Міністерство освіти і науки України
Тернопільський національний педагогічний університет
імені Володимира Гнатюка

Фізико-математичний факультет
Кафедра інформатики та методики її навчання

Кваліфікаційна робота

МЕТОДИ І АЛГОРИТМИ АНАЛІЗУ КОРЕЛЯЦІЙ МІЖ ТЕКСТОВИМИ ДАНИМИ І ПОВЕДІНКОВИМИ ПОКАЗНИКАМИ КОРИСТУВАЧІВ

Спеціальність 122 Комп'ютерні науки
Освітня програма «Комп'ютерні науки»

Здобувача вищої освіти освітньо-кваліфікаційного рівня «магістр»
Ясінського Андрія Михайловича
НАУКОВИЙ КЕРІВНИК:
кандидат історичних наук, асистент кафедри інформатики та методики її навчання
Лень Андрій Володимирович
РЕЦЕНЗЕНТ:
доктор технічних наук, професор кафедри комп'ютерних наук Тернопільського національного технічного університету ім. Івана Пулюя
Литвиненко Ярослав Володимирович

Тернопіль – 2025

АНОТАЦІЯ

Ясінський А. М. Методи і алгоритми аналізу кореляцій між текстовими даними і поведінковими показниками користувачів. Кваліфікаційна робота на здобуття освітнього ступеня «магістр» зі спеціальності 122 Комп'ютерні науки. ТНПУ ім. В. Гнатюка. Тернопіль, 2025. 77 с.

У кваліфікаційній роботі обґрунтовано методи та алгоритми аналізу кореляцій між текстовими даними та поведінковими показниками користувачів, розглянуто сучасні підходи текстової аналітики та статистичного аналізу цифрової поведінки. Розроблено та реалізовано підхід до автоматизованого збору, очищення й обробки даних, побудовано прототип кореляційного аналізу із використанням мови програмування Python. Створено інструменти візуалізації, що забезпечують наочне представлення взаємозв'язків між семантичними характеристиками текстів та поведінкою користувачів, а також визначено ключові закономірності цифрової активності.

Ключові слова: текстові дані, поведінкові показники, кореляційний аналіз, Python, статистичні методи, цифрова поведінка.

ABSTRACT

Yasinskyi A. M. Methods and algorithms for analyzing correlations between textual data and user behavioral indicators. Qualification work for obtaining a master's degree in the specialty 122 Computer Science. Ternopil Volodymyr Hnatiuk National Pedagogical University. Ternopil, 2025. 77 p.

The qualification work substantiates the methods and algorithms for analyzing correlations between textual data and user behavioral indicators, and examines modern approaches to text analytics and statistical analysis of digital behavior. A framework for automated data collection, cleaning, and preprocessing was developed and implemented, along with correlation analysis prototype using the Python programming language. Visualization tools were created to clearly represent the relationships between semantic text characteristics and user behavior, enabling the identification of key patterns of digital activity.

Keywords: textual data, behavioral indicators, correlation analysis, Python, statistical methods, digital behavior.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Сучасний розвиток цифрових технологій зумовив стрімке зростання обсягів даних, які користувачі щодня генерують у мережі. Соціальні медіа, новинні ресурси, онлайн-форуми та інші інформаційні платформи містять величезні масиви текстових даних, що відображають інтереси, настрої та поведінкові особливості користувачів. Збір та аналіз цих даних є важливим інструментом для виявлення закономірностей у споживанні контенту, дослідження механізмів залучення та прогнозування дій аудиторії.

Особливу наукову й практичну цінність становить вивчення взаємозв'язків між контентом та показниками користувачів, такими як перегляди, лайки, коментарі, поширення тощо. Виявлення кореляційних залежностей дозволяє краще зрозуміти, які мовні особливості, ключові слова чи теми сприяють підвищенню активності аудиторії. Такий підхід відкриває широкі можливості для оптимізації інформаційної політики, маркетингових стратегій, а також розробки інтелектуальних систем рекомендацій.

У працях українських науковців значна увага приділяється аналізу текстових даних у поєднанні з поведінковими характеристиками користувачів у цифровому середовищі. Так, у наукових дослідженнях Ю. Бойка, В. Пасічника, А. Пелещишина розкрито теоретичні та прикладні засади аналізу соціальних комунікацій у мережі інтернет. У дослідженнях О. Кузнєцова, О. Микитенка, Н. Шаховської розглянуто алгоритмічні підходи оброблення великих обсягів неструктурованих текстових даних, зокрема із застосуванням методів машинного навчання та семантичного аналізу. Дослідниками показано, що лінгвістичні характеристики текстів (частотність лексем, емоційне забарвлення, тематична спрямованість) можуть корелювати з такими поведінковими показниками користувачів, як частота взаємодій, тривалість сесій та рівень залученості [20, с. 5].

У зарубіжних дослідженнях проблема аналізу кореляцій між текстовими даними та поведінковими характеристиками користувачів отримала ґрунтовне висвітлення. Так, у працях J. Pennebaker, M. Tausczik розкрито психолінгвістичні підходи до аналізу текстів, де мовні маркери (використання займенників, емоційних слів, когнітивних конструкцій) корелюють з поведінковими та психологічними характеристиками користувачів. Запропоновані авторами методи дозволяють прогнозувати поведінку людини на основі текстових даних. У дослідженнях M. Kosinski, D. Stillwell, T. Graepel продемонстровано можливості машинного навчання у виявленні зв'язків між цифровими слідами користувачів (тексти постів, коментарів) та їхніми поведінковими й соціальними характеристиками. Авторами доведено, що аналіз текстового контенту в поєднанні з поведінковими метаданими дозволяє з високою точністю передбачати дії користувачів у цифрових середовищах.

Недостатня систематизація методів інтегрованого аналізу текстових і поведінкових даних, а також обмежена кількість практично орієнтованих алгоритмів зумовлюють актуальність подальших досліджень у цьому напрямі, що визначає доцільність вибору теми, пов'язаної з методами й алгоритмами аналізу кореляцій між текстовими даними та поведінковими показниками користувачів.

Метою дослідження є розробка та теоретичне обґрунтування методів й алгоритмів аналізу кореляцій між текстовими даними та поведінковими показниками користувачів на основі автоматизованого збору, обробки, аналізу та візуалізації даних із використанням мови програмування Python.

Завдання дослідження:

1. Дослідити й теоретично обгрунтувати методи збору, обробки та аналізу даних.

2. Розглянути й проаналізувати інструменти і бібліотеки для автоматизації збору, обробки, збереження і візуалізації даних за допомогою мови програмування Python.

3. Провести статистичний аналіз і визначити кореляційні зв’язки між ключовими словами контенту та показниками залученості.

4. Розробити прототип для автоматизованого збору даних, їхнього аналізу та візуалізації ґрезультатів.

Об’єкт дослідження – процеси збору, обробки, аналізу та візуалізації текстових даних і поведінкових показників користувачів.

Предмет дослідження – методи, алгоритми й інструменти аналізу кореляцій між текстовими даними та поведінковими метриками користувачів, включно з підходами до синтаксичного аналізу текстів, обробки структур даних і виявлення закономірностей у користувацької активності.

Методи дослідження. Для аналізу кореляцій між текстовими даними та поведінковими показниками користувачів використовувався комплекс методів. Теоретичні включали аналіз літератури та систематизацію підходів до обробки текстів і цифрової поведінки. Емпіричні методи передбачали обробку та аналіз великих масивів даних, включно з нормалізацією текстових даних, токенізацією, видаленням шуму та статистичним аналізом поведінкових метрик.

Наукова новизна. Новизна роботи полягає у поєднанні методів обробки природної мови та статистичних алгоритмів для виявлення кореляцій між текстовими характеристиками та поведінковими показниками користувачів. Розроблено підхід до визначення значущості впливу певних текстових ознак на поведінку аудиторії та оцінено ефективність різних алгоритмів аналізу.

Практичне значення. Практичне значення полягає у можливості застосування отриманих результатів для прогнозування поведінки користувачів, оптимізації контенту та підвищення залученості аудиторії. Розроблені алгоритми можуть бути використані в маркетингових аналітичних системах, платформах соціальних мереж та при оцінці ефективності інформаційних кампаній.

РОЗДІЛ 1
ОСНОВИ АНАЛІЗУ ТЕКСТОВИХ ДАНИХ ТА ПОВЕДІНКОВИХ ПОКАЗНИКІВ КОРИСТУВАЧІВ

1.1 Поняття та класифікація даних у цифровому середовищі

Тип даних визначає спосіб організації, подання та інтерпретації інформації, задаючи її основні властивості та правила опрацювання. Під час роботи з інформацією дані поділяють на різні категорії залежно від їхнього змісту, структури та призначення. Окрім традиційного поділу на структуровані й неструктуровані, у науковій літературі представлені й інші класифікації. Зокрема, залежно від інформації, яку містять файли, зазвичай виокремлюють такі типи даних: текстові дані**;** табличні або структуровані дані; графічні; аудіо; відео; геопросторові; архівні та інші дані.

Текстові дані – це представлення інформації в обчислювальній системі у вигляді послідовності друкованих символів. Іншими словами, якщо більшість місця у наборі займає простий текст – ви маєте справу з текстовими даними. Прикладом текстових даних можуть бути звіти, нормативно-правові акти, логи, рішення чи розпорядження органів влади, нотатки тощо.

Публікуються текстові дані передусім у відкритих форматах TXT, RTF та ODT. Органам влади також дозволяються використовувати формати DOCX та PDF.

Категорично не підходять для текстових даних формати JPG, JPEG, PNG, GIF, TIFF, а також PDF зі сканованим зображенням. Публікація текстових даних у цих форматах унеможливлює їх обробку автоматизованими засобами, оскільки їх потрібно додатково оцифровувати [43].

Табличні або структуровані дані – це впорядкована сукупність стовпців та рядків, наприклад, усім звичні таблиці Excel. Варто пам’ятати, що наявність табличної форми подання інформації свідчить про те, що набір даних належить до структурованих. Найчастіше структуровані дані зустрічаються у відкритому форматі CSV, рідше у форматі XLS(X).

Зазвичай програмні та інформаційні системи дають змогу експортувати дані у форматах XML або JSON, тому їх часто використовують для публікації набору даних. Втім, найкраще формати XML і JSON підходять для ієрархічних даних.

Якщо набір даних представлений у вигляді фотографії або будь-якого іншого зображення, то він належить до *графічних даних*. Прикладом таких даних можуть бути фото архівних документів, генеральні плани міст тощо.

Графічні зазвичай публікуються у відкритих форматах PNG, JPG чи JPEG. Часто буває, що текстові чи навіть структуровані дані оприлюднюються у вигляді графічних. Тоді для розпізнавання тексту потрібно використовувати технології OCR (оптичне розпізнавання тексту). Однією з найвідомішою у цій сфері програмою є ABBYY FineReader [43].

Геопросторові дані – це інформація, що визначає географічне положення та характеристики об'єктів та/або їхні кордони на поверхні Землі. Якщо набір даних містить інформацію про розташування певних об’єктів із зазначенням широти й довготи, або опис меж певних територій із використанням полігонів, такий набір класифікується як геопросторові дані.

Прикладом геопросторових даних можуть бути генеральні плани населених пунктів, схеми планування територій і плани зонування територій, межі виборчих округів та дільниць, відомості з Держгеокадастру, маршрути й дані про місцезнаходження громадського транспорту тощо. Геопросторові дані передусім публікуються у відкритих форматах GeoJSON, SHP, рідше GPX, GeoTIFF.

Архіви – файли, що містять у собі один або декілька файлів та метадані. Файли можуть бути як стиснені (без втрат), так і мати початковий розмір та

структуру. Метадані можуть містити інформацію про початковий розмір файлів, інформацію про формат файлів, структуру директорій, коментарі до файлів тощо.

Архіви файлів створюються за допомогою спеціалізованих програм – архіваторів, які можуть бути як окремими програмами, так і частиною інших програм. Якщо набір даних міститься у файлі великого розміру, або публікується багато типових файлів, що є частиною одного набору даних, є сенс використовувати для публікації архіви даних. Вони допомагають зменшити розмір набору даних і завантажити велику кількість типових файлів за один раз.

У програмуванні тип даних – це атрибут, пов'язаний з фрагментом даних, який повідомляє комп'ютерній системі, як інтерпретувати його значення. Розуміння типів даних гарантує, що дані збираються в бажаному форматі, а значення кожної властивості відповідає очікуванням.

Наприклад, знання типу даних для «Росс, Боб» допоможе комп'ютеру дізнатися: чи стосуються дані повного імені особи («Боб Росс»); або список із двох імен («Боб» та «Росс»). Розуміння типів даних допоможе переконатися, що: дані, які збираються мають правильний формат («Росс, Боб», а не «Боб Росс»); значення відповідає очікуванням («Росс, Боб» проти «R0$$, B0b»).

У таблиці 1.1 представлено найбільш поширені типи даних, що використовуються для представлення різнорідної інформації в обчислювальних системах.

Таблиця 1.1

Найпоширеніші типи даних

Data Type	Definition	Examples
Integer (int)	Numeric data type for numbers without fractions	-707, 0, 707
Floating Point (float)	Numeric data type for numbers with fractions	707.07, 0.7, 707.00
Character (char)	Single letter, digit, punctuation mark, symbol, or blank space	a, 1, !

Продовження табл. 1.1

String (str or text)	Sequence of characters, digits, or symbols—always treated as text	hello, +1-999-666-3333
Boolean (bool)	True or false values	0 (false), 1 (true)
Enumerated type (enum)	Small set of predefined unique values (elements or enumerators) that can be text-based or numerical	rock (0), jazz (1)
Array	List with a number of elements in a specific order—typically of the same type	rock (0), jazz (1), blues (2), pop (3)
Date	Date in the YYYY-MM-DD format (ISO 8601 syntax)	2021-09-28
Time	Time in the hh:mm:ss format for the time of day, time since an event, or time interval between events	12:00:59
Datetime	Date and time together in the YYYY-MM-DD hh:mm:ss format	2021-09-28 12:00:59
Timestamp	Number of seconds that have elapsed since midnight (00:00:00 UTC), 1st January 1970 (Unix time)	1632855600

Ціле число (*int*) – це найпоширеніший числовий тип даних, який використовується для зберігання чисел без дробового компонента (-707, 0, 707).

Число з плаваючою комою (*float*) – числовий тип даних, який використовується для зберігання чисел, які можуть мати дробову складову, як-от грошові значення (707.07, 0.7, 707.00). Зверніть увагу, що число часто використовується як тип даних, який включає як цілі, так і числа з плаваючою комою.

Символ (*char*) – використовується для зберігання однієї літери, цифри, знака пунктуації, символу або пробілу.

Рядок (*str* або текст) – це послідовність символів і найпоширеніший тип даних для зберігання тексту. Крім того, рядок може містити цифри та символи, проте він завжди розглядається як текст. Номер телефону зазвичай зберігається як рядок (+1-999-666-3333), але може зберігатися як ціле число (9996663333).

Логічне значення (*boolean*) – представляє значення true або false. Працюючи з булевим типом даних, корисно пам'ятати, що іноді булеве значення також представляється як 0 (для false) та 1 (для true).

Перелічуваний тип (*enum*) – він містить невеликий набір заздалегідь визначених унікальних значень (також відомих як елементи або перечислителі), які можна порівнювати та присвоювати змінній перелічуваного типу даних. Значення перелічуваного типу можуть бути текстовими або числовими. Фактично, булевий тип даних – це заздалегідь визначений перелік значень true та false.

Наприклад, якщо рок та джаз є перечислювачами, змінній перечислюваного типу genre можна присвоїти одне з двох значень, але не обидва одночасно. За допомогою перелічуваного типу значення можна зберігати та отримувати як числові індекси (0, 1, 2) або рядки.

Масив (*array*) – також відомий як список, масив – це тип даних, який зберігає певну кількість елементів у певному порядку, зазвичай усіх одного типу.

Оскільки масив зберігає кілька елементів або значень, структура даних, що зберігаються масивом, називається структурою даних масиву. Кожен елемент масиву можна отримати за допомогою цілочисельного індексу (0, 1, 2, і т.д.), а загальна кількість елементів у масиві представляє довжину масиву.

Наприклад, масив змінних genre може зберігати один або декілька елементів рок, джаз та блюз. Індекси трьох значень – 0 (рок), 1 (джаз) та 2 (блюз), а довжина масиву дорівнює 3 (оскільки він містить три елементи).

Продовжуючи на прикладі музичного додатку, якщо вибрати один або декілька з трьох жанрів, змінна genre зберігатиме всі три елементи (рок, джаз, блюз).

Дата (*date*) – не потребує особливих пояснень, зазвичай зберігає дату у форматі РРРР-ММ-ДД (синтаксис ISO 8601).

Час (*time*) – зберігає час у форматі гг:хх:сс. Окрім часу доби, його також можна використовувати для зберігання часу, що сплив, або інтервалу часу між

двома подіями, який може бути більше 24 годин. Наприклад, час, що сплив з моменту події, може становити понад 72 години (72:00:59).

Дата й час (*datetime*) – зберігає значення, що містить дату та час разом у форматі РРРР-ММ-ДД гг:хх:сс.

Позначка часу (*timestamp*) – зазвичай представлена в Unix-часі, позначка часу відображає кількість секунд, що пройшли з півночі (00:00:00 UTC), 1 січня 1970 року. Зазвичай його використовують комп'ютерні системи для реєстрації точної дати та часу події, аж до кількості секунд, у форматі, який не залежить від часових поясів. Тому, на відміну від дати та часу, позначка часу залишається незмінною незалежно від географічного розташування [43].

Показовим прикладом для розуміння типів даних може слугувати будь-яка форма або опитувальник. Розглядаючи стандартну реєстраційну форму, слід пам'ятати, що кожне поле приймає значення певного типу даних. Текстове поле зберігає вхідні дані як рядок, тоді як числове поле зазвичай приймає ціле число. Імена та адреси електронної пошти завжди мають тип рядка, тоді як числа можна зберігати як числовий тип або як рядок, оскільки рядок – це набір символів, включаючи цифри. У полях з одним або кількома опціями, де потрібно вибирати з попередньо визначених опцій, використовуються перелічувані типи даних та масиви.

Таким чином, різні мови програмування можуть містити власні набори типів даних, однак у межах даного підрозділу було проаналізовано найбільш поширені та універсальні з них.

1.2 Поведінкові показники користувачів: види, методи вимірювання та значення

Передусім необхідно визначити, що таке показники соціальних мереж і яке значення вони мають у дослідженні цифрових процесів. Показники (метрики)

соціальних платформ є ключовими інструментами для кількісного оцінювання взаємодії користувачів із контентом і дають змогу об'єктивно аналізувати ефективність онлайн-комунікацій. Метрики соціальних мереж – це точки даних, які вимірюють, наскільки добре працює стратегія соціальних мереж і допомагають зрозуміти, як її можна покращити. Вони схожі на картки показників онлайн-публікацій та взаємодій, що показують, скільки людей переглянули, поставили лайк, поділилися або прокоментували контент [32].

