УДК 004.94 **doi: 10.26906/SUNZ.2024.3.130**

А. А. Кочина, К. Н. Азімов

Харківський національний автомобільно-дорожній університет, Харків,Україна

РОЗРОБКА СИСТЕМИ АВТОМАТИЗОВАНОГО ПАРСИНГУ РАЗОВИХ ЗАМОВЛЕНЬ НА ВАНТАЖНІ ПЕРЕВЕЗЕННЯ У МІЖМІСЬКОМУ СПОЛУЧЕННІ

Анотація. Ефективне переміщення товарів на великі відстані є основою сучасної торгівлі. Автоматичні механізми збору даних стимулюють розвиток системи, яка збирає і оновлює дані в режимі реального часу, використовуючи такі технології, як аналіз даних, сканування даних і сенсорні мережі для створення динамічної платформи швидкого реагування. Цей підхід, заснований на даних, має величезний потенціал як для вантажовідправників, так і для перевізників. **Мета.** Визначення теоретичних основ автоматизованого збору даних для розробки парсингової системи для автоматизованого пошуку та обробки разових замовлень на міжміські вантажні перевезення. **Методологія**. Статистичне та імітаційне моделювання. **Постановка задачі** - виявлення та оцінка потенційних замовлень на онлайн-платформі DELLA, інтеграція в бізнес-процеси компанії, пов'язані з обробкою та аналізом інформації. **Результати**. У статті проаналізовано теоретичні основи розробки програмного забезпечення для автоматизованого збору вхідних даних. Розроблено автоматизовану систему парсингу для пошуку та обробки разових замовлень на міжміські вантажні перевезення. Розробка автоматизованої системи парсингу дозволила отримати матрицю разових замовлень на регіональному рівні України. **Висновки:** Для моделювання потоку разових замовлень запропоновано використовувати методи машинного навчання для створення парсеру даних, який структурує інформацію веб-сторінки та надає її у вигляді, готовому для аналізу. Пошук інформації здійснюється на найпопулярнішому логістичному сайті серед бірж вантажних перевезень в Україні DELLA, який має найбільшу цільову аудиторію в секторі автомобільних перевезень. **Оригінальність.** На основі аналізу програмного забезпечення для автоматизованого збору даних створено парсингову систему для автоматизованого пошуку та обробки разових замовлень на міжміські вантажні перевезення у міжміському сполученні. **Практична цінність**. В результаті обробки масиву даних сформовано матрицю разових замовлень, яка визначає попит на ринку міжміських автомобільних перевезень України.

Ключові слова: парсинг, попит, разові замовлення, системи автоматизованого пошуку даних.

Вступ

Обробка та аналіз даних передбачає вивчення передових методів очищення, структурування та аналізу цих наборів даних. Механізми автоматичного збору спонукають до розробки системи, яка збирає й оновлює дані в режимі реального часу, використовуючи такі технології, як парсинг даних, сканування даних і сенсорні мережі, для створення динамічної платформи швидкого реагування. Даний підхід на основі даних має величезний потенціал як для вантажовідправників, так і для перевізників. Невеликі компанії отримують доступ до більшої кількості перевізників, оптимізованих моделей ціноутворення та відстеження відправлень у реальному часі. У результаті транспортні компанії отримують вигоду від підвищення ефективності роботи, зменшення кількості порожніх пробігів і можливості задовольнити раніше невикористаний сегмент ринку.

Аналіз публікацій. Парсинг – це автоматизований збір та структурування інформації з сайтів за допомогою програми чи сервісу. Парсинг зазвичай застосовують, коли потрібно швидко зібрати великий обсяг даних. Область застосування парсингу можна звести до двох цілей [1]:

– аналіз конкурентів, щоб краще розуміти, як вони працюють, та запозичувати у них якісь підходи;

– аналіз власного майданчика для усунення помилок, швидкого впровадження змін тощо.

Під системою автоматизованого парсингу можна розуміти програмне забезпечення (ПЗ), яке використовується для збирання даних з різних джерел.

Для створення такого ПЗ існує два підходи.

Один з них передбачає використання готових систем парсингу з відкритим кодом, таких як Scrapy або Beautiful Soup і побудову на їх основі власного парсеру. Як правило, ці системи можуть легко адаптуватися до конкретних завдань.

