Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи та засоби парсингу веб-сайтів новин

(тема)

Виконав:

студент II курсу, групи СПм-20-2

Радченко А.В.
(прізвище, ініціали)

Спеціальність

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

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

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: доц. Носик А.М.
(посада, прізвище, ініціали)

Допускається до захисту

В.о. зав. кафедри ЕОМ _____ Волк М.О.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

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

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

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Радченку Антону Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи та засоби парсингу веб-сайтів новин

затверджена наказом по університету від " 24 " березня 2022 р. № 413 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18 травня 2022 р.

3. Вхідні дані до роботи

Парсинг, методи та засоби парсингу, веб-сайти новин, машинне навчання

4. Перелік питань, що потрібно опрацювати у роботі

Розглянути поняття парсингу

Дослідження методів пасрингу

Перегляд засобів парсингу

Оцінка переваг та недоліків методів та засобів парсингу

Створення власного програмного забезпечення для парсингу веб-сайтів новин

Оцінка якості роботи та виявлення недоліків та переваг власного продукту

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 13 слайдів

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

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Дослідження парсингу	29.03.22-31.03.22	
2	Дослідження методів та засобів парсингу	01.04.22-10.04.22	
3	Розробка власного програмного забезпечення	11.04.22-30.04.22	
4	Тестування програми	01.05.22-07.05.22	
5	Оформлення пояснювальної записки	08.05.22-13.05.22	
6	Подання кваліфікаційної роботи на	14.05.22-18.05.22	
	рецензування		

Дата видачі завдання 28 березня 2022 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Носик А.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 80 с., 37 рис., 1 дод., 24 джерел.

ПАРСИНГ, МЕТОДИ ПАРСИНГУ, ЗАСОБИ ПАРСИНГУ, АНАЛІЗ СТАТЕЙ, ВЕБ-САЙТИ, НОВИНИ, HTML, FAKE NEWS, REAL NEWS.

Метою кваліфікаційної роботи є дослідження сучаних методів та засобів парсингу веб-сайтів новин та створення програмного забезпечення для отримання та аналізу інформації з веб-сайтів новин.

У ході виконання кваліфікаційної роботи були проаналізовані існуючі методи парсингу та засоби для реалазції програмного забезпечення.

Наукова новизна результатів роботи полягає в доповненнях та покращеннях методів аналізу та отримання інформації з веб-сайтів новин, імпорту даних в зручному форматі для подальшого використання в інших програмах. Також програма доповнена модулем, який визначає правдивість новин.

Практичне значення одержаних результатів полягає в запропонованих технологічних рішеннях удосконалення процесів отримання інформації з веб-сайту та побудови модуля з використання машинного навчання, який був використаний для аналізу новин та виявлення серед них фейків.

ABSTRACT

Master’s thesis: 80 pages, 37 figures, 1 appendices, 24 sources.

PARSING, PARSING METHODS, PARSING MEANS, ANALYSIS OF ARTICLES, WEBSITES, NEWS, HTML, FAKE NEWS, REAL NEWS

The purpose of the qualification work is to study the current methods and tools for parsing news websites and to create software for obtaining and analyzing information from news websites.

During the qualification work, the existing parsing methods and tools for software implementation were analyzed.

The scientific novelty of the results of the work is the addition and improvements methods of analysis and retrieval of information from news websites and data import in a convenient format. The program is also supplemented by a module that determines the veracity of the news.

The practical significance of the obtained results lies in the proposed technological solutions to improve the processes of obtaining information from the website and building a module on the use of machine learning, which was used to analyze news and identify fakes among them.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 8
ВСТУП 9
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 11
1.1 Поняття парсера та парсингу 11
1.2 Законність парсингу 12
1.3 Види парсингу 14
1.4 Парсинг сайтів 16
1.5 Застосування парсингу 17
1.7 Обмеження парсингу 21
2 МЕТОДИ РОЗРОБКИ ПАРСИНГУ 24
2.1 Компоненти веб-додатків 24
2.2 Архітектура веб-парсеру 26
2.3 Інструменти веб-парсингу 28
2.4 Варіанти побудови власного веб-парсингу 30
2.4.1 Створення веб-парсеру за допомогою бібліотек 30
2.4.1 Headless браузери 32
2.4.2 SaaS рішення 34
2.5 Експорт даних 36
2.6 Проблеми побудови веб-парсеру 39
2.7 Вимоги для програми парсеру 41
2.8 Машинне навчання 42
3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 46
3.1 Основні використовувані інструменти та структура проетку 46
3.2 Запуск та опис проекту 47
3.3 Аналіз отриманих результатів 58
4 МАШИННЕ НАВЧАННЯ 61

4.1 Фейкові новини 61
4.2 Розробка програми виявлення феквоих новин 61
ВИСНОВКИ 69
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 70
ДОДАТОК А 73
Графічний матеріал кваліфікаційної роботи 73

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface – програмний інферфейс додатку.

CSS – Cascading Style Sheets – каскадні таблиці стилів.

CSV – Comma-Separated Values – текстовий формат, призначений для представлення табличних даних.

JS – JavaScript – мова програмування.

JSON – JavaScript Object Notation – текстовий формат обміну даними на основі JavaScript.

HTML – HyperText Markup Language – мова розмітки гіпертексту.

HTTP – HyperText Transfer Protocol – протокол передачі гіпертексту.

Saas – software as a service – програмне забезпечення як послуга.

TXT – Text File Type – файл, що містить текстові дані.

XML – eXtensible Markup Language – мова розмітки, що розширюється.

Xpath – XML Path Language – мова запитів до елементів XML-документу.

ВСТУП

На сьогоднішній день Інтернет є чимось більшим, ніж просто спосіб комунікації між людьми, це ще й найбільше джерело інформації у світі. І воно зростає на очах, адже щодня створюються численні веб-сайти та інтернет-портали, де розміщується важливий та цікавий контент. Однак опанувати таку кількість інформації людині не під силу, і тут на допомогу приходить комп'ютерна технологія веб-парсингу.

Веб-парсинг – винахід, покликаний значно спростити життя всім, хто так чи інакше стикається з необхідністю збору даних в Інтернеті. Пасринг – це программа, сервіс або скрипт, яка збира дані з вказаних веб-ресурсів,аналізує їх та видає в потрібному форматі.

Таким чином, можна отримувати, обробляти, систематизувати та зберігати у звичайному текстовому форматі дані веб-сторінок за лічені хвилини.

Метою атестаційної роботи є аналіз сучасних методів та засобів парсингу веб-сайтів новин та розробка програмного забезпечення для пошуку, отримання та аналізу статей. Об’єкт дослідження – процес парсингу веб-сторінок новин. Предмет досліджења є методи та засоби розробки програмного забезпечення для реалізації системи парсингу веб-сайтів новин.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- провести аналіз існуючих методів парсингу;
- проаналізувати існуючі програмні забезпечення та виявити їх недоліки та переваги;
- змоделювати систему та вибрати оптимальне програмне забезпечення;
- розробити та побудувати модель для вилучення потрібних даних з веб-сайтів новин;
- забезпечити експорт даних в інші програми;

- створити модуль для аналізу отриманих даних;
- розробити та протестувати програмне забезпечення.

Наукова новизна результатів роботи полягає в доповнені та покращені роботи методів пошуку,отримання інформації з веб-сайтів новин та створенні модуля за допомогою машинного навчання, який аналізує статі та виявляє серед них фейкові.

Практичне значення отриманих результатів полягає у створенні програмного забезпечення парсингу веб-сайтів новин. Результати розробленого програмного забезпечення для парсингу статей можуть бути використані іншими розробниками завдяки експорту даних в зручному форматі. Результати роботи опубліковані в матеріалах міжнародної науково-технічної конференції [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Тема, яка розглядається це дійсно важлива проблема, з якою стикаються багато компаній та інститутів. Веб-сканери майже такі ж старі, як і сама мережа. З постійною еволюцією та збільшенням кількості сторінок в інтернеті існувала необхідність у автоматичних програмах, які могли б сканувати та витягувати інформацію з Інтернету.

Крім того, пошукові системи стали необхідністю, і, як їх основну частину, потрібно було впровадити сканери та постійно проходити через всю можливу мережу, щоб зібрати всю необхідну інформацію, щоб зробити ці пошукові системи можливими. Інше використання — це інтелектуальний аналіз даних, де ці сканери повинні витягувати й аналізувати інформацію для статистичних цілей. Також згадуються деякі проблеми, з якими стикалися під час впровадження сканерів, як, наприклад, пошук повторюваних URL-адрес.

Як показано, багато досліджень було створено в цій області через незліченну кількість можливих застосувань. Також через величезну різноманітність Інтернету існує багато різних реалізацій. В Інтернеті є величезна кількість даних, представлених у дуже різноманітних способах і з багатьма різними конструкціями. Щоб мати можливість сканувати більшість даних, потрібно розуміти і вивчати всі подібні структури та можливість їх сканування та вилучення з них інформації.

1.1 Поняття парсера та парсингу

В сучасному світі інформація є найціннішим ресурсом, а її найбільшим джерелом є інтернет. В таких випадках мова йде про величезні обсяги даних, обробити які вручну – неможливо. Тому автоматизований збір і обробка даних є надзвичайно корисним і важливим процесом. Для швидкої обробки інформації використовуються парсери.

Парсер – це програмне забезпечення, яке здатне швидко обробляти інформацію згідно з алгоритмом і повертати потрібний результат. Він є ефективним рішенням для автоматизації збору і зміни інформації. Парсер здатний швидко обходити сотні веб-сторінок, знаходити необхідну інформацію і повертати її в визначеному вигляді [2].

Найпоширенішими парсерами є пошукові роботи, які використовуються пошуковими системами. Вони аналізують сторінки, зберігають інформацію про них в базі даних і потім, під час пошуку, повертають користувачу всю актуальну інформацію.

Для розробки парсерів використовується багато мов програмування, серед яких: Python, JS, Java, C#, Perl, Go та інші.

Парсинг – це процес, який автоматично збирає та витягує дані з Інтернету. Метою веб-скрейпінгу є вилучення великої кількості даних і збереження в локальному середовищі. На перших порах єдиним способом отримувати дані з веб-сайтів було копіювання та вставка того, що бачили на веб-сайті. Веб-скрейпінг стає популярним методом, оскільки він дозволяє новим стартапам швидко отримувати великі обсяги даних. Найбільш типовими прикладами веб-скрейпінгу є веб-сайти для порівняння цін і оглядів. Великі корпорації також використовують цю техніку для ранжування та індексації веб-сторінок.

Процес веб-скрейпінгу включає сканування та розбір. Сканери переходять до цільової URL-адреси, завантажують вміст сторінки та переходять за посиланнями у вмісті відповідно до реалізації розробника. Парсери аналізують завантажений вміст і витягують дані.

1.2 Законність парсингу

Хоча численні інструменти та технології були доступні для організації автоматизованого доступу до інтернет ресурсів за допомогою веб-парсингу, законність веб-скрапінгу все ще залишається «сірою зоною» в правовому полі.

Не існує законодавчих актів, які б безпосередньо стосувалися веб-скрапінгу. Наразі веб-скрапінг керується набором пов’язаних, фундаментальних правових теорій і законів:

- порушення авторських прав;
- порушення договору;
- закон «Про комп’ютерні шахрайства та зловживання» (CFAA);
- порушення прав власності.

Нижче наведені деякі конкретні подробиці того, як ці фундаментальні правові теорії [3] застосовуються до веб-парсингу:

- правила використання. У правовому полі часто говориться, що власник веб-сайту може ефективно запобігати програмному доступу до веб-сайту, явно забороняючи це в політиці «Правил використання», розміщеної на веб-сайті. Недотримання цих умов може призвести до порушення контракту зі сторони користувача веб-сайту. Щоб притягнути когось до відповідальності за порушення «умов використання», користувачеві веб-сайту необхідно укласти явну угоду з власником веб-сайту про дотримання політики «Загальних положень та умов» (наприклад, погодитись з цими правилами на сайті). Проста заборона веб-сканування і веб-парсингу на сайті не може перешкоджати комусь автоматизовано отримувати інформацію з юридичної точки зору;

- матеріал, захищений авторським правом. Веб-парсинг та повторна публікація матеріалів, якими володіє власник веб-сайту та які є явно захищені авторським правом власника веб-сайту, може призвести до відкриття справи про порушення авторських прав;

- веб-парсинг з визначеною метою. Будь-яке незаконне або шахрайське використання даних, отриманих за допомогою веб-скрапінгу, заборонено законом. Наприклад, особа, яка отримує доступ до даних з Інтернету, які є конфіденційними та захищеними, може бути притягнута до відповідальності за законом про комп’ютерні шахрайства та зловживання, якщо збиток перевищує 5000 доларів США. В Інтернеті це досить поширена практика, коли

людина свідомо отримує доступ до «преміум-контенту», а потім перепродає його, використовує у своїх цілях його або 30 продовжує отримувати доступ до контенту через несанкціонований канал навіть після отримання попереджувального листа від власника веб-сайту;

- завдання шкоди ресурсам та веб-сайту. Якщо веб-скрапінг перевантажує або пошкоджує веб-сайт або сервер, то особа, відповідальна за нанесені збитки, може бути притягнута до відповідальності за законом про порушення прав власності. Тим не менш, збиток повинен бути матеріальним і таким, що його можна буде легко довести в суді, для того щоб власник веб-сервера мав право на фінансову компенсацію;

- індивідуальна конфіденційність. Проекти, що використовують дані, зібрані з веб-сайтів, можуть ненавмисно порушити конфіденційність приватних осіб. Навіть якщо не порушено індивідуальну конфіденційність, проблема полягає в тому, що користувачі веб-сайту можуть не погодитися на використання їх даних третіми особами. Таким чином, використання цих даних без згоди є порушенням прав. Ці порушення конфіденційності та прав можуть призвести до серйозних наслідків для власника веб-сайту, враховуючи останні скандали в сфері конфіденційності та інформації про користувачів (справа Facebook та Cambridge Analytica);

- комерційна таємниця. Так само як і люди, компанії також мають право зберігати певні аспекти своєї діяльності в конфіденційності. З допомогою веб-скрапінгу можна ненавмисно розкрити комерційні таємниці або просто конфіденційну інформацію про організацію. Це може пошкодити репутацію компанії і призвести до матеріальних та фінансових втрат.

