УДК 004.41 DOI: 10.18664/ikszt.v30i2.335414

ФІЛІМОНЧУК Т. В., к.т.н., доцент,
КОПАЙЛО Я. Р., магістрант,
ПАРТИКА С. О.,
СЕВОСТЬЯНОВА О. М.
(Харківський національний університет радіоелектроніки)

Модель автоматизації збору даних із вебресурсів на основі генеративного штучного інтелекту

Анотація. Актуальність *дослідження зумовлена тим, що в сучасних умовах стрімкої диджиталізації, глобального розвитку інтернету та безперервного зростання обсягів даних, отриманих із різноманітних вебресурсів, зростає потреба в розробці ефективних інструментів для автоматизованого збирання, аналізу та структурування інформації. Вебскрапінг став невіддільною складовою процесів інформаційного забезпечення в багатьох галузях. Проте традиційні методи збору даних часто потребують складного ручного налаштування, що ускладнює масштабування систем та оперативне реагування на зміни структури вебсторінок. У зв'язку з цим використання універсальних правил та алгоритмів для створення вебскраперів набуває особливої актуальності. Особливу роль у цьому відіграють сучасні мовні моделі (LLM), здатні генерувати адаптивні рішення, зокрема у вигляді динамічних виразів XPath, що використовуються для ефективного вилучення потрібної інформації з HTML-структури вебсторінок. Такий підхід дає змогу створити масштабовані та самонавчальні системи збору вебданих, які здатні адаптуватися до різних форматів контенту та швидко підлаштовуватися до змін у вебінтерфейсах.* ***Об'єктом дослідження*** *є застосування великих мовних моделей (LLM) для створення автоматизованих механізмів збору вебданих, а також дослідження їх потенціалу у генерації високоточних запитів XPath.* ***Предметом дослідження*** *виступає модель, орієнтована на збір вебданих, яка передбачає багаторівневу взаємодію між незалежними функціональними модулями системи та забезпечує їх інтеграцію через уніфіковані інтерфейси та стандартизовані формати обміну даними.* ***Результати.*** *Запропонована розширена модульна модель системи збору вебданих, яка охоплює всі ключові етапи обробки вебконтенту: від виявлення релевантних джерел інформації, через генерацію та оптимізацію XPath-запитів, до збереження отриманих даних у масштабованій, структурованій базі даних. Запропоноване архітектурне рішення забезпечує високу адаптивність до змін, стабільність функціонування системи, мінімізацію витрат на технічну підтримку.* ***Висновки.*** *Використання генеративних мовних моделей для автоматизації створення вебскраперів відкриває нові можливості для гнучкого, точного та стабільного збору вебданих. Такий підхід сприяє ефективній інтеграції систем у мінливе вебсередовище, дозволяючи зменшити часові та фінансові витрати на підтримку інформаційно-аналітичних платформ.*

Ключові слова: *генеративний штучний інтелект, вебзбирання даних, автоматизація, модульна архітектура, база даних, скрапери, генеративна модель.*

Постановка проблеми

Інформаційні технології розвиваються дуже стрімко, впливаючи на всі сфери життя. Однією з основних сучасних технологій є вебскрапінг – процес автоматизованого збирання даних із вебсайтів, який суттєво спрощує отримання, аналіз і використання інформації, є дуже поширеним у бізнесі, ринковій аналітиці, моніторингу конкурентів та інших сферах, даючи змогу автоматизувати процеси збирання даних і мінімізувати витрати на ручну обробку.

На сьогодні вебскрапінг стикається з низкою серйозних перешкод, які обумовлені як технічними, так і правовими змінами в мережі Інтернет. Зростання популярності вебсторінок, що динамічно формуються з використанням JavaScript-фреймворків, ускладнює вилучення даних стандартними засобами. Слід зазначити, що сайти активно впроваджують захист від автоматизованих запитів: антибот-системи, CAPTCHA, обфускацію DOM-структур і часті зміни верстки, що робить скрапінг нестабільним.

Правове середовище також посилено: дотримання регламентних вимог (GDPR і CCPA), а також обмеження, що прописані в угодах користувачів, накладають додаткові бар'єри на автоматизоване збирання інформації. Тому традиційні підходи для вебскрапінгу стають усе менш ефективними, ресурсомісткими та потребують постійного ручного втручання, що створює необхідність у нових, більш гнучких та інтелектуальних підходах, які здатні адаптуватися до умов вебсередовища, що змінюються.