Другий підхід пропонує розробку власного парсеру за допомогою мов програмування, таких як Python або Java. Це дозволяє мати повний контроль за процесом парсингу та збору даних.

У порівнянні з ручним збиранням даних, парсинг дозволяє прискорити цей процес у десятки разів. Це відкриває нові можливості для дослідників та експертів. У порівнянні з ручним збиранням інформації, парсинг менш схильний до помилок. Але якщо було допущено неточності на етапі проектування (логічну помилку), це може призвести до спотворення даних та їх знецінення.

Мета та постановка задачі. Метою статті є визначення теоретичних основи розробки програмного забезпечення для автоматизованого збору даних. Розробка систем парсингу для автоматизованого пошуку та обробки разових замовлень на міжміські вантажні перевезення в міжміську сполученні. Постановка задачі – виявлення та оцінка потенційних замовлень на онлайн-майданчику DELLA, інтеграція в бізнес-процеси підприємства, пов’язані з обробкою та аналізом інформації.

Виклад основного матеріалу

Методика парсингу даних розробляється з метою автоматизації процесу пошуку разових замовлень на автомобільні міжміські перевезення. Враховуючи випадковий характер надходження разових

замовлень, процес їх обслуговування має певну складність при управлінні ланцюгами постачань. Адже цей фактор вимагає від перевізників постійного пошуку таких замовлень і швидкого реагування на їх появу. Крім того перевізник має обрати правильну для себе стратегію поведінки, а саме використовувати заздалегідь розроблені та обґрунтовані варіанти управлінських рішень або посилатись винятково на власний досвід [2].

Для полегшення даного процесу було запропоновано розробити програмний продукт, що значно скоротить час на аналіз про надходження разових замовлень, спростить його і, як результат, полегшить процес обслуговування замовлень.

Основну роль в даному процесі відіграє збирання вхідних даних.

Первинні дані – це дані, які дослідник збирає вперше. Зауважимо, що на відміну від терміну первинні дані, вхідні дані є більш загальним поняттям і використовується для опису функціонуючих систем. Так, для початку роботи програми необхідна наявність первинних даних. При цьому результати роботи програмного продукту можуть бути вхідними даними на наступних циклах її роботи, доки не виконаються умови, що дозволять завершити виконання програми.

Метод парсингу має кілька етапів для витягнення потрібної інформації із наукових сайтів:

а) збір контенту. Пошукова машина парсинг завантажує код сторінки сайту, під час такого процесу до нього долучається спеціальний скрипт, який розбиває весь код на частини та аналізує інформацію яка потрібна досліднику;

б) витяг інформації. Вся інформація, яка витягується зі сторінки, може бути не потрібна користувачеві, адже його цікавить тільки конкретне, наприклад, розділи статей, наукових робіт інших авторів в яких він може знайти для себе корисні дані та використати їх у своєму науковому дослідженні. Парсер буде знаходити ті сторінки чи розділи, де розміщена інформація про дану категорію, витягнувши все в кінцевий файл тільки тексти цих даних;

в) збереження результатів. Коли вся потрібна інформація витягнута з сайту, її потрібно зберегти. Зазвичай вносяться записи в базу даних, адже так зручніше аналізувати інформацію з даних інших наукових робіт.

Запропонована методика дасть можливість дослідити та проаналізувати отриману інформацію, чи потрібна вона дослідникові для внесення її у свою роботу. Для збору даних було запропоновано парсер для зазначеного веб-сайту за допомогою мови програмування Python та сторонніх бібліотек. В якості джерела інформації було обрано дані про вантажні заявки з веб-сайту della.ua.

Додаток був написаний з використанням мови програмування Python. Даний вибір пов’язаний з її популярністю в академічній сфері [3]. Python – це потужна мова програмування, яка проста у вивченні. Він має ефективні структури даних високого рівня та простий, але ефективний підхід до об’єктно-орієнтованого програмування. Елегантний синтаксис і динамічна типізація Python разом із інтерпретованою природою роблять його ідеальною мовою для сценаріїв і швидкої розробки додатків у багатьох галузях для поширених платформ.