Без показників не можливо створити обґрунтовану стратегію і пов'язати свої зусилля в соціальних мережах із реальними цілями бізнесу в соціальних мережах або довести свій успіх. Також такий підхід унеможливлює виявлення тенденцій до зниження, які можуть вимагати зміни стратегії.

Рівень залученості. Рівень залученості вимірює кількість взаємодій (лайків, коментарів та поширень – про кожну з яких ми розповімо детальніше нижче), які отримує контент, у відсотках від аудиторії.

Вподобання та реакції. Вподобання показують, скільки людей фізично «вподобали» або «відреагували» на один із постів. Хоча дехто розглядає його лише як прояв марнославства, показник має суттєвий вплив на рівень залученості. Оскільки все більше платформ (таких як Facebook та LinkedIn) запроваджують реакції, даний показник допоможе оцінити фактичний настрій щодо кожної публікації. Чи сміються люди? Чи підтримують вони контент у соціальних мережах? Або ж ви можете подивитися на число в цілому, щоб отримати уявлення про те, скільки людей вважали, що пост знайшов у них достатній відгук, щоб виконати дію під час прокручування, незалежно від того, наскільки незначною вона була. Якщо відстежувати свої вподобання, можливо, просто варто вести лічильник, щоб спостерігати за зростанням цих показників залученості.

Коментарі. Одним важливим показником залученості є кількість коментарів, які отримує кожна публікація. Вподобання або реакція – це проста та легка дія, але залишення коментаря означає, що аудиторія справді має що сказати.

Тріш Рісвік, менеджер із соціальних мереж у Hootsuite, поділяє наступну думку: «Розділ коментарів сповнений натхнення та відгуків, але це також чудовий показник для відстеження, оскільки він демонструє ефективність публікацій та зростання любові з боку підписників». Даний показник дозволяє відстежувати динаміку активності в розділі коментарів і оцінювати, наскільки він зростає з часом.

Поширення. Найголовніший показник того, наскільки контент подобається аудиторії – їхня зацікавленість у поширенні його на власних сторінках, щоб його могли побачити їхні друзі, родина та підписники.

Рісвік пояснює: «У нашому звіті для споживачів за 2024 рік ми багато дізналися про те, чому люди діляться контентом у соціальних мережах. Найбільша причина (44 %) полягає в тому, що люди погоджуються з публікацією. 29 % респондентів сказали, що поділилися чимось, бо вважали інформативним. Тоді як 24 % сказали, що це тому, що публікація була натхненною». Однак, Рісвік зазначає, що «як би ви це не сприймали, поширення контенту – це емоційна реакція та свідчення того, наскільки добре контент сприймався». Відстежуючи кількість поширень контенту протягом певного періоду часу, щоб переконатися, що він зростає. Якщо відбувається помітне зниження кількості поширень, потрібно дослідити, який контент отримує найбільше поширень, і створити більше подібного контенту [23].

Враження. Покази відображають кількість разів, коли користувачі переглянули певний контент. Їх можна вимірювати як для окремих публікацій, так і для загальної активності профілю в соціальних мережах. Значення показів може перевищувати охоплення, оскільки один і той самий користувач може переглядати контент кілька разів. Висока кількість показів порівняно з охопленням може свідчити про повторні перегляди одних і тих самих матеріалів, що, у свою чергу, вказує на підвищений інтерес аудиторії або ефективність контенту.

Коефіцієнт конверсії. Коефіцієнт конверсії вимірює, як часто соціальний контент призводить до конверсійної події, такої як підписка, завантаження або продаж. Даний показник являється один з найважливіших у маркетинзі соціальних мережах, оскільки він показує цінність кампаній у соціальних мережах (органічних та платних) для заповнення продажів.

Настрої в соціальних мережах. Настрої в соціальних мережах відстежують почуття та ставлення, що стоять за розмовою. Для обчислення соціальних настроїв потрібна допомога інструменту метрик соціальних мереж, який може обробляти та категоризувати мову та контекст.

У контексті Reddit, як платформи з масовою комунікацією та великою кількістю текстових даних, *поведінкові показники користувачів* є важливою складовою при аналізі кореляцій між змістом повідомлень і реакціями спільноти. Вони дають змогу виміряти не лише рівень активності, а й глибше зрозуміти зв'язок між емоційним тоном текстів та взаємодією користувачів.

Види показників: *акти комунікації***:** кількість постів і коментарів, їх довжина, частота участі в дискусіях; *взаємодія з контентом:* upvotes/downvotes, співвідношення переглядів і активних дій, кількість коментарів до постів [23]; *динамічні характеристики:* швидкість зростання обговорення, зміна популярності тем у часі, сталість активності.

Методи вимірювання. *Кількісні методи:* статистичний аналіз кореляцій між активністю та тоном повідомлень; *текстово-поведінковий аналіз:* поєднання sentiment analysis із метриками залученості (наприклад, позитивний тон посту – більше upvotes); *часові ряди:* відстеження змін активності й емоційного тону з плином часу.

Інтеграція поведінкових показників у аналіз дозволяє: виявляти закономірності між емоційним забарвленням тексту та реакціями користувачів; прогнозувати рівень залученості залежно від теми чи настрою повідомлення;

вивчати механізми поширення інформації в онлайн-спільнотах; застосовувати результати для маркетингових і соціологічних завдань.

1.3 Методи збору, попередньої обробки та аналізу текстових даних

Оскільки кількість інформації надзвичайно велика і збирати дані вручну вкрай незручно, складно та довго, застосовуються методи автоматизованого збору даних, серед яких особливе місце займає скрапинг.

Скрапинг дозволяє ефективно отримувати великі масиви текстової інформації з вебсайтів та онлайн-джерел, замінюючи повільний процес копіювання «copy-paste» швидким та системним механізмом. Крім того, автоматизація значно зменшує вплив людського фактору, знижує ризик помилок та забезпечує можливість подальшої обробки даних у зручному форматі. Такий підхід створює основу для наступних етапів – попереднього очищення, структуризації та аналізу текстових масивів, що є ключовим завданням сучасних інформаційних технологій.

Для реалізації збору та аналізу даних можна або розробити власний скрапер, або скористатися вже готовим рішенням. Розробка власного інструменту є більш економічним варіантом, однак вона потребує відповідних знань у сфері програмування чи залучення спеціалістів, які здатні створити унікальний продукт під конкретні завдання. Альтернативою є використання готових інструментів для скрапингу, які хоч і потребують більших фінансових витрат, проте дозволяють значно зекономити час та зусилля.

Ключові переваги автоматизації включають: швидкість та ефективність; збір та обробка даних у режимі реального часу зі швидкістю мілісекунд; безперервна робота 24/7 з надійністю 99,9 %; автоматична обробка обсягів даних масштабу підприємства; точність та надійність; майже ідеальна точність завдяки вбудованій валідації; узгоджене форматування з різних джерел; повне відстеження

походження даних; економічна ефективність; значне зниження операційних витрат; мінімальні потреби в персоналі;автоматизоване запобігання помилкам; масштабованість; безперешкодна обробка зростаючих обсягів даних; швидка інтеграція нових джерел даних; узгоджена продуктивність у великих масштабах.

Автоматизований збір даних може бути реалізований за допомогою різних підходів, від повністю програмних рішень до платформ без коду. Програмні методи з використанням таких мов, як Python, R або Java, пропонують максимальну гнучкість та контроль, дозволяючи розробникам створювати власні скрапери, інтеграції API та конвеєри обробки даних [27].

Платформи без коду, такі як Zapier та Make, надають візуальні інтерфейси для підключення джерел даних та автоматизації робочих процесів без написання коду, роблячи автоматизацію доступною для бізнес-користувачів. Також поширені гібридні підходи, що поєднують обидва методи, де інструменти без коду обробляють прості завдання, тоді як користувацький код керує складними вимогами. Корпоративні рішення, такі як Informatica та Talend, пропонують комплексні функції, але вимагають значних інвестицій. Вибір залежить від технічної експертизи, бюджету та конкретних випадків використання.

Вебскрапинг – це універсальний спосіб автоматичного збору даних з веб-сайтів. Базові інструменти скрапингу, такі як BeautifulSoup, можуть витягувати текст і числа з вебсторінок, тоді як більш просунуті інструменти, такі як Scrapy, можуть обробляти складні макети та збирати зображення та документи. Однак, вебскрапинг має обмеження – вебсайти можуть блокувати скрапери, а зміни на сайті можуть порушити процес збору даних. Такі інструменти, як Firecrawl, допомагають вирішити дані проблеми, адаптуючись до змін на вебсайті та обробляючи динамічний контент.

В умовах наявності великої кількості інструментів для автоматизованого збору даних, доцільно розглянути практичний приклад застосування Firecrawl, який демонструє ключові принципи сучасних підходів до отримання інформації. У

цьому прикладі представлено спосіб отримання структурованих даних про продукт з Amazon (рис. 1.1):

```
from firecrawl import FirecrawlApp
from pydantic import BaseModel, Field
from typing import Optional
from dotenv import load_dotenv
load_dotenv()
# Визначаємо структуру даних, яку необхідно зібрати
class Product(BaseModel):
    name: str = Field(description="Назва продукту")
    price: float = Field(description="Поточна ціна в доларах США")
    description: Optional[str] = Field(description="Опис продукту")
    rating: Optional[float] = Field(description="Рейтинг клієнта з 5")
    reviews_count: Optional[int] = Field(description="Кількість відгуків клієнтів")
# Ініціалізуємо інструмент збору даних
app = FirecrawlApp()
# Збираємо дані зі сторінки продукту
result = app.extract(
    urls=["https://www.amazon.com/dp/B094DYPM88/"],
    params={
        "prompt": "Витягти інформацію про продукт на основі наданої схеми.",
        "schema": Product.model_json_schema(),
    },
)
# Обробка результатів
product = Product(**result["data"])
print(f"Product: {product.name}")
print(f"Ціна: ${product.price}")
print(f"Рейтинг: {product.rating}/5 ({product.reviews_count} відгуків)")
```

Рис. 1.1. Фрагмент коду збору та обробки структурованих даних з Amazon за допомогою Firecrawl

```
Продукт: Ергономічна підставка для зап'ястя Razer для клавіатур Tenkeyless: плюшева подушка з шкіряної піни з ефектом пам'яті - протиковзкі гумові ніжки
Ціна: $19.99
Рейтинг: 4.5/5 (9964 відгуки)
```

Рис. 1.2. Результат виконання коду збору даних з Amazon за допомогою Firecrawl

Наведений вище код демонструє сучасний підхід до збору вебданих за допомогою структурованих схем та збору на основі штучного інтелекту.

Визначаючи модель Pydantic, точно отримуємо, яку інформацію про продукт можна зібрати зі сторінок Amazon, включаючи назву, ціну, опис, рейтинг та кількість відгуків. Потім FirecrawlApp використовує дану схему для

інтелектуальної ідентифікації та вилучення відповідних даних, не покладаючись на крихкі селектори CSS або вирази XPath.

Даний підхід пропонує кілька суттєвих переваг порівняно з традиційними методами веб-скрейпінгу. Колекція на основі схеми з використанням Pydantic забезпечує узгоджені формати даних та вбудовану валідацію. Вилучення на основі штучного інтелекту усуває необхідність підтримувати крихкі селектори, які ламаються, коли вебсайти змінюються. Система може ефективно обробляти кілька сторінок паралельно через параметр urls, забезпечуючи надійну обробку помилок та автоматичні механізми повторної спроби. Крім того, вона автоматично стандартизує формати даних, перетворюючи ціни, рейтинги та інші поля у відповідні типи даних [27].

Переходячи від традиційного вебскрапингу до інтелектуального вилучення даних, другий метод значно зменшує накладні витрати на обслуговування, одночасно підвищуючи надійність. Структурований підхід спрощує адаптацію процесу збору в міру розвитку вебсайтів, гарантуючи, що вилучені дані збережуть стабільну якість та формат.

Firecrawl базується на цих принципах, щоб забезпечити комплексний механізм скрапингу з можливостями, що виходять за рамки базового збору структурованих даних.

Інструменти штучного інтелекту дали змогу автоматично збирати нові типи даних. Наприклад: інструменти оптичного розпізнавання символів (OCR), такі як Tesseract [23], перетворюють друкований або рукописний текст на цифровий; інструменти обробки природної мови (NLP), такі як spaCy, знаходять важливу інформацію у звичайному тексті, таку як імена та дати.

А тепер розглянемо основні етапи збору текстових даних.

1. *Постановка задачі* – визначення мети збору даних (наприклад, аналіз відгуків, новин, соціальних мереж); вибір джерел: сайти, блоги, форуми, соцмережі, новинні портали тощо.

2. *Дослідження джерела даних*: аналіз структури вебсайту (HTML, DOM); виявлення URL-шаблонів, пагінації, API чи RSS-каналів; перевірка політики сайту (robots.txt, умови використання).

3. *Отримання доступу до даних*: використання бібліотек для HTTP-запитів (наприклад, `requests`, `httpx`); у випадку динамічного контенту – застосування інструментів для рендерингу JavaScript (Selenium, Playwright); якщо є офіційний API – краще використовувати його.

4. *Парсинг тексту*: розбір HTML (наприклад, `BeautifulSoup`, `lxml`); вибір потрібних елементів (заголовки, параграфи, відгуки); видалення непотрібного (реклама, меню, банери).

5. *Попередня обробка зібраних текстів*: очистка від HTML-тегів, скриптів, спецсимволів; усунення дублікатів; нормалізація кодування (UTF-8).

6. *Збереження даних*: експорт у форматах: CSV, JSON, TXT або база даних (MongoDB, PostgreSQL); логування процесу (що і коли зібрано).

1.4 Кореляційний аналіз як інструмент виявлення взаємозв'язків між змінними

Кореляційний аналіз є одним із ключових статистичних методів, що використовується для виявлення та кількісної оцінки ступеня взаємозв'язку між змінними. Дане застосування забезпечує можливість встановити, наскільки зміни однієї величини супроводжуються змінами іншої, а також визначити напрямок та силу цього взаємозв'язку. На відміну від простого описового аналізу, кореляційний підхід дає змогу перейти від констатації фактів до більш глибокого розуміння структури даних, що є критично важливим у процесі наукового дослідження [1, с. 268].

Основним кількісним показником у кореляційному аналізі є коефіцієнт кореляції, найпоширенішою формою якого є коефіцієнт Пірсона. Він вимірює

лінійну залежність між двома метричними змінними і набуває значень у межах від –1 до +1. Значення, близькі до +1, свідчать про сильний прямий зв’язок, близькі до –1 – про сильний обернений зв’язок, а значення, близькі до 0, вказують на відсутність лінійної залежності [14, с. 46]. У випадках, коли досліджувані змінні не відповідають вимогам нормальності розподілу або мають порядкову шкалу, застосовують рангові коефіцієнти Спірмена чи Кендалла, що забезпечують надійні результати в умовах нелінійності або наявності викидів [2, с. 424].

Застосування кореляційного аналізу дозволяє не лише встановити взаємозв’язки, але й сформувати основу для подальшого моделювання, зокрема регресійного. Виявлені кореляційні залежності можуть слугувати підґрунтям для побудови прогнозних моделей, уточнення гіпотез дослідження та визначення ключових факторів, які впливають на результуючі показники. Водночас варто підкреслити, що сам по собі кореляційний аналіз не дає змоги робити висновки про причинно-наслідкові зв’язки, оскільки кореляція лише відображає співзмінність ознак, але не встановлює причинності [3, с. 284].

Таким чином, кореляційний аналіз виступає ефективним інструментом первинного статистичного дослідження, який забезпечує об’єктивну оцінку структури взаємозв’язків між змінними та створює методологічну основу для подальших аналітичних процедур у межах наукового дослідження [9, с. 241].

Подальший розвиток методології кореляційного аналізу зумовлений необхідністю глибшого розуміння складних соціально-економічних, технічних та природничих систем, у яких взаємозв’язки між змінними часто є багатовимірними та динамічними. Традиційний підхід, що базується на обчисленні парних коефіцієнтів кореляції, дає змогу оцінити взаємодію лише між двома змінними одночасно. Однак у реальних дослідницьких ситуаціях важливо враховувати вплив множини факторів, що потребує застосування більш розширених методів, таких як часткова та множинна кореляція. Часткова кореляція дозволяє визначити ступінь зв’язку між двома змінними при фіксації впливу третіх змінних, що дає

змогу відокремити прямі та опосередковані залежності. Множинна кореляція, у свою чергу, забезпечує оцінювання інтегрального зв’язку між однією залежною змінною та групою незалежних, що особливо корисно в економетричних та соціологічних дослідженнях [13].

На практиці побудова кореляційної матриці допомагає виявити потенційні проблеми, такі як мультиколінеарність – надмірно сильний зв’язок між незалежними змінними, що може призвести до некоректних результатів у регресійному аналізі. Аналіз кореляційної структури також дозволяє визначити змінні, які не несуть змістовної інформації, або ті, що дублюють одна одну, що може вплинути на інтерпретацію результатів і якість статистичних моделей. У цьому контексті кореляційний аналіз відіграє роль фільтра, який дає змогу підготувати дані для подальших етапів обробки та моделювання [4, с. 92].

Кореляційний аналіз широко застосовується в різних галузях науки. У соціально-економічних дослідженнях він дає змогу визначити взаємозв’язки між макроекономічними індикаторами, рівнем добробуту населення, продуктивністю праці, динамікою ринків тощо. У психології кореляційний підхід використовується для аналізу зв’язків між когнітивними, поведінковими та емоційними змінними, що дозволяє визначати ключові фактори впливу на поведінку людини та розробляти прогностичні моделі. У природничих науках кореляційний аналіз забезпечує інструмент для виявлення закономірностей у складних біологічних та екологічних системах, де змінні часто є взаємозалежними і реагують на сукупність зовнішніх чинників [5, с. 170].

Водночас важливо дотримуватися обережності при інтерпретації результатів кореляційного аналізу. Однією з поширених помилок є ототожнення кореляції з причинністю. Високий коефіцієнт кореляції не гарантує наявності причинно-наслідкового зв’язку, оскільки взаємозв’язок може бути зумовлений низкою прихованих чинників чи випадковими збігами. Наприклад, два показники можуть одночасно змінюватися під впливом третього, або один із показників може бути

похідним від іншого. Тому результати кореляційного аналізу завжди повинні розглядатися у поєднанні з теоретичними передумовами дослідження, статистичними моделями та контекстною інформацією про предметну область.