Останні досягнення у сфері штучного інтелекту, зокрема великі мовні моделі (Large Language Models, LLM), значно розширюють можливості автоматизації технологічних процесів. Вони демонструють здатність аналізувати текстові дані, генерувати програмний код, структурувати інформацію та формувати складні правила, базовані на розпізнаванні різних патернів. Завдяки цьому LLM можуть зробити вебскрапінг ще гнучкішим, адаптивним і більш ефективним у роботі з вебданими. Автоматичний аналіз вебсторінок дасть змогу не лише ефективно структурувати інформацію, а й визначати головні закономірності у великих масивах даних. Застосування LLM у цьому процесі забезпечить розширені можливості оптимізації та створення більш ефективних алгоритмів збирання інформації.

Впровадити ці рішення допоможе мова програмування Python, яка пропонує широкий набір бібліотек для вебскрапінгу, обробки даних та інтеграції з LLM. Простий синтаксис, гнучкість і активна спільнота розробників роблять її ідеальним вибором для масштабованих і продуктивних систем. Додаткове використання таких інструментів, як RabbitMQ (система черг повідомлень), Prometheus (система моніторингу) і Grafana Loki (система логування), допоможуть забезпечити надійний обмін даними, підвищити гнучкість архітектури та спростити масштабування систем за необхідності.

Аналіз останніх досліджень і публікацій

У межах дослідження автоматизованих методів збирання та аналізу вебданих потрібно звернути увагу на низку наукових робіт і статей, що висвітлюють сучасні підходи до вебскрапінгу та обробки інформації. Зокрема, вивчають інструменти на базі мови програмування Python, такі як Scrapy, BeautifulSoup і Selenium, які дають змогу ефективно отримувати, структурувати та аналізувати дані. Окрему увагу приділено ролі штучного інтелекту, включаючи NLP (Natural Language Processing), і машинному навчанню у вдосконаленні процесів обробки текстових і структурованих даних. Також розглядають технічні аспекти розроблення масштабованих архітектур для збирання 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 [1] досліджено сучасні підходи для розроблення та впровадження ефективних методів вебскрапінгу для автоматизованого збирання та обробки даних із використанням мови програмування Python. Автори акцентують на застосуванні бібліотек BeautifulSoup, Selenium і Scrapy, що забезпечують високу швидкість і точність отримання інформації з різних вебджерел, зокрема вторинних ринків, особливий на автоматизації процесу за допомогою створення скриптів і використання планувальників завдань, таких як cron, що дає змогу безперервно оновлювати бази даних без втручання людини. Окремо розглянуто методи обробки та очищення зібраних даних, включаючи фільтрацію непотрібної інформації, дублікатів і шумів, що сприяє підвищенню якості даних. Практичне застосування результатів дослідження демонструє їхню цінність для аналізу ринку, оцінювання попиту і прогнозування якості, підкреслюючи важливість запропонованих методів.

У роботі [2] подано всебічний огляд сучасних методів вебскрапінгу та їх застосувань у різних галузях. Автори аналізують основні компоненти вебскрапінгу, включаючи дизайн вебскраперів, та обговорюють різні підходи для збирання даних із вебсайтів. Особливу увагу приділено порівнянню методів вебскрапінгу, таких як парсинг HTML, використання API та автоматизація браузерів, із зазначенням їхньої ефективності та придатності для різних типів вебресурсів. Крім того, розглядають технології, що підтримують процес вебскрапінгу, включаючи мови програмування та фреймворки. Результати дослідження дають корисні рекомендації для науковців і практиків щодо вибору та впровадження оптимальних методів вебскрапінгу для ефективного вилучення та аналізу онлайн-даних.

Також у роботі досліджено застосування методу HTML DOM для вебскрапінгу з метою автоматизованого збору наукових статей із Google Scholar. У дослідженні описано процес ідентифікації специфічних елементів DOM-структури вебсторінок Google Scholar і розроблення скриптів на PHP для ефективного вилучення даних. Зібрану інформацію зберігають у базі даних MySQL, що дає змогу формувати звіти про профілі дослідників, їхні афіліації, кількість цитувань і списки публікацій. Під час експериментів було успішно зібрано значний обсяг даних про дослідників і їхні публікації, що демонструє ефективність запропонованого підходу.

Результати підкреслюють практичну цінність методу для оцінювання наукової діяльності та можуть бути використані для подальшого аналізу продуктивності дослідників та установ.

У роботі [3] досліджено синергію використання мови програмування Python, API та автоматизації для ефективного вебскрапінгу. Автори звертають увагу на застосування бібліотек BeautifulSoup і Scrapy, які спрощують парсинг і вилучення даних із HTML-текстів. Використання API підвищує ефективність процесу, надаючи прямий доступ до структурованих даних та оптимізуючи конвеєр вилучення інформації. Автоматизація процесів дає змогу масштабувати операції скрапінгу, що сприяє швидкому та систематичному збиранню даних із різних джерел. Окремо виділено етичні аспекти вебскрапінгу, зокрема дотримання політик і 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 і сталого використання в наукових цілях.