Інтерпретатор Python і обширна стандартна бібліотека доступні у вихідному або двійковому вигляді для всіх основних платформ на веб-сайті Python. Інтерпретатор Python легко розширюється за допомогою нових функцій і типів даних, реалізованих у C або C++ (або інших мовах, похідних від C). Python також підходить як мова розширення для настроюваних програм.

Головною перевагою Python є те, що він йде в пакеті з дуже великою бібліотекою стандартних пакетів (тобто не треба їх додатково інсталювати, щоб створювати навіть досить складні проєкти) та ще більшим списком пакетів Open Source, доступних в публічних репозиторіях (для легкого встановлення). У тому числі потужні пакети для обробки зображення, створення графіків, статистичного аналізу даних, мережевої комунікації, штучного інтелекту (особливо Deep Learning) та Machine Learning, опрацювання натуральної мови, взаємодії з базами даних, створення вебдодатків, графічних алгоритмів, витягування даних з вебсайтів [4]. За замовчуванням у Python доступна стандартна бібліотека, яка містить велику кількість функцій, що багаторазово використовуються. Крім того, доступно більше 137 000 бібліотек Python для різних завдань, серед яких інтернет-розробка, наука про дані та машинне навчання (ML).

Розглянемо найбільш популярні бібліотеки, що дозволяють спростити процес написання програмного продукту: Pandas, Requests та Beatifulsoup. Вони ліцензовані BSD-ліцензію або містять її розділи. Ліцензія BSD є дозвільною, тобто вона накладає мінімальні обмеження на використання та розповсюдження ліцензійного програмного забезпечення. Основною вимогою ліцензії BSD є те, що будь-яке поширення програмного забезпечення має включати копію ліцензії та відмову від відповідальності. Багато розробників програмного забезпечення та компанії використовують цю ліцензію для того, щоб їхня робота була доступна широкому колу користувачів при збереженні прав на програмне забезпечення.

Для роботи з бібліотеками та зчитування інформації з веб-сайту необхідно розуміти як комп’ютер завантажує веб-сторінки та отримує інформацію від сервера. Коли користувач відвідує сторінку в Інтернеті, комп'ютер використовує протокол передачі гіпертексту (HTTP) для завантаження цієї сторінки.

Принцип роботи HTTP протоколу можна описати наступним чином. Коли ми хочемо переглянути веб-сторінку, ми можемо використовувати різні типи девайсів: ноутбук, стаціонарний комп'ютер або телефон. Головне, щоб на пристрої була програма браузера. Користувач або вводить уніфікований покажчик ресурсу (URL) у пошуковий рядок браузера, або переходить за посиланням з відкритої сторінки. URL-адреса починається з HTTP. Це сигнал браузеру, що йому необхідно використовувати протокол HTTP для отримання документа за цією адресою. Зазвичай IP-адреси містять зручні та читабельні для людини

назви доменів, наприклад highload.today або wikipedia.org.

На другому кроці браузер шукає потрібну IP-адресу. Для цього браузер відправляє запит до DNS-сервера для зіставлення імені домену з його IP-адресою. Як тільки браузер визначив IP-адресу комп'ютера, на якому розміщений запит URL, він відправляє HTTP-запит. На четвертому корці, коли хост-комп'ютер отримує HTTP-запит, він відправляє клієнту відповідь із змістом та метаданими. Після виконання всіх кроків, браузер, отримує всю необхідну інформацію для відображення запитаного документа, відтворює його на екрані користувача.

Ознайомившись з принципом роботи веб-сторінки доцільно розглянути способи збереження та відтворення інформації. Зберігання даних – це спосіб організації даних на носіях інформації для її подальшої обробки. Найбільш поширеними типами даних є текстові, графічні, аудіо та відео дані. У науковій сфері найбільш поширеним є текстовий формат зберігання даних.

Текстові формати - це формати зберігання даних, які використовуються для зберігання тексту. Вони можуть бути простими або форматованими.

