C., Yere T., Rajeshirke R., Jadhav R. Web Scraping: Leveraging the Power of Python, APIs, and Automation. International Journal of Novel Research and Development. 2024; 9(3): b383–b389

19. Bhute T., Raut S., Kannake S., Guda B., Sheikh M. Web Scraping and Automation. International Journal of Creative Research Thoughts. 2024; 12(5): 285-290

20. Weerasinghe K., Maduranga M.W.P., Kawya M.M.V.T. Enhancing Web Scraping with Artificial Intelligence: A Review. 4th Research Symposium of Faculty of Computing, 2024. URL: https://www.researchgate.net/publication/379024314_Enhancing_Web_Scraping_with_Artificial_Intelligence_A_Review

21. Huang C.-J. The Synergy of Automated Pipelines with Prompt Engineering and Generative AI in Web Crawling. arXiv preprint arXiv:2502.15691, 2024. doi:10.48550/arXiv.2502.15691

22. Lotfi C., Srinivasan S., Ertz M., Latrous I. Web Scraping Techniques and Applications: A Literature Review. SCRS Conference Proceedings on

Intelligent Systems, 2021. URL: https://www.researchgate.net/publication/367719780_Web_Scraping_Techniques_and_Applications_A_Literature_Review

23. Філімончук Т.В., Копайло Я.Р., Партика С.О., Севостьянова О.М. Модель автоматизації збору даних з вебресурсів на основі генеративного штучного інтелекту // Інформаційно-керуючі системи на залізничному транспорті. Харків: УкрДУЗТ, 2025. Випуск №2 (подано до друку).

24. Ahluwalia A., Wani S. (2024). Leveraging Large Language Models for Web Scraping. URL: https://arxiv.org/abs/2406.08246

25. Миколайчук Р.А., Старинський І.М., Миколайчук В.Р. Аналіз технологічних аспектів реалізації веб-скрапінгу статичних і динамічних сайтів // Сучасні інформаційні технології у сфері безпеки та оборони. Київ, Україна, 2024. 51(3), С. 80-88.