У роботі [4] висвітлено принципи вебскрапінгу та автоматизації, які спрямовані на створення системи збирання та публікації технічних новин на вебсайті. Автори аналізують застосування інструментів, таких як BeautifulSoup, Selenium і Scrapy, для автоматичного збирання, обробки та збереження даних у структурованих форматах. Окремо розглянуто розроблення інтегрованої системи, що включає логін і 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 збирання даних та оновлення контенту, що є важливим інструментом для конкурентної переваги в динамічному середовищі IT-індустрії.

У роботі [5] розглянуто роль штучного інтелекту у вдосконаленні процесу вебскрапінгу. Авторка детально аналізує сучасні AI-методи, включаючи машинне навчання, обробку природної мови (Natural Language Processing, NLP) і комп'ютерний зір, які допомагають підвищити точність, продуктивність і масштабованість збирання даних із вебресурсів. Окремо досліджено проблеми традиційного вебскрапінгу, зокрема обробку динамічного контенту, механізми захисту сайтів від скрапінгу та обробку неструктурованої інформації. Загальний висновок підкреслює значний потенціал AI у підвищенні ефективності збирання та аналізу вебданих.

У роботі [6] проаналізовано передові методи вебскрапінгу, інтегровані з штучним інтелектом, що дають змогу значно підвищити точність і ефективність збирання інформації з динамічних вебресурсів. Автори детально розглядають застосування NLP для вилучення основних даних, використання моделей LSTM (Long Short-Term Memory) і 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 і обходити механізми захисту від скрапінгу.

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

$$ADCM = \{DCM, DPSM, AnM, SCI\}, \quad (1)$$

де DCM (data collection module) – модуль збирання 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 і системою збирання даних, проте для вирішення більш складних завдань вона не може бути використана і потребує розширення. Наприклад, простота базової моделі не враховує специфічні нюанси інтеграції AI-алгоритмів із традиційними методами обробки даних, що може впливати на стабільність роботи системи та обмежувати можливості її масштабування.

Розроблення 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 головну роль відіграє архітектура моделі, оскільки саме вона визначає логіку реалізації функціональних компонентів, встановлює часові рамки розроблення та формує орієнтовну структуру витрат. Особливої актуальності набуває застосування динамічної архітектури, яка забезпечує гнучкість і масштабованість рішення. Це у свою чергу суттєво

підвищує продуктивність і адаптивність моделі, що є критично важливим у контексті інтеграції модулів вебскрапінгу з сучасними технологіями штучного інтелекту.

Мета дослідження

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 і підвищенню продуктивності за рахунок використання розподілених обчислень.

Викладення основного матеріалу

На основі аналізу сучасних підходів для автоматизованого збирання даних із вебресурсів було виявлено, що чимало робіт зосереджено або на ручному створенні правил для створення скраперів, або на інтерактивних мовних агентах. Такий поділ спричиняє обмеження в масштабуванні, адже традиційні «обгортки» потребують постійного ручного доопрацювання, а агентно-орієнтовані рішення можуть бути надмірно ресурсомісткими і не завжди забезпечують повторне використання створених рішень.

Для вирішення цієї проблеми запропоновано інтегрувати великі мовні моделі (Large Language Models, LLM) в окремі програмні модулі для прискорення процесів автоматичного створення правил збирання даних, суттєвого підвищення рівня автоматизації та покращення автономності роботи всієї моделі. LLM – це генеративні моделі штучного інтелекту, які навчаються на великому обсязі текстових даних і здатні генерувати текст, програмний код чи правила вилучення інформації на основі вхідних запитів (промптів). Завдяки здатності аналізувати контекст і виявляти складні патерни, LLM демонструють високу точність у створенні алгоритмів і правил для автоматичного вилучення інформації. Якість згенерованого коду та правил дає змогу суттєво знизити кількість помилок, підвищити стабільність роботи і ефективність обробки вебданих. Вибір оптимальної LLM моделі забезпечує максимальну продуктивність і автономність системи, що робить інтеграцію таких моделей перспективним та ефективним рішенням для автоматизації процесів збирання даних.

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

$$ADCM = \{CE, AM, RM, DCM, DPSM, AnM, SCI\}, \quad (2)$$

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 з моделі (1) зазнали суттєвих змін. Наприклад, модуль аналітики, який у попередній версії забезпечував лише перегляд інформації, тепер розширено, щоб давати змогу налаштовувати Python-скрипти для обробки даних і генерувати графічне подання зібраної інформації, що значно підвищує аналітичний потенціал системи. Аналогічно модуль збирання 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 дає змогу працювати з більш складними структурами інформації.