До найпоширеніших текстових форматів можна віднести:

– TXT простий текстовий формат, який не підтримує форматування. Він є стандартним текстовим форматом, який підтримується більшістю програм;

– DOC текстовий формат, який підтримує форматування, створений Microsoft Word. Він є популярним форматом для зберігання документів, таких як листи, звіти та презентації;

– DOCX текстовий формат, який підтримує форматування, сумісний з Microsoft Word 2007 і пізнішими версіями. Він є покращеною версією формату DOC;

– RTF текстовий формат, який підтримує форматування, створений Microsoft. Він є універсальним текстовим форматом, який підтримується більшістю програм;

– ODT текстовий формат, створений OpenDocument для ОС Linux. Він є відкритим форматом, який підтримується більшістю офісних програм.

Прості текстові формати, такі як TXT, мають ряд переваг. Вони є простими у використанні, а їх легко відкрити та редагувати в будь-якій програмі, яка підтримує текстові формати. Однак вони не дозволяють зберігати форматування тексту, що може бути недоліком для деяких завдань.

Форматовані текстові формати, такі як DOC, DOCX та RTF, дозволяють зберігати форматування тексту, що може бути корисним для завдань, які вимагають форматування, наприклад, для створення документів, таких як листи, звіти та презентації. Однак вони можуть бути більш складними у використанні, ніж прості текстові формати.

Вибір текстового формату залежить від конкретних потреб. Для простих завдань, таких як зберігання тексту без форматування, можна використовувати простий текстовий формат, такий як TXT. Для завдань, які вимагають форматування, можна використовувати форматований текстовий формат, такий як DOC, DOCX або RTF.

Серед текстових форматів розрізняють такі, що мають деякі особливості, які відрізняють його від інших. До них відносяться:

– HTML (HyperText Markup Language) використовується для створення веб-сторінок. HTML-файли складаються з тегів, які описують структуру та вигляд веб-сторінки;

– XML (Extensible Markup Language) використовується для зберігання структурованих даних. XML-файли складаються з елементів, які описують структуру даних;

– JSON (JavaScript Object Notation) використовується для зберігання об'єктів JavaScript. JSON-файли складаються з пар ключ-значення, які описують властивості об'єкта;

– YAML (YAML Ain't Markup Language) також використовується для зберігання структурованих даних. На відміну від XML, YAML-файли складаються з ключів, значень, списків та інших елементів, які описують структуру даних;

– INI (Initialization File) використовується для зберігання конфігураційних даних. INI-файли складаються з розділів, які описують різні параметри конфігурації;

– Markdown. Це полегшена мова розмітки даних, яку створено з ухилом на прочитність та зручність у публікації з подальшим перетворенням. Використовується для створення форматованого тексту, такого як заголовки, списки, посилання та інші. Markdown-файли можуть бути перетворені на HTML, PDF та інші формати.

Окрему увагу варто приділити CSV формату. CSV (Comma-Separated Values) – це текстовий формат, який використовується для зберігання табличних даних. У таких файлах дані зберігаються у вигляді рядків, де кожний рядок представляє одну запис у таблиці. Поля в записі розділяються за допомогою роздільника, як правило, коми. CSV-файли є популярним форматом для обміну даними між різними програмами, оскільки вони є простими для обробки та зрозумілими. CSV-файли також можна легко відкрити та редагувати в будь-якому текстовому редакторі.

Крім того вони можуть використовуватися для зберігання різних типів даних, включаючи текст, числа та дати. Однак важливо пам'ятати, що CSV-файли не підтримують форматування даних, наприклад, шрифти, розміри шрифтів, кольори, вирівнювання та інші. Вони також можуть бути більш складними у використанні, ніж форматовані текстові формати, такі як DOC, DOCX та RTF.

Вибір CSV-формату залежить від конкретних потреб. Для простих завдань, таких як зберігання табличних даних без форматування, CSV-формат є хорошим вибором. В даній роботі для зберігання проміжних даних використовувався CSV-формат.

Розглянемо більш детальний опис засобів та бібліотек для обробки інформації на Python, а також

особливості бібліотек для парсингу, які були застосовані у даній роботі.

Pandas – це бібліотека з відкритим вихідним кодом, ліцензована BSD, що надає високопродуктивні, прості у використанні структури даних та інструменти аналізу даних для мови програмування Python. Одним з основних інструментів, що був використаний у даній роботі – це клас DataFrame. Він слугує для уявлення двовимірної таблиці зі змінюваною розмірністю. Ця структура даних є основною для *pandas* дозволяє персоналізувати позначення на осях (заголовки рядків та стовпців), виконувати арифметичні операції для рядків і стовпців табиці. У визначеннях мови Python, DataFrame можна розглядати як dict-подібний контейнер для об'єктів Series.

