МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ЛУК'ЯНЧУК ОЛЕКСАНДРА В'ЯЧЕСЛАВІВНА

Допускається до захисту:
завідувач кафедри
інформаційних технологій
д. т. н., доцент
_____ Т.В. Нескородєва
«_____» _____ 2022р.

РОЗРОБКА ВЕБ-ДОДАТКУ ДЛЯ АНАЛІЗУ МЕТА-ДАНИХ САЙТУ

Спеціальність 122 «Комп'ютерні науки»
Кваліфікаційна (магістерська) робота

Науковий керівник:
Січко Т.В. канд. техн. наук, доцент
кафедри інформаційних технологій

Оцінка: _____/_____
(бали за шкалою ЄКТС/за національною шкалою)
Голова ЕК: _____
(підпис)

Вінниця – 2022

ЗМІСТ

АННОТАЦІЯ

Лук’янчук О.В. Розробка веб-додатку для аналізу мета-даних сайту. Спеціальність 122 «Комп’ютерні науки», Освітня програма «Data science». Донецький національний університет імені Василя Стуса, Вінниця, 2022.

У кваліфікаційній (магістерській) роботі розроблено аналізатор якості сайтів з боку пошукової оптимізації. Показані кроки розробки, отримані результати, використані інструменти, підходи до розробки, порівняння із наявними аналогами, перспективи подальшого розвитку даної системи.

Ключові слова: SEO, пошукова оптимізація, аналіз даних, веб-додаток, система для аналізу.

Lukianchuk O. Development of a web application for site meta data analysis. Specialty 122 "Computer science", Educational program "Data science". Vasyl’ Stus Donetsk National University, Vinnytsya, 2022.

In the qualification (master's) work was developed the SEO site analyzer. Development steps, obtained results, used tools, approaches to development, comparison with existing analogues, prospects for further development of this system are shown.

Keywords: SEO, search engine optimization, data analysis, web application, analysis system.

ВСТУП

Актуальність роботи. Пошукова оптимізація (SEO) є важливою складовою у просуванні сайту в Інтернеті. SEO (Search Engine Optimization) – це оптимізація пошуку, тобто процес покращення сайту, що підвищить його видимість у пошукових системах [1]. Вживши всіх необхідних заходів, таких як усунення помилок, оптимізація картинок, тексту, виправлення пошкоджених посилань та інше сайт зможе піднятись в рейтингу та зайняти вищі позиції в результаті пошуку за певним запитом.

Чим вища позиція за результатами пошуку, тим більша ймовірність збільшити трафік відвідувачів сайту.

Даний програмний продукт спрямований на виявлення описаних вище недоліків та буде корисним помічником для власників сайтів, спеціалістів з пошукової оптимізації та веб-розробників.

Мета дослідження. Метою роботи є розробка програмного засобу для SEO недоліків сайту для подальшої пошукової оптимізації.

Об’єкт дослідження. Аналіз мета-даних, структури та наповнення сайту; дослідження процесу ранжування сторінок в пошукових системах.

Предмет дослідження. Методи оптимізації.

Апробація. Результати кваліфікаційної (магістерської) роботи апробовано на III Всеукраїнській науково-практичній конференції «Комп’ютерні технології обробки даних», яка відбулась 8 грудня 2022 року на базі кафедри інформаційних технологій Донецького національного університету імені Василя Стуса. Тези на тему: «Розробка веб-додатку для аналізу мета-даних сайту» було опубліковано в електронному збірнику наукових праць.

Структура роботи. Кваліфікаційна (магістерська) робота складається зі вступу, трьох розділів, списку літератури та додатків.

Перший розділ присвячений ознайомленню з поняттям пошукової оптимізації (SEO). Розглянуто дані з якими працює SEO та етапи пошукової

оптимізації та просування сайту, проведено розбір процесу ранжування сторінок в мережі Інтернет, приведений порівняльний аналіз додатків які займаються виявленням недоліків пошукової оптимізації.

Другий розділ описує основні задачі які повинен вирішувати розроблений в рамках кваліфікаційної роботи додаток. Досліджено предметну область, виконано постановку задачі, описано методи та підходи до вирішення поставленої задачі.

Третій розділ присвячений розробці додатку для аналізу мета-даних сайту. У даному розділі розглядаються використані при розробці інструменти, структура та створений функціонал додатку, наведено екранні форми з результатами роботи програми та описано подальші перспективи розвитку створеного додатку.

Додатки містять фрагменти коду, які допомагають краще зрозуміти особливості реалізації програмного засобу.

Кваліфікаційна робота включає в себе 73 сторінки, 11 рисунків, 51 літературне джерело та 3 додатки.