По-друге, до моделі було додано нові модулі, яких не було в початковій версії, що значно розширює її функціональність. Зокрема, додано модуль керуючого елемента (CE), який є центральною ланкою координації між усіма іншими модулями та забезпечує більш ефективний розподіл завдань. Також впроваджено модуль аналізу сайту (AM), завдання якого полягає в попередньому аналізі вебсторінок із метою ідентифікації релевантного контенту, що дає змогу 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 збирання інформації. Нові модулі дають моделі змогу не лише оперативно адаптуватися до змін у структурі вебресурсів, забезпечуючи більшу гнучкість, ефективність і масштабованість системи порівняно з першою версією, але і значно підвищують рівень автоматизації всієї моделі, впроваджуючи автоматичне оновлення правил збирання даних, моніторинг ефективності збирання та автоматичного корегування процесів вилучення інформації, що мінімізує потребу ручного втручання та забезпечує безперервну роботу моделі.

Порівняно з традиційними рішеннями запропонована модель дає змогу організувати подієво-орієнтовану систему, яка забезпечує автоматичний аналіз сайтів і виявлення контентних структур на них, автоматизоване вилучення і структурування вебданих, асинхронну взаємодію між модулями через чергу повідомлень, динамічну

генерацію правил збирання 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 і адаптивність роботи системи, гарантуючи узгоджену взаємодію між усіма елементами.

CE складається з кількох основних частин, кожна з яких використовує критично важливий функціонал для роботи всієї інфраструктури. Керуючий елемент побудований на множині складових

$$CE = \{MPS, CM, MS, LS, MS, CS\}, \quad (3)$$

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 основним елементом комунікаційної інфраструктури системи. Він забезпечує асинхронний обмін даними між компонентами за допомогою механізму черг повідомлень RabbitMQ. Використовуючи підхід подієво-орієнтованої архітектури, система підписана на такі події, як «виявлено новий сайт», «завершено збирання даних» або «помилка з отриманням 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, і реалізує механізми автоматичного перезапуску процесів у разі збоїв. Координація виконання завдань відбувається через систему черг повідомлень, що дає змогу мінімізувати затримки і оптимізувати використання ресурсів. Цей механізм контролює інтерфейс налаштування системи (SCI), який дає змогу задавати правила виконання операцій, налаштовувати пріоритети і 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 забезпечено коректне виконання паралельних процесів і стабільність функціонування системи.

Сервіс логування (LS) відповідає за збирання, зберігання та аналіз історії подій у системі. Він інтегрований із централізованою системою логування Grafana Loki, щоб 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 дає змогу швидко реагувати на збої та покращувати продуктивність системи.

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

Сервіс конфігурації (CS) забезпечує управління параметрами системи, включаючи визначення списку вебресурсів для збирання даних, налаштування шаблонів парсингу та обмеження для кожного джерела. Конфігураційні файли зберігають 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 збирання даних і налаштовувати кількість скраперів, що одночасно працюють.

Керуючий центр виступає головною ланкою між усіма компонентами системи, забезпечуючи їхню безперебійну та узгоджену роботу. Завдяки подієво-орієнтованій архітектурі та розподіленій обробці повідомлень він дає змогу легко масштабувати систему, швидко адаптувати її до змін у структурі вебресурсів і гарантувати стабільність її функціонування навіть у високонавантажених середовищах.

Модуль аналізу сайту (AM) є основним елементом процесу автоматизованого збирання даних, що забезпечує проходження по всіх сторінках вебресурсу, ідентифікацію контентних сторінок і

відправлення цієї інформації для подальшої обробки. Його основна функція – визначення, які сторінки містять корисний контент, а які є технічними або інформаційними. Структурно він складається з таких основних компонентів:

$$AM = \{SC, FM, ClS, RTM\}, \quad (4)$$

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 передавання результатів.

Механізм обходу сайту (SC) забезпечує ітеративне сканування вебресурсу, переходячи за посиланнями та формуючи карту сторінок. Для цього використано бібліотеку Scrapy, яка дає змогу виконувати швидкий обхід у багатопотоковому режимі. Важливим аспектом є визначення структури сайту і 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 дає змогу оптимізувати обчислювальні ресурси та зменшити кількість запитів до LLM.

Сервіс класифікації сторінок (ClS) використовує LLM для аналізу вмісту сторінки та визначення її типу. Для аналізу застосовують заздалегідь підготовлені prompts (інструкції для мовної моделі), щоб модель могла оцінювати, чи є сторінка контентною або вона містить лише технічну інформацію. На основі цього визначення формують список сторінок, які можуть бути використані для подальшого збору даних.