Для виконання основних функцій кожний клас передбачає опис конструктора класу. Він може містити чисельну кількість параметрів.

Перший параметр класу DataFrame є dict-словарем, що може містити такі структури даних, як ряди, масиви, константи, класи даних або спискові об'єкти. Якщо дані є словниками dict, порядок стовпців відповідає порядку полів словника. Якщо dict містить ряди, які мають визначений індекс, структура вирівнюється за його індексом. Це вирівнювання також відбувається, якщо дані є серією або самим DataFrame. Вирівнювання виконується на входах Series/DataFrame. Якщо дані є списком словників dicts, порядок стовпців відповідає порядку розташування словників у списку

Якщо вхідні дані є списком dicts, порядок стовпців структури буде відповідати порядку вставки.

Один з додаткових вхідних параметрів конструктора DataFrame відповідає за заголовки створюваної структури. За замовчуванням мітки стовпців для результуючого фрейма, якщо дані не містять їх, будуть числами від 0 до n. Якщо дані містять мітки стовпців, вони будуть відтворені у заголовках стовпців. В даному випадку фреймом називається таблиця, що має певний набір функцій для її обробки.

Як тільки фрейм буде заповнено даними доцільно зберегти його на локальному носії. Розглянемо як можна експортувати фрейм до CSV файлу за допомогою *to_csv()* методу пакету *pandas*. За замовчуванням, *to_csv()* метод експортує фрейм до CSV файлу з рядковим індексом як перша колонка і комою в якості сепаратора даних [5].

Для персоналізації результатів у цього методу є численна кількість параметрів. Так, щоб встановити роздільник поля для вихідного файлу слугує параметр *sep=','* за замовчуванням рівний комі. Для відтворення заголовків у вихідному файлі можна встановити параметр *header=True*. Даний параметр варто відключити при повторному записі того самого файлу. Параметр *index= True* повідомляє програмі пронумерувати кожний рядок. Для повноцінної роботи функції першим параметром має бути назва файлу для запису інформації.

Розглянемо наступну бібліотеку – requests. Це елегантна та проста бібліотека HTTP для Python, створена для людей. Вона дозволяє надзвичайно легко надсилати запити HTTP/1.1. Немає потреби вручну додавати рядки запиту до URL-адрес або кодувати дані PUT і POST у формі – достатньо застосувати метод json.

Зробити запит за допомогою Requests дуже просто. Для цього достатньо зробити імпорт модуля та створити запит GET для отримання веб-сторінки. Після того як запит виконано у програма має об'єкт Response під назвою r. З цього об'єкта ми можемо отримати всю необхідну інформацію.

Бібліотека забезпечує широкий спектр функцій для взаємодії з HTTP-серверами. Ось деякі з основних функцій [6]:

а) відправка HTTP-запитів будь-якого типу – означає підтримку всіх основних типів HTTP-запитів, включаючи GET, POST, PUT, DELETE та HEAD;

б) обробка відповідей HTTP – дозволяє отримувати інформацію про відповіді HTTP, включаючи статус-коди, заголовки та контент;

в) підтримка різних методів аутентифікації, таких як Basic, Digest та OAuth;

г) налаштування параметрів HTTP-запитів.

Бібліотека requests дозволяє налаштовувати параметри HTTP-запитів, такі як заголовки, cookies та тимчасові файли. Вона є потужним інструментом, який може бути використаний для вирішення різних завдань, пов'язаних з HTTP. Вона є простою у використанні та забезпечує широкий спектр можливостей.

Останньою бібліотекою, яку використано для парсингу є Beautiful Soup. Це бібліотека Python для вилучення даних із файлів HTML та XML. Вона працює з будь-яким парсером та надає прості та звичні способи навігації, пошуку та зміни дерева розбору. Вона економить програмістам години та дні роботи [7].