РОЗДІЛ 1
ЗАГАЛЬНІ ВІДОМОСТІ ПРО АНАЛІЗ МЕТА-ДАНИХ САЙТУ, СФЕРУ ЗАСТОСУВАННЯ ТА ОБРОБКУ ДАНИХ

1.1 Поняття та цілі аналізу мета-даних сайту

Аналіз мета-даних сайту це частина процесу пошукової оптимізації (SEO). Пошукова Оптимізація (search engine optimization, SEO) — комплекс заходів щодо внутрішньої та зовнішньої оптимізації для підняття позицій сайту в результатах видачі пошукових систем за певними запитами користувачів, з метою збільшення мережного трафіку (для інформаційних ресурсів) та потенційних клієнтів (для комерційних ресурсів) та наступної монетизації з цього трафіку [1]. Зазвичай, чим вище позиція сайту в результатах пошуку, тим більше зацікавлених відвідувачів переходить на нього з пошукових систем.

Основні цілі пошукової оптимізації полягають в тому щоб за допомогою зібраних мета-даних сайту визначити «слабкі місця» сайту з боку структури, наповнення, стилів, зручності користування та на основі отриманих і проаналізованих даних вжити ряд заходів, що допоможуть сайту в просуванні у пошукових системах.

Пошукові системи враховують безліч внутрішніх та зовнішніх параметрів сайту при обчисленні його релевантності (ступеня відповідності введеному запиту):

- щільність ключових слів (складні алгоритми сучасних пошукових систем дозволяють проводити семантичний аналіз тексту, щоб відсіяти пошуковий спам , в якому ключове слово зустрічається дуже часто;
- індекс цитування («ІЦ») та тематичний індекс цитування («ТІЦ») залежать від кількості та авторитетності веб-ресурсів, що посилаються на даний сайт. Багатьма пошуковими системами не враховуються взаємні посилання (один на одного). Спосіб

нарощування числа сайтів-донорів, що посилаються на сайт, що просувається, називається лінкбілдінг;

- водність тексту - показник, що визначає наявність малозначущих слів, які не несуть жодної корисної інформації та служать для розведення тексту (стоп-слова);
- поведінкові фактори (внутрішні) — ряд усіляких дій користувачів, які вони можуть зробити на сайті: вхід, загальний час, проведений користувачем на сайті, кількість сесій одного користувача на сайті, перегляд сторінок, кількість переглянутих користувачем сторінок, повернення користувача на сайт, кліки на посилання у тексті, переходи за посиланнями у меню;
- поведінкові фактори (зовнішні) - основним зовнішнім показником якості поведінки користувача при взаємодії з сайтом є відмова від подальшого пошуку за ключовою фразою у пошуковій системі;
- індекс якості сайту ("ІКС") - це показник того, наскільки корисний конкретний сайт для користувачів. ІКС введено в 2018 році замість тематичного індексу цитування (ТІЦ), який враховував тематику сайтів, що посилаються.
- швидкість завантаження сайту – показник швидкості завантаження сайту. Використовується кілька параметрів для характеристики швидкості завантаження сайту – завантаження до появи першого контенту, завантаження першого контенту до взаємодії, швидкість відповіді сервера на запит, довжина HTML коду. Загальноприйнятим стандартом швидкості завантаження сайту прийнято вважати сервіс Google PageSpeed .

1.2 Етапи пошукової оптимізації та просування сайту

Для того, щоб просування ресурсу було успішним (тобто щоб він опинився у верхніх рядках пошукової видачі), можна піти кількома шляхами. Наприклад, можна оплатити просування за допомогою реклами. В такому разі, посилання на сайт буде з'являтись найвище серед результату пошуку та маркуватись позначкою «Реклама». Інший варіант - провести оптимізаційні роботи для сайту. Цей спосіб вважається умовно безкоштовною можливістю просування, так як власнику сайту не потрібно платити пошуковим системам за місце в рейтингу. Процес оптимізації складається з наступних етапів:

1. Вивчення специфіки інтернет-ресурсу та конкурентного середовища на ринку та в пошукових системах. На даному етапі докладно досліджуються особливості сфери, до якої належить сайт, вивчаються основні потреби його потенційних користувачів, виявляються ключові конкурентні переваги та відмінності продукту. Все це необхідно для правильного підбору семантичного ядра та розробки якісного сайту, здатного зайняти топові позиції в пошукових системах.

2. Збір та кластеризація семантичного ядра. Цей етап передбачає: підготовку масок (пошук високочастотних запитів), підбір за зібраними масками ключових фраз, що входять до СЯ, чищення СЯ від ключів, які не мають відношення до тематики сайту, збір частотності ключових фраз за допомогою спеціальних ресурсів (наприклад, Wordstat), кластеризацію запитів СЯ за групами.