Механізм передавання 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 і генерації правил збирання 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 і систему фільтрації унікальних сторінок, він забезпечує швидке і 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 і генерує правила на основі аналізу їхнього DOM-дерева, використовуючи інтеграцію з великими мовними моделями (LLM). RM забезпечує масштабованість і повторне використання правил, а також адаптацію до змін у структурі вебсайтів:

$$RM = \{RCM, GAM, RVS, RURS\}, \quad (5)$$

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 і 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 застосовано адаптивні алгоритми, що дають змогу 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 фактора на точність отриманих результатів. Під час дослідження 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 і швидке навчання на мінімальній кількості промптів. Щоб забезпечити точність і гнучкість вилучення даних, використовують спеціальні правила (промпти) для LLM. Вони не лише налаштовують LLM для генерації XPath, але й оцінюють їхню ефективність, порівнюючи з уже 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 і самостійно ідентифікувати патерни та залежності у DOM-структурі сторінок, що забезпечує високу адаптивність до змін структури вебсайтів.

Процес генерації XPath заснований на підході багатоступінчастої побудови правил: спочатку створюють 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 корегування параметрів. Це дає змогу покроково тестувати кожен етап побудови XPath-запитів, виявляти вузькі місця в логіці генерації та адаптувати правила до змін у структурі вебсторінки. Такий підхід дає змогу мінімізувати випадкові помилки та підвищити стабільність роботи скрапера навіть зі зміною 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 перевіряють узгодженість згенерованих правил із реальними вебсторінками. Використовуючи набір тестових сторінок, система перевіряє правильність знаходження контенту, визначаючи, чи виділено потрібні елементи. Якщо виявлено некоректні елементи, система корегує правила перед подальшим використанням. Тут важливу роль також відіграє генеративний штучний інтелект, який дає змогу автоматизовано аналізувати результати тестування та класифікувати помилки за типами, що спрощує їх подальше виправлення.

Після успішного аналізу виконують валідацію правил за допомогою відповідного сервісу (RVS), де тестують правила на ширшому наборі сторінок сайту. Це забезпечує генерацію універсальних правил, які можна застосовувати для всіх подібних сторінок. Додатково перевіряють на стабільність і 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 перевіряють за тим самим алгоритмом і в разі успішного проходження тестів зберігають у модулі збереження даних (DPSM) для подальшого використання.

Механізм створення правил є критично важливим компонентом системи збирання даних, оскільки дає змогу автоматизувати адаптацію скрапінгових правил і забезпечує високу ефективність вилучення даних навіть у динамічних вебсередовищах.

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

$$DCM = \{REM, MM, DGS, DTM\}, \quad (6)$$

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 передавання даних.

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

Механізм багатопотокової обробки (MM) забезпечує паралельне збирання 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 зберігають у проміжному форматі (JSON), після чого передають на подальшу обробку.

Передавання даних здійснюється через RabbitMQ, що забезпечує ефективний обмін

інформацією між DCM і DPSM. У разі виявлення помилок або змін у структурі сайту DCM генерує відповідний івент і надсилає його до CE, який перенаправляє його в RM для оновлення правил.

Отже, DCM відіграє важливу роль у процесі автоматизованого збирання даних, забезпечуючи точність, адаптивність і ефективну інтеграцію з іншими модулями системи. Він виконує функції обходу, вилучення та передавання 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, використовуваних у процесі збирання та аналізу вебконтенту, включаючи правила вилучення, унікальні сторінки, метаінформацію вебресурсів і сам контент. DPSM виконує критично важливу роль у підтримці цілісності даних і забезпечує швидкий доступ до необхідної інформації для інших модулів системи. Основною базою даних для DPSM є MongoDB, оскільки вона забезпечує гнучкість, масштабованість і ефективну обробку неструктурованих даних. Використання NoSQL-бази виправдане через динамічний характер збереження інформації, зокрема правил вилучення, структур вебсторінок і контентних даних. MongoDB дає змогу 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, щоб підвищити продуктивність системи. DPSM складається з таких компонентів:

$$DPSM = \{DClM, NM, TS, SCM, AdM, CM, IS\}, \quad (7)$$

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

NM (normalization mechanism) – механізм нормалізації форматів;

TS (transfer service) – сервіс передавання 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 і некоректних записів. Він використовує алгоритми фільтрації, регулярні вирази та автоматичне виправлення помилок у структурі даних, що підвищує їхню якість перед збереженням.