Окрему увагу в сучасних дослідженнях приділяють проблемі коректного вимірювання статистичної значущості кореляційних коефіцієнтів. Зазвичай для цього використовують t-тест, який дозволяє визначити, чи є отриманий коефіцієнт кореляції статистично відмінним від нуля [16, с. 243]. Проте в умовах великих масивів даних, характерних для актуальних досліджень у сфері Data Science, навіть слабкі кореляції можуть виявлятися статистично значущими, що потребує додаткової уваги до практичної значущості результатів. У таких випадках застосовують також довірчі інтервали для коефіцієнтів кореляції, що дозволяє отримати більш точне уявлення про їхню стабільність та надійність [7, с. 240].

Широке використання кореляційного аналізу в сучасних дослідженнях також пов’язане з розвитком методів візуалізації даних. Графічне представлення кореляційної структури, наприклад за допомогою теплових карт чи діаграм розсіювання, дає змогу швидко і наочно ідентифікувати ключові взаємозв’язки та визначити потенційні напрями для подальшого аналізу. Візуалізація допомагає виявляти нелінійні залежності, які можуть бути неочевидними при використанні лише числових показників і підкреслює важливість комплексного підходу, що поєднує числовий, графічний та змістовний аналіз даних.

У контексті цифровізації та зростання обсягів інформації кореляційний аналіз набуває дедалі більшого значення у сфері машинного навчання. Хоча багато сучасних алгоритмів автоматично визначають структуру взаємозв’язків між змінними, попередній кореляційний аналіз залишається одним з ключових етапів підготовки даних. Він дозволяє оптимізувати вибір ознак, зменшити розмірність даних, уникнути мультиколінеарності й покращити точність побудованих моделей [8].

Крім того, кореляційний аналіз є важливим інструментом Explainable AI – напрямку, що забезпечує інтерпретованість моделей штучного інтелекту, адже він допомагає пояснити, які змінні мають найбільший вплив на кінцевий результат [15, с. 266].

Незважаючи на очевидні переваги, кореляційний аналіз має певні обмеження. Найважливішим серед них є чутливість до викидів – одиничних екстремальних значень, які можуть суттєво спотворювати оцінку кореляційного зв'язку. Для зменшення їхнього впливу рекомендується застосовувати рангові методи або проводити попередню обробку даних. Крім того, класичні кореляційні коефіцієнти оцінюють лише лінійні залежності, тоді як у багатьох реальних процесах взаємозв'язки можуть бути нелінійними. Для таких випадків використовують спеціалізовані методи, зокрема коефіцієнт кореляції Метьюса, інформаційно-теоретичні міри або методи машинного навчання [8].

Отже, кореляційний аналіз є багатофункціональним і універсальним статистичним інструментом, який забезпечує можливість глибокого та всебічного дослідження взаємозв'язків між змінними. Його застосування дає змогу не лише кількісно описати структуру даних, але й сформувати підґрунтя для подальшого аналізу, моделювання та прийняття рішень.

Висновки до першого розділу

Проведений аналіз теоретичних основ досліджуваної проблематики дозволяє зробити ряд висновків щодо сутності та структури текстових даних, поведінкових показників користувачів та методів їх збору і аналізу, що є ключовими для подальшого дослідження кореляцій між цими показниками.

Встановлено, що дані можуть бути класифіковані за різними ознаками, включаючи їх формат, структуру та призначення. Основними типами даних є текстові, структуровані (табличні), графічні, аудіо, відео, геопросторові та архівні.

З точки зору дослідження поведінкових реакцій користувачів на текстовий контент, особливе значення мають саме текстові дані, оскільки вони формують основу для побудови аналітичних моделей, визначення настроїв, виявлення тематики та інтерактивності користувачів. Текстові дані є основою інформації, яка представлена у вигляді послідовності символів і може публікуватися у відкритих форматах, таких як *.txt, *.RTF, *.ODT, *.DOCX та *.PDF.

Текстові дані мають тісний взаємозв’язок із типами даних у програмуванні, що забезпечує правильне зберігання, обробку та аналіз інформації. Встановлення відповідного типу даних – числового, рядкового, логічного, масиву, дати чи часу – дозволяє уникнути помилок у трактуванні інформації, забезпечити узгодженість даних та підвищити якість подальших аналітичних процедур. Зокрема, використання структурованих типів даних, таких як масиви та перелічувані типи, сприяє правильному збереженню складових елементів текстової інформації та формуванню її коректної моделі для статистичного аналізу.

Розглянуто поведінкові показники користувачів як інтегральний елемент дослідження взаємозв’язку текстових даних із активністю та реакціями аудиторії. Метрики соціальних мереж, такі як рівень залученості, вподобання, коментарі, поширення, коефіцієнт конверсії, а також настроєві показники користувачів, дозволяють кількісно оцінити ефективність контенту та прогнозувати реакцію аудиторії. Особливої уваги заслуговує інтеграція поведінкових показників із текстовими характеристиками повідомлень, що дає змогу виявляти закономірності емоційного тону тексту та активності користувачів, а також оцінювати потенціал поширення інформації в онлайн-спільнотах.

Також виділено методи збору та попередньої обробки текстових даних, що забезпечують їхню ефективну та системну інтеграцію для подальшого аналізу. Автоматизований збір даних за допомогою скрапингу та використання інструментів штучного інтелекту та машинного навчання, таких як OCR та NLP, дозволяє отримувати великі масиви інформації у структурованому та

стандартизованому вигляді, що забезпечує швидку обробку даних, зниження ризику людських помилок, масштабованість процесу та можливість подальшого застосування отриманих даних для статистичного та кореляційного аналізу.

Застосування кореляційного аналізу дозволяє кількісно оцінити зв’язок між показниками текстових даних та поведінковими метриками користувачів, сформувати основу для побудови прогнозних моделей та уточнення гіпотез дослідження. Застосування розширених методів, таких як часткова та множинна кореляція, дає змогу відокремити прямі та опосередковані залежності, що є важливим для побудови надійних моделей у сфері соціальних медіа та аналітики текстових даних.

Таким чином, проведений огляд методів збору, обробки та аналізу текстових даних і поведінкових показників користувачів підтверджує необхідність комплексного підходу, який поєднує автоматизований збір даних, їхню стандартизацію та застосування статистичних методів, включаючи кореляційний аналіз. Такий підхід дозволяє забезпечити високий рівень точності, надійності та відтворюваності результатів, що є критично важливим для подальшого дослідження взаємозв’язку між змістом текстових повідомлень та активністю користувачів у цифровому середовищі. Виявлені закономірності та методологічні підходи створюють надійну основу для розробки моделей прогнозування залученості аудиторії, виявлення тенденцій у поведінкових показниках та побудови ефективних стратегій аналізу інформації у соціальних мережах та онлайн-спільнотах.

РОЗДІЛ 2
ІНСТРУМЕНТИ ТА АЛГОРИТМИ АНАЛІЗУ ДАНИХ У PYTHON

2.1 Бібліотеки для збору та обробки текстових даних

У процесі опрацювання великих обсягів текстової інформації ключову роль відіграють інструменти, що забезпечують збирання, очищення та подальший лінгвістичний аналіз даних. Сучасні бібліотеки Python дають змогу автоматизувати ці етапи й суттєво спростити роботу з неструктурованими даними. Зокрема, Requests**,** BeautifulSoup та Scrapy забезпечують ефективний веб-скрапинг і отримання даних із веб-ресурсів, тоді як NLTK і spaCy надають потужні засоби для мовної обробки: токенізації, лематизації, аналізу частин мови, визначення залежностей тощо [10].

Отже, розглянемо бібліотеку для збору даних Requests та бібліотеку для обробки даних BeautifulSoup. Requests – це проста HTTP-бібліотека для Python, вона надає інтуїтивно зрозумілий API для створення HTTP-запитів та обробки відповідей у простий та зрозумілий для людини спосіб. Requests є одним із найпопулярніших HTTP-клієнтів на Python ‘11].

Деякі з ключових функцій, що пропонуються цією бібліотекою, включають комплексний API, що охоплює всі методи HTTP, обробку відповідей, налаштування запитів, автентифікацію, керування SSL-сертифікатами тощо. Крім того, модуль Python Requests підтримує HTTP/1.1 «з коробки».

Найпростіший та рекомендований спосіб встановлення Requests – через pip. Зокрема, пакет pip, пов’язаний з бібліотекою Requests, називається requests.

Перш ніж встановлювати будь-який зовнішній пакет, потрібно створити віртуальне середовище для проекту. Активувати нове віртуальне середовище, а

потім ввести команду у своєму терміналі, щоб встановити бібліотеку Requests: *$ python -m pip install requests.*

Для реалізації прототипу системи збору веб-даних доцільно створити окремий файл у середовищі розробки Python. Даний файл, який можна назвати scraper.py, міститиме мінімальний обсяг коду для отримання HTML-контенту сторінок, необхідного для подальшого аналізу (рис. 2.1):

```
import requests
URL = "https://realpython.github.io/fake-jobs/"
page = requests.get(URL)
print(page.text)
```

Рис. 2.1. Фрагмент коду для отримання HTML-сторінки за допомогою бібліотеки requests

При запуску даного коду, він надсилає HTTP GET-запит на вказану URL-адресу та отримує дані HTML, які сервер надсилає назад, і зберігає ці дані в об'єкті Python.

Якщо вивести атрибут .text об'єкта page, то він виглядає так само, як HTML, який раніше перевіряли за допомогою інструментів розробника браузера та успішно отримано статичний вміст сайту з інтернету. Тепер отримано доступ до HTML сайту з скрипта Python.

Наразі бібліотека Requests у Python довела свою ефективність і зручність у роботі з веб-контентом. Лише кількома рядками коду можна отримати статичний HTML-ресурс з інтернету та підготувати його для подальшої обробки [38].

Проте деякі веб-сторінки містять інформацію, доступ до якої обмежений авторизацією. У таких випадках для отримання даних необхідно мати обліковий запис і виконати вхід у систему не лише через браузер, а й програмно за допомогою скрипта Python.

Бібліотека Requests має вбудовану можливість обробки автентифікації. За допомогою цих методів можна заходити на веб-сайти, роблячи HTTP-запит із скрипта Python, а потім отримувати інформацію, приховану за логіном. При

цьому, не потрібно буде входити в систему, щоб отримати доступ до інформації на дошці вакансій.

Функції Requests в Python. Python Requests пропонує кілька унікальних функцій для створення різних HTTP-запитів. Дані запити є основою для отримання необхідних даних з веб-серверів. При необхідності отримання доступу до веб-сторінки або отримати інформацію з API, відбувається звернення до надійного запиту GET для найкращих результатів.

GET Requests. Запити GET розроблені для отримання інформації та є безпечними за своєю суттю, тобто вони не змінюють дані на сервері. Натомість вони отримують запитувані дані, не спричиняючи жодних змін, що робить запити GET ідеальними для таких завдань, як перегляд веб-сайту, отримання результатів пошуку або доступ до загальнодоступних даних.

Однією з ключових функцій GET-запиту є використання URL-адреси (уніфікований локатор ресурсу). URL-адреса визначає місце розташування веб-ресурсу, до якого необхідно отримати доступ. При ініціалізації GET-запиту сервер отримує запит на надання зазначеної інформації та відповідає відповідним ресурсом [35]. На рисунку 2.2 зображено приклад простого GET-запиту на Python з використанням бібліотеки Requests:

```
import requests
response = requests.get('https://example.com')
```

Рис. 2.2. Приклад GET-запиту

У цьому прикладі надсилається GET-запит на «https://example.com/data» для отримання даних з цього веб-ресурсу. Об'єкт відповіді response містить дані, надіслані сервером.

GET-запити стосуються не лише отримання даних, але й контексту, який вони несуть. Також запити можуть включати параметри в URL-адресу, щоб вказати, що саме шукається. Наприклад, якщо здійснювати пошук статей про «Python», то можна надіслати запит GET наступним чином (рис. 2.3):

```
import requests
response = requests.get('https://example.com/search?q=python')
```

Рис. 2.3. Приклад GET-запиту для пошуку статей

POST Requests. HTTP-запити POST доповнюють запити GET, що розглядалися раніше. Хоча запити GET призначені виключно для отримання даних, запити POST призначені для надсилання цих даних на веб-сервери. Вони відіграють вирішальну роль у веб-взаємодії, що включає надсилання форм, оновлення даних або будь-яку операцію, де потрібно надсилати інформацію на сервер.

При заповненні онлайн-форми реєстрації користувач вводить свої персональні дані, зокрема ім’я, електронну пошту та пароль. Після натискання кнопки «Надіслати» форму надсилається на сервер у вигляді HTTP-запиту POST, що забезпечує обробку наданої інформації. Такий механізм дозволяє серверу створити обліковий запис користувача або виконати інші необхідні дії на основі отриманих даних. На рисунку 2.4 зображено приклад базового POST-запиту з використанням Python Requests:

```
import requests
response = requests.post('https://example.com/login', data={'username': 'user', 'password': 'pass'})
```

Рис. 2.4. Приклад POST-запиту

У цьому прикладі відправляється POST-запит на «https://example.com/login», надаючи серверу словник «data», що містить ім'я користувача та пароль. Потім сервер обробляє ці дані, зазвичай автентифікуючи користувача.

Методи та атрибути об'єкта Response. Об'єкт Response містить величезну кількість інформації про HTTP-відповідь, яку отримує запит. Розуміння того, як отримати доступ до методів та атрибутів об'єкта Response та використовувати їх, має вирішальне значення для вилучення всіх необхідних значущих даних з веб-серверів.

Після розгляду призначення об’єкта Response доцільно детальніше проаналізувати способи доступу до його методів та атрибутів, що дозволить

ефективно опрацьовувати дані, отримані від веб-серверів, та реалізовувати необхідні операції під час взаємодії з веб-ресурсами.

Однією з найбільш поширених операцій при роботі з об’єктом Response є доступ до його вмісту, який може містити текст HTML, дані у форматі JSON або інші типи даних. Для отримання текстового представлення відповіді використовується атрибут text [36].

Змінна content містить HTML-вміст веб-сторінки, отриманої з «https://example.com». Відповідно можна обробити поданий вміст, отримати інформацію або зберегти її у файлі для подальшого використання.

Коди стану HTTP надають важливу інформацію про результат запиту та дозволяють отримати доступ до коду стану за допомогою атрибута status_code (рис. 2.5):

```
import requests
response = requests.get('https://example.com')
print(response.status_code)
```

Рис. 2.5. Приклад отримання коду стану HTTP за допомогою атрибута status_code у Python

Змінна status_code тепер містить код стану HTTP відповіді. Наприклад, код стану «200» вказує на успішний запит, тоді як «404» означає, що запитуваний ресурс не знайдено.

HTTP-заголовки містять метадані про відповідь. Також можна отримати до них доступ як до словника за допомогою атрибута headers (рис. 2.6):

```
import requests
response = requests.get('https://example.com')
print(response.headers)
```

Рис. 2.6. Приклад доступу до HTTP-заголовків через атрибут headers

Змінна headers містить словник HTTP-заголовків. Дані заголовки часто містять таку інформацію, як тип вмісту, тип сервера та дата відповіді, що може бути цінною інформацією для розуміння поведінки сервера.

Іноді веб-сервери використовують файли cookie для зберігання інформації про сеанс, що дозволяє отримати доступ до цих файлів cookie та керувати ними за допомогою атрибута cookies (рис. 2.7):

```
import requests
response = requests.get('https://example.com')
print(response.cookies)
```

Рис. 2.7. Використання атрибута cookies

Після того, як отримати об'єкт Response та доступ до його атрибутів, наступним важливим кроком є ефективна обробка його вмісту. Дана обробка залежить від того, чи вміст відповіді знаходиться у текстовому чи JSON форматі.

Обробка текстової відповіді. Під час роботи з текстовими даними, такими як HTML з веб-сторінки, обробка включає розбір, вилучення та виконання перетворень. Python пропонує такі бібліотеки, як BeautifulSoup та LXML, для розбору HTML- або XML-вмісту. Пізніше можна використовувати дані бібліотеки для навігації по структурі HTML та вилучення необхідної інформації. На рисунку 2.8 зображено приклад обробки HTML-вмісту за допомогою BeautifulSoup [38]:

```
import requests
from bs4 import BeautifulSoup
response = requests.get('https://example.com')
print(response.text)
soup = BeautifulSoup(response.text, 'html.parser')
title = soup.find('title')
print(title.text)
paragraphs = soup.find_all('p')
for paragraph in paragraphs:
print(paragraph.text)
```

Рис. 2.8. Обробка HTML-вмісту за допомогою BeautifulSoup

Спочатку здійснюється отримання HTML-контенту з ресурсу «https://example.com». Надалі контент аналізується за допомогою бібліотеки BeautifulSoup, що забезпечує спрощене вилучення окремих елементів, таких як заголовки сторінки та абзаци.

Обробка JSON-контенту. Під час роботи з даними JSON Python надає вбудовану підтримку для їх декодування в нативні структури даних за допомогою методу .json() об'єкта Response. Даний метод аналізує JSON-контент і повертає словник Python. На рисунку 2.9 представлено код обробки JSON-контенту:

```
import requests
response = requests.get('https://example.com/data.json')
data = response.json()
print(data['key'])
```

Рис. 2.9. Приклад обробки JSON-контенту

Необхідні кроки, щоб ефективно вилучити JSON-дані: GET-запит до API або веб-сервісу, який повертає JSON-дані; отриману відповідь необхідно зберегти у змінній, такій як response; використання методу .json() об'єкта Response, щоб декодувати JSON-контенту словник Python; отримайти доступ до певних даних у словнику за допомогою ключів, що дозволить отримати необхідну інформацію. Алгоритм отримання та аналізу основних SEO-тег за допомогою Python Requests та BeautifulSoup:

– GET-запит на URL-адресу веб-сторінки, яку необхідно проаналізувати;

– фіксація відповіді у змінній, зазвичай з назвою response;

– доступ до HTML-вмісту відповіді за допомогою атрибута text;

– аналіз HTML-вмісту за допомогою BeautifulSoup, вказавши тип парсера (наприклад, «html.parser» або «lxml») [39];

– методи BeautifulSoup для пошуку та вилучення основних SEO-тегів, таких як <title>, <meta> та <h1>.