1.3 Види парсингу

Відповідно до розмежування між бізнес-аналітикою та бізнес-розвідкою, відповідні аналітичні методи можна розділити на загальні категорії [4] . Існує поділ на чотири категорій:

- описовий парсинг;
- дослідницький парсинг;
- діагностичний парсинг;
- прогнозуючий парсинг.

Описовий парсинг, також відомий як описовий аналіз даних, зосереджений даних минулих років. Він організовує та структурує емпіричні дані. Аналіз даних спрямовано отримання відповіді питання: «Що сталося?». Наприклад, у ньому міститься така інформація, як обсяг продажу за останній квартал або тип та кількість запитів на обслуговування. Для отримання таких результатів описовий аналіз може отримати дані з різних джерел і узагальнити, систематизувати і структурувати інформацію. Однак описовий аналіз не дає відповіді на такі питання, як «Чому щось трапилося?». Описовий аналіз даних часто комбінується з іншими методами аналізу.

Метою дослідницького аналізу даних є пошук зв'язків у даних та генерація гіпотез. До проведення цього виду парсингу існують обмежені знання про взаємозв'язок даних та змінних. Типовою сферою застосування аналізу розвідувальних даних є видобуток даних. Виявлення кореляцій за допомогою аналізу розвідувальних даних дозволяє зробити висновки про причини процесів.

Аналіз діагностичних даних стосується саме питання «Чому щось трапилося?». Порівнюючи історичні та інші дані, виявляючи закономірності та виявляючи взаємини, він знаходить причини чи взаємини. За допомогою аналізу діагностичних даних організації можуть вирішувати конкретні проблеми з виявлення їх корінних причин.

Прогнозований парсинг, також відомий як прогностичний аналіз, дозволяє зазирнути у майбутнє. Він відповідає питанням: «Що станеться?». Щоб зробити правильний прогноз, під час аналізу даних використовуються результати описаних раніше методів описового, дослідницького чи діагностичного аналізу, а також алгоритмів та методів штучного інтелекту та машинного навчання . Пошук кореляцій, причин та тимчасових тенденцій

робить майбутні тенденції передбачуваними. Імовірність та точність прогнозування багато в чому залежить від якості даних, закономірностей, знайдених кореляцій та тенденцій, а також від інтелекту алгоритмів. Наприклад, можна передбачити майбутні продажі чи поведінку покупців.

1.4 Парсинг сайтів

Парсинг сайтів - це автоматичний спосіб отримання великих обсягів даних із веб-сайтів.Більшість цих даних є неструктуровані дані у форматі HTML, які потім перетворюються на структуровані дані в електронній таблиці або базі даних, щоб їх можна було використовувати в різних додатках. Існує безліч різних способів виконання веб-пасрингу для отримання даних із веб-сайтів. До них відносяться використання онлайн-сервісів, визначених API або навіть створення коду для веб-скрейпінгу з нуля. Багато великих веб-сайтів, таких як Google, Twitter, Facebook, StackOverflow і т. д. у них є API, які дозволяють вам отримувати доступ до їх даних у структурованому форматі. Це найкращий варіант, але є й інші сайти, які не дозволяють користувачам отримати доступ до великих обсягів даних у структурованій формі або просто не настільки технологічні. У цій ситуації краще використовувати веб-парсер для пошуку даних на веб-сайті.

Веб-парсинг можна розділити на 2 типи.

1 Тип. Технічний парсинг сайту, яким переважно користуються SEO фахівці для виявлення різних проблем сайту:

- пошук битих посилань та некоректних 30* редиректів;
- виявлення дублів або інших проблем із мета-тегами Title, Description та заголовками h1;
- для аналізу правильної роботи Robots.txt;
- перевірка налаштування мікророзмітки на сайті;
- виявити небажані сторінки, які відкриті для індексації;
- інші технічні завдання.

На основі отриманих даних фахівець складає технічні завдання усунення виявлених проблем.

2 Тип. Парсинг сайту для розвитку бізнесу. Ось деякі приклади таких завдань:

- збір інформації про асортимент конкурентів;
- парсинг назв товарів, артикулів, цін та іншого для наповнення власного інтернет-магазину. Це може бути як одноразове завдання, так і на основі регулярного моніторингу;
- аналіз структури сайтів-конкурентів з метою покращення та розвитку власної структури.

Принцип роботи парсерів для веб-сторінок однаковий, зазвичай складається з 3 етапів.

1 Етап. Запит-відповідь.Перший крок — запитати цільового сайту вміст певної URL-адреси. Натомість парсер отримує запитану інформацію у форматі HTML.

2 Етап. Розбір та вилучення. Синтаксичний аналіз зазвичай застосовується до будь-якої комп'ютерної мови. Це процес розпізнавання коду у вигляді тексту та створення структури у пам'яті, яку комп'ютер може зрозуміти і з якою буде працювати. Парсер бере HTML-код і витягує звідти відповідну інформацію - таку, як заголовок сторінки, абзаци, підзаголовки, посилання, виділення жирним, потрібні теми і так далі, проводячи парсинг тексту [5].

3 Етап. Завантаження даних. Отримані дані завантажуються та зберігаються. Формат файлу визначається таким чином, щоб його можна було відкрити в іншій потрібній програмі. Для Google Таблиць це, наприклад, CSV, для парсингу бази даних - JSON і таке інше.

1.5 Застосування парсингу

У роздрібній торгівлі існує багато можливостей використання парсингу.

Наприклад, моніторинг цін конкурентів або аналітика ринку, де парсинг використовують для обробки даних та вилучення з них цінної для маркетологів інформації.

Так, для електронної комерції може знадобитися незліченну кількість зображень та описів товарів. Їх не можна просто створити за пару-трійку днів, тому що навіть просто скопіювати і вставити кожен займе певний час. Набагато простіше та швидше створити парсинг та швидко «вичепити» все потрібне. Або взяти аналітику ринкових цін - регулярний парсинг веб-сторінок конкурентів допоможе своєчасно помічати та враховувати всі зміни на ринку.

Раніше аналіз фондового ринку обмежувався вивченням фінансової звітності компаній та, відповідно, інвестуванням у найбільш підходящі цінні папери. Сьогодні кожна новина чи зміни настроїв у політиці та суспільстві важливі для визначення поточних трендів. Як отримувати такі альтернативні дані? Тут допомагає парсинг. Він дозволяє отримати всю сукупність інформації, пов'язаної з ринком, та побачити загальну картину. Не кажучи вже про те, що витягувати річні звіти та всі стандартні фінансові дані з будь-якого сайту набагато простіше та швидше за допомогою парсингу.

По суті кожен парсер проходить навчання. Це дозволяє штучному інтелекту виявляти закономірності. Однак для того, щоб встановити потрібні зв'язки, необхідно передати в комп'ютерний розум багато даних та допомогти зв'язати одне з одним. Часто парсери застосовуються у технологіях AI, щоб забезпечити регулярний потік навчальної інформації.

Парсинг електронної пошти дозволяє аналізувати вхідні та вихідні повідомлення. Потім їх вміст можна інтегрувати в різні програми за допомогою програмного інтерфейсу API або зібрати для подальшого аналізу.

Електронна пошта – одна з найбільш завантажених даними форм сучасного спілкування. Звичайне надсилання одного електронного листа збирає, передає та інтерпретує близько 100 Кб даних. Помножте на мільярди і ви зрозумієте, чому компаніям може бути складно керувати такими обсягами інформації. На щастя, вирішення проблеми сьогодні беруть він спеціальні

парсери.

Більшість компаній використовують рішення на основі API для трьох основних видів додатків.

1 Підтримка клієнтів. Парсинг забезпечує фіксацію взаємодій з клієнтами електронною поштою, доставку потрібних повідомлень користувачам, збирання та аналіз вхідних повідомлень для зберігання та відображення в програмах підтримки;

2 Програми для керування відносинами з клієнтами, CRM. Програми CRM часто записують все листування з клієнтом, щоб цінні дані з історії стосунків збереглися. А надсилаючи відповіді клієнтів через службу парсингу, програми CRM можуть отримувати попередньо проаналізовані дані.

3 Соціальні програми. Програми для соцмереж часто дозволяють спілкуватися електронною поштою, щоб полегшити користувачам відстеження бесіди. Парсинг вхідної пошти — простий і швидкий спосіб налаштування поштових програм таким чином, щоб вони розширювалися в міру масштабування листування в соцмережах.

Парсер телефонів це програма або хмарне програмне забезпечення, яке може збирати дані про потенційних покупців або клієнтів з різних веб-сайтів, відкритих джерел та інших матеріалів, ... За допомогою парсера телефонів Ви зможете: зібрати телефонні номери потенційних клієнтів або партнерів; вказати лише список сайтів, з яких Ви хочете зібрати телефони та запустити парсер; зберегти зібрану інформацію у будь-якому зручному форматі (Excel, TXT, WordPress, MySQL тощо).

Багато компаній потребують постійно розшукувати можливих покупців чи партнерів з метою просування послуг і товарів, або тих чи інших маркетингових даних. Ручна обробка займає дуже багато часу і безліч сил.

Для того, щоб зберегти свій час, необхідно використовувати саме автоматизований збір інформації - парсери. Парсери в лічені хвилини проаналізують всю інформацію про клієнтів, зберт її і відформатують для зручного використання, наприклад, в таблиці Excel. Це набагато зручніше, ніж

користуватися ручним копіюванням і пошуком даних.

Серед переваг парсингу можна виділити:

- програми, скрипти, а також онлайн-сервіси дозволяють збирати важливу статистичну інформацію для отримання конкурентної переваги у вибраній бізнес-ніші. Наприклад, вивчати товарний асортимент конкурентів, збирати актуальні ціни, отримувати інформацію про діючі знижки, акції або розпродажі. Потрапляючи до рук грамотного маркетолога, ці дані становлять величезний інтерес для бізнесу, т.к. є основою щодо маркетингових досліджень;
- автоматизація виключає людську працю, що знижує навантаження і дозволяє оптимізувати бюджет підприємства з допомогою скасування великого відділу аналітики. Крім того, на відміну від людини, алгоритми програм здатні обробляти сотні, тисячі сторінок протягом доби, виконуючи місячний обсяг роботи кваліфікованого спеціаліста за кілька годин;
- скріпінг (парсинг плюс) - процес максимально точний. Адже скрипт отримує лише зазначену інформацію, ігноруючи різний «інформаційний шум». А з появою та підключенням нейромереж (алгоритмів машинного навчання при вибірці необхідних даних), парсери навчилися структурувати їх не гірше за людину [6].

Серед недоліків парсингу можна виділити:

- тривалість обробки зібраних даних (залежить від обчислювальних потужностей комп'ютера або сервер онлайн-сервісу);
- складнощі під час аналізу. Умовний недолік, т.к. сучасні парсери плюс здатні формувати прості та зрозумілі звіти;
- обмеження за швидкістю. При підозріло частому зверненні до сервера система захисту сайту може заблокувати скрипт, що перерве збір інформації.

1.7 Обмеження парсингу

З точки зору розробника веб-сайтів, парсери можна сприймати як неприємність. Вони маскуються під справжніх відвідувачів веб-сайту і можуть здійснювати багато запитів, що значно збільшує навантаження на сервер.Для того, щоб контролювати кількість та типи поданих запитів, багато доменів містять robots.txt файл, який повідомляє розробникам, як вони повинні взаємодіяти з їхнім сайтом. Приклад файлу robots.txt наведено нижче на рисунку 1.1.

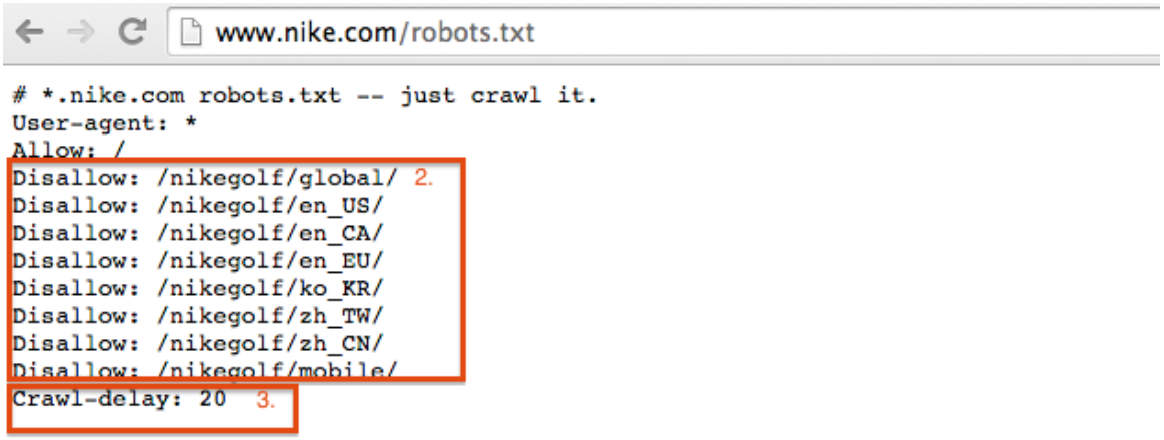

Рисунок 1.1 – Приклад robots.txt

Більшість основних пошукових систем таких як: Bing, Google, Yahoo виконують даний файл. Проте файл є суто консультативним і спирається на чесність розробників парсеру.