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

Сервіс передавання даних (TS) здійснює комунікацію з іншими модулями через керуючий елемент (CE). DPSM отримує структуровані дані, обробляє їх і 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. Зі створенням схеми враховують типи отриманих даних, їхню ієрархію та можливість динамічного розширення, що забезпечує масштабованість системи без необхідності ручного оновлення.

Механізм адаптації структури (AdM) оновлює схему бази з виявленням нових атрибутів у вхідних даних, що дає змогу підтримувати гнучкість і розширюваність системи, адаптуючи її до зміни вхідного 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, дає змогу аналізувати інформацію в реальному часі, забезпечуючи її доступність для аналітичних модулів.

Отже, DPSM виконує повний цикл обробки, нормалізації та збереження даних, забезпечуючи їхню чистоту, узгодженість і ефективну інтеграцію в загальну систему збирання 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 і візуалізації даних через бібліотеки Python. Основне завдання 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 таких компонентів:

$$AnM = \{DQM, SM, DVS, GS\}, \quad (8)$$

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 вибраного вебресурсу, після чого надсилає запит на отримання конкретних значень для подальшої обробки, що дає змогу модулю формувати коректні запити на аналітику.

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

Сервіс візуалізації даних (DVS) відповідає за обробку результатів аналізу та побудову графіків, діаграм і таблиць. Для цього використовують бібліотеку Matplotlib та інші інструменти візуалізації Python. Отримані візуальні зображення формують відповідно до запиту користувача.

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

Отже, модуль аналітики забезпечує повний цикл: від отримання даних до їх обробки та формування PDF-звітів. Його інтеграція з LLM дає змогу динамічно адаптувати запити і 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 збирання даних, контроль ресурсів і генерація аналітичних звітів на основі отриманої інформації. SCI складається з таких компонентів:

$$SCI = \{PCM, CM, RGM\}, \quad (9)$$

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

CM (configuration mechanism) – механізм конфігурації обробників;

RGM (report generation mechanism) – механізм генерації звітів.

Механізм управління обробкою (PCM) відповідає за додавання нових сайтів у процес збирання 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) дає змогу задавати кількість активних процесів збору даних, встановлювати ліміти на використання ресурсів та управляти параметрами виконання завдань. Це забезпечує гнучкість і 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 в текстовому форматі, після чого система автоматично формує відповідний звіт.

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

Результати та їх обговорення

Розглянемо декілька прикладів використання великих мовних моделей LLM для підвищення ефективності вебскрапінгу.

Приклад 1: збирання даних про товари з п’яти великих e-commerce платформ. З використанням традиційного підходу процес потребує значних ресурсів: зазвичай задіяні два розробники, які витрачають близько трьох днів на написання та налагодження аналітичного аналізатора (парсера). Такий підхід передбачає створення понад тисячі рядків коду мовою Python із використанням інструментів типу Scrapy і XPath. За будь-якої зміни в структурі хоча б одного сайту необхідне ручне доопрацювання, що робить процес трудомістким і таким, що слабко масштабований.

Альтернативний шлях із застосуванням моделей LLM дає змогу радикально спростити завдання. Замість команди розробників достатньо одного фахівця, який за один день за допомогою Python-скрипту та виклику моделі LLM може обробити HTML-сторінки та отримати потрібну інформацію. При цьому промт легко адаптований до всіх п'яти сайтів без необхідності змінювати код. Такий підхід скорочує трудомісткість до 70 % та значно знижує технічний поріг: тепер цю роботу може виконати не розробник, а аналітик.

Приклад 2: класифікація відгуків користувача за тональністю є однією із найчастіших завдань в e-commerce та клієнтській аналітиці. Традиційно його вирішують за допомогою машинного навчання: команда з одного-двох ML-інженерів витрачає близько п'яти днів на збирання та розмітку навчальної вибірки, попередню обробку тексту, навчання моделі (наприклад на базі scikit-learn або Hugging Face) і налаштування всього пайплайну. Такий підхід потребує наявності якісних даних, часу на експерименти з гіперпараметрами, а зі зміною предметної сфери – повного перенавчання моделі.

Використання LLM (наприклад GPT-4) робить процес простішим і швидшим. Один спеціаліст із базовими навичками Python за день може реалізувати скрипт, який надсилає текст відгуків до LLM із заздалегідь підготовленим промтом. Така схема не потребує навчання моделі та адаптується до різних тематик товарів без додаткового налаштування, що дає змогу скоротити витрати часу до 80 % і виконати завдання силами не інженерів, а маркетолога чи аналітика.