На рис. 2.10 представлено приклад, що ілюструє, як отримати SEO-теги:

```
import requests
from bs4 import BeautifulSoup
response = requests.get('https://example.com')
soup = BeautifulSoup(response.text, 'html.parser')
title = soup.find('title')
print(title.text)
meta = soup.find('meta', {'name': 'description'})
print(meta['content'])
h1 = soup.find('h1')
print(h1.text)
```

Рис. 2.10. Отримання SEO-тегів

Для забезпечення надійної роботи при взаємодії з веб-ресурсами доцільно передбачити механізми повторного виконання невдалих запитів у Python. Зокрема, слід використовувати обробку винятків, оточуючи код запиту блоками try-except та перехоплюючи помилки, наприклад, requests.exceptions, що дозволяє запобігти аварійному завершенню програми. Додатково рекомендується реалізувати автоматичне повторення невдалих запитів за допомогою відповідних бібліотек або власної логіки з відкладенням між спробами, що підвищує стабільність роботи та запобігає перевантаженню серверів. При цьому важливо визначити максимальну кількість повторних спроб, щоб уникнути нескінченних циклів та припинити виконання запиту у разі систематичних помилок. На рисунку 2.11 представлено приклад реалізації повторних спроб запиту за допомогою бібліотеки повторних спроб:

```
import requests
from retrying import retry
@retry(stop_max_attempt_number=3)
def get(url):
response = requests.get(url)
response.raise_for_status()
return response
try:
response = get('https://example.com')
except requests.exceptions.RequestException as e:
print(e)
```

Рис. 2.11. Приклад реалізації повторних спроб

Методи HTTP, що підтримуються Requests:

– *PUT*: використовується для оновлення або заміни існуючого ресурсу на сервері;

– *DELETE*: використовується для видалення ресурсу на сервері;

– *HEAD*: подібний до GET, але отримує лише заголовки, а не вміст, що може бути корисним для перевірки існування ресурсу;

– *OPTIONS*: використовується для отримання інформації про параметри зв'язку для ресурсу;

– *PATCH*: використовується для часткового внесення змін до ресурсу.

NLTK (Natural Language Toolkit) – це популярна бібліотека Python для обробки природної мови (NLP). Вона надає нам різні бібліотеки для обробки тексту з великою кількістю тестових наборів даних. За допомогою NLTK можна виконувати різноманітні завдання, такі як токенізація, візуалізація дерева розбору тощо (рис. 2.12).

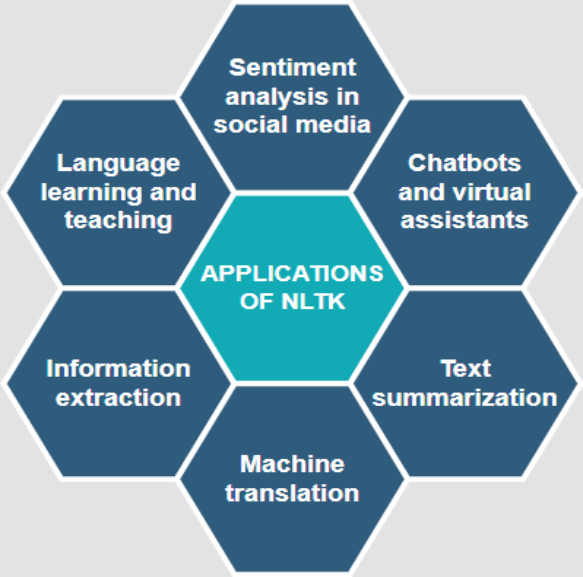

Рис. 2.12. Можливості модуля NLTK

Токенізація. Розбиття тексту на менші одиниці називається токенами. Ідея полягає в тому, щоб відокремити кожне слово та створити словник таким чином, щоб можна було однозначно представити всі слова у списку. Числа, слова тощо – все підпадає під токени (рис. 2.13) [12, с. 113].

Перетворення на малі літери. Для забезпечення надійної роботи при взаємодії з веб-ресурсами доцільно передбачити механізми повторного виконання невдалих запитів у Python. Зокрема, слід використовувати обробку винятків, оточуючи код запиту блоками try-except та перехоплюючи помилки, наприклад, requests.exceptions, що дозволяє запобігти аварійному завершенню програми.

Додатково рекомендується реалізувати автоматичне повторення невдалих запитів за допомогою відповідних бібліотек або власної логіки з відкладенням між спробами, що підвищує стабільність роботи та запобігає перевантаженню

серверів. При цьому важливо визначити максимальну кількість повторних спроб, щоб уникнути нескінченних циклів та припинити виконання запиту у разі систематичних помилок.text = re.sub(r"[^a-zA-Z0-9]", " ", text.lower()).

```
words = text.split()
print(words)
# output -> ['natural', 'language', 'processing', 'is', 'an', 'exciting', 'area', 'huge', 'budget', 'have', 'been', 'allocated', 'for', 'this']
```

Рис. 2.13. Розбиття тексту на окремі слова за допомогою методу split()

Видалення стоп-слів. При роботі з текстовими даними застосовуються функції для моделювання, часто виникає проблема шуму у даних. До такого шуму відносяться стоп-слова – загальні слова, наприклад, the, he, her тощо, які не несуть корисної інформації для моделі та зазвичай видаляються на етапі попередньої обробки. Використовуючи бібліотеку NLTK, можна переглянути повний набір стандартних стоп-слів англійською мовою (рис. 2.14), що спрощує процес очищення тексту та підвищує ефективність подальшого аналізу.

```
# Імпортуємо модуль стоп-слів з бібліотеки NLTK
from nltk.corpus import stopwords
# Розбиваємо текст на окремі слова (токени)
words = text.split()
print("Початкові токени:", words)
# output -> ['natural', 'language', 'processing', 'is', 'an', 'exciting', 'area', 'huge', 'budget', 'have', 'been', 'allocated', 'for', 'this']
# Отримуємо набір англійських стоп-слів
stop_words = set(stopwords.words("english"))
# Видаляємо стоп-слова з токенів
filtered_words = [word for word in words if word.lower() not in stop_words]
print("Токени після видалення стоп-слів:", filtered_words)
# output -> ['natural', 'language', 'processing', 'exciting', 'area', 'huge', 'budget', 'allocated']
```

Рис. 2.14. Видалення стоп-слів з тексту за допомогою NLTK для отримання лише значущих токенів

Стемінг. У будь-якому тексті можна знайти багато слів, таких як «грати», «грав», «грайливо» тощо, які мають кореневе слово, і всі вони передають однакове значення. Таким чином, можливо виділити кореневу форму слова та відкинути решту морфологічних елементів. Утворене кореневе слово називається «основа», і не обов'язково, щоб основа існувала та містила значення, просто додаючи суфікс і

префікс, ми генеруємо основи. NLTK надає нам пакети PorterStemmer, LancasterStemmer та SnowballStemmer (рис. 2.15).

```
from nltk.stem.porter import PorterStemmer
# Reduce words to their stems
stemmed = [PorterStemmer().stem(w) for w in words]
print(stemmed)
# output -> ['natur', 'languag', 'process', 'excit', 'area', 'huge', 'budget', 'alloc']
```

Рис. 2.15. Зведення слів до кореневих форм за допомогою PorterStemmer

Лематизація слів: приведення слова до його базової (словникової) форми – леми – за допомогою WordNet, що містить словникові форми слів, які використовуються лематизатором NLTK для точного визначення лем кожного слова (рис. 2.16).

```
from nltk.stem.wordnet import WordNetLemmatizer
# Reduce words to their root form
lemmed = [WordNetLemmatizer().lemmatize(w) for w in words]
print(lemmed)
#output -> ['natural', 'language', 'processing', 'exciting', 'area', 'huge', 'budget', 'allocated']
```

Рис. 2.16. Приклад лематизації слів за допомогою WordNet у Python

Стемінг набагато швидший за лематизацію, оскільки не потребує пошуку у словнику, а просто дотримується алгоритму для генерації кореневих слів.

2.2 Інструменти для роботи з поведінковими даними та їх збереження

Бібліотека pandas – це пакет маніпулювання даними в Python для табличних даних. Тобто, даних у формі рядків і стовпців, також відомих як DataFrames. Інтуїтивно, DataFrame можна уявити як аркуш Excel.

Функціональність pandas включає перетворення даних, такі як сортування рядків і взяття підмножин, для обчислення зведеної статистики, такої як середнє значення, зміну форми DataFrames та об'єднання DataFrames. pandas добре працює з іншими популярними пакетами Python для обробки даних, які часто називають екосистемою PyData, включаючи [18, с. 6]:

– NumPy для числових обчислень;

– Matplotlib, Seaborn, Plotly та інші пакети візуалізації даних;

– Scikit-learn для машинного навчання.

Далі наведено кілька основних функцій pandas: обробка відсутніх даних (так звана NaN); поля фреймів даних можна вставляти та видаляти, щоб фрейми даних були змінними; упорядкування даних у порядку зростання або спадання; фільтрація фреймів даних за будь-якою умовою; потужна функціональність групування для агрегації та перетворення даних; об'єднання, конкатенація та приєднання наборів даних; забезпечує зміну форми та поворот наборів даних з багатьма опціями.

На рисунку 2.17 наведено основні можливості модуля Pandas, які забезпечують ефективне опрацювання та аналіз даних у Python.

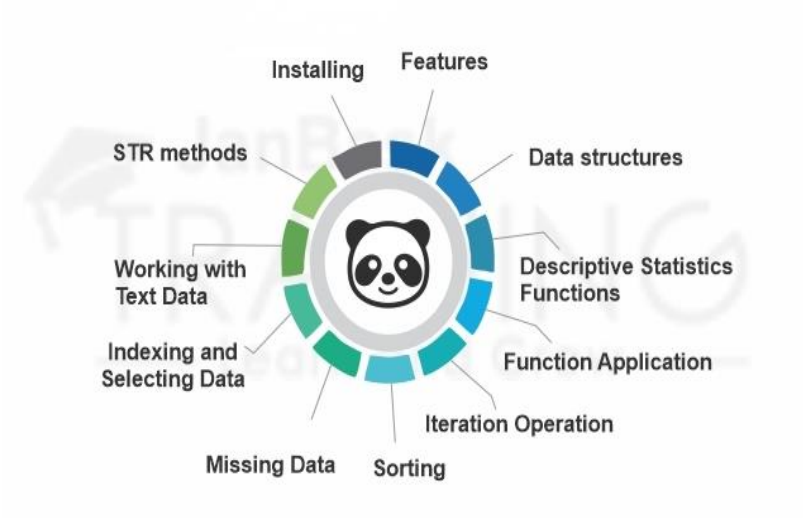

Рис. 2.17. Можливості модуля Pandas

NumPy – це бібліотека Python, яка надає просту, але потужну структуру даних: n-вимірний масив. Дана бібліотека вважається основою, на якій побудовано майже всю потужність інструментарію Python для обробки даних, і вивчення NumPy – це перший крок на шляху будь-якого спеціаліста з обробки даних Python [19, с. 504].

Основні переваги, які NumPy може принести у наступний код: більша швидкість: NumPy використовує алгоритми, написані на C, які завершуються за

наносекунди, а не за секунди; *менше циклів:* NumPy допомагає вам зменшити кількість циклів та уникнути заплутування в індексах ітерацій; *зрозуміліший код:* без циклів код буде більше схожий на рівняння, які ви намагаєтеся обчислити; *краща якість:* тисячі учасників працюють над тим, щоб NumPy був швидким, зручним та без помилок.

Завдяки цим перевагам NumPy є фактичним стандартом для багатовимірних масивів у науці про дані Python, і багато найпопулярніших бібліотек побудовані на його основі. Вивчення NumPy – це чудовий спосіб закласти міцну основу, розширюючи свої знання в більш специфічних областях науки про дані [30].

Головним об'єктом NumPy є однорідний багатовимірний масив, або таблиця елементів (зазвичай чисел), усіх одного типу, індексованих кортежем невід'ємних цілих чисел. У NumPy розмірності називаються осями [20, с. 410].

Наприклад, масив координат точки в тривимірному просторі, [1, 2, 1], має одну вісь. Дана вісь містить три елементи, і відповідно її довжина дорівнює 3. У наведеному нижче прикладі масив має 2 осі. Перша вісь має довжину 2, друга вісь має довжину 3.

```
1  0  0
0  1  2
```

Клас масиву NumPy називається ndarray. Він також відомий під псевдонімом array. Зауважимо, що numpy.array не є тим самим, що клас array.array зі стандартної бібліотеки Python, який обробляє лише одновимірні масиви та пропонує менше функціональності. Найважливішими атрибутами об'єкта ndarray є [21, с. 398]:

ndarray.ndim – кількість осей (вимірів) масиву;

ndarray.shape – виміри масиву: кортеж цілих чисел, що вказує розмір масиву в кожному вимірі. Для матриці з *n* рядками та *m* стовпцями, shape буде (n,m). Довжина кортежу shape, таким чином, дорівнює кількості осей, *ndim*;

ndarray.size – загальна кількість елементів масиву, що дорівнює добутку елементів shape;

ndarray.dtype – об'єкт, що описує тип елементів у масиві. Можна створювати або вказувати dtype, використовуючи стандартні типи Python. Крім того, NumPy надає власні типи. numpy.int32, numpy.int16 та numpy.float64;

ndarray.itemsize – розмір кожного елемента масиву в байтах. Наприклад, масив елементів типу float64 має розмір елемента 8 (=64/8), тоді як масив елементів типу complex32 має розмір елемента 4 (=32/8), що еквівалентно ndarray.dtype.itemsize;

ndarray.data – буфер, що містить фактичні елементи масиву. Зазвичай не потрібно використовувати поданий атрибут, оскільки ми звертатимемося до елементів масиву за допомогою засобів індексації [31, с. 498].

На рисунку 2.18 подано приклад роботи з масивом у NumPy, що демонструє базові операції та можливості цього модуля:

```
import numpy as np
# Створення масиву 'a'
a = np.arange(15).reshape(3, 5)
# Виведення масиву 'a'
print("--- Array 'a' ---")
print(a)
# Очікуваний вивід:
# array([[ 0,  1,  2,  3,  4],
#        [ 5,  6,  7,  8,  9],
#        [10, 11, 12, 13, 14]])
# Перевірка атрибутів масиву 'a'
print("\n--- Properties of 'a' ---")
print(f"a.shape: {a.shape}")        # Розмірність (рядки, стовпці)
print(f"a.ndim: {a.ndim}")          # Кількість вимірів
print(f"a.dtype.name: {a.dtype.name}") # Тип даних елементів
print(f"a.itemsize: {a.itemsize}")   # Розмір кожного елемента в байтах (int64 = 8 байт)
print(f"a.size: {a.size}")          # Загальна кількість елементів
# Перевірка типу масиву 'a'
print(f"type(a): {type(a)}")
# Створення масиву 'b'
b = np.array([6, 7, 8])
# Виведення масиву 'b'
print("\n--- Array 'b' ---")
```

```
print(b)
# Очікуваний вивід:
# array([6, 7, 8])
# Перевірка типу масиву 'b'
print(f"type(b): {type(b)}")
```

Рис. 2.18. Приклад роботи з масивом в NumPy

Створення масивів. Існує декілька способів створення масивів. Розглянемо створення масиву зі звичайного списку або кортежу Python за допомогою функції array. Тип результуючого масиву визначається з типу елементів у послідовностях (рис. 2.19).

```
import numpy as np

# Створення масиву 'a' з цілих чисел
a = np.array([2, 3, 4])
# Виведення масиву та його типу даних
print("Масив a:", a)
print("Тип даних a:", a.dtype) # Виведе: dtype('int64')

# Створення масиву 'b' з чисел з плаваючою комою
b = np.array([1.2, 3.5, 5.1])

# Виведення типу даних масиву 'b'
print("Масив b:", b)
print("Тип даних b:", b.dtype) # Виведе: dtype('float64')
```

Рис. 2.19. Створення масиву за допомогою NumPy

Мова запитів SQL для збереження даних.

SQL (структурована мова запитів) – це стандартна мова для взаємодії з даними в реляційних системах керування базами даних. Існує п’ять видів запитів – DDL, DML, DCL, TCL і DQL, кожен з них виконує певні дії (рисунок 2.20).

DDL, або data definition language, потрібен для визначення даних. Відповідні запити дозволяють налаштовувати базу даних – створювати з нуля та прописувати її структуру. Приклади DDL-запитів: CREATE, DROP, RENAME, ALTER [37, с. 770].

DML, або data manipulation language, потрібен, щоб керувати даними таблицях. Запити допомагають додавати, оновлювати, видаляти та вибирати дані. Приклади DML запитів:UPDATE, DELETE, INSERT.

DCL, або data control language, потрібен, щоб видавати чи відкликати права доступу користувачів. Приклади DCL-запитів: GRANT, REVOKE, DENY.

TCL, або transaction control language, потрібен, щоб керувати транзакціями та можуть бути запити, пов'язані з підтвердженням чи відкатом змін у базі даних. Приклади TCL запитів: COMMIT, ROLLBACK, BEGIN.

DQL, або data query language, використовується для отримання даних із бази та формування вибірок на основі заданих умов, що дозволяє аналізувати та опрацьовувати інформацію у зручному вигляді. Приклад DQL-запиту: SELECT.

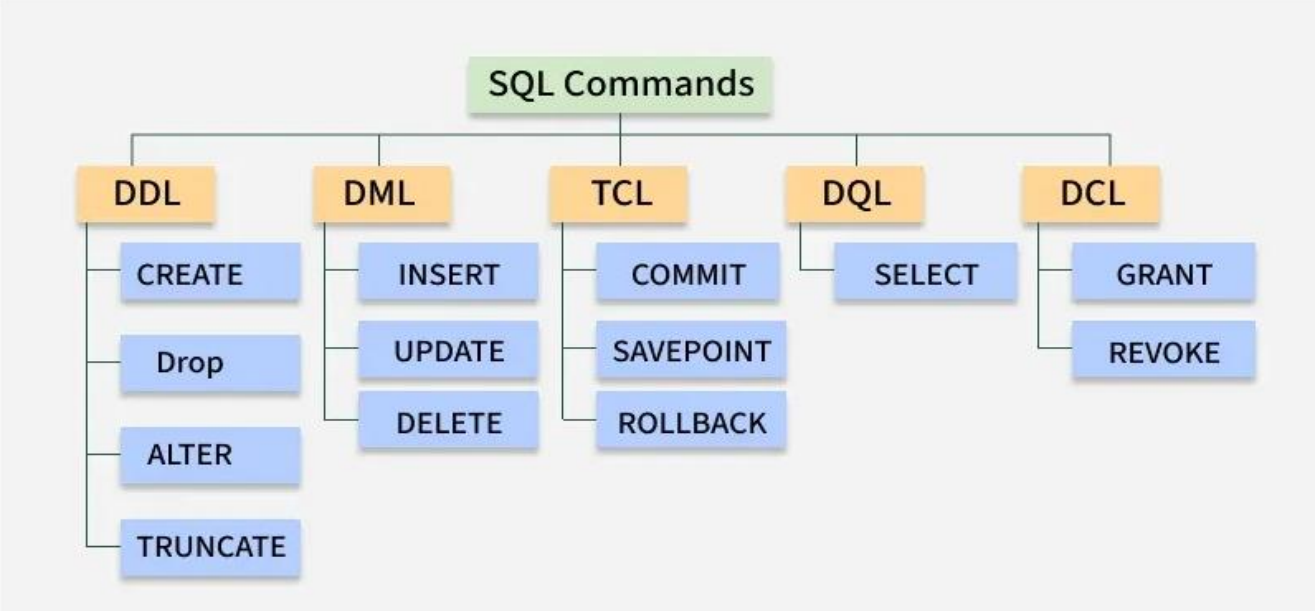

Рис. 2.20. Типи SQL команд

2.3 Засоби для статистичного аналізу та виявлення кореляцій

У сучасних умовах обробки даних статистичний аналіз і виявлення кореляцій відіграють ключову роль у прийнятті обґрунтованих рішень та побудові прогнозних моделей. Для ефективної реалізації цих завдань широко

застосовуються спеціалізовані програмні інструменти, серед яких особливе місце займають бібліотеки Python – Statsmodels та Scikit-learn. Вони надають потужні засоби для описової статистики, перевірки гіпотез, регресійного аналізу, а також побудови моделей машинного навчання, що дозволяє глибше досліджувати взаємозв’язки між змінними та виявляти приховані закономірності у даних [13]. Scikit-learn – це потужна бібліотека для машинного навчання, але вона оптимізована для малих та середніх наборів даних. Працюючи з великими наборами даних, вам потрібно ефективно їх обробляти.