Протокол Sitemaps дозволяє веб-розробникам інформувати парсери про URL-адреси веб-сайту, які доступні для сканування. Також дозволяє включати додаткову інформацію про кожну URL-адресу: коли вона востаннє оновлювалася, як часто вона змінюється та наскільки є важливою у відношенні до інших URL-адрес веб-сайту. Даний протокол, наприклад, дозволяє пошуковим системам ефективніше сканувати сайт та знаходити URL-адреси, які можуть бути ізольованими від решти вмісту сайту. Протокол Sitemaps – це

протокол включення або виключення URL-адреси та доповнення до robots.txt. Формат протоколу Sitemap складається з XML тегів. Сам файл повинен бути закодований UTF-8. Sitemaps також можуть бути просто текстовим списком URL-адрес стиснутим у форматі .gz. Максимальний розмір файлу Sitemaps становить 50 Мб або 50 000 URL-адрес. Так само, як і robots.txt, даний протокол носить лише консультативний характер і спирається на чесність розробників [7]. Приклад протоколу sitemaps зображено на рисунку 1.2.

```
<?xml version="1.0" encoding="UTF-8"?>

<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">

   <url>

      <loc>https://www.example.com/</loc>

      <lastmod>2018-08-24</lastmod>

      <changefreq>weekly</changefreq>

      <priority>0.5</priority>

   </url>

</urlset>
```

Рисунок 1.2 – Приклад sitemaps

На відміну від вищенаведених обмежень, даний метод є одним із найефективніших та найбільш часто використовуваних способів захисту від частих автоматизованих запитів. Captcha (повністю автоматизований публічний тест Тюринга для розрізнення комп’ютерів та людей) – комп’ютерний тест типу запит-відповідь, який використовується для визначення, хто користується системою – комп’ютер чи людина, на рисунку 1.3. У найпоширеніших варіантах Captcha, від користувача вимагається введення символів, які зображені в спотвореному вигляді на пропонованому малюнку, інколи з доданням шуму або напівпрозорості. Також одним з поширених варіантів є розпізнавання зображень.

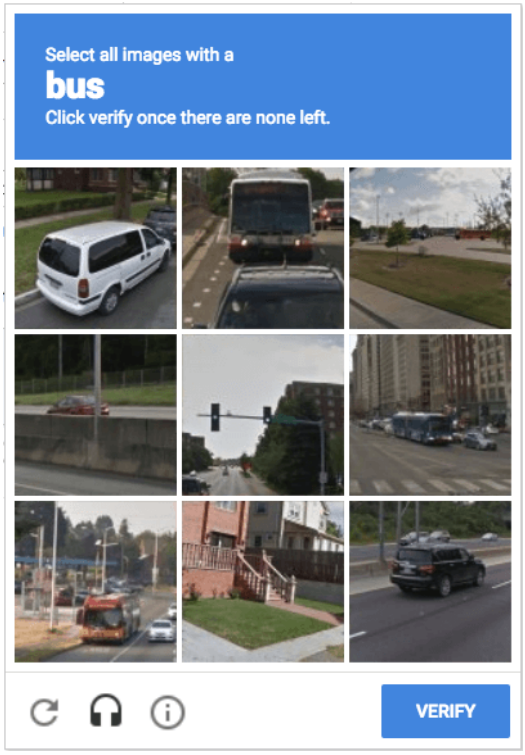

Рисунок 1.3 – Приклад Captcha

2 МЕТОДИ РОЗРОБКИ ПАРСИНГУ

2.1 Компоненти веб-додатків

Веб-сайти можуть мати різні компоненти та структури. Однак загалом виділяють 4 компоненти: HTML, CSS, JavaScript та мультимедійні ресурси. Крім того, об’єктна модель документа, селектор CSS і Xpath надають розробнику можливість контролювати веб-контент. Опишемо коротко їх:

- HTML – це мова розмітки, яка визначає структуру веб-додатка. Елементи HTML є основними компонентами сторінки HTML;

- CSS – описує, як документ HTML представляється кінцевим користувачам;

- JavaScript – це мова сценаріїв, яка здебільшого використовується в Інтернеті. У більшості веб-додатків JavaScript дає змогу користувачам взаємодіяти з компонентами програми, перевіряти введені дані та здійснювати виклик API на серверний сервер;

- мультимедійні файли, як правило, є конфіденційними даними, які не слід завантажувати та зберігати в базі даних Web Scraping. Однак зберігання посилань на мультимедійні файли може бути прийнятним, оскільки саме це робить веб-браузер, коли користувач відвідує веб-сторінки, і великі компанії, такі як Google, також зберігають їх;

- Xpath – це потужна мова запитів для отримання вузлів із XML-документа. Оскільки HTML є підмножиною XML, XPath також можна використовувати для отримання елемента HTML [8].

Незважаючи на те, що XPath є потужним, його важко читати та писати. Селектор CSS є більш зручною альтернативою, яка дозволяє вибрати елемент HTML на основі стилів CSS, імені класу або ідентифікатора, пов’язаного з елементом. На рисунку 2.1 показаний приклад HTML-розмітки сторінки на веб-сайті з CSS класами.

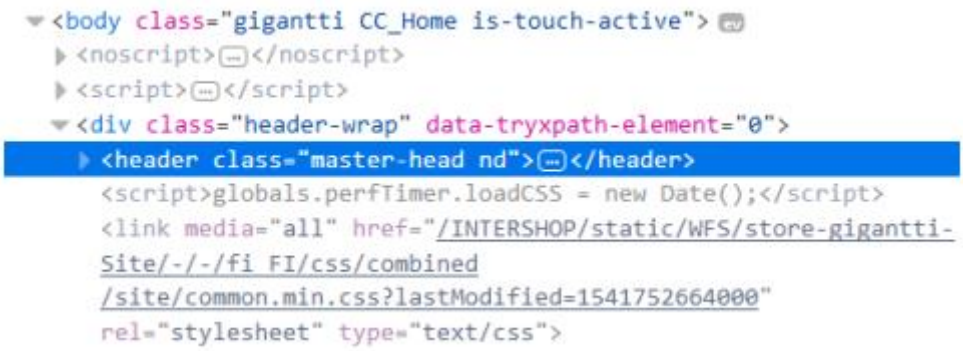

Рисунок 2.1 – HTML-розмітки сторінки на веб-сайті з CSS класами

Проаналізувавши рисунок 2.1, для вибору елемента div з ім’ям класу обгортання заголовка є принаймні 2 способи вибрати в Xpath:

- `/html/body/div` : отримати перший div всередині тегу html;
- `//div[contains(@class,'header-wrap')]` : отримати всі теги div, які мають заголовок класу ”div.header-wrap”.

У цьому випадку обидва повертають однаковий результат. Згідно з кількома вимірюваннями тесту з веб-сайту Elemental Selenium, у старих браузерах селектори XPath також дають більш швидку продуктивність при отриманні елементів HTML. Однак у сучасних браузерах із розумною оптимізацією швидкість HTML Element майже однакова як XPath, так і CSS.

За допомогою селектора CSS ”div.header-wrap” це легше зрозуміти. Однак створити більш розширені критерії вибору елементів HTML складніше.

Важливим компонентом також являється DOM — це інтерфейс для мов програмування для доступу до документів HTML. Він описує ці документи як вузли і об’єкти дерева, що дає можливість мовам програмування, таким як JavaScript, вибирати вузли за допомогою селекторів XPath або CSS, щоб динамічно змінювати структуру веб-сторінки на основі взаємодії з користувачем. На рисуноку 2.2 зображено приклад ієрархічного дерева веб-документа [9].

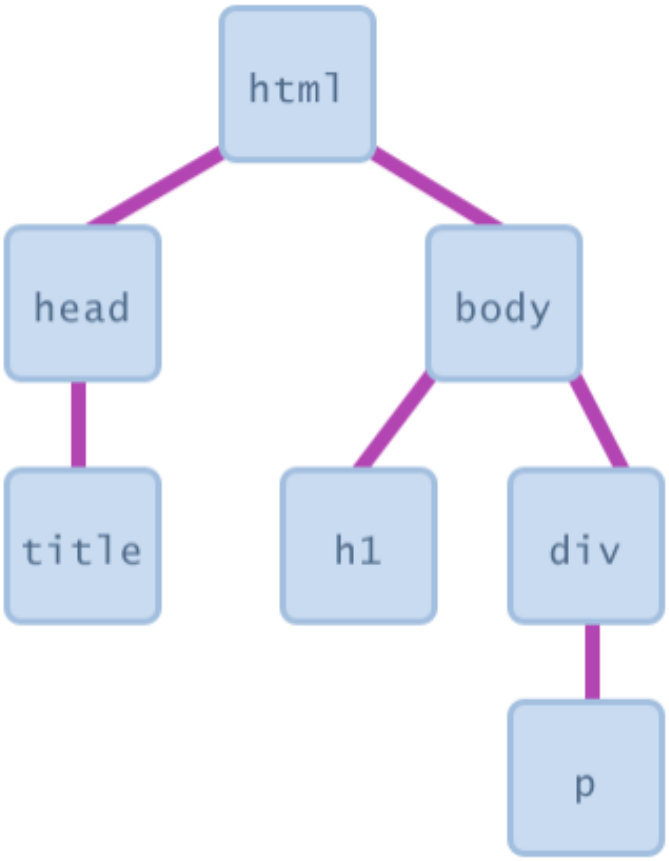

Рисунок 2.2 – Ієрархічне дерево веб-документа

2.2 Архітектура веб-парсеру

Існує багато фреймворків веб-скрейпінгу, написаних різними мовами програмування. Серед них Scrapy є одним із найзріліших фреймворків веб-скрапінгу, написаних на Python. Як правило, більшість систем веб-скрепінгу мають схожу архітектуру та компоненти.

На рисунку 2.3 представлені важливі компоненти архітектури. Кожен компонент відповідає лише за конкретне завдання. Таким чином, написання тестів простіше, а скрепер легше підтримувати.

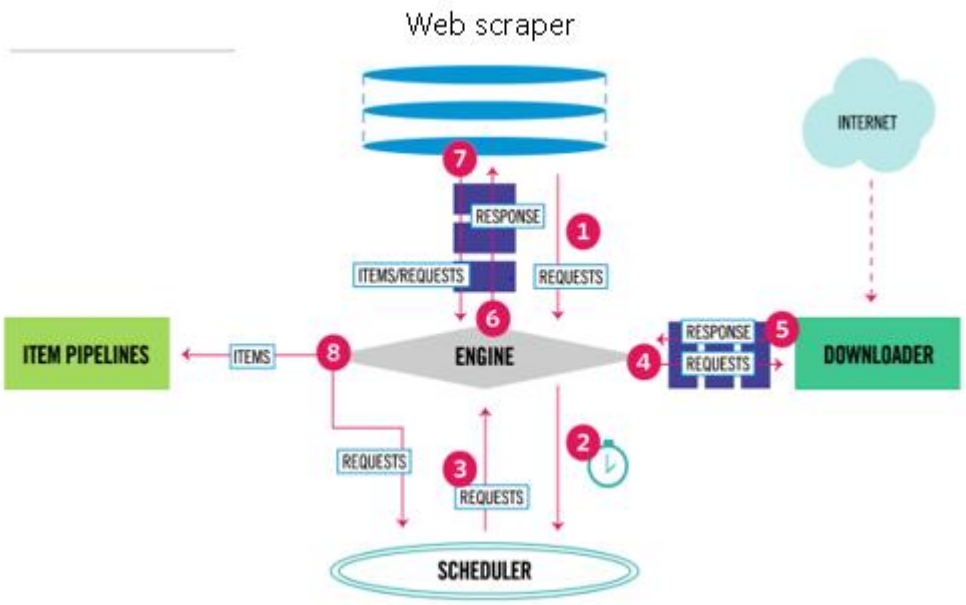

Рисунок 2.3 – Архітектура Scrapy

Двигун є центром фреймворку Scrapy. Він керує потоком між компонентами Scrapy. Він також відповідає за прослуховування та ініціювання подій, щоб реагувати на певні дії, такі як помилка запиту, помилка відповіді, винятки [10].

Планувальник керує, коли завдання має бути запущено, і він має прямий зв’язок із чергами завдань. Він може регулювати, скільки затримок має кожен запит.

Завантажувач — це місце, де виконуються HTTP-запити. Потім він зберігає та віддає вміст відповіді HTTP назад до механізму в звичайному випадку, коли реальний браузер не використовується. У випадку, якщо для виконання запитів використовується справжній браузер, проміжне програмне забезпечення, яке може керувати браузером, повністю замінить завантажувач.

Скрепер — це класи, написані розробниками, щоб визначити, які дії має виконувати парсер, щоб отримати та проаналізувати певний веб-вміст.

Item Pipeline обробляє дані, які повертає скрапер, і виконує перевірку, користувацьку трансформацію, очищення та збереження даних у сховищі даних, наприклад Redis, або Postgres.

2.3 Інструменти веб-парсингу

Існують різні інструменти веб-скрапінгу та послуги, що пропонують компанії, дуже важко визначити які з них є більш ефективними та надійніші в порівнянні з іншими. Основуючись на тому, як вони працюють, нижче наведено класифікацію інструментів веб-скрапінгу рисунок 2.4, доступних на ринку [11].

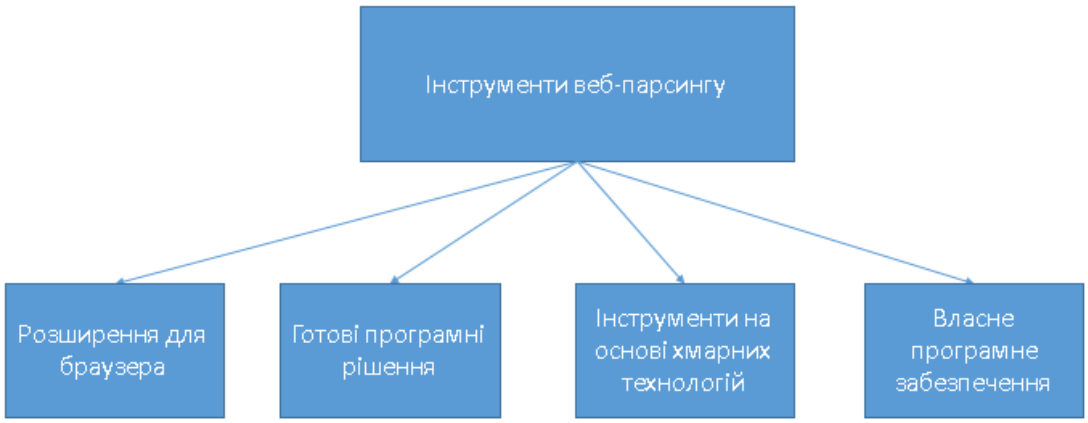

Рисунок 2.4 – Види інструментів веб-парсингу

1 Розширення для браузера. Розширення для браузера є прекрасним інструментом, щоб витягти невеликі частини даних. Веб-скрапінг за допомогою плагіна для браузера, а не окремого програмного забезпечення, встановленого на ПК є найбільш ефективним рішенням для перегляду, аналізу та подальшої модифікації даних. Розширення для браузера дозволяє встановити його і вибрати спосіб, за допомогою якого потрібно отримати дані з вибраного веб-сайту. Дані буде завантажено у вибраному форматі (наприклад CSV, JSON) або в будь-якому іншому зручному форматі. Розширення для браузера майже не вимагають зусиль для встановлення та налаштування. Такі розширення мають лише мінімальні налаштування для отримання даних з веб-сайтів. Також, через свою простоту, такі рішення мають

певні обмеження. Розширення для браузера може витягти дані лише з однієї сторінки за один раз. Для обробки великої кількості даних є інші типи інструментів для веб-парсингу.