Приклад 3: завдання вилучення основних даних (назва компанії, дата, сума та ІПН із сканів договорів та актів) традиційно потребує чималих ресурсів. Зазвичай до роботи залучені двоє розробників і аналітик, а весь процес займає від тижня до десяти днів. Використовують інструменти для OCR (наприклад Tesseract), після цього набір регулярних виразів і кастомних шаблонів під конкретні формати документів. Для будь-якої зміни структури документа потрібне ручне доопрацювання коду.

З використанням LLM, таких як GPT-4, завдання вирішують швидше та простіше. Один спеціаліст, навіть без глибоких навичок програмування, може за один-два дні вирішити це завдання: текст документів розпізнають через стандартний OCR, потім передають до GPT-моделі з універсальним промтом. Використовуючи таке рішення, немає необхідності писати складні правила або регулярні вирази, тому що модель сама адаптується до різних форматів і стилів оформлення документів. У результаті досягають близько 75 % економії часу, знижується технічний поріг із рівня команди розробників до рівня офіс-аналітика чи юриста.

У таблиці та на рисунку наведено порівняння традиційного вебскрапінгу (синім кольором) і вебскрапінгу з використанням мовних моделей LLM (помаранчевим кольором) за основними метриками: швидкість налаштування, стійкість, точність, обсяг коду та загальний час.

Таблиця

Порівняння підходів вебскрапінгу

Метрика порівняння	Підхід	
	традиційний	з LLM
1. Час налаштування, год	3	0,5
2. Частота поломок, %	40	10
3. Точність вилучення, %	85	97
4. Код, рядки	300	40
5. Загальний час, год	40	12

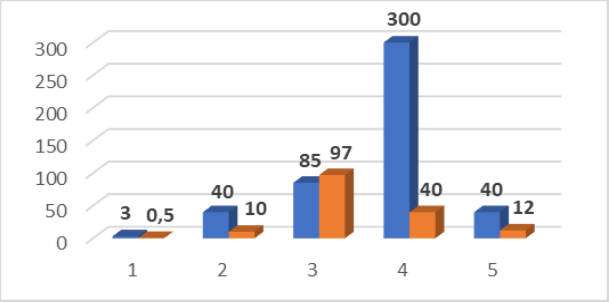

Рис.

Наведені приклади доводять, що використання мовних моделей LLM радикально спрощують і прискорюють рутинні завдання, для яких раніше потрібна була участь розробників або фахівців із машинного навчання. Загальна вигода від використання LLM є очевидною: економія часу – до 70-80 %, зниження технічного порогу з рівня розробника до рівня аналітика чи предметного експерта, підвищення гнучкості за рахунок використання промтів, які працюють відразу для різних вхідних даних без необхідності доопрацювання коду. Усі ці переваги відкривають нові можливості для автоматизації бізнес-процесів у тих сферах, де раніше були потрібні ресурсоємні технічні рішення.

Висновки

Проведений аналіз показав, що запропонована модульна архітектура вебскрапінгової системи значно перевершує сучасні рішення за основними параметрами: ефективністю, масштабованістю і адаптивністю. Використання подієво-орієнтованого підходу забезпечує асинхронну взаємодію між компонентами моделі, мінімізуючи затримки та підтримуючи безперервну роботу системи.

Основною перевагою запропонованої моделі порівняно з попередньою (1) є автоматизація процесу створення скраперів завдяки інтеграції великих мовних моделей (LLM), що дає змогу швидко генерувати правила для вилучення інформації, суттєво знижуючи потребу в ручному втручанні. Для підтвердження цього було проведено експеримент, у якому проаналізовано набір із 10 вебмагазинів. У рамках цього експерименту встановлено, що використання традиційної моделі, яка передбачає ручне розроблення скраперів, потребувало в середньому близько 8-12 годин роботи одного спеціаліста для кожного сайту. Натомість запропонована модель дала змогу автоматично генерувати повністю функціональні правила для збирання аналогічної інформації за 20-30 хвилин, що скорочує час розроблення більш ніж у 16 разів.

Крім швидкості, якість і точність зібраних даних у запропонованій моделі також суттєво покращені завдяки адаптивності до змін у структурі вебсторінок. Це забезпечують механізми автоматичної генерації та оновлення XPath-запитів, що дає змогу оперативно реагувати на зміни структури контенту. Використання MongoDB як масштабованої бази даних підвищує швидкість доступу до збережених даних, даючи змогу ефективно працювати з великими обсягами інформації.

Отже, запропонована модель не лише скорочує витрати часу та ресурсів на розроблення скраперів, але й забезпечує високу якість, гнучкість і стабільність у роботі. Вона може бути ефективно застосована в широкому спектрі завдань, включаючи маркетингову аналітику, моніторинг конкурентів, дослідження штучного інтелекту та інші галузі, для яких потрібне швидкі та точні збирання структурованих вебданих.