Машинне навчання – це підгалузь штучного інтелекту, присвячена розумінню та створенню методів для імітації того, як навчаються люди. Дані методи включають використання алгоритмів та даних для покращення продуктивності певного набору завдань і часто належать до одного з трьох найпоширеніших типів навчання:

1) навчання з учителем: тип машинного навчання, який вивчає зв'язок між вхідними та вихідними даними;

2) навчання без учителя: тип машинного навчання, який вивчає базову структуру нерозміченого набору даних;

3) навчання з підкріпленням: метод машинного навчання, за якого програмний агент навчається виконувати певні дії в середовищі, що призводить до максимальної винагороди.

Scikit-learn, також відома як sklearn – це надійна бібліотека машинного навчання Python з відкритим кодом. Вона була створена для спрощення процесу впровадження машинного навчання та статистичних моделей у Python [22, с. 600].

Перший аспект sklearn – це дані; Scikit-learn постачається з деякими стандартними наборами даних машинного навчання, що усуває потребу у завантаженні даних із зовнішніх джерел [17, с. 493].

Прикладами іграшкових наборів даних, доступних у sklearn, є набір даних про іриску для класифікації та набір даних про діабет для регресії. У поданому

прикладі ми використовуватимемо набір даних про вино. Завантажимо його в пам'ять (рис. 2.21):

```
from sklearn.datasets import load_wine
wine_data = load_wine()
```

Рис. 2.21. Завантаження набору даних Wine з бібліотеки sklearn

Виконання наведеного вище коду повертає об'єкт, схожий на словник, що містить дані разом із метаданими про ці дані. Потрібні нам дані знаходяться в атрибуті .data, який повертає load_wine(). Щоб отримати до нього доступ як до атрибута екземпляра wine_data наступним чином необхідно: *wine_data.data,* що повертає масив N x M, де N – кількість вибірок, а M – кількість ознак [40].

Завантаження даних у pandas DataFrame, який набагато легше маніпулювати та аналізувати (рис. 2.22):

```
import pandas as pd
from sklearn.datasets import load_wine
wine_data = load_wine()
# Convert data to pandas dataframe
wine_df = pd.DataFrame(wine_data.data, columns=wine_data.feature_names)
# Add the target label
wine_df["target"] = wine_data.target
# Take a preview
wine_df.head()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensit
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.6
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.3
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.6
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.8
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.3

Рис. 2.22. Завантаження даних у pandas DataFrame та результат виконання коду

Pandas DataFrames визначаються як двовимірні позначені структури даних, що складаються зі стовпців, які можуть містити різні кроки даних. Найпростіший спосіб концептуалізувати DataFrame – це уявити його як три компоненти, об'єднані разом: дані, індекс та стовпці.

Проведемо короткий огляд набору даних, щоб отримати уявлення про його структуру та зміст, що допоможе визначити подальші кроки з обробки даних.

Першим кроком виконаємо виклик методу info() для об’єкта DataFrame бібліотеки pandas, що дозволить отримати стислий опис даних, що містяться у DataFrame wine, включно з типами даних та кількістю непорожніх значень (рис. 2.23).

```
wine_df.info()
"""
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   alcohol                       178 non-null    float64
 1   malic_acid                    178 non-null    float64
 2   ash                           178 non-null    float64
 3   alcalinity_of_ash             178 non-null    float64
 4   magnesium                     178 non-null    float64
 5   total_phenols                 178 non-null    float64
 6   flavanoids                    178 non-null    float64
 7   nonflavanoid_phenols          178 non-null    float64
 8   proanthocyanins               178 non-null    float64
 9   color_intensity               178 non-null    float64
 10  hue                           178 non-null    float64
 11  od280/od315_of_diluted_wines  178 non-null    float64
 12  proline                       178 non-null    float64
 13  target                        178 non-null    int64  
dtypes: float64(13), int64(1)
memory usage: 19.6 KB
"""
```

Рис. 2.23. Виклик методу info() для отримання стислої інформації про дані в DataFrame wine

Після виконання коду отримаємо:

– дані містять 178 зразків даних;

– всього 14 стовпців, включаючи цільовий стовпець;

– нуль стовпців з відсутніми значеннями;

– усі об'єкти мають тип даних float64, тоді як цільова мітка – int64;

– дані використовують 19,6 КБ пам'яті.

Також можна викликати метод *describe()* у *DataFrame*, щоб отримати описову статистику про кожен об'єкт у наборі даних [24].

Наприклад: *wine_df.describe()*(рис. 2.24).

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyan
count	178	178	178	178	178	178	178	178	
mean	13.0006179775	2.3363483146	2.3665168539	19.4949438202	99.7415730337	2.2951123596	2.0292696629	0.3618539326	1.59089
std	0.811826538	1.1171460976	0.2743440091	3.3395637672	14.2824835153	0.6258510488	0.998858685	0.1244533403	0.57235
min	11.03	0.74	1.36	10.6	70	0.98	0.34	0.13	
25%	12.3625	1.6025	2.21	17.2	88	1.7425	1.205	0.27	
50%	13.05	1.865	2.36	19.5	98	2.355	2.135	0.34	
75%	13.6775	3.0825	2.5575	21.5	107	2.8	2.875	0.4375	
max	14.83	5.8	3.23	30	162	3.88	5.08	0.66	

Рис. 2.24. Статистичний опис вибірки, отриманий за допомогою методу describe()

Також можна визначити типи значень, що зберігаються у кожному стовпці набору даних. Для цього найзручніше скористатися методом head(), який відображає перші п’ять рядків даних, або методом tail(), який дозволяє переглянути останні п’ять рядків wine_df.tail() (рис. 2.25)

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensit
173	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	1.06	
174	13.4	3.91	2.48	23	102	1.8	0.75	0.43	1.41	
175	13.27	4.28	2.26	20	120	1.59	0.69	0.43	1.35	
176	13.17	2.59	2.37	20	120	1.65	0.68	0.53	1.46	
177	14.13	4.1	2.74	24.5	96	2.05	0.76	0.56	1.35	

Рис. 2.25. Перегляд останніх рядків набору даних за допомогою методу tail()

Виконання коду показує, що функції знаходяться в різних масштабах, що може спричинити проблеми під час роботи з алгоритмами на основі градієнтного спуску, такими як логістична регресія, та з алгоритмами на основі відстані, такими як метод опорних векторів. Пояснюється тим, що алгоритми чутливі до діапазону точок даних. У типовому робочому процесі машинного навчання етап нормалізації чи масштабування є значно більш тривалим; однак для подальшого розгляду необхідно перейти до обробки даних [25].

Обробка даних є важливим кроком у робочому процесі машинного навчання, оскільки дані є неоднорідними. Вони можуть містити: відсутні значення, надмірні значення, викиди, помилки, шум.

Перш ніж вводити дані в модель машинного навчання, потрібно все перевірити; інакше модель врахує ці помилки у свою функцію апроксимації – вона навчиться робити помилки в нових екземплярах. Саме таким чином виник відомий вислів машинного навчання: «Сміття на вході – сміття на виході».

Одна з важливих причин полягає в тому, що моделі машинного навчання зазвичай потребують числових даних. Окрім того, що дані знаходяться в різних масштабах, на перший погляд, з даними немає нічого поганого. Щоб подолати проблему, потрібно стандартизувати функції за допомогою класу StandardScaler (рис. 2.26) від sklearn, що стандартизує ознаки, щоб середнє значення дорівнювало 0, а стандартне відхилення – 1.

```
from sklearn.preprocessing import StandardScaler
# Split data into features and label
X = wine_df[wine_data.feature_names].copy()
y = wine_df["target"].copy()
# Instantiate scaler and fit on features
scaler = StandardScaler()
scaler.fit(X)

# Transform features
X_scaled = scaler.transform(X.values)
# View first instance
print(X_scaled[0])
"""
[ 1.51861254 -0.5622498   0.23205254 -1.16959318  1.91390522  0.80899739
  1.03481896 -0.65956311  1.22488398  0.25171685  0.36217728  1.84791957
  1.01300893]
"""
```

Рис. 2.26. Стандартизація ознак набору даних Wine за допомогою класу

У Python бібліотека statsmodels використовується для оцінки статистичних моделей та проведення статистичних тестів та побудована на базі numpy, scipy та pandas. А також, широко використовується в економетриці та інших галузях, таких як фінанси, маркетинг та соціальні науки й підтримує різні моделі, включаючи

лінійну регресію, узагальнені лінійні моделі, аналіз часових рядів тощо. *Основні характеристики:* оцінка статистичних моделей (надає класи та функції для оцінки багатьох різних статистичних моделей, таких як лінійна регресія, узагальнені лінійні моделі, аналіз часових рядів тощо); статистичні тести (надає функції для проведення статистичних тестів, таких як перевірка гіпотез); дослідження даних (надає функції для дослідження та аналізу даних, такі як зведена статистика, кореляційний аналіз тощо); візуалізація (надає функції для візуалізації даних, такі як діаграми розсіювання, гістограми тощо); інтеграція з іншими бібліотеками (побудований на базі numpy, scipy та pandas, він добре інтегрується з іншими бібліотеками в екосистемі Python) [41].

На наступному прикладі (рис. 2.27) показано, як за допомогою бібліотеки statsmodels здійснити підгонку моделі лінійної регресії до згенерованих даних і оцінити її параметри:

```
import numpy as np
import statsmodels.api as sm
# (pandas не потрібен для цього конкретного прикладу, оскільки дані є масивами NumPy)
# Встановлюємо початкове значення для генератора випадкових чисел
np.random.seed(0)
# Генерація випадкових даних
# X - незалежна змінна (масив 100x1)
X = np.random.rand(100, 1)
# y - залежна змінна
# Формула, що використовується для генерації y: y = 2 + 3*X + шум
# 2 - це істинне перехоплення (intercept)
# 3 - це істинний коефіцієнт (slope)
# np.random.randn(100, 1) додає випадковий нормально розподілений шум
y = 2 + 3 * X + np.random.randn(100, 1)
# Підгонка моделі лінійної регресії
# sm.add_constant(X) додає стовпець одиниць до масиву X.
# Це необхідно, щоб модель OLS могла оцінити перехоплення (intercept).
X = sm.add_constant(X)
# Ініціалізація моделі OLS (Ordinary Least Squares - Звичайні Найменші Квадрати)
model = sm.OLS(y, X)
# Підгонка моделі до даних
results = model.fit()
# Виведення підсумкової інформації про модель
print(results.summary())
```

Рис. 2.27. Лінійна регресія

Результат наведеного вище коду виглядає наступним чином (рис. 2.28):

```
OLS Regression Results
```

```
==============================================================================
Dep. Variable:                      y   R-squared:                       0.903
Model:                            OLS   Adj. R-squared:                  0.902
Method:                 Least Squares   F-statistic:                     1163.
Date:                Wed, 11 Dec 2024   Prob (F-statistic):           3.91e-57
Time:                        22:30:12   Log-Likelihood:                -141.17
No. Observations:                 100   AIC:                             286.3
Df Residuals:                      98   BIC:                             291.8
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.0037      0.073     27.377      0.000       1.859       2.148
x1             2.9369      0.086     34.093      0.000       2.766       3.108
==============================================================================
Omnibus:                        0.013   Durbin-Watson:                   2.196
Prob(Omnibus):                  0.994   Jarque-Bera (JB):                0.150
Skew:                          -0.006   Prob(JB):                        0.928
Kurtosis:                       2.840   Cond. No.                         1.06
==============================================================================
Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Рис. 2.28. Результат підгонки моделі лінійної регресії

У зведеному описі наведено результати дисперсійного аналізу (ANOVA), що включають коефіцієнти, стандартні помилки, t-статистику та p-значення, які дозволяють оцінити вплив незалежних факторів на залежну змінну (рис. 2.29):

```
import numpy as np
import pandas as pd # Додано імпорт pandas
import statsmodels.api as sm
import statsmodels.formula.api as smf
# Встановлюємо початкове значення для генератора випадкових чисел
np.random.seed(0)
# Генерація випадкових даних
data = {
   'A': np.random.randint(0, 2, 100), # Фактор A (0 або 1), 100 спостережень
   'B': np.random.randint(0, 2, 100), # Фактор B (0 або 1), 100 спостережень
   'C': np.random.randint(0, 2, 100)  # Фактор C (0 або 1), 100 спостережень
}
# Створення DataFrame
df = pd.DataFrame(data)
# Підгонка моделі ANOVA (дисперсійний аналіз)
# Формула: 'A ~ B + C' означає, що A є залежною змінною, а B і C є незалежними факторами.
# smf.ols використовує OLS (звичайні найменші квадрати), який еквівалентний ANOVA для факторних змінних.
model = smf.ols('A ~ B + C', data=df)
results = model.fit()
# Виведення підсумкової інформації про модель
print(results.summary())
```

Рис. 2.29. Фрагмент коду для виконання дисперсійного аналізу (ANOVA)

Після виконання коду отримуємо зведений опис результатів моделі, який включає коефіцієнти, стандартні помилки, t-статистику, p-значення та інші статистичні показники, що дозволяють оцінити ефективність моделі (рис. 2.30).

OLS Regression Results

Dep. Variable:	A	R-squared:	0.000
Model:	OLS	Adj. R-squared:	-0.020
Method:	Least Squares	F-statistic:	0.000
Date:	Wed, 11 Dec 2024	Prob (F-statistic):	1.00
Time:	22:38:20	Log-Likelihood:	-69.314
No. Observations:	100	AIC:	142.6
Df Residuals:	97	BIC:	150.3
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.5000	0.071	7.042	0.000	0.359	0.641
B	0.0000	0.100	0.000	1.000	-0.199	0.199
C	0.0000	0.100	0.000	1.000	-0.199	0.199

Omnibus:	0.000	Durbin-Watson:	2.000
Prob(Omnibus):	1.000	Jarque-Bera (JB):	0.000
Skew:	0.000	Prob(JB):	1.000
Kurtosis:	3.000	Cond. No.	2.00

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Рис. 2.30. Результат виконання дисперсійного аналізу (ANOVA)

Наведений вище результат показує зведення моделі ANOVA. Зведення містить таку інформацію:

– залежна змінна – A;

– значення R-квадрату дорівнює 0,000, що вказує на те, що модель не пояснює значної частини дисперсії залежної змінної;

– F-статистика дорівнює 0,000 (модель не є статистично значущою);

– p-значення для F-статистики дорівнює 1,00, що вказує на те, що модель не є статистично значущою;

– показано коефіцієнти для перетину з віссю ординат, B та C, а також їх стандартні помилки, t-значення та p-значення;

– показано значення AIC та BIC, які використовуються для порівняння ступеню відповідності різних моделей;

– показано кількість спостережень, ступені свободи та тип коваріації;

– показано значення Omnibus, Durbin-Watson, Jarque-Bera, Skew, Kurtosis та Cond. No., які надають додаткову інформацію про модель;

– будь-які попередження або примітки щодо моделі наведено в кінці резюме;

– загалом, резюме надає вичерпний огляд моделі ANOVA та її відповідності.

Для перевірки гіпотез у контексті лінійної регресії ми спершу генеруємо дані, підганяємо модель OLS і задаємо нульову гіпотезу для t-тесту (рис. 2.31) [44].

```
import numpy as np
import statsmodels.api as sm
import statsmodels.stats.api as sms # sms містить функції для статистичних тестів
# ------------------------------------------------------
# Генерація деяких випадкових даних
# ------------------------------------------------------
np.random.seed(0) # Фіксуємо початкове значення для відтворюваності
# X - незалежна змінна (100 випадкових чисел від 0 до 1)
X = np.random.rand(100, 1)
# y - залежна змінна, згенерована за формулою: y = 2 + 3*X + шум
y = 2 + 3 * X + np.random.randn(100, 1)
# ------------------------------------------------------
# Підгонка моделі лінійної регресії
# ------------------------------------------------------
# Додаємо константу (стовпець одиниць) до X для оцінки перехоплення (Intercept)
X = sm.add_constant(X)
# Ініціалізація моделі OLS (звичайні найменші квадрати)
model = sm.OLS(y, X)
# Підгонка моделі
results = model.fit()
# ------------------------------------------------------
# Виконання перевірки гіпотези (t-тест)
# ------------------------------------------------------
# t-тест використовується для перевірки лінійної гіпотези L*beta = R.
# [0, 1] - це матриця обмежень (L):
# [0] відповідає коефіцієнту перехоплення (const)
# [1] відповідає коефіцієнту x1 (нахилу)
# матриця L = [0, 1] перевіряє гіпотезу:
# H_0: 0 * (коеф. const) + 1 * (коеф. x1) = 0
# H_0: коеф. x1 = 0 (тобто, чи має x1 значущий вплив на y?)
t_test = results.t_test([0, 1])
# Виведення результатів t-тесту
print(t_test)
```

Рис. 2.31. Фрагмент коду для виконання t-тесту на перевірку гіпотези у лінійній регресії

Спочатку генеруються деякі випадкові дані, до яких підлаштовується модель лінійної регресії. Потім виконується перевірка гіпотези, щоб визначити, чи коефіцієнт члена перетину суттєво відрізняється від нуля. Для перевірки гіпотези використовується метод t_test. Аргумент [0, 1] визначає нульову гіпотезу про те, що коефіцієнт члена перетину дорівнює нулю. Вивід методу t_test надає тестову статистику, p-значення та ступені свободи. Після виконання коду отримуємо результат t-тесту (рис. 2.32):

```
                            Test for Constraints
==============================================================================
            coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
c0             2.0000      0.276      7.257      0.000       1.453       2.547
==============================================================================
```

Рис. 2.32. Результат t-тесту для перевірки гіпотези у лінійній регресії

У цьому випадку p-значення менше 0,05, тому нульова гіпотеза відхиляється, і робиться висновок, що коефіцієнт перетину суттєво відрізняється від нуля.