Beautiful Soup підтримує парсер HTML, включений до стандартної бібліотеки Python, а також ряд сторонніх парсерів на Python. Одним із них є парсер *lxml*. Інша альтернатива – написаний виключно на Python парсер *html5lib*, який розбирає HTML так само, як це робить веб-браузер. У кожного метода є свої переваги та недоліки, але найкращим з них є *lxml* парсер. Він виокремлюється від інших своєю швидкістю та глучкістю роботи.

Варто зауважити, якщо документ є невалідним, різні парсери будуть генерувати дерево Beautiful Soup для цього документа по-різному. При роботі з веб-додатками коли говорять, що документ є невалідним, це означає невідповідність його компонування та розмітки стандартам. Ці стандарти розроблюються консорціумом Всесвітньої павутини (W3C). Найновішим стандартом є HTML5.

Перевірка на валідність HTML документів є важливою для забезпечення того, щоб HTML-код був правильно написаний і міг бути правильно інтерпретований браузерами. Валідність HTML-коду також допомагає покращити його доступність для користувачів з обмеженими можливостями.

Для перевірки на валідність HTML документів можна використовувати різні інструменти. Деякі з них є безкоштовними та доступними в Інтернеті, інші є платними та вимагають встановлення на комп'ютер. Нижче наведено деякі з найпопулярніших інструментів для перевірки на валідність HTML документів:

– W3C Markup Validation Service. Безкоштовний онлайн-інструмент, який дозволяє перевірити HTML-код на відповідність стандарту HTML5;

– HTML Tidy. Безкоштовний інструмент, який дозволяє перевірити HTML-код на відповідність стандарту HTML5 і виправити деякі помилки;

– Codesniffer. Безкоштовний інструмент, який дозволяє перевірити HTML-код на відповідність стандартам W3C і іншим стандартам кодування.

Для перевірки на валідність HTML документів можна також використовувати вбудовані засоби перевірки в таких текстових редакторах, як Atom, Sublime Text та Visual Studio Code.

Ознайомившись з компонентами проекту можемо перейти до його детального опису. Розглянемо шлях створення додатку.

На першому етапі було встановлено інтерпретатор Python для написання та запуску парсера, за допомогою *pip* системи управління пакетами Python встановлено 3 бібліотеки:

– pip install beautifulsoup4;
– pip install requests;
– pip install pandas.

Як було зазначено раніше, кожна бібліотека має своє призначення:

– requests відповідає за отримання даних (у вигляді html-сторінки) з веб-сайту на локальний комп'ютер;

– beatifulsoup4 за інструменти обробки отриманих даних;

– pandas за зберігання оброблених даних на локальному носії у вигляді csv-файлу.

Встановлення пакетів здійснено через командний рядок Windows. За допомогою цих трьох бібліотек можна почати аналіз нашої веб-сторінку.

Врахувавши простоту синтаксису мови Python відносно інших мов програмування та запуску програм, в якості середовища розробки обрано Notepad++, середовище перевірки коду – командний рядок Windows.

Для початку необхідно було виконати аналіз HTML-коду веб-сторінки для виявлення елементів з корисною інформацією.

Для цього було виконано пошук потрібної веб-сторінки з потрібною інформацією.

На головній сторінці сайту потрібно натиснути на вкладку «пошук вантажів і транспорту», розміщену в правій частині сайту. Перед нами відкриється нова сторінка, на якій пропонується знайти попутні вантажі, а також вільний і попутний транспорт (рис. 1).

Нас цікавить інформація про міжміські вантажні перевезення, тому радіоперемикач має бути встановлено на пункті «Вантажі», що включено за замовчуванням. У параметрах «звідки», «куди» обираємо «Україна».

Для визначення терміну виконання заявки налаштовуємо пункт «Дата». Решту параметрів можна залишити за замовчуванням.

Після виконання налаштувань пошуку натискаємо кнопку «Знайти».

В результаті відкриється нова сторінка з потрібною інформацією.

Надалі працюємо з даною сторінкою.

За допомогою інструменту «Переглянути код» в інструментах веб-браузера Chrome (рис. 2) було отримано код веб-сторінки, який збережено в текстовому форматі.

Отриманий код було «передано» до бібліотеки *beatifulsoup4*, яка виконала його структурування.

Проаналізувавши код, було виділено елементи, що містять необхідну інформацію. Результатом аналізу стала можливість здійснювати пошук інформації по ключовим елементам, що дало змогу автоматизувати пошук.