2 Готові програмні рішення. Оскільки попит на дані та інформацію зростає з кожним днем, IT компанії взялися за розробку повноцінного програмного забезпечення для веб-парсингу. Як і будь-яке інше програмне забезпечення, програмне забезпечення для веб-скрапінгу передбачає його повноцінне встановлення на ПК. Більшість таких програм повністю сумісні з найбільш поширеними операційними системами (Windows, MacOS). Таке програмне забезпечення має більш широкий спектр налаштувань, тому таке рішення може бути більш гнучким і ефективним. Для початку роботи достатньо лише підлаштувати веб-скрапер під свої потреби. Повноцінні програми для веб-скрапінгу використовують найбільш популярні і зручні у подальшому використанні формати для зберігання зібраних даних, такі як JSON, CSV. 23 На відміну від розширень для браузерів, програми для ПК дозволяють одночасно сканувати декілька сторінок і можуть обробляти більшу кількість інформації. Вони розраховані на невеликі та середні сайти.

3 Інструменти на основі хмарних технологій. У порівнянні з іншими інструментами, веб-скрапінг на основі хмарних технологій вважається найбільш ефективним та надійним рішенням. Використовуючи інструменти на основі хмарних технологій не потрібно встановлювати жодного програмного забезпечення. Зазвичай налаштування в таких системах дуже прості і водночас мають найбільше варіацій порівняно з двома попередніми типами веб-скраперів. Одразу після налаштування система починає роботу. Дані можна отримати через API у веб-браузері і завантажити у вибраному форматі. Хмарні технології не обмежують веб-скрапінг певним об’ємом даних завдяки використанню декількох обчислювальних середовищах. Таким чином можна зробити висновок, що використання хмарних технологій для веб-парсингу є найбільш доцільним при роботі з великими об’ємами даних та складними веб-сайтами.

У порівнянні з іншими інструментами, які вимагають втручання людини на деяких етапах веб-скрапінгу, інструменти веб-скрапінгу з використанням хмарних технологій можуть полегшити процес підтримки веб-скрапінгу та зробити його повністю автоматизованим і безпроблемним.

4 Власне програмне забезпечення. Потрібно взяти до уваги, що можна створити своє власне програмне забезпечення для веб-скрапінгу використовуючи майже будь яку мову програмування. Таке рішення має свої переваги. Наприклад, користувач може налаштувати програму повністю під себе і створити найкращі умови для збору інформації саме з тих ресурсів, які йому потрібні.

2.4 Варіанти побудови власного веб-парсингу

2.4.1 Створення веб-парсеру за допомогою бібліотек

Цей підхід вимагає розуміння процесу формування запитів та логіки роботи програми, що тягне за собою додаткові витрати на вивчення низькорівневої роботи сайту. Можливо, це доречно і виправдано для одиничного сайту, але якщо потрібно написати кілька парсерів для декількох різнорідних сайтів за обмежений час - навряд чи.

До таких інструментів належать численні бібліотеки для різних мов програмування: jSoup для Java, Simple HTML Dom для PHP, lxml.html для Python та інші.Давайте розглянемо одну з них.

jSoup це бібліотека, використана для аналізу документа HTML. Soup надає API для отримання та маніпулювання даними з URL або файлів HTML. Використовує методи схожі на DOM, CSS, JQuery, щоб отримати дані та маніпулювати ними [12].

У цьому прикладі програми на лістингу 2.1 показано, як отримати сторінку за URL-адресою; отримувати посилання, зображення та інші покажчики:

Лістинг 2.1– Приклад коду jSoup

```
package org.jsoup.examples;
import org.jsoup.Jsoup;
import org.jsoup.helper.Validate;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
import java.io.IOException;
/**
 * Example program to list links from a URL.
 */
public class ListLinks {
    public static void main(String[] args) throws IOException {
        Validate.isTrue(args.length == 1, "usage: supply url to
fetch");
        String url = args[0];
        print("Fetching %s...", url);
        Document doc = Jsoup.connect(url).get();
        Elements links = doc.select("a[href]");
        Elements media = doc.select("[src]");
        Elements imports = doc.select("link[href]");
        print("\nMedia: (%d)", media.size());
        for (Element src : media) {
            if (src.normalName().equals("img"))
                print(" * %s: <%s> %sx%s (%s)",
                        src.tagName(), src.attr("abs:src"),
src.attr("width"), src.attr("height"),
                        trim(src.attr("alt"), 20));
            else
                print(" * %s: <%s>", src.tagName(),
src.attr("abs:src"));
        }
        print("\nImports: (%d)", imports.size());
        for (Element link : imports) {
            print(" * %s <%s> (%s)",
link.tagName(),link.attr("abs:href"), link.attr("rel"));
        }
        print("\nLinks: (%d)", links.size());
        for (Element link : links) {
            print(" * a: <%s>  (%s)", link.attr("abs:href"),
trim(link.text(), 35));
        }
    }
    private static void print(String msg, Object... args) {
        System.out.println(String.format(msg, args));
    }
```

Продовження Лістингу 2.1

```
    private static String trim(String s, int width) {
        if (s.length() > width)
            return s.substring(0, width-1) + ".";
        else
            return s;
    }
}
org/jsoup/examples/ListLinks.java
```

Виведення результату зображено на рисунку 2.5.

```
Fetching http://news.ycombinator.com/...

Media: (38)
 * img: <http://ycombinator.com/images/y18.gif> 18x18 ()
 * img: <http://ycombinator.com/images/s.gif> 10x1 ()
 * img: <http://ycombinator.com/images/grayarrow.gif> x ()
 * img: <http://ycombinator.com/images/s.gif> 0x10 ()
 * script: <http://www.co2stats.com/propres.php?s=1138>
 * img: <http://ycombinator.com/images/s.gif> 15x1 ()
 * img: <http://ycombinator.com/images/hnsearch.png> x ()
 * img: <http://ycombinator.com/images/s.gif> 25x1 ()
 * img: <http://mixpanel.com/site_media/images/mixpanel_partner_logo_borderless.gif> x

Imports: (2)
 * link <http://ycombinator.com/news.css> (stylesheet)
 * link <http://ycombinator.com/favicon.ico> (shortcut icon)

Links: (141)
 * a: <http://ycombinator.com>  ()
 * a: <http://news.ycombinator.com/news>  (Hacker News)
 * a: <http://news.ycombinator.com/newest>  (new)
 * a: <http://news.ycombinator.com/newcomments>  (comments)
```

Рисунок 2.5 – Результати програми

1.4.1 Headless браузери

Цей підхід дозволяє обробляти сторінку в браузері з підтримкою JavaScript, що дозволяє писати свої сценарії для отримання необхідної інформації і навіть використовувати JavaScript бібліотеки на зразок jQuery для отримання інформації зі сторінки, що прискорює розробку парсерів. Відсутність графічного інтерфейсу дозволяє запускати дані браузери навіть

серверах, підтримують лише консольний режим.

До таких інструментів можна віднести PhantomJS,SlimerJS. Розглянемо як приклад SlimerJS [13].

SlimerJS — це браузер із сценаріями. Він дозволяє маніпулювати вебсторінкою за допомогою зовнішнього сценарію Javascript: відкривати вебсторінку, натискати посилання, змінювати вміст... Корисно виконувати функціональні тести, автоматизацію сторінки, моніторинг мережі, захоплення екрана тощо. В наступному коді на лістингу 2.2 було використано CasperJS, який є утилітою для написання сценаріїв для PhantomJS і SlimerJS. За допомогою цього коду буде виведено список останніх опублікованих статей.

Лістинг 2.2 – Приклад коду

```
var casper = require('casper').create();
// Tell CasperJS to visit the homepage
casper.start('http://blog.badacadabra.net');
// Wait for articles to be loaded
casper.waitForSelector('article', function () {
  this.echo('BLOG TITLE:\n' + this.getTitle());
  this.echo('\nLATEST ARTICLES:');
  // Each title in the "titles" array returned by "evaluate"...
  this.each(this.evaluate(function () {
    var articles = document.querySelectorAll('article'),
        titles = [];
    for (var i = 0; i < articles.length; i++) {
      titles.push(articles[i].querySelector('.post-
title').textContent);
    }
    return titles;
  }), function (self, title) {
    // ... should be displayed in console!
    this.echo(title);
  });
});
casper.run();
```

Також на рисунку 2.6 наведені результати роботи програми.

```
BLOG TITLE:
Hocus Pocus Numericus | Blog de Baptiste Vannesson (Badacadabra)

LATEST ARTICLES:
Les robots ont-ils des droits ?
Stress, anxiété et phobies à l'ère du numérique
Le règne de l'autodidaxie
Les limites du transhumanisme
L'ambiguïté du référencement payant
```

Рисунок 2.6 – Результати програми з використанням CasperJS

1.4.2 SaaS рішення

Дані сервіси надають графічний інтерфейс, за допомогою якого можна вказати адресу сторінки, вказати блоки, з яких потрібно отримати інформацію, а також створити ряд правил щодо вилучення даних. Такі послуги не мають тієї гнучкості, яку надають низькорівневі рішення. Деякі з них коштують досить дорого, проте ними просто користуватися. До таких послуг можна віднести Mozenda, Octoparse (безкоштовний), Import.io (безкоштовний).

Octoparse це безкоштовний інструмент призначений для веб-парсингу. Він дозволяє отримувати дані з інтернету без рядка коду і перетворювати веб-сторінки в структуровані дані всього за один клік. Завдяки автоматичної ротації IP-адрес для запобігання блокування та можливості планування подальшого скрапінга цей інструмент є одним з найефективніших [14]. На рисунку 2.7 наведено приклад роботи програми Octoparse.

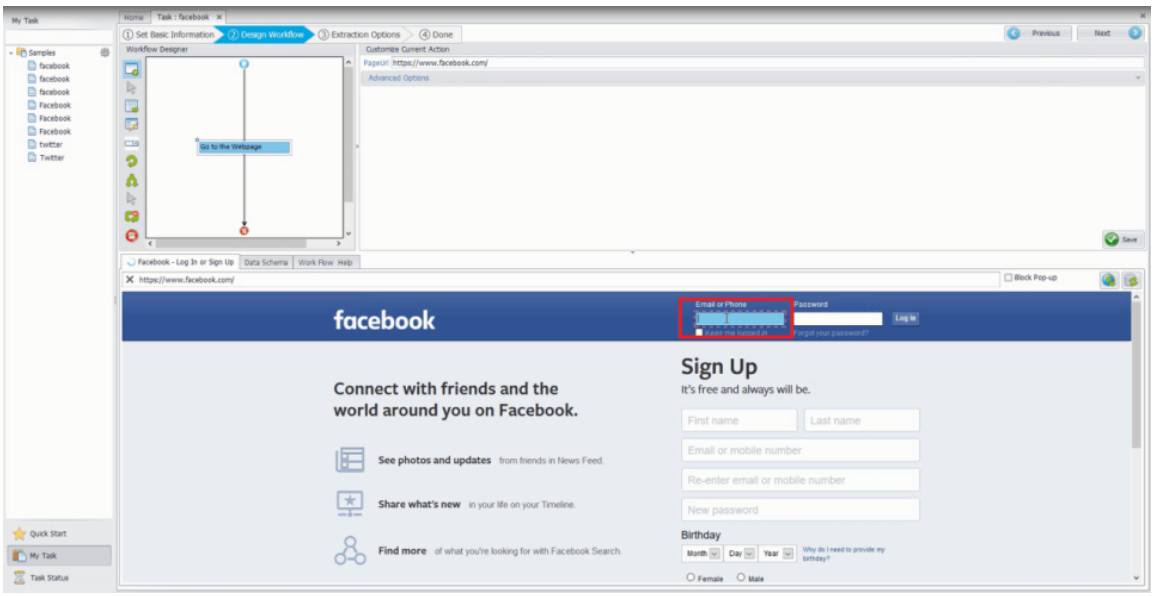

Рисунок 2.7 – Зовнішній вигляд програми Octoparse

Mozenda це корпоративне програмне забезпечення розроблене для всіх видів завдань по вилученню даних. Цій компанії довіряють тисячі підприємств і більше 30% компаній зі списку Global Fortune 500. Це один із кращих інструментів для парсинга веб-сторінок, який допоможе за лічені хвилини 36 створити скрапер агента. Mozenda також пропонує функції Job Sequencer and Request Blocking для збору веб-даних в реальному часі і кращий сервіс для роботи з клієнтами [15]. На рисунку 2.8 наведено приклад роботи програми Mozenda.

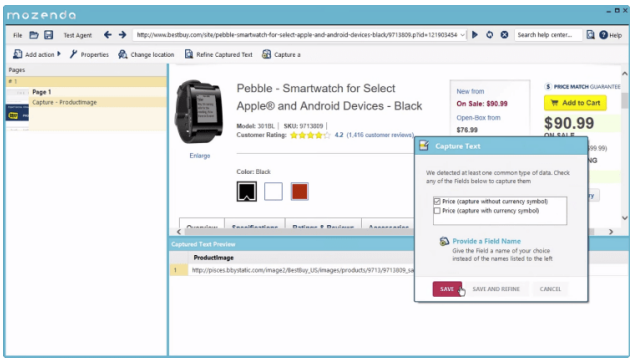

Рисунок 2.8 – Програма Mozenda

Import.oi – SaaS платформа, яка дозволяє перетворювати напів-структуровані веб-дані в структуровані. Платформа надає можливість вилучення даних в реальному часі за допомогою API-інтерфейсів JSON REST і потокової 35 передачі, а також легко інтегрується з багатьма мовами програмування і інструментами для аналізу даних [16]. На рисунку 2.9 наведено приклад програми Import.io.