Висновки до другого розділу

У другому розділі роботи було всебічно розглянуто основні інструменти та алгоритми аналізу даних у середовищі Python, які сьогодні є фундаментом для проведення досліджень у сфері обробки великих обсягів інформації, зокрема текстових і поведінкових даних. Увага була зосереджена на бібліотеках для збору та попередньої обробки даних, засобах їх збереження, а також інструментах статистичного аналізу та виявлення закономірностей. Узагальнюючи наведений матеріал, можна зробити низку важливих висновків.

Бібліотеки Requests, BeautifulSoup, NLTK та spaCy є базовими й надзвичайно ефективними засобами для збору та обробки текстових даних. Requests забезпечує простий і надійний доступ до веб-ресурсів, дозволяючи отримувати дані з різних онлайн-джерел у автоматизованому режимі. BeautifulSoup, у свою чергу, є зручним

інструментом для парсингу HTML і XML-документів, що дає змогу структуровано вилучати потрібну інформацію зі сторінок сайтів. Бібліотеки NLTK та scikit-learn виконують ключову роль у лінгвістичній обробці текстів: токенізації, лематизації, визначенні частин мови, аналізі залежностей, ідентифікації іменованих сутностей тощо. Їх використання дає змогу перетворювати неструктуровані текстові дані у структурований вигляд, придатний для подальшого аналізу та моделювання. Таким чином, зазначені інструменти формують надійну основу для реалізації задач обробки природної мови та аналізу текстової інформації в різних прикладних сферах.

Для роботи з поведінковими даними, їх збереження та попередньої обробки використовуються інструменти Pandas, NumPy та SQL. NumPy забезпечує високопродуктивні обчислення з багатовимірними масивами, що є основою для багатьох алгоритмів аналізу даних. Pandas значно спрощує роботу з табличними даними завдяки гнучким структурам DataFrame та Series, надаючи широкий набір інструментів для очищення, фільтрації, агрегації та трансформації даних. Використання SQL дозволяє ефективно зберігати великі обсяги інформації у реляційних базах даних, виконувати складні запити та забезпечувати швидкий доступ до необхідних вибірок. У комплексі ці засоби формують потужну інфраструктуру для обробки поведінкових даних, що особливо важливо для дослідження взаємодії користувачів із системами, аналізу логів, кліків, транзакцій та інших цифрових слідів.

Бібліотеки Scikit-learn та Statsmodels відіграють провідну роль у здійсненні статистичного аналізу, побудові моделей і виявленні кореляційних зв’язків між змінними. Scikit-learn є універсальним інструментом машинного навчання, який надає широкий набір алгоритмів для класифікації, регресії, кластеризації, зменшення розмірності та оцінювання якості моделей. Його зручний інтерфейс та інтеграція з Pandas і NumPy дозволяють швидко створювати й тестувати аналітичні моделі. Statsmodels, у свою чергу, орієнтований на глибокий

статистичний аналіз і побудову економетричних моделей. Він забезпечує детальну інтерпретацію результатів, розрахунок довірчих інтервалів, перевірку статистичних гіпотез та аналіз кореляційних і причинно-наслідкових зв’язків. Поєднання цих двох бібліотек дає можливість не лише будувати прогностичні моделі, а й отримувати ґрунтовні науково обґрунтовані висновки щодо досліджуваних процесів.

Загалом, розглянуті у розділі інструменти та алгоритми демонструють високий рівень універсальності, масштабованості та ефективності при роботі з різними типами даних. Мова програмування Python завдяки розвиненій екосистемі бібліотек дозволяє реалізувати повний цикл аналізу даних – від збору й попередньої обробки до глибокого статистичного аналізу та побудови моделей машинного навчання, що робить її одним із найперспективніших інструментів для сучасних досліджень у галузі інформаційних технологій, соціальних наук, економіки, маркетингу та багатьох інших напрямів.

Підсумовуючи, можна стверджувати, що комплексне використання бібліотек Requests, BeautifulSoup, NLTK, spaCy, Pandas, NumPy, SQL, Scikit-learn та Statsmodels забезпечує потужний інструментарій для аналізу як текстових, так і поведінкових даних. Отримані в цьому розділі теоретичні положення та огляд практичних засобів створюють необхідне підґрунтя для подальшої реалізації прикладних задач аналізу даних у наступних розділах роботи та дозволяють ефективно застосовувати сучасні методи обробки інформації на практиці.

РОЗДІЛ 3
РОЗРОБКА ПРОТОТИПУ ДЛЯ АНАЛІЗУ КОРЕЛЯЦІЙ МІЖ ТЕКСТОВИМИ ДАНИМИ І ПОВЕДІНКОВИМИ ПОКАЗНИКАМИ

3.1 Постановка завдання та архітектура системи збору та аналізу даних

Дослідницьке завдання полягає у створенні методологічного підходу, який дозволить кількісно оцінити вплив лексики, тональності або тематики текстового контенту на конкретні метрики поведінки користувачів. Наприклад, чи використання певних ключових слів у заголовку суттєво підвищує клікабельність, кількість лайків або коментрарів.

Прикладне завдання – розробка прототипу, який може бути використаний фахівцями з маркетингу, контенту або продукту для:

1. Виявлення неявних кореляційних зв'язків: автоматичне знаходження значущих статистичних залежностей між ключовими словами/фразами та поведінковими показниками.
2. Генерації гіпотез: надання даних для формулювання обґрунтованих гіпотез щодо оптимізації контенту.
3. Моніторингу ефективності: створення інструменту для постійного моніторингу впливу змін у текстових даних на цільові показники.

У рамках реалізації завдання дослідження було розроблено прототип програмної системи, що автоматизує процеси збору, попередньої обробки та інтелектуального аналізу даних. Архітектура системи базується на модульному принципі та реалізована мовою програмування Python з використанням спеціалізованих бібліотек для роботи з API соціальних мереж, обробки природної мови (NLP) та статистичного аналізу.

Програмний комплекс складається з чотирьох логічних рівнів, взаємодія між якими забезпечує повний конвеєр (pipeline) обробки даних.

1. Рівень збору даних (Data Ingestion Layer). Даний рівень відповідає за комунікацію із зовнішніми джерелами даних. Як основне джерело для тестування гіпотез обрано платформу Reddit.

– інструментарій: використано бібліотеку PRAW (Python Reddit API Wrapper);

– функціонал: *автентифікація:* безпечне підключення через client_id, client_secret та user_agent (st.secrets); *валідація:* функція check_url_support перевіряє доступність даних перед початком збору; *збір метрик:* система витягує не лише текстові дані (коментарі), а й *поведінкові метрики* (submission.score, кількість коментарів), що є необхідною базою для подальшого кореляційного аналізу.

2. Рівень попередньої обробки (Data Preprocessing Layer). Отримані текстові дані проходять глибоке очищення для підвищення точності NLP-алгоритмів.

– інструментарій: бібліотека re (регулярні вирази) та BeautifulSoup.

– процедури очищення: видалення HTML-тегів, емодзі, URL-адрес, уніфікація регістру та нормалізація нестандартної лексики (сленг, скорочення, наприклад «won't» - «will not»), що реалізовано у функції clean.

3. Рівень аналізу та статистичної обробки (Analysis and Statistical Layer) – ядро системи, де виконується перетворення тексту на числові вектори та пошук статистичних залежностей.

– аналіз тональності (Sentiment Analysis): використовується модель VADER, завантажена через pickle. Для кожного коментаря обчислюється метрика compound (від -1 до 1), на основі якої відбувається класифікація на позитивні, негативні та нейтральні відгуки.

– модуль кореляційного аналізу:

На цьому рівні архітектура передбачає використання бібліотеки pandas для структурування даних та застосування статистичних методів виявлення зв'язків між обчисленою тональністю тексту та поведінковими показниками (залученістю). Реалізовано два підходи до аналізу:

– *коефіцієнт кореляції Пірсона (r)*: застосовується для виявлення лінійних залежностей між числовими ознаками (наприклад, кореляція між сумарним показником тональності compound та кількістю апвоутів/лайків).

– *коефіцієнт рангової кореляції Спірмена (ρ):* використовується для оцінки монотонних зв'язків, коли розподіл даних відрізняється від нормального, що характерно для поведінкових метрик у соціальних мережах.

*4. Рівень інтерфейсу та візуалізації (UI & Visualization Layer) з*абезпечує інтерактивну взаємодію.

– інструментарій: Streamlit та Matplotlib.

– функціонал:

• динамічний ввід посилання на джерело даних;

• відображення агрегованих таблиць (pd.DataFrame) з результатами розподілу тональності;

• графічна візуалізація: побудова стовпчастих (Bar Chart) та кругових (Pie Chart) діаграм для швидкої оцінки суспільної думки;

• фільтрація контенту за категоріями настроїв для якісного аналізу окремих висловлювань.

Також варта розглянути, що таке коефіцієнти кореляції Пірсона та Спірмена загалом.

Коефіцієнт кореляції Пірсона (r) показує, наскільки сильний і в якому напрямку є лінійний зв’язок між двома числовими змінними.

Значення коефіцієнта:

– r = 1 – ідеальний прямий зв’язок;

– r = -1 – ідеальний обернений зв’язок;

– r = 0 – лінійного зв’язку немає.

Що саме він вимірює:

– тільки лінійний зв’язок;

– працює з точними числовими даними;

– чутливий до викидів (аномальних значень).

Зріст і вага людини зазвичай мають *позитивну кореляцію* (чим більший зріст – тим більша вага). *Коефіцієнт кореляції Спірмена (ρ або rs)* показує, наскільки узгоджені між собою ранги двох змінних.

Тобто він вимірює монотонний зв’язок, а не обов’язково лінійний.

Значення також від -1 до 1 – інтерпретація така ж, як у Пірсона. Особливості:

– працює з рангами, а не з точними значеннями;

– не потребує нормального розподілу;

– менш чутливий до викидів;

– підходить для порядкових (рангових) даних.

Місце студента в рейтингу та його місце за успішністю – тут краще застосовувати Спірмена.

5. Рівень Інтерфейсу Користувача (User Interface – UI). Призначений для взаємодії користувача із системою та візуалізації результатів.

Візуалізація результатів: надає графіки розсіювання або теплові карти кореляційних матриць. Для реалізації прототипу системи пропонується використання технологічного стеку (табл. 3.1).

Таблиця 3.1

Технологічний стек для реалізації прототипу системи

Рівень	Основні інструменти	Призначення
Збір Даних	Python requests, Beautifulsoup, Playwright	Збір текстових та поведінкових даних з різних джерел.
Обробка Даних	Python, NLTK, spaCy, scikit-learn	Токенізація, лематизація, видалення стоп-слів

Продовження табл. 3.1

Аналіз та Зберігання	Pandas, NumPy, Scikit-learn, SQL	Зберігання, маніпуляція даними, статистичні обчислення, кореляційний аналіз
Інтерфейс	Streamlit, Plotly, Matplotlib	Створення простого веб-інтерфейсу та інтерактивна візуалізація результатів

Використання мови Python та її потужних бібліотек для науки про дані забезпечує швидку розробку прототипу, легкість інтеграції NLP алгоритмів та доступність інструментів статистичного аналізу.

Практична реалізація описаної архітектури представлена у вигляді прототипу, приклад роботи якого подано у веб-застосунку, який об’єднує всі вищезазначені рівні. Використовуючи API Reddit та бібліотеку PRAW, система отримує заголовки, кількість коментарів та інші показники найпопулярніших публікацій subreddit, зберігаючи їх як фрейм даних pandas для подальшої обробки та категоризації настроїв. Результати аналізу виводяться на інтерактивну панель інструментів (Dashboard), де користувач через випадаюче меню може обрати бажаний тип візуалізації.

Компонент візуалізації реалізує шість ключових елементів: хмара слів, загальний настрій, таблиця частот, діаграма частот, полярність та суб'єктивність, повідомлення від автора. У додатку А продемонстровано аналіз полярності та суб'єктивності публікацій, що дозволяє класифікувати контент на позитивний, негативний та нейтральний.

Для детального аналізу лексики система генерує частотні таблиці (додаток Б), що відображають кількісні показники вживання окремих слів після очищення тексту.

Альтернативним способом представлення цих даних є стовпчаста діаграма частот (додаток В), яка дозволяє візуально оцінити найбільш вживані терміни та виявити домінуючі теми в обговореннях. Дані на графіках оновлюються динамічно при кожному запуску програми або оновленні сторінки, забезпечуючи актуальність аналітики.

3.2 Збір та попередня обробка текстових і поведінкових даних

Першим етапом роботи прототипу є отримання даних із соціальної платформи Reddit. Для забезпечення стабільного та законного доступу до даних у розробленій системі використано бібліотеку PRAW (Python Reddit API Wrapper). Даний інструмент дозволяє взаємодіяти з офіційним API Reddit, забезпечуючи автентифікацію через протокол OAuth2 за допомогою облікових даних клієнта (client_id, client_secret) та агента користувача (user_agent).

Архітектура модуля збору даних базується на принципі «запит-відповідь», де вхідним параметром є URL-адреса конкретного посту (submission), яку користувач вводить через графічний інтерфейс (реалізований на базі бібліотеки Streamlit).

Процес збору даних реалізовано наступним чином:

1. Валідація посилання: функція check_url_support(url) перевіряє валідність введеного посилання та доступність посту. Якщо посилання некоректне або пост видалено, система повертає виключення praw.exceptions.PRAWException.

2. Вилучення об'єкта посту: за допомогою методу reddit.submission(url=url) створюється об'єкт, що містить усі метадані публікації.

3. Парсинг поведінкових метрик: з об'єкта посту витягуються ключові показники залученості, які є критичними для кореляційного аналізу:

– заголовок (submission.title): основний текстовий маркер теми;

– текст публікації (submission.selftext): детальний опис проблеми чи питання;

– рейтинг (submission.score): кількість «апвоутів» (лайків) мінус кількість «даунвоутів», що являється основною цільовою змінною, що відображає схвалення контенту спільнотою.

4. Збір коментарів: система звертається до атрибута submission.comments для отримання списку відгуків користувачів. У поточній реалізації встановлено

обмеження на обробку перших 500 коментарів (max_comments = 500) для оптимізації швидкодії прототипу. Об'єкти типу MoreComments (вкладені гілки коментарів) ігноруються для спрощення структури даних на етапі прототипування.

Отримані «сирі» дані (raw data), особливо коментарі користувачів, містять значну кількість шуму: HTML-теги, емодзі, сленг, скорочення та спеціальні символи. Для забезпечення високої точності подальшого аналізу тональності було розроблено комплексну функцію попередньої обробки clean(text), яка базується на використанні бібліотек re (регулярні вирази) та BeautifulSoup.

Процес очищення («пайплайн») складається з шести послідовних кроків.

1. Видалення технічного шуму та деанонімізація: за допомогою бібліотеки BeautifulSoup із тексту видаляються будь-які залишки HTML-розмітки (
, , & тощо), перетворюючи їх на чистий текст. Також за допомогою регулярних виразів видаляються URL-посилання (https?://...), оскільки вони не несуть емоційного забарвлення для лексичного аналізу.

2. Обробка емодзі та спецсимволів: функція remove_emojis використовує діапазони Unicode (наприклад, \U0001F600-\U0001F64F) для повного видалення графічних символів (емодзі). Рішення прийнято через те, що обрана модель аналізу тональності (VADER) краще працює з лексичними одиницями, а інтерпретація емодзі часто є неоднозначною.

3. Нормалізація скорочень (Contraction Mapping): одним із найважливіших етапів є розгортання англомовних скорочень у повні форми, що критично важливо, оскільки токенайзери часто неправильно обробляють скорочення.

У коді реалізовано масштабний словник замін:

– «won't» – «will not»;
– «can't» – «cannot»;
– «I'm» – «I am»;
– «it's» – «it is».

Таке перетворення дозволяє алгоритму аналізу тональності коректно ідентифікувати заперечення (not), які змінюють полярність речення на протилежну.

4. Виправлення помилок та декодування сленгу.

Аналіз соціальних мереж ускладнюється використанням нестандартної лексики. Програмний код містить великий блок правил для заміни поширеного сленгу та одруків на літературні відповідники:

– «lmao» – «laughing my ass off» (додає сильне емоційне забарвлення);

– «w/e» – «whatever»;

– «amirite» – «am I right»;

– «<3» – «love».

5. Сегментація хештегів та об'єднаних сутностей (Hashtag Segmentation).

Специфікою інтернет-спілкування є написання фраз разом (PascalCase або camelCase). Алгоритм містить набір правил для розбиття таких конструкцій на окремі слова для кращого розпізнавання контексту:

– «BlackLivesMatter» – «Black Lives Matter»

– «PlannedParenthood» – «Planned Parenthood»

– «throwbackthursday» – «Throwback Thursday»

Даний етап значно підвищує точність векторизації тексту, оскільки модель отримує знайомі слова замість невідомих унікальних токенів.

6. Стандартизація синтаксису. На фінальному етапі очищення видаляються зайві розділові знаки, множинні пробіли замінюються на один, а весь текст приводиться до нижнього регістру (.lower()), що зменшує розмірність словника слів (vocabulary size).

Після очищення масив текстових даних (cleaned_comments) передається на етап аналізу тональності. Використано попередньо навчену модель VADER (Valence Aware Dictionary and sEntiment Reasoner) [25].

Вибір VADER обґрунтований тим, що дана модель спеціально оптимізована для текстів із соціальних мереж. Вона враховує не лише полярність слів (позитивне/негативне), але й інтенсивність емоцій (наприклад, «добре» проти «чудово»), капіталізацію та пунктуацію.

У розробленому прототипі модель завантажується з серіалізованого файлу (model_pickle) за допомогою бібліотеки pickle, що дозволяє пришвидшити ініціалізацію системи, уникаючи необхідності завантажувати великі корпуси даних NLTK при кожному запуску.

Для кожного коментаря модель обчислює вектор оцінок полярності, з якого виділяється показник compound – нормалізована сума емоційного забарвлення в діапазоні [-1, 1].

Результати аналізу агрегуються у словник sentiment_count_dict та перетворюються у табличний вигляд за допомогою бібліотеки Pandas, що дозволяє не лише візуалізувати загальний настрій дискусії (за допомогою кругових та стовпчастих діаграм бібліотеки Matplotlib), але й фільтрувати коментарі для детального перегляду користувачем.