На третьому етапі було організовано запис отриманої інформації до csv-файлу.

Так було визначено назви та послідовність заголовків, формат даних тощо.

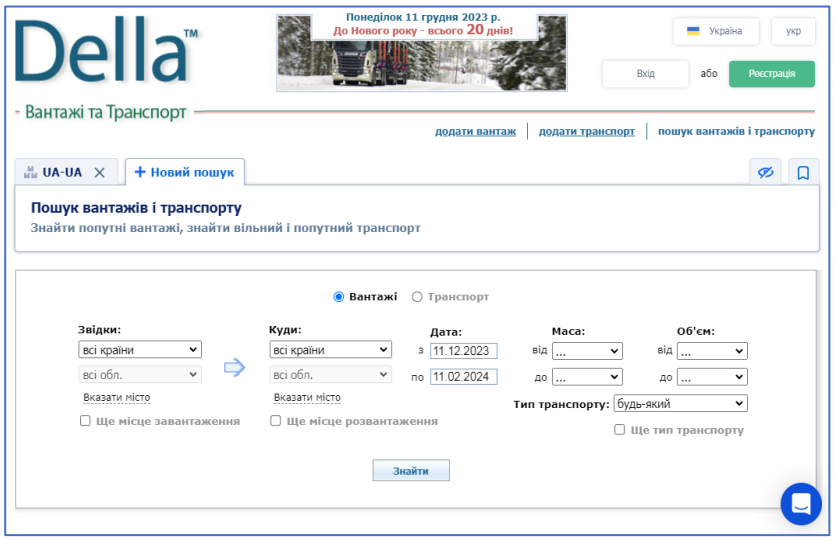

Рис. 1. Веб-сторінка «Пошук вантажів і транспорту»

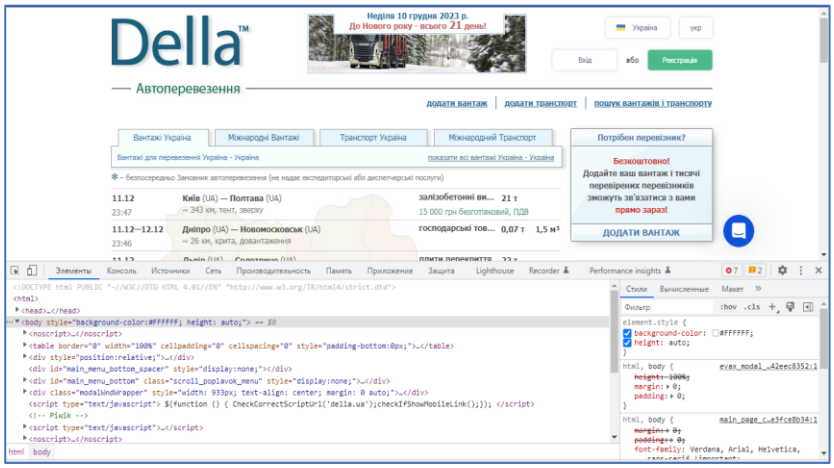

Рис. 2. Інструмент «Переглянути код» в нижній частині вікна браузера

Загалом, роботу парсера можна описати наступним чином.

На першому кроці відбувається завантаження інформації з веб-ресурсу за допомогою методів бібліотеки requests.

Отримавши відповідь на запит, завантажена інформація конвертується в текстовий формат та передається на обробку до бібліотеки *beatifulsoup4*, яка відбудовує структуру веб-сторінки локально для подальшої навігації по ключовим елементам.

На останньому кроці за допомогою методів навігації по елементам веб-сторінки завантажена інформація оброблюється та зберігається в словнику Python, після чого записується до csv-файлу.

Для запису до файлу використовується бібліотека *pandas*.

В результаті виконання парсингу веб-ресурсу було отримано інформацію про вантажні перевезення по Україні за період з 30.08.2023 по 05.09.2023.

Отримавши оброблені дані було виконано підрахунок кількості заявок, що припадають на кожну область та сформовано матрицю разових замовлень на міжміські вантажні перевезення [8].

Для автоматизації цього процесу було напитано скрипт на Python.