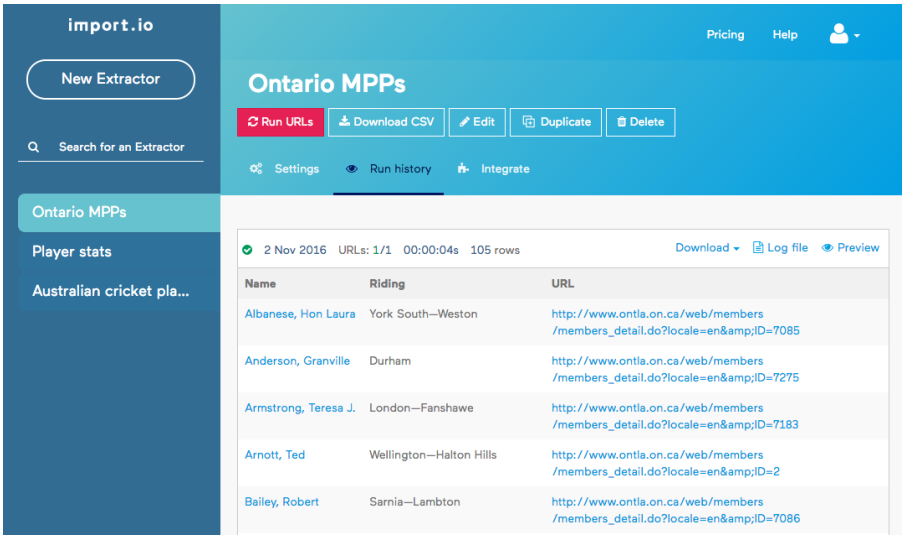

Рисунок 2.9 – Програма Import.io

2.5 Експорт даних

Матеріал, отриманий з розпарсенного сайту, необхідно упакувати у вигляді, придатному для подальшого використання. Конкретний формат залежить від того як в подальшому буде оброблятися зібрана інформація.

Найчастіше це бази даних MySQL або PostgreSQL. Заливати в БД можна не тільки за допомогою запитів SQL, але і за допомогою JSON через Ajax. У багатьох випадках із спарсенного контенту за допомогою XML формується RSS-потік, що вельми зручно при використанні даних, без процедури рерайтинга. Іноді результат парсинга поміщають в CSV-файл - оскільки цей текстовий формат дуже простий у подальшій обробці, легко конвертується в

SQL-запити і без проблем відкривається в Excel. У спеціальних випадках потрібно, щоб кінцеві дані були представлені у вигляді електронних таблиць XLS.

При парсингу кожної сторінки сайту-донора, витягнуті дані, як правило, тут же заносяться в базу даних. Для повної реалізації даної задачі нам не обійтися без впевнених знань про MySQL і PostgreSQL - двох найбільш використовуваних СУБД для веб-розробок.

Зазвичай у вигляді БД спарсенний контент і потрібний для перенесення на інший веб-ресурс. Однак часто наповненням бази даних справа не закінчується. Інформацію з SQL доводиться конвертувати в інші формати: JSON, текстовий CSV (який, в свою чергу, є зручним буфером для подальшої обробки), таблично-процесорний XLS (в PHP є спеціальні додаткові бібліотеки для перетворення SQL-даних в електронні таблиці Excel), XML (часом фінальним акордом парсерной сюїти є веб-серверний RSS-агрегатор).

Ну, і звичайно ж, доводиться іноді перетворювати даних з MySql в PostgreSQL і назад, бо бажана веб-програмістом і необхідна для конкретного завдання СУБД можуть не збігатися.

CSV (англ. Comma-Separated Values - значення через кому) - формат текстових файлів для зберігання табличної інформації. Один рядок - один запис в таблиці. Роздільник значення полів у записі - кома. Втім, замість коми можна використовувати і інші символи.

Формат CSV популярний оскільки дуже простий і універсальний. Його можна легко:

- правити в будь-якому текстовому редакторі;
- відкривати в Microsoft Excel і OpenOffice.org Calc;
- змінювати функціями PHP для роботи з файлами;
- конвертувати в SQL і назад.

При парсингу досить-таки часто у Вас можуть запитати спарсені дані в цьому форматі. Що не викликає ні найменших труднощів і дуже зручно для подальшої обробки інформації. Приклад CSV-файлу на рисунку 2.10.

	A	B	C
1	First Name	Last Name	Email Address
2	Jane	Doe	jane@gmail.com
3	John	Doe	john@gmail.com
4	Chris	Perkins	cperk@gmail.com
5	Eva	Davis	eva@gmail.com
6	Mitchell	Tarver	mitch@gmail.com
7	Nathan	Woodward	nathanwoodward@gmail.com

Рисунок 2.10 – Приклад CSV-файлу

XML-документи (як технологія для RSS) в парсингу використовується в двох протилежних задачах.

RSS-потік - це спарсена інформація з потрібного нам сайту. Якщо канал є джерелом контенту, то завдання програміста зводиться до нескладної фільтрації або об'єднання декількох RSS-feed'ів з різних джерел в один загальний. По суті справи, парсер в даному випадку представляє з себе веб-серверний RSS-агрегатор, настроєний на вирішення конкретного завдання.

З іншого боку, спарсена інформація часто сама є основою для створення RSS-каналу. На основі отриманих даних формується стрічкове зведення,інформуюче про наповнення сторонніх сайтів і періодичних змінах на їхніх сторінках. Приклад RSS-файлу на рисунку 2.11.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<rss version="2.0">
  <channel>
    <title>NYT &gt; World Business</title>
        <item>
          <title>Russia and Ukraine Reach Compromise</title>
          <link>http://www.nytimes.com/2006/01/05/05ukraine.html</link>
          <description>The solution allowed both nations...</description>
          <author>ANDREW E. KRAMER</author>
          <pubDate>Thu, 05 Jan 2006 00:00:00 EDT</pubDate>
        </item>
    </title>
  </channel>
</rss>
```

Рисунок 2.11 – Приклад RSS-файлу

Іноді розпарсенний матеріал потрібно представити у вигляді електронної таблиці Excel. Безумовно, електронні таблиці неможливо редагувати як звичайний текстовий файл. Досить відкрити документ XLS в текстовому редакторі (Notepad ++, Sublime Text, наприклад) і виявити там набір кракозябри, щоб прийти до висновку про марність стандартних функцій для роботи з файлами. Просто дописати в кінець файлу чергові дані (як у випадку з CSV), витягнуті з сайту-донора, не вийде.

Що ж, якщо стандартні функції не підходять, то на зміну їм використовують нестандартні. Існує чимало бібліотек, покликаних полегшити роботу з книгами Excel. Підключивши їх, можна як зчитувати інформацію з готових електронних таблиць, так і засобами мов веб-програмування формувати XLS-документи. Додавати нові записи в таблицю можна безпосередньо відразу після вилучення контенту з сайту-донора. Або ж спарсити матеріал в базу даних SQL, а потім з БД сформувати електронну таблицю.

У міру обходу сторінок сайту-донора часто зібрана інформація тут же за допомогою запитів SQL заноситься в базу даних. Альтернативою є JSON - текстовий формат обміну даними оснований на JavaScript.

У даного методу чимало безперечних переваг перед тим же SQL. JSON відрізняють мовонезалежність, крайня простота і лаконічність синтаксису, наявність готових рішень для обробки даних. Але головна перевага - це можливість прямої взаємодії браузера і сервера, що дозволяє реалізовувати 24 алгоритми пов'язані з фоновим занесенням інформації в базу даних, роботою з файловим кешем, серіалізацією, десеріалізацією динамічних структур та ін.

2.6 Проблеми побудови веб-парсеру

Вивчивши найрізноманітніші рішення, можна сміливо заявити: ідеального парсера не існує, через велику кількість можливих проблем на шляху:

- жоден сайт не має ідеальної верстки з точки зору догматів веб-дизайну, саме це робить кожен сайт унікальним і привабливим для користувачів;

- кожен веб-розробник (якщо він не працює в солідній IT компанії зі своїми правилами і стайл-гайдами) пише код під себе або просто, як вміє. Далеко не завжди код виходить грамотним і якісним. Найчастіше в ньому можна знайти величезну кількість помилок. У тому числі граматичних. Все це робить "самописний" код абсолютно нечитабельним для скраперів (мова, в першу чергу, йде про верстку) [17];

- маса веб-ресурсів використовує HTML5, де кожен елемент може бути абсолютно унікальним;

- деякі ресурси містять різноманітні системи захисту від копіювання даних, а значить і від скрапінга. Це виражається в багаторівневої верстці, використанні JavaScript для рендерингу контенту, перевірки user-agent і т.д.;

- залежно від сезону або тематики цільового матеріалу на сайті можуть бути використані різні макети. Періодично це стосується навіть типових сторінок (сезонні акції, преміум статті і т.д.);

- крім корисних блоків, веб-сторінка часто рясніє "сміттям" у вигляді реклами, коментарів, додаткових елементів навігації і т.д.;

- вихідний код може містити посилання на одні й ті ж картинки різних розмірів, наприклад - для попереднього перегляду;

- сайт може визначити країну, в якій знаходиться ваш сервер і віддати інформацію не необхідною мовою;

- у всіх сайтів може бути різне кодування, що не отримується у відповіді на запит.

Перераховані вище фактори серйозно ускладнюють процес веб-скрапінга. В результаті якість контенту може впасти до 20% і навіть до 10%, що абсолютно неприйнятно. Не забуваємо – повнота інформації є її найважливішим критерієм.

Оскільки проблем справді багато, в даній ситуації можна зробити

висновок, наступним чином:

- при необхідності отримувати дані з невеликої кількості джерел, краще написати свій скрапер і налаштувати його під потрібні сайти (якість одержуваного контенту - близько 100%);
- Використовувати комплексний підхід у виборі рідера (парсера), якщо потрібно отримувати інформацію з великої кількості джерел (до 95% якості).

2.7 Вимоги для програми парсеру

Нижче представлено список вимог для програми парсеру:

- програма повинна вміти вилучати дані з різних типів джерел xml, так і розташованих у мережі: сторінок сайтів, стрічок новин, результатів API запитів і т.д.;
- програма повинна вміти отримувати дані із файлів великого розміру;
- програма повинна мати зручні інструменти для аналізу інтернет-джерел. Ці інструменти повинні надавати інформацію як у графічному, так і консольному варіантах;
- налаштування програми для вилучення даних має бути максимально простим і не вимагати від користувача поглиблених знань структури XML формату. Тобто для того, щоб витягти дані, Вам не потрібно, наприклад, знати, що таке xpath і як його використовувати;
- програма повинна працювати на комп'ютерах з різними операційними системами (windiws,macos), бути невибагливою до ресурсів системи (працювати навіть на малопотужних комп'ютерах) та підтримувати як локальне так і віддалене управління [18];
- програма повинна надавати інструменти для збереження вилучених даних у затребувані формати.

2.8 Машинне навчання

Машинне навчання - це програма , яка дає системі можливість вивчати речі без явного програмування. Машинне навчання працює з даними, і воно вчитиметься на деяких даних. Машинне навчання дуже відрізняється від традиційного підходу. У машинному навчанні ми подаємо дані і машина генерує алгоритм. Машинне навчання має три типи навчання [22]:

- контрольоване навчання;
- неконтрольоване навчання;
- навчання з підкріпленням.

Уявіть себе учнем, який сидить у класі, де ваш учитель спостерігає за вами, «як ви можете вирішити проблему» або «правильно ви робите чи ні». Так само в контрольованому навчанні введення надається як позначеного набору даних, модель може вчитися у ньому, щоб легко надати результат проблеми.У цьому проекті виявлення фейкових новин ми використовуємо контрольоване навчання.

Неконтрольоване навчання. Цей алгоритм навчання повністю протилежний до навчання з учителем. У неконтрольованому навчанні немає повного та чистого розміченого набору даних. Неконтрольоване навчання – це самоорганізоване навчання. Його основна мета – вивчити основні закономірності та передбачити результат. Тут ми в основному надаємо машині дані та просимо шукати приховані функції та групувати дані таким чином, щоб це мало сенс.

Начання з підкріпленням. Він не заснований ні на навчанні з учителем, ні на навчанні без учителя. Понад те, тут алгоритми вчаться самостійно реагувати на довкілля. Він швидко зростає і, крім того, виготовляє безліч алгоритмів навчання. Ці алгоритми корисні у сфері робототехніки, ігор тощо.

Для агента навчання завжди є початковий стан та кінцевий стан. Проте досягнення кінцевого стану може бути інший шлях. У задачі навчання з підкріпленням агент намагається маніпулювати довкіллям. Агент

переміщається з одного стану до іншого. Агент отримує винагороду (оцінку) у разі успіху, але не отримує жодної винагороди чи оцінки у разі невдачі. Таким чином, агент навчається у навколишнього середовища.

В данному проекті для виявлення фейкових новин буде використовуватись технологія векторів,яка буде описана далі.

Машини опорних векторів, скорочено SVM, навчаються за допомогою навчання з наглядом, тобто дані, на яких навчається модель, повинні бути позначені.

SVM у N-вимірному контексті класифікує дані шляхом розбиття N-вимірного простору гіперплощиною, щоб якомога більше точок класифікувалися правильно. Якщо дані лінійно розділені, тобто можна розділити точки даних площиною та правильно їх класифікувати, це прагне максимізувати запас, поле — це простір між площиною та вузлами, найближчими до площини.

Як показано на рисунку 2.12, є два способи розділяйте дані, однак один має більший запас. Більше поле дозволяє краще класифікувати викиди, що видно з класифікації квадратів за різними лініями.

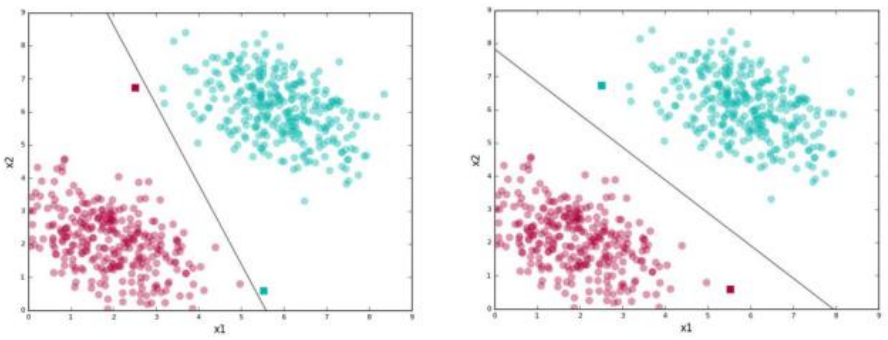

Рисунок 2.12 – Приклад оптимізації маржі

Дані не завжди так легко відокремити, як дані на рисунку 2.12, і вимога до 100% точності даних навчання може призвести до поганої продуктивності тестових даних. Використання того, що називається м’яким маржем, щоб

забезпечити певну кількість помилок у класифікації, водночас отримання більшого запасу може підвищити точність навчальних даних, це може бути особливо корисно, коли набори даних мають викиди, які перекриваються. На рисунку 2.13 параметр C можна використовувати для контролю дозволеної класифікації помилок за ціною/прибутком маржі. Приклади того, як параметр C впливає на продуктивність, показані на рисунку 2.13. Більш широкий запас, ймовірно, буде працювати краще у випадках, коли переобладнання є проблемою, і навпаки.

Якщо дані не є лінійно розділеними в їх поточних розмірах, можна використовувати функцію ядра. Метою функцій ядра є взяти дві точки в поточному просторі та обчислити крапковий добуток на спроектований простір. Якщо до даних застосувати правильну функцію ядра, вони можуть стати лінійно розділеними зображено на рисунку 2.14.

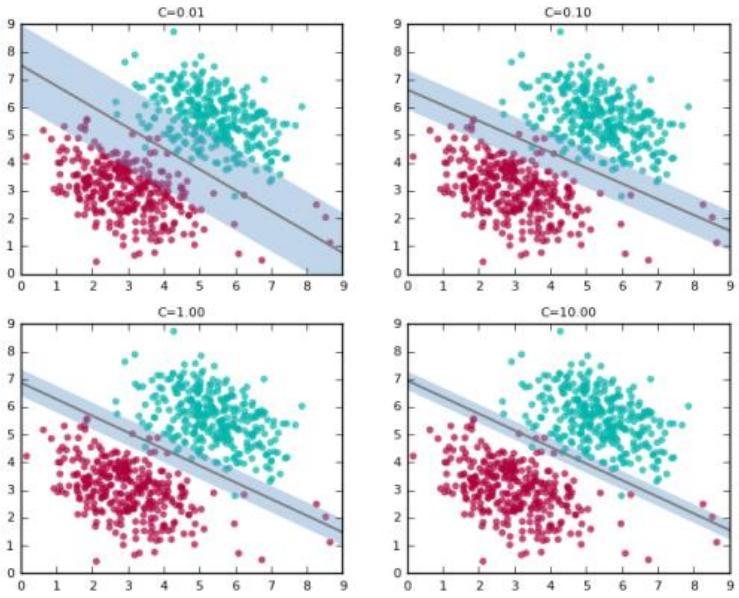

Рисунок 2.13 – Приклад компромісу між великим запасом і високою точністю навчальних даних

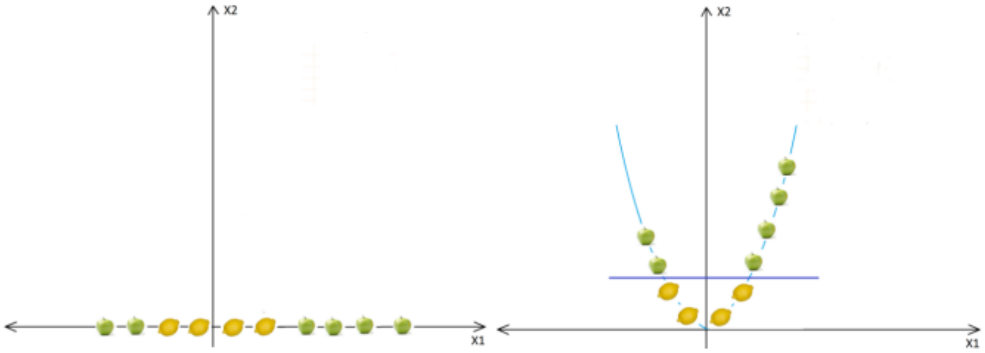

Рисунок 2.14 – Відображення нелінійно розділеного набору даних, ліворуч, перетвореного за допомогою функції ядра: f(x) = x2 до лінійно роздільного набору даних, праворуч

Спосіб обчислення координат точок у новому вимірі залежить від функції ядра. Коли проекція не використовується, а крапкові добутки обчислюються в вихідному просторі, це називається лінійним ядром. Лінійне ядро є вигідним, коли дані мають велику кількість функцій, це пов’язано з тим, що малоймовірно, що в просторі ознак буде багато накладок, отже, цілком імовірно, що дані лінійно розділені. Це також набагато ефективніше обчислювально. Було виявлено, що машини лінійних опорних векторів добре працюють для класифікації тексту [19].

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Основні використовувані інструменти та структура проетку

В ході розробки програми було використано наступні інструменти:

- середовище розробки Visual Studio Code, Colab;
- мова програмування Python;
- мова розмітки HTML;
- структура данних JSON,CSV,TXT;
- бібліотеки Pandas, Beautifulsoup4, Selenium [20];
- машинне навчання;

Файлова структура проекту наведена на рисунку 3.1 та в неї входять наступні файли:

- файл Init.py використовується для запуску програми;
- файл Articles.py відповідає за парсинг та вилучення інформації з сайту;
- файл Helpers.py містить в собі ряд хелперів або допоміжних функцій;
- файл Sources.py відповідає за коректність та правильність URL;
- файл Utils.py відповідає за підключення різних бібліотек та окремі функції.

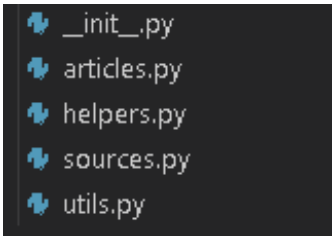

Рисунок 3.1 – Структура проекту

3.2 Запуск та опис проекту

Для того щоб запустити проект достатньо імпортувати мою бібліотеку та запустити наступний код, який міститься в Init.py :

Лістинг 3.1– Приклад коду для запуску програми