3. Проектування структури веб-сайту. Для ефективної оптимізації сайту потрібно розробити його структуру так, щоб кожен кластер запитів мав свою посадкову сторінку (лендінг). Це дасть користувачеві можливість швидко знайти в системі навігації всі необхідні розділи. Також грамотно спроектована структура сайту дозволяє покращити показники його індексації та охопити великий обсяг посадкових сторінок під пошукові запити.

4. Внутрішня (технічна) оптимізація. Даний вид роботи над сайтом орієнтований на виконання технічних вимог, що висуваються пошуковими системами до ресурсів. Проведення внутрішньої оптимізації передбачає перевірку:

- верстки сайту, код якого повинен відповідати встановленим стандартам та не повинен містити помилок;
- наявність на кожній сторінці сайту її заголовку в пошуковій видачі (title), опису (description) з урахуванням рекомендованої довжини кожного мета-тега (в середньому 70-80 символів для title, 200 для description), тематичних заголовків (h1), перерахованих ключових слів (keywords), опису зображень (alt);
- оптимізації зображень (стиснення їх розмірів та ваги під певні параметри залежно від формату). Оптимальними форматами вважаються webp та svg.

Крім того, сайт повинен швидко завантажуватися, містити входження (згадування) ключових запитів, не мати сторінок, що дублюються, і битих (неробочих) посилань.

Для технічної оптимізації сайту також слід встановити всі потрібні скрипти (виведення стилів, шрифтів, метрик) таким чином, щоб їх завантаження не сповільнювало роботу користувача з сайтом, перевірити правильність відображення сайту на різних пристроях, налаштувати людино-зрозумілі URL (адреси сторінок).

5. Оптимізація юзабіліті. До цього комплексу заходів включаються:

- забезпечення адаптивності сайту під усі пристрої та зручності роботи з сайтом;
- виділення активних пунктів меню;
- створення пошуку по сайту за допомогою якого користувач швидко та легко знайде товар/статтю/інформацію. Також корисними елементами є фільтри та пагінація;

- розміщення логотипу бренду у зручному місці;
- додавання блоку меню із зазначенням аналогічних та рекомендованих товарів, розділу з інформацією про компанію;
- вказівка розташування офісів компанії на картах Google Яндекс;
- вказує розміри та формати файлів для скачування.

6. Оптимізація контенту. Проведення оптимізаційних робіт з текстовим наповненням сторінки. Розміщення ключових фраз, видалення водного тексту, який несе мало змісту.

7. Перелінкування сайту (розстановка внутрішніх посилань на сторінки ресурсу).

8. Робота з комерційними факторами (налаштування калькуляторів та кошиків для інтернет-магазинів).

9. Реєстрація сайту. Бажаною є реєстрація сайту у вебмайстрах - для отримання даних стосовно індексації сайту, також інформації стосовно різних помилок, таких як помилки обробки сайту, недоліки системних файлів та ін.; Також рекомендованою є реєстрація у системах аналітики – для перевірки відвідуваності ресурсу, поведінкових характеристик, конверсії та інших параметрів сайту. Реєстрація у довідниках для бізнесу допоможе позначити регіональну приналежність компанії та розмістити докладну інформацію про неї.

10. Зовнішня оптимізація, тобто отримання посилань на сайт із різних зовнішніх ресурсів. При цьому зовнішні посилання бувають платними (орендними та «вічними»), і їх можна придбати на спеціалізованих біржах та безкоштовними – які можна отримати, зареєструвавшись у соцмережах, на сайтах відгуків, у довідниках та каталогах.

Розберемо детальніше види оптимізації. Усі фактори, що впливають на положення сайту у видачі пошукової системи, можна розбити на зовнішні та внутрішні.

До зовнішньої оптимізації відноситься Клауд-Маркетинг, Outreach, оптимізація посилань на сайтах-донорах та ін.

До внутрішньої оптимізації (що стосується виключно внутрішньої системи сайту) відноситься робота, спрямована на загальне підвищення якості сайту, користі, яку він приносить відвідувачу. Сюди можна віднести роботу над структурою проекту, над полегшенням сприйняття контенту та безпосередньо над якістю цього контенту.

Внутрішня оптимізація включає роботу із заголовками сторінки, які містяться в коді з тегами <h1>, <h2>, <h3>, мета-тегами, написом, який висвічується на вкладці браузера - Title, і створенням унікального тексту на цих сторінках. Також важливо приділити увагу мета-тегу description, оскільки саме його користувач найчастіше бачить під url сайту у пошуковій видачі.