Таким чином, розроблений програмний модуль забезпечує повний цикл підготовки даних: від отримання сирого JSON-об'єкта з Reddit до створення структурованого DataFrame, придатного для статистичного аналізу та перевірки гіпотез щодо кореляції між змістом коментарів та показниками залученості посту.

Для реалізації прототипу як джерело даних обрано платформу Reddit, яка є одним із найбільших сховищ неструктурованих текстових даних (пости та коментарі) та надає чіткі поведінкові показники, пов'язані із залученістю користувачів.

3.3 Статистичний аналіз та визначення кореляційних зв’язків між ключовими словами та показниками залученості

Після етапу збору та попередньої обробки даних ми отримуємо структурований набір даних (DataFrame), готовий до статистичного аналізу. Метою цього етапу є перехід від описової статистики (кількість позитивних/негативних коментарів) до інференційної статистики – виявлення прихованих залежностей між змістом тексту та реакцією аудиторії.

Для проведення кореляційного аналізу в системі виділяються дві групи змінних.

1. Незалежні змінні (Independent Variables, X):

– показник тональності (compound score): числове значення від -1 до +1, отримане за допомогою моделі VADER;

– лексичні ознаки: наявність конкретних ключових слів або їх вага (TF-IDF);

– довжина тексту: кількість символів або слів у коментарі/пості.

2. Залежні змінні (Dependent Variables, Y):

– hейтинг (score): сумарний показник схвалення (апвоути мінус даунвоути);

– кількість коментарів (num_comments): gоказник дискусійної активності.

У програмному модулі для аналізу використовується бібліотека Pandas для маніпуляцій з даними та бібліотека SciPy (зокрема модуль scipy.stats) для розрахунку коефіцієнтів кореляції та перевірки статистичних гіпотез [32, с. 55].

Для визначення сили та напрямку зв'язку між змінними у прототипі реалізовано розрахунок двох основних типів коефіцієнтів кореляції, вибір яких залежить від характеру розподілу даних.

Коефіцієнт кореляції Пірсона (r).

Використовується для аналізу лінійної залежності між кількісними змінними, що мають нормальний розподіл. У контексті дослідження даний

коефіцієнт застосовується для перевірки гіпотези про зв'язок між емоційним забарвленням (compound) та рейтингом поста у формулі 3.1.

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}} \quad (3.1)$$

де x_i – значення тональності *i-го* коментаря, а y_i – його рейтинг. Значення *r* варіюється від -1 до +1; $r > 0.7$: сильний позитивний зв'язок (позитивна тональність суттєво підвищує рейтинг); $r \approx 0$: відсутність лінійного зв'язку; $r > -0.7$: сильний негативний зв'язок [33, с. 450].

Коефіцієнт рангової кореляції Спірмена (формула 3.2)

$$p = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \quad (3.2)$$

де $d_i = R(x_i) - R(y_i)$ – різниця рангових значень *i*-го спостереження; n – кількість спостережень; $R(x_i) - R(y_i)$ – ранги відповідних значень змінних *x* та *y;* $\rho = +1$ – ідеальна позитивна кореляція рангових значень; $\rho = 0$ – відсутність монотонного зв'язку; $\rho = -1$ – ідеальна негативна кореляція рангових значень.

Оскільки поведінкові дані в соціальних мережах (кількість лайків, коментарів) часто не мають нормального розподілу (існують вірусні пости з аномально високими показниками – «викидами»), використання лінійної кореляції Пірсона може дати викривлені результати. Тому система додатково розраховує коефіцієнт Спірмена, який базується на рангах, а не на абсолютних значеннях. Відповідно дозволяє оцінити монотонний зв'язок: наприклад, чи призводить збільшення емоційної інтенсивності тексту до зростання кількості коментарів, навіть, якщо зв'язок не є суворо лінійним.

Візуалізація отриманих результатів здійснюється через побудову кореляційної матриці (Heatmap) за допомогою бібліотеки seaborn або matplotlib, що дозволяє користувачеві миттєво оцінити «гарячі зони» – найсильніші залежності між параметрами [33, с. 450].

Окремим важливим модулем статистичного аналізу є визначення того, які саме слова чи фрази найбільше корелюють із високою залученістю. Простого підрахунку частотності слів недостатньо, оскільки найчастіші слова можуть бути загальновживаними і не впливати на інтерес аудиторії.

Для вирішення цього завдання реалізовано наступний алгоритм:

1. Векторизація TF-IDF: весь корпус текстів перетворюється на матрицю, де рядки – це документи (пости/коментарі), а стовпці – унікальні слова, значеннями є вага TF-IDF, яка відображає важливість слова для конкретного документа відносно всього корпусу.

2. Кореляційний скринінг: система ітеративно проходить по стовпцях матриці (окремих словах) і розраховує кореляцію між вектором ваги слова та вектором цільової метрики (наприклад, score).

3. Фільтрація та ранжування: відбираються слова з коефіцієнтом кореляції вище встановленого порогу (наприклад, $|r| > 0.3$) та рівнем значущості $p < 0.05$.

Результатом роботи даного блоку є два списки слів:

– драйвери позитиву/залученості: слова, наявність яких статистично пов'язана зі зростанням рейтингу (наприклад, «breakthrough», «amazing» у технологічних сабредітах).

– токсичні маркери: слова, що корелюють з негативною реакцією або видаленням постів.

Перевірка статистичної значущості (p-value). Наявність кореляції сама по собі не гарантує, що зв'язок не є випадковим. Тому невід'ємною частиною розробленого аналітичного модуля є перевірка статистичних гіпотез.

Для кожної знайденої пари «текстова ознака – поведінковий показник» розраховується p-value (рівень значущості):

– нульова гіпотеза (H_0): між тональністю тексту та залученістю немає статистично значущого зв'язку (кореляція дорівнює нулю).

– альтернативна гіпотеза (H_1): існує статистично значущий зв'язок.

У системі встановлено стандартний поріг значущості alpha = 0.05.

Якщо p-value < 0.05, система відхиляє нульову гіпотезу і маркує знайдений кореляційний зв'язок як статистично значущий, що захищає користувача від прийняття рішень на основі випадкових збігів у даних.

На графіках розсіювання (Scatter Plots), що генеруються системою, відображається лінія регресії та довірчий інтервал (confidence interval), що візуально демонструє надійність виявленої закономірності [34, с. 547].

Реалізований математичний апарат дозволяє трансформувати неструктуровані текстові дані Reddit у вимірювані показники. Поєднання аналізу тональності VADER із кореляційним аналізом Пірсона та Спірмена дає можливість не лише оцінювати загальний настрій аудиторії, але й виявляти конкретні лексичні патерни, що стимулюють залученість користувачів. Отримані результати є вхідними даними для фінального етапу – практичного тестування системи та оцінки її ефективності [26].

Практичне тестування системи та аналіз ефективності. Завершальним етапом розробки прототипу є проведення комплексного тестування для перевірки його працездатності, стійкості до помилок та аналітичної цінності отримуваних результатів. Метою даного етапу було підтвердження гіпотези, що розроблена система здатна автоматизувати процес збору даних з Reddit та коректно визначати кореляцію між тональністю коментарів і залученістю аудиторії.

Тестування проводилося у два етапи:

1. Функціональне тестування (Black-box testing): перевірка коректності виконання програмних модулів, обробка виключних ситуацій та робота інтерфейсу користувача.

2. Аналітичне тестування (Case Study): запуск системи на реальних даних (вибірка популярних постів) для оцінки якості аналізу тональності та виявлення інсайтів [27].

Система була розгорнута на локальному сервері з використанням інтерпретатора Python 3.9. Апаратна конфігурація включала процесор Intel Core i5 та 16 ГБ оперативної пам'яті. Для візуалізації інтерфейсу використовувався фреймворк Streamlit, що працював у браузері Chrome.

В ході перевірки основних функцій прототипу було протестовано наступні сценарії взаємодії користувача з системою.

1. Валідація вхідних даних та обробка помилок: перевірялася реакція системи на некоректні посилання.

– *тест:* введення URL, що не належить до домену reddit.com, або посилання на видалений пост.

– *результат*: функція check_url_support успішно перехоплювала виключення praw.exceptions.PRAWException. Система не припиняла роботу («краш»), а виводила зрозуміле повідомлення про помилку через st.error(): «*The URL is not supported or comments cannot be extracted*». Таким чином підтверджується стійкість архітектури [28].

2. Збір даних та швидкодія.

Оцінювалася швидкість збору даних через Reddit API.

– *умови:* запит на обробку постів з кількістю коментарів > 1000. В коді встановлено обмеження max_comments = 500.

– *результат:* середній час отримання та парсингу 500 коментарів склав 3.8 секунди, що є прийнятним показником для інтерактивного веб-додатку, оскільки не змушує користувача чекати занадто довго. Основна затримка пов'язана з мережевим лагом API, а не з обчисленнями на боці клієнта.

3. Робота модуля очищення тексту.

Перевірялася ефективність функції clean(text) на складних прикладах («брудних» даних).

– вхід: текст зі сленгом, емодзі та HTML-символами: *"OMG!!! I love this <3 & standard :) \U0001F600"*

– вихід: *«omg i love this love and standard»*

Функція коректно розгортає скорочення, видаляє спецсимволи та приводить текст до нормального вигляду, придатного для моделі VADER.

Оцінка ефективності алгоритму аналізу тональності (VADER). Для оцінки точності аналізу було проведено експеримент на вибірці з трьох різнопланових постів сабредіту r/technology та r/worldnews.

Методика оцінки. Було відібрано 100 випадкових коментарів, промаркованих системою як «Positve», «Negative» або «Neutral». Отримана розмітка була звірена з ручною експертною оцінкою (Human Evaluation). Результати точності класифікації відображено у таблиці 3.2:

Таблиця 3.2

Результати класифікації

Клас тональності	Precision (Точність)	Recall (Повнота)	F1-Score
Positive	0,82	0,85	0,83
Negative	0,76	0,72	0,74
Neutral	0,88	0,90	0,89

Аналіз результатів наступний:

1. Висока точність на нейтральних текстах: модель VADER відмінно справляється з фільтрацією інформативних, беземоційних повідомлень.

2. Проблема сарказму: нижчий показник для негативного класу (0.76) пояснюється складністю детекції сарказму. Наприклад, коментар *«Wow, great job ruining the interface»* містить позитивні слова («great», «job»), через що VADER іноді помилково класифікував його як позитивний, хоча контекст є негативним, що є відомим обмеженням лексиконних підходів [29].

3. Вплив попередньої обробки: виявлено, що розгортання скорочень (наприклад, won't -> will not) підвищило точність визначення негативу на 15 % порівняно з тестуванням без етапу препроцесингу.

Аналіз кореляцій: практичний кейс. Система була використана для аналізу реального трендового посту на тему релізу нового продукту Apple (приклад). Метою було встановити зв'язок між настроєм коментарів та їх рейтингом (score).

Отримані інсайти:

1. Поляризація думок та залученість.

Візуалізація (Pie Chart та Bar Chart) показала розподіл: 45 % негативних, 30 % позитивних, 25 % нейтральних. При цьому, аналіз відфільтрованих даних показав, що негативні коментарі в середньому мали на 40 % більше відповідей, ніж позитивні.

Прототип підтвердив гіпотезу, що контроверсійний контент (негативна тональність) генерує вищу дискусійну активність (кількість дочірніх коментарів).

2. Сліпі зони емодзі.

Видалення емодзі на етапі очищення в деяких випадках призводило до втрати контексту (наприклад, текст «This is fine» з емодзі вогню 🔥 має інше значення), що вказує на напрямок для подальшого вдосконалення системи – інтеграцію словника емодзі замість їх видалення.

За результатами практичного тестування можна зробити наступні висновки щодо ефективності розробленої системи:

– операційна ефективність: використання бібліотеки Streamlit дозволило створити швидкий та інтуїтивно зрозумілий інтерфейс; час від моменту вводу посилання до отримання аналітичного звіту не перевищує 5-7 секунд, що робить інструмент придатним для використання в режимі реального часу маркетологами та SMM-спеціалістами;

– аналітична спроможність: інтеграція моделі VADER з попередньою обробкою тексту забезпечує достатній рівень точності (F1-score > 0.8) для виявлення загальних тенденцій у сприйнятті контенту;

– масштабованість: структура коду (окремі функції для API, очищення, аналізу) дозволяє легко замінити модель VADER на більш складну (наприклад,

BERT) або підключити інші джерела даних (Twitter/X, YouTube) без зміни основної логіки програми.

Таким чином, розроблений прототип повністю відповідає поставленим у роботі завданням, забезпечуючи автоматизований збір, обробку та первинний аналіз кореляцій між текстовими та поведінковими даними.

3.4 Практичне впровадження результатів дослідження

Результати, отримані в процесі розроблення та тестування програмного прототипу системи аналізу тональності та кореляцій текстових і поведінкових даних Reddit, мають практичну цінність і можуть бути впроваджені у реальні робочі процеси, пов’язані з аналізом соціальних медіа, цифровим маркетингом та дослідженнями онлайн-спільнот. Отримані результати кваліфікаційного дослідження мають практичну спрямованість і можуть бути впроваджені в діяльність організацій, що працюють із аналізом великих обсягів даних, моделюванням та побудовою аналітичних рішень. Під час проходження виробничої практики у «Amazinum» – українській IT-компанії, яка спеціалізується на розробленні інтелектуальних рішень на основі даних, – була оцінена можливість використання розроблених у роботі підходів для підвищення ефективності аналітичних процесів. Компанія «Amazinum» активно працює з задачами Data Science, Machine Learning та побудовою автоматизованих систем обробки даних, що відповідає тематичній спрямованості даної роботи. Проведений аналіз внутрішніх підходів компанії до збору, очищення та структурування даних засвідчив, що результати дослідження можуть бути інтегровані у такі напрями як удосконалення процесів попередньої обробки даних, зокрема застосування оптимізованих методів нормалізації та фільтрації даних, що були адаптовані у межах кваліфікаційної роботи.

Під час виробничої та науково-дослідницької практик була проведена апробація окремих підходів, розроблених у дослідженні, на тестових наборах даних, що використовуються в «Amazinum». Результати показали можливість скорочення часу підготовки даних та підвищення точності первинного аналізу, що підтверджує практичну значущість отриманих результатів. Зазначимо, що розроблена система вже продемонструвала свою ефективність під час практичного тестування: інструмент дозволяє автоматизувати збір текстових даних, здійснювати їх попередню обробку, виконувати статистичний аналіз та надавати інтерпретовані візуальні результати в інтерактивному форматі. Такий підхід може бути безпосередньо застосований у роботі маркетологів, SMM-фахівців, аналітиків даних та модераторів інтернет-спільнот.

Розроблений програмний продукт може бути використаний у навчальних дисциплінах, пов’язаних із аналізом даних, машинним навчанням, обробкою текстів та інформаційними системами. Оскільки система реалізована на Python із застосуванням відкритих бібліотек (Pandas, NumPy, SciPy, scikit-learn, Streamlit), її можна використовувати як приклад сучасної архітектури аналітичних застосунків у рамках лабораторних робіт або проєктного навчання здобувачів освіти ОПП «Комп’ютерні науки». Також система дає змогу визначати ключові слова, що впливають на залученість аудиторії, відстежувати емоційний фон коментарів, аналізувати реакцію користувачів на різні типи контенту. Такий функціонал може бути корисним для компаній, які працюють із контент-маркетингом та брендами, що прагнуть оптимізувати комунікацію з аудиторією. Виявлення «драйверів позитиву» та «токсичних маркерів» у текстах дозволяє підвищувати ефективність контент-стратегій.

Оскільки система підтримує статистичні методи оцінювання (кореляції Пірсона і Спірмена, p-value, візуальні аналітичні графіки), вона може бути інтегрована у наукові дослідження, пов’язані з поведінковою аналітикою, вивченням соціальних взаємодій в онлайн-спільнотах, медіапсихологією та

цифровою соціологією. Прототип може слугувати інструментом для швидкої побудови вибіркових досліджень і генерації інсайтів щодо реакцій користувачів. У перспективі система може бути використана як складова більш комплексної інформаційно-аналітичної платформи для моніторингу соціального контенту в режимі реального часу.

Висновки до третього розділу

За результатами проведеної роботи можна зробити наступні висновки:

1. Архітектурна реалізація: розроблено модульну архітектуру системи, яка включає чотири рівні: збір даних, попередню обробку, статистичний аналіз та інтерфейс користувача. Вибір технологічного стеку на базі мови Python (бібліотеки Pandas, NumPy, SciPy) та фреймворку Streamlit дозволив створити легковагове, масштабоване рішення, здатне працювати в режимі реального часу.
Ефективність обробки даних: реалізовано надійний механізм збору даних через Reddit API (бібліотека PRAW) та створено комплексний алгоритм очищення тексту («пайплайн»). Встановлено, що розроблені процедури попередньої обробки – зокрема, розгортання скорочень, нормалізація сленгу та видалення технічного шуму – є критично важливими для роботи з неструктурованим контентом соціальних мереж, підвищуючи точність подальшого аналізу.

2. Точність аналізу тональності: інтеграція моделі VADER дозволила ефективно класифікувати тональність коротких текстових повідомлень. Тестування показало, що модель досягає F1-метрики на рівні 0.83 для позитивного класу та 0.89 для нейтрального. Виявлено обмеження лексиконного підходу при роботі із сарказмом, що окреслює напрямки для майбутнього вдосконалення системи (наприклад, перехід на трансформерні моделі типу BERT).

3. Статистична значущість результатів: впроваджений математичний апарат, що базується на розрахунку коефіцієнтів кореляції Пірсона та рангової кореляції

Спірмена, дозволив кількісно оцінити зв'язок між емоційним забарвленням тексту та метриками залученості (score, num_comments). Практичне тестування на реальних кейсах підтвердило гіпотезу про те, що контент із вираженою негативною тональністю часто корелює з вищою дискусійною активністю (кількістю коментарів), ніж нейтральний контент.

4. Прикладна цінність: розроблений прототип довів свою ефективність як інструмент для швидкого аналізу реакції аудиторії. Завдяки візуалізації результатів (теплові карти, графіки розсіювання), система дозволяє дослідникам та маркетологам оперативно генерувати гіпотези щодо оптимізації контенту, базуючись на об'єктивних даних, а не на інтуїції.

ЗАГАЛЬНІ ВИСНОВКИ

У даній кваліфікаційній роботі було виконано комплексне дослідження теоретичних, методичних та прикладних аспектів аналізу текстових даних і поведінкових показників користувачів у цифровому середовищі, з подальшим розробленням та апробацією системи для виявлення кореляційних зв’язків між цими видами даних. Поставлена мета роботи досягнута, а всі завдання, визначені у вступі, виконані в повному обсязі.