```
import mylib
articles = mylib.source("https://www.bbc.com/")
for article in articles[:10]:
    news = mylib.read(article["url"], lang=True)
    print(news)
```

В данному лістингу продемонтровано початок роботи програми спочатку запускається функція source, яка відповідає за аналіз та перевірку коректності URL адреси. Потім йде цикл в якому ми можемо вказати скільки статей ми будемо парсити, в даному випадку 10, але є можливість вказати всі доступні статті на сайті. Далі функція Read зчитує всі статті з сайту та поміщає їх в news, змінна lang відповідає за визначення мови статті. На рисунку 3.2 зображено результат роботи програми.

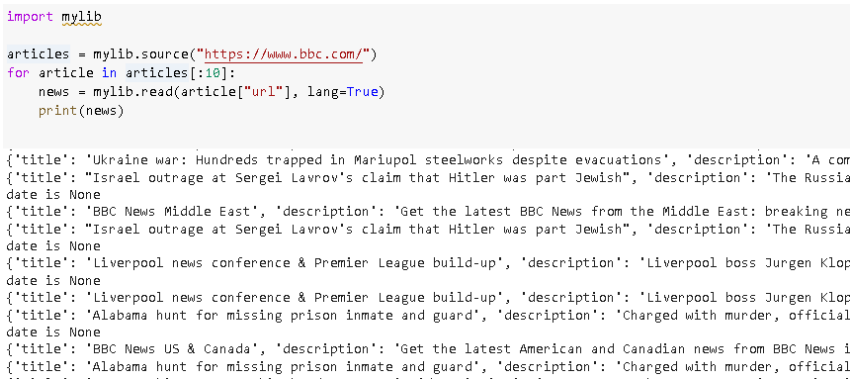

Рисунок 3.2 – Результати програми

На рисунку 3.2 зображено консольне виведення статей, проте в програмі ще доступні такі формати як CSV,JSON,TXT. Це зроблено для експорту даних

в інші програми рисунок 3.3.

Рисунок 3.3 – Дані в форматі txt

Файл Utils.py, як вже говорилось раніше відповідає за підключення бібліотепк та інших корисних функцій, наприлад функція отримання мови статті яку розглянуто на лістингу 3.2.

Лістинг 3.2– Функція отримання мови статті

```
def get_lang(title, desc, content):
    from langdetect import detect
    from langdetect.lang_detect_exception import LangDetectExcep
tion
    try:
        if desc is not None:
            return detect(title)
        if title is not None:
            return detect(title)
        if content is not None:
            return detect(content)
    except LangDetectException:
        return None
```

На лістингу 3.2 зображено метод в якому аналізується загаловок та тіло статті та визначається мова на якій написана статя. В цьому методі використовується бібліотека langdetect, яка підтрумує понад 55 мов по всьому світу, в тому числі й українську.

Також на лістингу 3.3 зображена ще одна корисна функція, а саме

функція отримання стоп-адрес.

Лістинг 3.3– Функція отримання мови статті

```
def get_stop_urls():
    stop_urls_filename = os.path.join(
        directory, "ressources/stop_urls.txt")
    with open(stop_urls_filename) as f:
        return set(f.read().split())
```

Ця функція відповідає за отримання слів зі спеціального файлу stop_urls.txt, який містить в собі спеціальні адреси, які не потрібно враховувати при парсингу оскільки вони не мають якоїсь користі для нас. Наприклад вкладка логіну чи конфіденційних умов є на кожному сайті але вона нам не потрбіна. На рисунку 3.4 зображено цей файл.

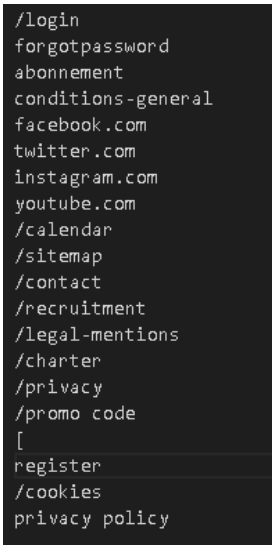

Рисунок 3.4 – Файл зі стоп-адресами

Наступна функція, яка буде зображена на лістингу 3.4 призначена для валідації URL адреси.Бувають випадки, коли користувач набрав некоректне посилання випадково чи навмисно. Ця функція допоможе відредагувати адресу та перевірити її працездатність

Лістинг 3.4 – Функція валідації

```
def get_source(url):
    source = re.match(
        r'^(?:https?:\/\/)?(?:[^@\/\n]+@)?(?:www\.)?([^:\/?\n]+)
', url)
    if source is not None:
        return source.group(0).strip()
    return None
```

Спочатку за допомогою Regex перевіряється посилання на наявність протоколу https або чи містить воно www, потім зайва частина відрізається для подальшого парсингу.

Файл Helpers.py містить ряд допоміжних функцій розглянемо деякі з них.Наприклад функція, продемонтсрована в лістингу 3.5 покликана прибрати UTM параметри з нашої ссилки.

Лістинг 3.5 – Функція видалення UTM параметрів

```
def remove_campaign_from_url(url):
    parsed = urlparse(url)
    qd = parse_qs(parsed.query, keep_blank_values=True)
    filtered = dict((k, v) for k, v in qd.items() if not
k.startswith('utm_'))
    newurl = urlunparse([
        parsed.scheme,
        parsed.netloc,
parsed.path,
        parsed.params,
        urlencode(filtered, doseq=True),
        parsed.fragment
    ])
    return newurl
```

UTM параметри це параметри, які додаються до URL-адреси для отримання детальної інформації про трафік. Вони нам не потрібні тому ми позбуваємося за допомогою словника в якому міститься наша адреса і перевіряється кожен її елемент, якщо він починається з utm то він видаляється та повертається модифікована адреса без utm [21].

На лістингу 3.6 продемонтсровано додаткові перевірки адреси на якоря, стоп-адреси та перевірки коректності структури адреси.

Лістинг 3.6 – Функція видалення UTM параметрів

```
def simple_filter(source, url, stop_urls):
    if len(url) == 0:
        return False
    if url[0] == "#":
        # if anchor, remove url
        return False
    if url == "/":
        # if home, return url
        return False
    if any((x in url for x in stop_urls)):
        # if url in stop_urls
        return False
    if len(source) + 16 >= len(url):
        return False
    return True
def correct_url(source, protocol, url):
    url = url.strip()
    if url[:2] == "//":
        return protocol + url
    elif url[:1] == "/":
        return source + url
    elif "http" not in url[:5]:
        return source + "/" + url
    return url
```

Як вже згадувалось раніше файл Source.py відповідає за коректність та правильність URL.Проте основні методи валдації були розглянуті в попередніх файлах одже настав час застосувати їх в (лістингу 3.7).

Лістинг 3.7 – Методи файлу Source.py

```
def get_urls(root, url):
    stop_urls = get_stop_urls()
    source = get_source(url)
    protocol = "https:" if "https:" in url else "http:"
    urls = root.xpath("//a/@href")
    urls = [correct_url(source, protocol, x) for x in urls]
sources = [get_source(x) for x in urls]
    filters = [simple_filter(source, x, stop_urls) for x in
```

Продовження Лістингу 3.7

```
urls]
    data = []
    for i in range(len(urls)):
        if filters[i]:
            data.append({
                "url": urls[i],
                "source": sources[i]
            })
    return data

def build(url):
    html, root, url = request(url)
    return get_urls(root, url)
```

В даному міститься метод get_urls який отримує шлях та url сайту, та перевіряє його на стоп-адресу, на коректність, на наявність протоколу.Потім ми отримуємо всі гкд з сайту, фільтруємо їх та перевіряємо,якщо все ок то додаємо до нашого масиву.

Як вже говорилось раніше файл Articles.py відповідає за парсинг та вилучення інформації. Метод read який міститься в цьому файлі відовідає за початок процесу парсингу.На (лістингу 3.8) зображено цей метод.

Лістинг 3.8 – Методи файлу Source.py