Список використаних джерел

1. 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. *Матеріали X Міжнар. наук.-практ. конф. «Інформатика. Культура. Техніка». Одеса: Національний університет «Одеська політехніка», Інститут комп'ютерних систем*. 2024. Т. 1, № 1. С. 62-68. doi: https://doi.org/10.15276/ict.01.2024.08.

2. Lotfi C., Srinivasan S., Ertz M., Latrous I. Web Scraping Techniques and Applications: A Literature Review. *SCRS Conference Proceedings on Intelligent Systems*. 2021. P. 381-394. doi: https://doi.org/10.52458/978-93-91842-08-6-381.

3. 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. Vol. 9, Iss. 3. P. b383-b389. URL: https://www.ijnrd.org/papers/IJNRD2403143.pdf.

4. Bhute T., Raut S., Kannake S., Guda B., Sheikh M. Web Scraping and Automation. *International Journal of Creative Research Thoughts*. 2024. Vol. 12, Iss. 5. P. 285-290.

5. 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.

6. Huang C.-J. The Synergy of Automated Pipelines with Prompt Engineering and Generative AI in Web Crawling. 2024.

doi:10.48550/arXiv.2502.15691. URL: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/ https://www.arxiv.org/pdf/2502.15691.

T. Filimonchuk, Y. Kopailo, S. Partyka, O. Sevostianova

Model of automated data collection from web resources based on generative artificial intelligence

Abstract. *The relevance of the research lies in the fact that, under the current conditions of rapid digitalization, global internet development, and the continuous growth of data volumes obtained from various web resources, there is an increasing need to develop efficient tools for automated data collection, analysis, and structuring. Web scraping has become an integral part of information support processes across many industries. However, traditional data collection methods often require complex manual configurations, which complicate system scalability and hinder timely adaptation to changes in website structures. Therefore, the use of universal rules and algorithms for building web scrapers is becoming particularly relevant. A key role in this process is played by modern large language models (LLMs), which are capable of generating adaptive solutions, particularly in the form of dynamic XPath expressions used to efficiently extract relevant data from the HTML structure of web pages. This approach enables the creation of scalable and self-learning web data collection systems that can adapt to various content formats and quickly respond to changes in web interfaces.* ***The object of the study*** *is the application of large language models (LLMs) in developing automated mechanisms for web data collection, as well as exploring their potential in generating high-precision XPath queries.* ***The subject of the study*** *is a model focused on web data collection that involves multi-level interaction between independent functional modules and ensures their integration through unified interfaces and standardized data exchange formats.* ***Results.*** *The proposed extended modular model of the web data collection system covers all key stages of web content processing: from identifying relevant information sources, through generating and optimizing XPath queries, to storing the extracted data in a scalable, structured database. The proposed architectural solution ensures high adaptability to changes, system stability, and minimized technical support costs.* ***Conclusions.*** *The use of generative language models for automating the creation of web scrapers opens up new possibilities for flexible, accurate, and stable web data collection. This approach facilitates the effective integration of such systems into dynamic web environments, reducing both time and financial costs associated with maintaining information and analytical platforms.*

Keywords: *generative artificial intelligence, web data collection, automation, modular architecture, database, scrapers, generative model, large language model.*

Філімончук Тетяна Володимирівна*, кандидат технічних наук, доцент кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна. E-mail: tetiana.filimonchuk@nure.ua. ORCID: 0000-0002-4380-504X.*

Копайло Ярослав Русланович*, магістрант, Харківський національний університет радіоелектроніки, Харків, Україна. E-mail: yaroslav.kopailo@nure.ua.*

Партика Станіслав Олександрович*, старший викладач кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна. E-mail: stanislav.partyka@nure.ua. ORCID: 0000-0002-7376-8980.*

Севостьянова Олена Миколаївна, *старший викладач, Харківський національний університет радіоелектроніки, Харків, Україна. E-mail: olena.sevostianova@nure.ua. ORCID: 0009-0008-2595-5133.*

Tetiana Filimonchuk*, PhD, Associate Professor of the Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine. E-mail: tetiana.filimonchuk@nure.ua. ORCID: 0000-0002-4380-504X.*

Yaroslav Kopailo*, master's student, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine. E-mail: yaroslav.kopailo@nure.ua.*

Stanislav Partyka*, Senior Lecturer of the Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine. E-mail: stanislav.partyka@nure.ua. ORCID: 0000-0002-7376-8980.*

Olena Sevostianova, *Senior lecturer, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine. E-mail: olena.sevostianova@nure.ua. ORCID: 0009-0008-2595-5133.*