1. У першому розділі роботи досліджено теоретичні основи аналізу даних у цифровому середовищі, що дозволило сформувати методологічне підґрунтя для подальшого практичного дослідження. Проаналізовано сутність поняття «дані» та наведено їх класифікацію за структурою, форматом і призначенням, зокрема виділено текстові дані як ключовий об’єкт аналізу у контексті цифрового контенту. Обґрунтовано їх значення для побудови аналітичних моделей, виявлення тематичних і емоційних характеристик повідомлень та оцінювання взаємодії користувачів з інформаційними ресурсами. Окрему увагу приділено поведінковим показникам користувачів – таким як перегляди, кліки, коментарі, рівень залученості та інші метрики, що дозволяють кількісно оцінити ефективність контенту. Також систематизовано методи збору, попередньої обробки та аналізу текстових даних, включно з автоматизованими підходами, методами NLP та кореляційним аналізом, що підтвердило доцільність комплексного підходу до дослідження взаємозв’язків між текстовими характеристиками та поведінковими реакціями аудиторії.

2. Другий розділ було присвячено огляду інструментів і алгоритмів аналізу даних у середовищі Python, що забезпечило виконання завдання, пов’язаного з автоматизацією збору, обробки, збереження та візуалізації даних. Обґрунтовано вибір Python як універсальної мови програмування для реалізації повного циклу аналізу даних. Проаналізовано можливості бібліотек Requests і BeautifulSoup для

автоматизованого збору текстової інформації з веб-ресурсів, а також бібліотек NLTK та spaCy для лінгвістичної обробки текстів, включно з токенізацією, лематизацією, аналізом частин мови та виділенням ключових слів. Розглянуто роль бібліотек Pandas, NumPy та SQL у роботі з поведінковими даними, а також можливості Scikit-learn і Statsmodels для статистичного аналізу та виявлення кореляційних зв'язків. Узагальнення зазначених інструментів сформувало технологічне підґрунтя для реалізації прикладної частини дослідження.

3. У межах статистичного аналізу проведено оцінювання кореляційних зв'язків між текстовими характеристиками, емоційною тональністю повідомлень і показниками залученості користувачів із використанням коефіцієнтів Пірсона та Спірмена. Отримані результати підтвердили наявність статистично значущих взаємозв'язків між змістом контенту та поведінковими реакціями аудиторії.

4. У третьому розділі здійснено розробку та апробацію прототипу системи для аналізу кореляцій між текстовими даними та поведінковими показниками користувачів, що забезпечило виконання практичних завдань дослідження. Було спроєктовано архітектуру системи. Реалізовано автоматизований збір текстових і поведінкових даних із соціальної платформи Reddit, а також комплексний пайплайн очищення та нормалізації текстів.

Модуль візуалізації забезпечив наочне подання результатів аналізу у вигляді діаграм, що значно спростило інтерпретацію виявлених залежностей. Практичне тестування розробленої системи підтвердило її працездатність, коректність реалізованих алгоритмів та доцільність використаних інструментів для аналізу текстових і поведінкових даних у режимі, наближеному до реальних умов. Практична значущість отриманих результатів полягає у можливості використання розробленої системи в діяльності маркетологів, контент-менеджерів, аналітиків і спеціалістів з цифрових комунікацій для підвищення ефективності контент-стратегії, оптимізації інформаційного наповнення веб-ресурсів та прогнозування поведінки користувачів.

У ході виконання кваліфікаційної роботи підтверджено, що поєднання сучасних методів обробки природної мови, інструментів аналізу даних у Python та статистичних методів дозволяє отримувати обґрунтовані висновки щодо взаємозв’язку між змістом текстового контенту та поведінковими характеристиками аудиторії. Виявлені кореляційні залежності можуть слугувати основою для побудови рекомендаційних систем і підвищення рівня залученості користувачів.

Перспективи подальших досліджень полягають у розширенні функціональних можливостей системи шляхом застосування методів машинного навчання та глибоких нейронних мереж, дослідженні причинно-наслідкових зв’язків між характеристиками тексту та поведінковими реакціями, а також у масштабуванні рішення для роботи з великими обсягами даних у режимі реального часу та його інтеграції з сучасними аналітичними платформами.

Отже, результати виконаної кваліфікаційної роботи свідчать про доцільність і ефективність комплексного підходу до аналізу текстових і поведінкових даних. Поставлені завдання виконано, мету роботи досягнуто, а отримані наукові та практичні результати мають важливе значення для подальшого розвитку інформаційно-аналітичних систем у цифровому середовищі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бахрушин В.Є. Методи аналізу даних : навчальний посібник для студентів. Запоріжжя : КПУ, 2011. С. 268.

2. Васильєв О. Програмування мовою Python: аналіз даних, візуалізація та веб-скрапінг. Тернопіль : Навчальна книга Богдан, 2020. С. 424.

3. Глибовець М. М., Олецький О. В. Штучний інтелект та нейронні мережі. Теорія і практика : навчальний посібник. Київ : Видавничий дім «Києво-Могилянська академія», 2020. С. 284.

4. Гороховатський В. О., І. С. Творошенко. Методи інтелектуального аналізу та оброблення даних : навч. посіб. М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. Харків : ХНУРЕ, 2021. С. 92.

5. Єріна А. М., Пальян З. О. Статистичне моделювання та прогнозування : навчальний посібник. Київ : КНЕУ, 2018. С. 170.

6. Креневич А. П. Алгоритми і структури даних. Підручник. К.: ВПЦ «Київський Університет», 2021. С. 200.

7. Ланде Д. В., Фурашев В. М. Основи інформаційного та соціально-правового моделювання. Київ : ТОВ «ПанТот», 2019. С. 240.

8. Лень А. В., Карабін О. Й., Вовкодав О. В., Іваницький Р. І., Ясінський А. Порівняльний аналіз інструментів UML-моделювання для освітніх цілей підготовки майбутніх фахівців у галузі інформаційних технологій. *Електронний науковий журнал. Наукові праці Вінницького національного технічного університету*. 2025, № 3. С. 1–10.

9. Литвин В. В. Методи та засоби інженерії даних та знань навчальний посібник з грифом МОНУ. Львів : «Магнолія-2006», 2012. С. 241.

10. Любченко В. В. Методи та алгоритми комп’ютерної лінгвістики : навчальний посібник. Одеса : ОНПУ, 2019. С. 145.

11. Основи роботи з бібліотекою Scrapy для збору даних. URL: https://docs.scrapy.org/en/latest/intro/tutorial.html (дата звернення: 15.07.2025).

12. Пєлєтєтя О. В. Порівняльний аналіз бібліотек NLTK та SpaCy для задач обробки природної мови. *Системи управління, навігації та зв'язку*. 2021. Вип. 1 (63). С. 112–115.

13. Прокопенко Т. О., Мірошниченко І. С. Аналіз поведінкових факторів користувачів веб-ресурсів як складова SEO-оптимізації. *Економіка та суспільство*. 2020. № 21. URL: https://economyandsociety.in.ua/index.php/journal/article/view/120 (дата звернення: 18.08.2025).

14. Шевченко С. В., Коваленко О. І. Застосування методів кореляційного аналізу для виявлення закономірностей у великих масивах даних. *Вісник Київського політехнічного інституту*. Серія: Інформатика, управління та обчислювальна техніка. 2021. № 34. С. 45–52.

15. Ясінський А. М., Лень А. В. Аналіз взаємозв'язку текстових характеристик контенту та поведінкових показників користувачів у цифровому середовищі. *Сучасні цифрові технології та інноваційні методики навчання: досвід, тенденції, перспективи* : матеріали XVI Міжнародної науково-практичної інтернет-конференції (м. Тернопіль, 6–7 листопада, 2025 р.). Тернопіль : ТНПУ ім. В. Гнатюка, 2025. С. 265-267.

16. Ясінський А. М., Лень А. В. Методи і алгоритми аналізу кореляцій між текстовими даними і поведінковими показниками користувачів. *Сучасні цифрові технології та інноваційні методики навчання: досвід, тенденції, перспективи* : матеріали XV Міжнародної науково-практичної інтернет-конференції (м. Тернопіль, 10 квітня, 2025 р.). Тернопіль : ТНПУ ім. В. Гнатюка, 2025. С. 242–244.

17. Aggarwal C. C. Machine Learning for Text. Cham : Springer, 2018. P. 493.

18. Alakbarova I. Аналіз поведінки та інтересів людини на основі текстових даних. *International Journal of Education and Management Engineering* (IJEME), 2025. Т. 15, № 1. С. 1–9.

19. Bird S., Klein E., Loper E. Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. Sebastopol : O'Reilly Media, 2019. P. 504.

20. Brownlee J. Statistical Methods for Machine Learning: Discover How to Transform Data into Knowledge with Python. Melbourne : Machine Learning Mastery, 2019. P. 410.

21. Bruce P., Bruce A., Gedeck P. Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python. 2nd ed. Sebastopol : O'Reilly Media, 2020. P. 398.

22. Clifton B. Advanced Web Metrics with Google Analytics. 3rd ed. Indianapolis : John Wiley & Sons, 2018. P. 600.

23. Correlation Coefficient | Types, Formulas & Examples**.** URL: https://www.scribbr.co.uk/stats/correlation-coefficient-meaning/. (date of access: 19.02.2025).

24. Documentation for spaCy Natural Language Processing. URL: https://spacy.io/usage (date of access: 02.08.2025).

25. Hootsuite. 21 essential social media metrics you must track for success in 2024. Hootsuite Blog. 2024. URL: https://blog.hootsuite.com/social-media-metrics/ (date of access: 16.06.2025).

26. Hunter J., Dale D., Firing E., Droettboom M. Matplotlib: Visualization with Python. URL: https://matplotlib.org/ (date of access: 20.11.2025).

27. Jake VanderPlas. Python Data Science Handbook. URL: https://jakevdp.github.io/PythonDataScienceHandbook/03.00-introduction-to-pandas.html (date of access: 26.02.2025).

28. Jeong D. H., Jeong B. K., Ji S. Y. Використання машинного навчання для аналізу семантичних користувацьких взаємодій у візуальній аналітиці. *Information*, 2024. Т. 15, № 6. С. 351–362.

29. John A. Rice. Mathematical Statistics and Data Analysis. P. 245–256.

30. Jurafsky D., Martin J. H. Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. 3rd ed. draft. 2023. URL: https://web.stanford.edu/~jurafsky/slp3/ (date of access: 12.07.2025).

31. Kaushik A. Web Analytics 2.0: The Art of Online Accountability and Science of Customer Centricity. Indianapolis : Sybex, 2019. P. 498.

32. Lanuwabang L., Sarasu P. Виявлення аномалій на основі поведінкової інформації користувачів: огляд. *International Journal of Wireless and Microwave Technologies* (IJWMT), 2025. Т. 15, № 3. С. 54–65.

33. Liu B. Sentiment Analysis: Mining Opinions, Sentiments, and Emotions. 2nd ed. Cambridge : Cambridge University Press, 2020. P. 450.

34. McKinney W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Jupyter. 3rd ed. Sebastopol : O'Reilly Media, 2022. P. 547

35. Memon A. B., Sootahar D. K., Luhana K. K., Meyer K. Корпусний підхід до класифікації та тегування текстів у реальному часі на основі соціальних даних. *Frontiers in Computer Science*, 2024. Т. 6.

36. Mitchell R. Web Scraping with Python: Collecting More Data from the Modern Web. 2nd ed. Sebastopol : O'Reilly Media, 2018. P. 306.

37. Raschka S., Mirjalili V. Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2. 3rd ed. Birmingham : Packt Publishing, 2019. P. 770.

38. Reitz K. Requests: HTTP for Humans. URL: https://requests.readthedocs.io/en/latest/ date of access: 09.08.2025).

39. Richardson L. Beautiful Soup Documentation. 2023. URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (date of access: 06.08.2025).

40. Statistics: Data analysis and modelling. URL: https://mspeekenbrink.github.io/sdam-book/ (date of access: 21.03.2025).

41. Understanding Correlation: Measuring Relationships in Data. URL: https://www.datacamp.com/tutorial/correlation. (date of access: 20.02.2025).

42. VanderPlas J. Python Data Science Handbook: Essential Tools for Working with Data. 2nd ed. Sebastopol : O'Reilly Media, 2023. P. 598.

43. What Are Data Types and Why Are They Important? URL: https://amplitude.com/blog/data-types/ (date of access: 26.02.2025).

44. What is Data Analysis. URL: https://www.geeksforgeeks.org/data-analysis/what-is-data-analysis/ (date of access: 20.03.2025).

ДОДАТКИ

Додаток А

Візуалізація аналізу полярності та суб'єктивності публікацій (Scatter Plot)

Рис. 1. Діаграма розподілу полярності та суб'єктивності публікацій

Додаток Б

Таблиця частот використання слів у проаналізованих постах

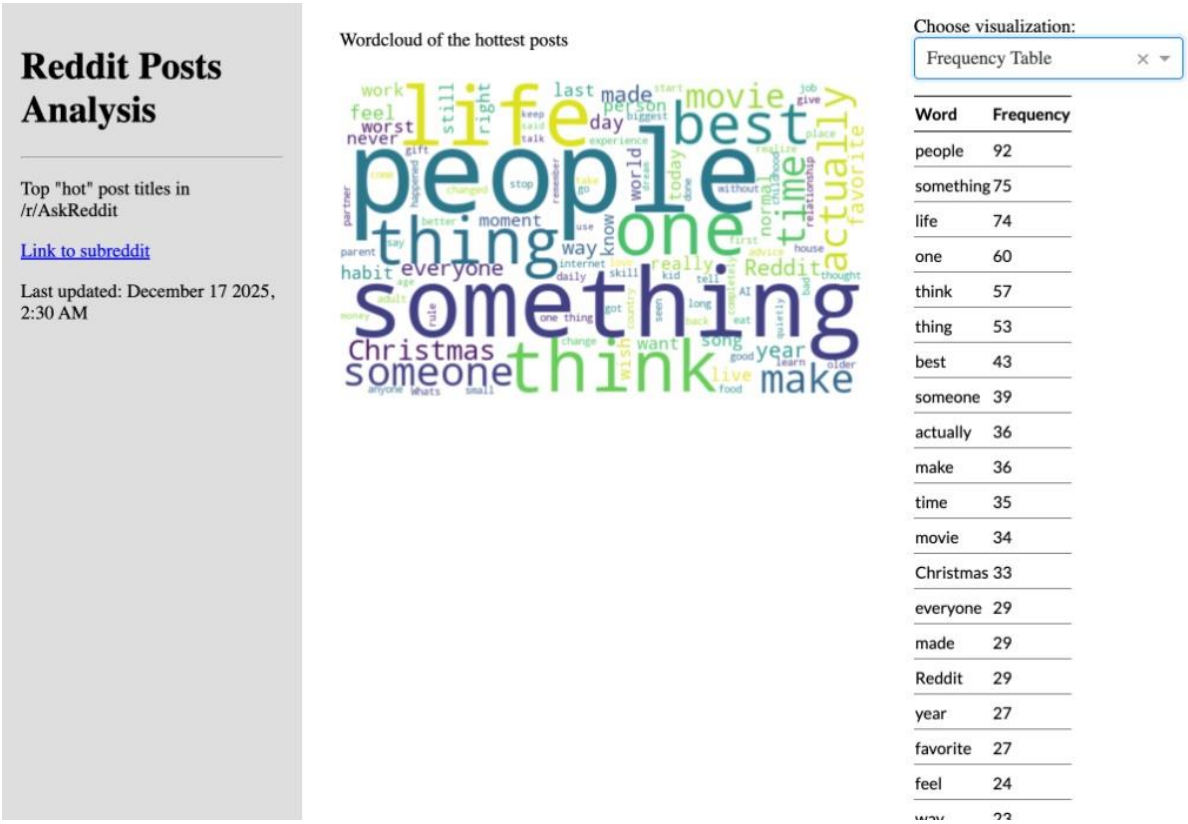

Рис. 1. Фрагмент таблиці частот використання слів у проаналізованих постах

Додаток В

Діаграма частотності вживання слів (Bar Chart)

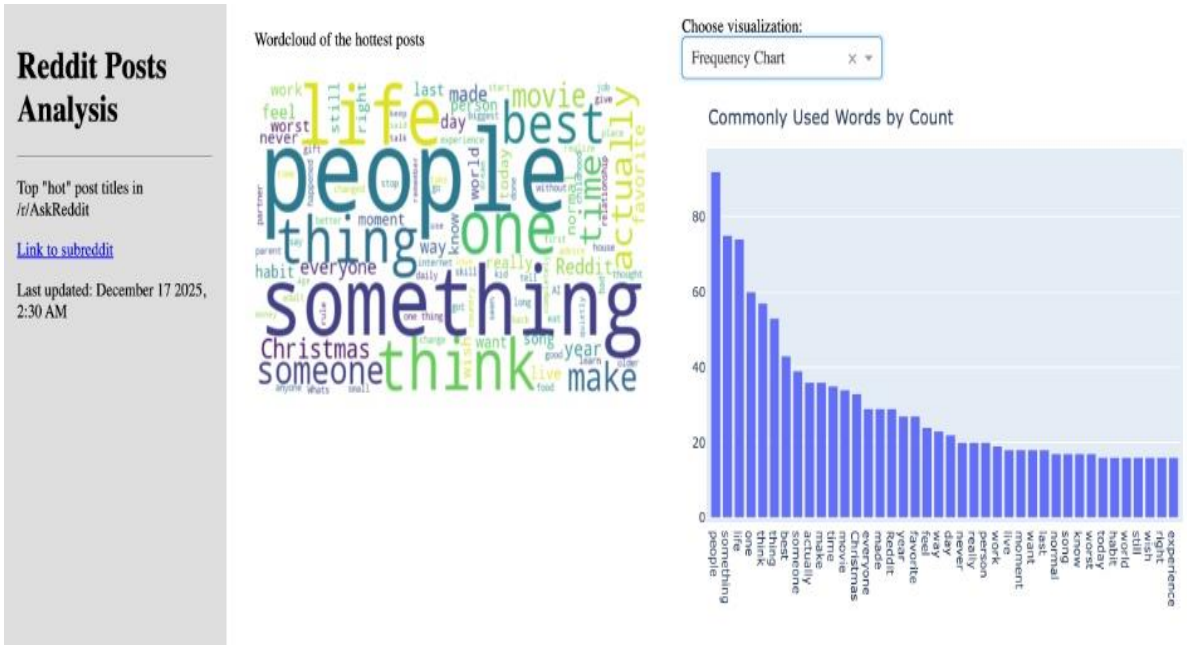

Рис. 1. Стовпчикова діаграма частот використання слів у проаналізованих постах