```
def read(url, lang=True):
    url = remove_campaign_from_url(url)
    try:
        html, root, url = request(url)

        title = get_title(html, root)
        desc = get_description(root)
        created_time = get_created_time(html, root)
        img = get_image(root)
        content = get_content(root)
        author = get_author(root)
        if lang:
            detected_lang = get_lang(title, desc, content)
        else:
            detected_lang = None
```

Продовження Лістингу 3.8

```
        return {
            "title": title,
            "description": desc,
            "created_time": created_time,
            "img": img,
            "content": content,
            "lang": detected_lang,
            "author": author,
            "url": url
        }
    except ParseError:
        return None
    except ConnectionError:
        return None
```

Метод read приймає url та надсилає запит на отримання html файлу, як тільки ми його отримали, вилучаємо фнормацію, яка нам потрібна це автори, заголовки, час коли була створена статья, зображення якщо вони є, та встановлюємо мову статті, за допомогою метода get_lang про який ми згадували раніше.

Далі за допомогою бібліотеки Selenium ми будемо парсити html сторінку.На (лістингу 3.9) зображено методи get_title та get_description, де за допомогою xpath локатора ми вказуємо, елементи в який зберігаються заголовки та описи статей.

Лістинг 3.9 – Методи get_title та get_description

```
def get_title(html, root):
    t = root.xpath("//h1")
    if len(t) == 0:
        t = root.xpath("//title")
    if len(t) == 0:
        raise ParseError("title not found")
    return to_str(t[0])
@clean_str
def get_description(root):
    t = root.xpath("//p[@class='article__desc']")
    if len(t) > 0:
```

Продовження Лістингу 3.9

```
return to_str(t[0])

    t = root.xpath("//meta[@name='description']")
    if len(t) > 0:
        return t[0].attrib.get("content", None)

    t = root.xpath("//div[contains(@class, 'gr-article-
teaser')]")
    if len(t) > 0:
        return to_str(t[0])

    t = root.xpath("//meta[@property='og:description']")
    if len(t) > 0:
        return t[0].attrib.get("content", None)

    t = root.xpath("//meta[@property='twitter:description']")
    if len(t) > 0:
        return t[0].attrib.get("content", None)

    return None
```

Аналогічно методам get_title та get_description отримуємо авторів та картинки зображень витягуючи їх або з мета даних або з атрибутів автора та картинки.На лістингу 3.10 зображено методи get_author та get_image.

Лістинг 3.10 – Методи get_title та get_description

```
def get_author(root):
    t = root.xpath("//meta[@name='author']")
    if len(t) > 0:
        return t[0].attrib.get("content", None)
    t = root.xpath("//meta[@itemprop='author']")
    if len(t) > 0:
        return t[0].attrib.get("content", None)
    t = root.xpath("//meta[@property='author']")
    if len(t) > 0:
        return t[0].attrib.get("content", None)
def get_image(root):
    t = root.xpath("//meta[@property='og:image']")
    if len(t) > 0:
        t = t[0].attrib.get("content", None)
        return t
```

Продовження Лістингу 3.10

```
t = root.xpath("//meta[@property='twitter:image']")
    if len(t) > 0:
        t = t[0].attrib.get("content", None)
        return t

    t = root.xpath("//meta[@property='facebook:image']")
    if len(t) > 0:
        t = t[0].attrib.get("content", None)
        return t
```

Також перевіряємо мета данні та атрибути часу,які можуть бути і різних тегах, які перераховані в функції get_created_time. До речі за допомогою бібліотеки dataparser, ми також можемо отримати не тільки час публікації статі, але й також час модифікації. Метод get_created_time зображений на лістингу 3.11.

Лістинг 3.11 – Метод get_ created_time

```
def get_created_time(html, root):
    t = root.xpath("//meta[@itemprop='dateModified']")
    if len(t) > 0:
        created_time =
dateparser.parse(t[0].attrib.get("content", ""))
        return created_time

    t = root.xpath("//meta[@itemprop='datePublished']")
    if len(t) > 0:
        created_time =
dateparser.parse(t[0].attrib.get("content", ""))
        return created_time
    t =
root.xpath("//meta[@property='og:article:modified_time']")
    if len(t) > 0:

        created_time =
dateparser.parse(t[0].attrib.get("content", ""))
        return created_time

    t = root.xpath("//meta[@property='article:modified_time']")
    if len(t) > 0:
        created_time =
dateparser.parse(t[0].attrib.get("content", ""))
        return created_time
```

Продовження Лістингу 3.11

```
 t =
root.xpath("//meta[@property='twitter:article:modified_time']")
    if len(t) > 0:
        created_time =
dateparser.parse(t[0].attrib.get("content", ""))
        return created_time

    t =
root.xpath("//meta[@property='og:article:published_time']")
    if len(t) > 0:
        created_time =
dateparser.parse(t[0].attrib.get("content", ""))
        return created_time

    t = root.xpath("//meta[@property='article:published_time']")
    if len(t) > 0:
        created_time =
dateparser.parse(t[0].attrib.get("content", ""))
        return created_time

    t =
root.xpath("//meta[@property='twitter:article:published_time']")
    if len(t) > 0:
        created_time =
dateparser.parse(t[0].attrib.get("content", ""))
        return created_time

    t = re.findall(r'dateModified\":[\s]+\"(.+?)\"', html)
    if len(t) > 0:
        created_time = dateparser.parse(t[0])
        return created_time

    t = re.findall(r'datePublished\":[\s]+\"(.+?)\"', html)
    if len(t) > 0:
        created_time = dateparser.parse(t[0])
        return created_time
 t = root.xpath("//*[@class='date']")
    if len(t) > 0:
        created_time = dateparser.parse(to_str(t[0]))
        return created_time

    t = root.xpath("//time")
    if len(t) > 0:
        created_time = dateparser.parse(t[-
1].attrib.get("datetime", ""))
        return created_time

    t = root.xpath("//*[contains(@class, 'date')]")
    if len(t) > 0:
        created_time = dateparser.parse(to_str(t[0]))
```

Продовження Лістингу 3.11

```
return created_time

    t = re.findall(r"updat=([^\s]+)", html)
    if len(t) > 0:
        created_time = dateparser.parse(t[0])
        return created_time

    t = re.findall(r"publicationData\":\"([^\"]+)", html)
    if len(t) > 0:
        created_time = dateparser.parse(t[0])
        return created_time
```

Далі за допомогою метода get_content ми витягуємо тіло статті, та анлазіуємо батьківські та дочірні елементи в яких зберігається потрібна нам інформація. Метод get_content зображено на лістингу 3.12.

Лістинг 3.12 – Метод get_content

```
def get_content(root):
    t = root.xpath("//*/*[self::p or self::h2 or
self::div[@class='article__body']]")
    paragraphs = defaultdict(list)
    length = defaultdict(int)
    max_length = 0
    id_max = None
    for item in t:
        parent = item.getparent()
        id_parent = get_element_id(parent)
        paragraphs[id_parent].append(item)
content_strip = re.sub(r"[\s\n]", "", to_str(item))
        length[id_parent] += len(content_strip)
        if "article" in id_parent:
            length[id_parent] += len(content_strip)

        if length[id_parent] > max_length:
            max_length = length[id_parent]
            id_max = id_parent

    if id_max is not None:
        return "\n\n".join([to_str(y) for y in
paragraphs[id_max]])
    return None
```

Ми познайомилися з соновноми функціями та їх реалізацією, проте в програмі є ще інші методи, результати яких буде показано далі.

3.3 Аналіз отриманих результатів

За результатами програми ми отримали парсер, який здатний аналізувати веб-сайти новин та вилучати з них необхідну інфоормацію. На рисунку 3.5 зображена інформація, яка була отримана при парсингу на прикаді однієї статті в форматі json.


```
"headline": "Facebook is spending $5.7 billion to capitalize on internet boom",
"author": ["Sherisse Pham", "Cnn Business"],
"date_publish": "2022-04-22 04:33:39",
"date_modify": "None",
"image_url": "None",
"filename": "https://www.cnn.com/2020/04/22/tech/facebook-reliance-jio/index.html.json",
"description": "Facebook is investing $5.7 billion into Jio Platforms, the digital techno
"publication": "CNN",
"category": "tech",
"source_domain": "www.cnn.com",
"article": "Hong Kong (CNN Business) Facebook (FB) is spending billions of dollars for a
"summary": "Hong Kong (CNN Business) Facebook (FB) is spending billions of dollars for a
"keyword": [
  "mobile","billion", "platforms","57", "partnership", "boom","indias", "stake", "users",
],
"title_page": "None",
"title_rss": "None",
"url": "https://www.cnn.com/2020/04/22/tech/facebook-reliance-jio/index.html"
```


Рисунок 3.5 – Прикад статті в форматі json

Програма видає різноманітну кількість інформації про статью, а саме 16 категорій в яких є все від заголовка до url шляху в той час як в інших парсерах інформації менше, що наведено на рисунку 3.6.


```
"data":[
{
"Headline": "Facebook is spending $5.7 billion to capitalize on internet boom",
"Author": [ "Cnn Business"],
"Datepublish": "2022-04-22 04:33:39",
"Datemodify": "None",
"Imageurl": "None",
"Content": "Hong Kong (CNN Business) Facebook (FB) is spending billions of dollars for

"Url": "https://www.cnn.com/2020/04/22/tech/facebook-reliance-jio/index.html"
}
]
```


Рисунок 3.6 – Прикад однієї статті в форматі json в іншій програмі

Також як і в решти програм є зручний експорт даних для їх подальшого використання в форматах:

- txt;
- json;
- csv.

Щодо швидкості роботи програми були проведені досліди та порівняння з іншими програмами і результати приємно здивовують. Вони наведені на рисунку 3.7.

Рисунок 3.7 – Порівняння швидкості роботи програм

Проаналізувавши гістаграму виявлено, що швидкіст роботи нашої програми перевищує інших в півтори рази.

Також було проведено експеремент щодо виявлення статей в порівнянні з іншими програмамию. Результати будуть зображені на рисунку 3.8, де видно, що точність виявлення статі сягнув 95 відстоків, хоча це не багато у зрівнянні з іншими 90 і 87 відстоків, проте все ж таки невелика перевага є.

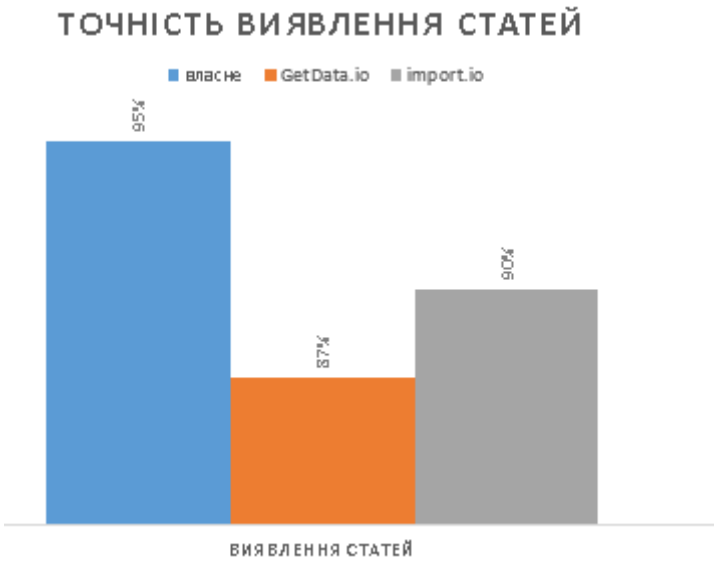

Рисунок 3.8 – Точність виявлення статей

Отже за рахунок власного алгоритму, який був розроблений в ході написання програми вдалося перевершити існуючі засоби парсингу, в деяких аспектах значно краще, в інших з невеликим покращенням, але власну програму можна вважати успішною.

4 МАШИННЕ НАВЧАННЯ

4.1 Фейкові новини

Простий зміст фейкових новин полягає в тому, щоб включати інформацію, яка веде людей хибним шляхом. В даний час фейкові новини розповсюджуються як вода, і люди діляться цією інформацією, не перевіряючи її. Це часто робиться для просування чи нав'язування певних ідей та часто досягається за допомогою політичних програм.

Засобам масової інформації можливість залучати глядачів на свої веб-сайти потрібна для отримання доходів від онлайн-реклами. Тому необхідно виявляти фейкові новини.

4.2 Розробка програми виявлення феквоих новин

Дані машинного навчання працюють лише з числовими функціями, тому нам потрібно перетворити текстові дані на числові стовпці. Отже, ми маємо попередньо обробити текст, і це називається обробкою природної мови.

У попередній обробці тексту ми очищаємо наш текст шляхом обробки парою, лемматизації, видаляємо стоп-слова, видаляємо спеціальні символи та числа і т. д. Після очищення даних ми повинні передати ці текстові дані у векторизатор, який перетворює ці текстові дані на числові функції.

Перш за все нам потрібні набори даних для навчання нашої програми. Ви можете знайти безліч наборів даних для виявлення підроблених новин на Kaggle або на багатьох інших сайтах. Я завантажую ці набори даних із Kaggle. Існує два набори даних: один для фейкових новин, а інший для справжніх новин. У справжніх новинах 21417 новин, а у фейкових новинах 23481 новина. Обидва набори даних мають стовпець міток, у якому 1 відповідає фальшивим новинам, а 0 - правдивим новинам. Ми об'єднали обидва набори даних за

допомогою вбудованої функції pandas. На рисунку 4.1 зображено по 5 перших новин з фейкових та правдивих наборів.

