заперечувати важливість вибору відповідних методів для оцінки кореляції та необхідність використання більш точних методів, як-от методи підтверджуючого факторного аналізу і дослідницького моделювання структурних рівнянь.

**Список використаних джерел**

1. Hair, J. F., Hult, T., Ringle, C. M., Sarstedt, M. (2022). A Primer on Partial Least Squares Structural Equation Modeling (PLS-SEM), 3rd edn. New York, NA: Sage.

2. Morin, S., Arens, K., Tran, A., Caci, H. (2016). Exploring sources of construct-relevant multidimensionality in psychiatric measurement: a tutorial and illustration using the composite scale of morningness. *Int. J. Methods Psychiatr. Res*. 25, 277–288. DOI: 10.1002/mpr.1485.

3. Alamer, A., Marsh, H. (2022). Exploratory structural equation modeling in second language research: an applied example using the dualistic model of passion. *Stud. Second Lang. Acquis, 1–24.* DOI: 10.1017/S0272263121000863.

**УДК 004.6+005.7**

*Крохмалюк В. В., здобувач 2 курсу спеціальності 122 Комп'ютерні науки ОС «Магістр»,*

*Потапова Н. А., канд. екон. наук, доцент, доцент кафедри інформаційних технологій*

## ВИКОРИСТАННЯ ІНСТРУМЕНТІВ ДЛЯ ЗБОРУ ДАНИХ У ДОСЛІДНИЦЬКІЙ ДІЯЛЬНОСТІ ЯК ЕФЕКТИВНИЙ МЕТОД ОТРИМАННЯ ІНФОРМАЦІЇ

*Донецький національний університет імені Василя Стуса, м. Вінниця*

Парсинг – це процес аналізу вхідних даних з метою виділення з них значимих елементів та перетворення їх у структурований формат для подальшої обробки. У програмуванні парсинг часто використовується для обробки текстових даних, як-от дані у форматі CSV, JSON або XML, а також для обробки коду мов програмування [2].

Парсинг може виконуватись із допомогою спеціальних програм, які називають парсерами. Парсер зчитує вхідні дані, аналізує їх за заданою граматикою та перетворює їх у структурований формат. Із допомогою парсера можна виділити з великого обсягу текстової інформації лише ту, що потрібна для подальшої обробки. У веброзробці парсинг часто використовується для збору даних із вебсторінок. Основна його ідея полягає в автоматичному зборі даних із вебсторінок за допомогою скриптів [1].

Парсинг може бути використаний для збору інформації про товари, послуги, ціни, новини тощо. Отже, парсинг – це процес аналізу вхідних даних з метою їх

обробки та перетворення у структурований формат [2]. У програмуванні парсинг часто використовується для обробки текстових даних, збору даних із вебсторінок. Для виконання парсингу використовуються спеціальні бібліотеки та інструменти, які дають змогу зчитувати та аналізувати дані з різних джерел, як-от файлові системи, бази даних, вебсторінки, електронні документи та інші джерела.

Розглянемо принцип роботи на прикладі парсера, який збирає дані з вебсторінок. Парсер аналізує HTML-код сторінки та виконує процес розбору (парсингу) на основі заздалегідь встановлених правил. Основний принцип роботи парсера полягає в тому, що він сканує HTML-код сторінки та визначає різні типи тегів, як-от заголовки, текстові блоки, посилання тощо. Після того, як парсер визначив теги та їх вміст, він може зібрати ці дані та структурувати їх у зручний формат для подальшого використання. Наприклад, якщо парсеру потрібно зібрати інформацію зі списку товарів на вебсторінці, то він буде шукати тег <ul> та його елементи <li>. Після знаходження цих тегів парсер може отримати дані про кожен товар, зокрема його назву, опис, зображення та ціну [3].

Одним із найпоширеніших методів парсингу є парсинг із використанням регулярних виразів. Регулярні вирази дають змогу шукати та витягувати певні частини тексту на основі певного шаблону. Цей метод дуже ефективний для простих парсерів, але не дуже надійний для складних структур сторінок, оскільки може не враховувати всі варіації розмітки HTML [1].

Існує також інший метод парсингу – парсинг з використанням бібліотеки для обробки HTML-коду. Цей метод допомагає парсеру більш точно визначати структуру сторінки, зокрема, з допомогою визначення CSS-селекторів. Завдяки цьому методу парсер може точніше зібрати інформацію з вебсторінки та зменшити ймовірність помилок під час парсингу.

Інструменти, з допомогою яких найчастіше створюються парсери, залежать від використовуваної мови програмування та типу даних, які потрібно парсити. Найбільш популярними інструментами для створення парсерів є:

1. Beautiful Soup: це бібліотека Python, призначена для парсингу HTML- та XML-даних. Beautiful Soup дає змогу зчитувати дані з вебсторінок та обробляти їх, знаходячи необхідні теги й атрибути [3].

2. lxml: це бібліотека Python для обробки XML- та HTML-даних, яка має високу швидкодію. lxml дає змогу використовувати XPath-запити для пошуку необхідних даних.

3. Scrapy: це фреймворк Python для парсингу вебсторінок. Scrapy дає змогу створювати скрапери (вебпавуки), які можуть збирати дані з кількох вебсторінок одночасно, виконувати асинхронні запити та оброблювати результати.

4. Selenium: це інструмент для автоматизації браузерів, який дає змогу створювати скрипти для взаємодії з вебсторінками. Selenium може бути використаний для автоматичного заповнення форм, натискання кнопок та збору даних із динамічних вебсторінок.

Парсер є корисним інструментом для науковців, які працюють із великими об’ємами даних. Завдяки парсингу вебсторінок вони можуть отримувати необхідну інформацію з різних джерел, аналізувати її та використовувати для своїх наукових досліджень.

Основна перевага використання парсера полягає в тому, що він може значно зменшити витрати часу та зусиль на пошук, обробку й аналіз даних. До того ж використання парсера допомагає отримати більш точну та повну інформацію, ніж під час ручного збору даних.

Із допомогою парсера науковці можуть зібрати та проаналізувати великий об’єм даних, що дасть змогу зробити більш точні висновки й отримати нові знання в різних галузях науки. До того ж використання парсера допомагає автоматизувати процес збору даних, що зменшує ймовірність помилок і забезпечує швидкий та ефективний доступ до необхідної інформації.

Отже, користування парсером може допомогти науковцям ефективніше й точніше збирати та аналізувати дані, що дає змогу робити більш досконалі дослідження та отримувати нові знання в різних галузях науки.

**Список використаних джерел**

1. Mitchell R. Web Scraping with Python: Collecting More Data from the Modern Web 2nd Edition, 2018. 306 p.

2. Heydt M. Python Web Scraping Cookbook: Over 90 proven recipes to get you scraping with Python, micro services, Docker and AWS, 2018. 364 p.

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

**УДК 004.85:004.93(043.2)**

*Бевз Д. М., здобувач вищої освіти 2 курсу ОС «Магістр» спеціальності 122 Комп’ютерні науки,*

*Римар П. В., старший викладач кафедри інформаційних технологій*

## ЕФЕКТИВНІСТЬ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ НА ПРИКЛАДІ ЗАДАЧІ КЛАСИФІКАЦІЇ ТЕКСТІВ

*Донецький національний університет імені Василя Стуса, м. Вінниця*

У сучасному світі обсяги текстових даних зростають із неймовірною швидкістю, що створює потребу в ефективних методах їх обробки та аналізу. Особливо актуальним є завдання класифікації текстів, яке має широке застосування в різних сферах, від виявлення спаму до аналізу настроїв.