Висновки

Для моделювання потоку разових замовлень запропоновано використовувати методи машинного навчання. Ці методи дозволяють враховувати складні залежності між різними чинниками, які впливають на попит перевезень.

У результаті пошуку способів збирання даних виявлено чисельну кількість різноманітних підходів, що ґрунтуються на роботі програмних продуктів. Одним з таких є парсер даних, який виконує структурування інформації веб-сторінки і надає їх у готовому для аналізу вигляді.

Пошук інформації відбувається на найбільш популярному серед вантажних автотранспортних бірж в Україні логістичному сайті "DELLA", що відповідно має найбільшу цільову аудиторію у сфері автомобільних перевезень. В результаті обробки масиву даних було сформовано матрицю разових замовлень, яка визначає попит на ринку міжміських автомобільних перевезень України.

СПИСОК ЛІТЕРАТУРИ

1. Мокін В. Б., Романюк О. Н., Войтко В. В., Сторчак В. Г. та Гавенко О. В. Моделювання параметрів транспортної мережі в середовищі автоматизованої системи пошуку оптимальних рішень. *Інформаційні технології та комп'ютерна інженерія*. URL: https://www.researchgate.net.
2. Кочина А.А. Визначення закономірностей потоку разових замовлень у міжміському сполучення при управлінні ланцюгами постачань. *Сучасні технології в машинобудуванні та транспорті*. 2023. Вип. 10. С. 151-159.
3. Систематичне накопичення дослідних матеріалів. *Підручники для вузів онлайн*: веб-сайт. URL: https://pidru4niki.com/1089030461409/pedagogika/ sistematichne_nakopichennya_doslidnih_materialiv.
4. Олексій Волошин Переваги і недоліки Python. *Блог компанії Hillel*: веб-сайт. URL: https://blog.ithillel.ua/ru/articles/preimushchestva-i-nedostatki-yazyka-python.
5. Mrinalwalia. Saving a Pandas Dataframe as a CSV. *Geeksforgeeks*: веб-сайт. URL: https://www.geeksforgeeks.org/saving-a-pandas-dataframe-as-a-csv/.
6. Kenneth Reitz. Requests: HTTP for Humans. URL: https://requests.readthedocs.io/en/latest/.
7. Beautiful Soup Documentation. URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/.
8. Мосьпан Н. В. Формування стратегій автотранспортних підприємств по обслуговуванню разових замовлень на перевезення вантажів у міжміському сполученні : автореф. дис. ... канд. техн. наук : 05.22.01. Харків, 2018. 212 с.

Received (Надійшла) 26.05.2024
Accepted for publication (Прийнята до друку) 14.08.2024

Development of an automated parsing system one-time orders for freight transport in long-distance traffic

A. Kochina, K. Asimov

Abstract. Problem. The efficient movement of goods over vast distances is the backbone of modern trade. Automatic collection mechanisms are driving the development of a system that collects and updates data in real time, using technologies such as data parsing, data scanning and sensor networks to create a dynamic, rapid response platform. This data-driven approach has huge potential for both shippers and carriers. **Goal.** Defining the theoretical foundations of automated data collection software for the development of a parsing system for automated search and processing of one-time orders for long-distance freight transport. **Methodology**. Statistical and simulation modelling. Task statement – Identification and evaluation of potential orders on the DELLA online platform, integration into the company's business processes related to information processing and analysis. **Results.** The article analyses the theoretical foundations of software development for automated input data collection. An automated parsing system was developed to search and process one-time orders for long-distance freight transport. The development of the automated parsing system allowed to obtain a matrix of one-time orders at the regional level of Ukraine. Conclusions: To model the flow of one-time orders, it is proposed to use machine learning methods to create a data parser that structures the information of a web page and provides it in a form ready for analysis. The information is searched for on the most popular logistics website among freight transport exchanges in Ukraine, DELLA, which has the largest target audience in the road transport sector. **Originality.** Based on the analysis of automated data collection software, a parsing system was created for automated search and processing of one-time orders for long-distance freight transport in intercity traffic. **Practical value.** As a result of processing the data set, a matrix of one-time orders was formed, which determines the demand in the Ukrainian intercity road transport market.

Keywords: parsing, demand, one-off orders, automated data retrieval systems.