```
True_news.head()
```

	title	text	subject	date	label
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017	0
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017	0
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017	0
3	FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	politicsNews	December 30, 2017	0
4	Trump wants Postal Service to charge 'much mor...	SEATTLE/WASHINGTON (Reuters) - President Donal...	politicsNews	December 29, 2017	0

```
Fake_news.head()
```

	title	text	subject	date	label
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017	1
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017	1
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017	1
3	Trump Is So Obsessed He Even Has Obama's Name...	On Christmas day, Donald Trump announced that ...	News	December 29, 2017	1
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017	1

Рисунок 4.1– Набір правдивих та фейкових новин

У цьому наборі даних немає пропущених значень, інакше нам доведеться видалити цю інформацію або нам доведеться поставити якесь значення. Наш остаточний набір даних є збалансованим, тому що обидві категорії мають приблизно однакове число прикладів.

Наступний етап – очищення даних. Ми не можемо використовувати текстові дані безпосередньо, тому що в них є непридатні слова, спеціальні символи та багато іншого. Якби ми використовували його безпосередньо без очищення, алгоритм машинного навчання було б дуже складно виявити закономірності в цьому тексті, а іноді він також видавав би помилку. Отже, ми повинні завжди спочатку очищати текстові дані. У цьому проекті ми робимо одну функцію cleaning_data, яка очищає дані, яка зображена на рисунку 4.2.

```
def cleaning_data(row):

    # convert text to into lower case
    row = row.lower()

    # this line of code only take words from text and remove number and special character using RegX
    row = re.sub('[^a-zA-Z]' , ' ' , row)

    # split the data and make token.
    token = row.split()

    # lemmatize the word and remove stop words like a, an , the , is ,are ...
    news = [ps.lemmatize(word) for word in token if not word in stopwords]

    # finaly join all the token with space
    cleanned_news = ' '.join(news)

    # return cleanned data
    return cleanned_news
```

Рисунок 4.2 – Функція cleaning_data

В цій функції ми приводимо весь текст до нижнього регістру, потім видаляємо числа та спеціальні символи використовуючи Regx. Розділяємо дані та створюємо токен. Далі використовуємо Лематизацію - це процес перетворення слова або токена в його базову форму. Під час стемінгу слова "швидко", "швидкий", "швидкі" будуть перетворені до форми "швидк". А слова "бігом", "бігаю", "бігати" взагалі до кореня слова "біг". Далі ми видаляємо стоп-слова - це слова, які зустрічаються дуже часто і не вважаються інформативними.

Поділ даних – найважливіший крок у машинному навчанні. Ми навчаємо нашу модель на поїзді та перевіряємо наші дані на тестовому наборі. Ми поділяємо наші дані в train та тестуємо, використовуючи функцію train_test_split із навчання Scikit, яка зображена на рисунку 4.3.

```
from sklearn.model_selection import train_test_split
train_data , test_data , train_label , test_label = train_test_split(X , y , test_size = 0.2 ,random_state = 0)
```

Рисунок 4.3– Функція cleaning_data

Ми розділили наші 80% даних для навчального набору та 20% даних для тестового набору, що залишилися.

Наступний етап створення вектору Tfidf.

Tfidf-Vectorizer: (Частота терміну * Зворотна частота документа).

Частота термінів: кількість разів, коли слово з'являється у документі, поділену на загальну кількість слів у документі. Кожен документ має власну частоту термінів. Формула частоти термінів:

$$tf_{i,j} = \frac{n_{i,j}}{\sum k n_{i,j}} \tag{4.1}$$

Зворотна частота документів: логарифм кількості документів, поділений на кількість документів, що містять слово w . Зворотна частота даних визначає вагу рідкісних слів у всіх документах корпусу:

$$idf(w) = \log(\frac{N}{df_t}) . \tag{4.2}$$

Нарешті, векторизатор Tfidf:

$$w_{i,j} = tf_{i,j} \log(\frac{N}{df_t}) . \tag{4.3}$$

Тепер ми створимо об'єкт TfidfVectorizer [23] із деякими аргументами, які зображено на рисунку 4.4.

```
vectorizer = TfidfVectorizer(max_features = 50000 , lowercase=False , ngram_range=(1,2))
```

Рисунок 4.4 – Об'єкт TfidfVectorizer

CountVectorizer вибере слова/функції, які найчастіше зустрічаються, щоб бути в його словнику, і відкине все інше.

ngram_range : n-грама – це просто рядок із n слів поспіль. Наприклад,

пропозиція «Я є Грут» містить 2 грами «Я є» та «є Грут». Пропозиція сама по собі є 3-грамовою. Встановіть параметр ngram_range=(a,b), де a це мінімальний розмір, a b — максимальний розмір ngram, який ви хочете включити у свої функції. За замовчуванням діапазон ngram_range дорівнює (1,1).

У цьому проекті, я виявив, що включення 2-грамів як ознак значно підвищило передбачувану силу моєї моделі. Це інтуїтивно зрозуміло; Багато назв посад, такі як «вчений за даними», «інженер даних» та «аналітик даних», складаються з двох слів.

Тепер ми поміщаємо цей векторизатор на наш набір даних для навчання і перетворіть його значення на набір даних для навчання та тестування стосовно векторизатора. Код зображений на рисунку 4.5.

```
vec_train_data = vectorizer.fit_transform(train_data)

vec_train_data = vec_train_data.toarray()

vec_test_data = vectorizer.transform(test_data).toarray()
```

Рисунок 4.5 – Код перетворення вектора

Після векторизації даних він поверне розріджену матрицю, тому алгоритмів машинного навчання ми повинні перетворити їх у масиви. toarray функція зробить цю роботу за нас.

Наступний етап - Баєсовський метод використовується для класифікації, коли знаходиться у дискретній формі. Це дуже корисно для обробки тексту. Кожен текст буде перетворено на вектор кількості слів. Він може працювати з негативними числами.

Він визначений у бібліотеці Scikit Learn. Таким чином, ми можемо

імпортувати цей клас до нашого проекту, а потім створити об'єкт поліноміального байєсівського класу, що зображено на рисунку 4.6.

1 Зіставте класифікатор з нашими векторизованими даними про поїзди.

2 Коли класифікатор успішно підходить до навчального набору, ми можемо використовувати метод прогнозування, щоб передбачити результат тестового набору.

```
from sklearn.naive_bayes import MultinomialNB
```

```
clf = MultinomialNB()
```

```
clf.fit(training_data, train_label)
y_pred  = clf.predict(testing_data)
```

Рисунок 4.6 – Об'єкт поліноміального баєсівського класу

Щоб перевірити, наскільки добре наша модель, ми використовуємо деякі показники для визначення точності нашої моделі. У Scikit Learn є безліч типів метрик класифікації:

- матриця плутанини;
- оцінка точності;
- точність;
- відгук;
- F1-рахунок.

Матриця плутанини: в основному це показники, скільки результатів передбачено правильно і скільки результатів передбачено неправильно

Оцінка точності: це кількість правильних прогнозів, порівняно із загальною кількістю пророцтв, всі деталі зображені на рисунку 4.7.

```
print(classification_report(test_label , y_pred))
```

	precision	recall	f1-score	support
0	0.96	0.94	0.95	3326
1	0.95	0.96	0.96	3674
accuracy			0.95	7000
macro avg	0.95	0.95	0.95	7000
weighted avg	0.95	0.95	0.95	7000

Рисунок 4.7 – Метрика класифікації

Як бачите, у нас дуже хороші показники Precision, Recall та F1 Score. Таким чином, ми можемо сказати, що наша модель добре працює з невидимими даними. Оцінка точності у тестовому наборі даних становить 95%, що дуже добре.

Тепер ми бачимо звіт про класифікацію на тренувальному наборі, який зображений на рисунку 4.8.

```
y_pred_train = clf.predict(training_data)
print(classification_report(train_label , y_pred_train))
```

	precision	recall	f1-score	support
0	0.96	0.95	0.96	13385
1	0.96	0.96	0.96	14615
accuracy			0.96	28000
macro avg	0.96	0.96	0.96	28000
weighted avg	0.96	0.96	0.96	28000

Рисунок 4.8 – Результати роботи моделі

Ми також отримуємо дуже добрий показник точності на тренувальному наборі. Оцінка точності на тренувальному та тестовому наборі, зображені на рисунку 4.9.

```
accuracy_score(train_label , y_pred_train)
```

```
0.9584642857142858
```

```
accuracy_score(test_label , y_pred)
```

```
0.9531428571428572
```

Рисунок 4.9 – Результати точності

Проте висока точність на тренувальному та тестововму наборі очікувано буде високою проте, яку точність покаже наша модель на реальному наборі новин, які були отримані в розділі 3, зараз побачимо на рисунку 4.10.

	precision	recall	f1-score	support
0	0.86	0.89	0.88	3213
1	0.90	0.87	0.88	3522
accuracy			0.88	6735
macro avg	0.88	0.88	0.88	6735
weighted avg	0.88	0.88	0.88	6735

Рисунок 4.10 – Результати точності на реальному наборі

Аналізуючи дані отримаємо результат становить 88%, що не такий високий показник,як на тренувальному наборі, але все ж таки достатньо високий та ефективний.

ВИСНОВКИ

Для досягнення поставленої мети кваліфікаційної роботи було проаналізовано методи парсингу новин, існуючі програмні забезпечення, виявлені їх недоліки та переваги, змодельовано систему та вибрані оптимальні інструменти для розробки. Був забезпечений експорт даних, розроблена модель для вилучення необхідної інформації, розроблений та протестований модуль для аналізу отриманих даних на їх правдивість.

Результатом цих заходів стало створення програмного забезпечення пасрингу веб-сайтів новин. Розроблене програмне забезпечення у повній мірі використовує основні можливості новітніх технологій для вирішення прикладної задачі.

На підставі проведеного дослідження була запропонована та розроблена програма, парсингу та аналізу веб-сайтів новин з можливістю чіткого виявлення статей, отримання всієї доступної та необхідної інформації про неї, можливістю зберігання інформації у різних форматах для експорту в інші програми. Також було розроблено аналіз статей на їх правдивість за допомогою машинного навчання.

В ході дослідження була проведена оцінка роботи програми, а саме, її час роботи, точність, та обє’м отриманої інформації, яка показала свою високу ефективність та покращила результати своїх аналогів у півтора рази.

Функціональні можливості спроектованого та розробленого веб-застосунку можуть бути доопрацьовані та розширені у відповідності до вимог тієї сфери, де буде застосовано розроблену систему.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Радченко А. В. МЕТОДИ ТА ЗАСОБИ ПАРСИНГУ ВЕБ-САЙТІВ НОВИН / А. В. Радченко, А. М. Носик // СУЧАСНІ НАПРЯМИ РОЗВИТКУ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЗАСОБІВ УПРАВЛІННЯ / А. В. Радченко, А. М. Носик.. – С. 84–85.

2. A Comparative Study on Web Scraping [Електронний ресурс]. – 2015 – Режим доступу до ресурсу: http://ir.kdu.ac.lk/bitstream/handle/345/ 1051/com-059.pdf?sequence=1&isAllowed=y

3. What is a Parser? [Електронний ресурс] – Режим доступу до ресурсу: https://www.techopedia.com/definition/3854/parser.

4. The dangers of web scraping [Електронний ресурс]. – 2016 – Режим доступу до ресурсу: https://www.information-age.com/dangers-web-scraping-123461971/

5. Види парсингу [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/446488/.

6. Мельник К. В. Автоматичний збір інформації (парсинг) в мережі / К. В. Мельник, В. М. Мельник, А. М. Григоришин. // Науковий журнал "Комп’ютерно-інтегровані технології: освіта, наука, виробництво". – 2020. – №39.

7. Advantages and disadvantages of parsing [Електронний ресурс] – Режим доступу до ресурсу: https://www.promptcloud.com/blog/web-scraping-advantages-and-dis-advantages/.

8. Sitemaps [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Sitemaps.

9. Css Vs. X Path [Електронний ресурс] – Режим доступу до ресурсу: http://elementalselenium.com/tips/34-xpath-vs-css-revisited-2.

10. Document_Object_Model [Електронний ресурс] – Режим доступу до ресурсу:https://developer.mozilla.org/enUS/docs/Web/API/Document_Object_Mo

del.

11. Scrapy architecture [Електронний ресурс] – Режим доступу до ресурсу: https://doc.scrapy.org/en/latest/.

12. Types of Web Scraping Tools [Електронний ресурс]. – 2017 – Режим доступу до ресурсу: https://medium.com/prowebscraper/types-of-web-scraping-tools-940f824622fb

13. Jsoup: HTML-парсер Java [Електронний ресурс] – Режим доступу до ресурсу: https://jsoup.org/cookbook/extracting-data/example-list-links.

14. Scraping a Ghost blog with PhantomJS and SlimerJS using CasperJS [Електронний ресурс] – Режим доступу до ресурсу: https://badacadabra.github.io/Scraping-a-Ghost-blog-with-PhantomJS-and-SlimerJS-using-CasperJS/.

15. Octoparse [Електронний ресурс] – Режим доступу до ресурсу: https://www.octoparse.com/.

16. Mozenda [Електронний ресурс] – Режим доступу до ресурсу: https://www.mozenda.com/.

17. Import i.o. [Електронний ресурс] – Режим доступу до ресурсу: https://www.import.io/.

18. What are the Different Scraping Techniques [Електронний ресурс]. – 2017 – Режим доступу до ресурсу: https://www.shieldsquare.com/what-are-the-different-scraping-techniques

19. Web Scraping Tools to Extract Online Data [Електронний ресурс]. – 2019 – Режим доступу до ресурсу: https://www.hongkiat.com/blog/web-scraping-tools/

20. Thorsten Joachims. “Text categorization with Support Vector Machines: Learning with many relevant features”. In: Machine Learning: ECML98. Ed. by Claire Nédellec and Céline Rouveirol. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 137–142. isbn: 978-3-540-69781- 7.

21. Selenium [Електронний ресурс] – Режим доступу до ресурсу: https://www.selenium.dev/documentation/.

22. UTM, what is it [Електронний ресурс] – Режим доступу до ресурсу: https://blog.hubspot.com/marketing/what-are-utm-trackinght#:~:text=UTM%20(Urchin%20Tracking%20Module)%20codes,a%20link%20to%20this%20URL.

23. Supervised Learning [Електронний ресурс] – Режим доступу до ресурсу: https://www.aitude.com/supervised-vs-unsupervised-vs-reinforcement/

24. TfidfVectorizer [Електронний ресурс] – Режим доступу до ресурсу:https://scikitlearn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html.