Теги HTML схожі на ключові слова, які визначають, як веб-браузер форматуватиме та відображатиме вміст. За допомогою тегів веб-браузер може відрізнити вміст HTML від простого вмісту. Теги HTML містять три основні частини: відкриваючий тег, вміст і закриваючий тег. Але деякі теги HTML є тегами що не потребують закриття. Коли веб-браузер читає документ HTML, він читає його зверху вниз і зліва направо. Теги HTML використовуються для створення документів HTML і відтворення їхніх властивостей. Кожен тег HTML має різні властивості.

Атрибути HTML - це спеціальні слова, які надають додаткову інформацію про елементи, або атрибути є модифікатором елемента HTML. Кожен елемент або тег може мати атрибути, які визначають поведінку цього елемента.

HTML дозволяє вказувати метадані – додаткову важливу інформацію про документ різними способами. Елементи meta можна використовувати для включення пар ім’я/значення, що описують властивості HTML-документа, такі як автор, термін дії, список ключових слів, тощо.

Тег <meta> використовується для надання такої браузеру інформації. Цей тег є порожнім елементом і тому не має закриваючого тегу, але він містить інформацію в своїх атрибутах. Ви можете включити один або кілька

метатегів у свій документ залежно від того, яку інформацію ви хочете зберегти у своєму документі. Загалом метатеги не впливають на зовнішній вигляд документа, тому з точки зору зовнішнього вигляду не має значення, чи ви додасте їх чи ні. Але, якщо говорити про пошукову оптимізацію, то вони грають важливу роль для браузерів та пошукових роботів. Інформація, що міститься в атрибутах цих тегів дає браузеру зрозуміти, як правильно обробляти сторінку, яка її мова тощо.

Крім роботи з мета-тегами, у внутрішню оптимізацію сайту входить комплекс заходів, по поліпшенню його використання користувачем (простота та зрозумілість під час користування).

Оптимізація зображень: часто саме картинки мають найбільший розмір тим самим уповільнюють швидкість роботи, тому потрібно масштабувати зображення під потрібні розміри, використовувати лише відповідний формат. Додатково можна обмежити палітру кольорів і застосовуйте дельта-кодування, це знизить вагу зображень в кілька разів. У картинок є свої мета-теги alt та title. Заповнення тегів допоможе роботам в індексації.

Внутрішня перелінковка. Це проставлення посилань усередині сайту з одного матеріалу на інший. Проводиться зазвичай для зручності відвідувачів сайту та покращення юзабіліті. Коли відвідувач читає статтю, він бачить посилання на пов'язані матеріали та може одразу перейти на них. Йому не потрібно шукати додаткову інформацію через пошук. Або на сайті інтернет-магазину в огляді продукції вказують прямі посилання на товар. Для SEO цілей перелінковування потрібне при розподілі ваги між сторінками або просуванні за низькочастотними запитами.

Технічна оптимізація проводиться, щоб усунути помилки на сайті. Спочатку варто перевірити наявність «битих» посилань. Якщо сторінка веде в нікуди, то вага сторінки знижується, а за ним і загальна вага сайту. Такі посилання потрібно видалити або замінити на такі, що ведуть до існуючих сторінок.

Усування дублювання інформації. Знайдіть дублювання мета-тегів title, description та h1. Зробити їх унікальними без втрати сенсу. Знайти та виправити дубльований контент. Зверніть увагу, що інформація на шапці сайту, у підвалі, меню не відноситься до дубльованого контенту, хоч і повторюється на кожній сторінці. У інформаційних технологіях вона називається common content.

Швидкість завантаження впливає показник відмов. Якщо сайт завантажується понад 3 секунди, зазвичай, користувачі покидають такі сторінки.

Файл robots.txt або індексний файл — звичайний текстовий документ у кодуванні UTF-8, що діє для протоколів http, https, а також FTP. Файл дає пошуковим роботам поради: які сторінки/файли варто сканувати. Якщо файл містить символи не в UTF-8, а в іншому кодуванні, пошукові роботи можуть неправильно їх обробити. Правила, перелічені у файлі robots.txt, дійсні лише щодо того хоста, протоколу та номера порту, де розміщено файл.

Файл повинен розташовуватись у кореневому каталозі у вигляді звичайного текстового документа та бути доступним за адресою: https://site.com.ua/robots.txt .

Отже, методи внутрішньої пошукової оптимізації:

- валідація проекту;
- збільшення швидкості роботи сайту;
- адаптація під мобільні пристрої;
- складання семантичного ядра (СЯ);
- аналіз сайтів-конкурентів;
- написання якісного контенту та створення правильної структури сайту;
- коригування текстової релевантності усієї сторінки;
- оптимізація тегів H1, Title, мета-тегів та Description;

- внутрішня перелінковка.
- оптимізація картинок та відео

1.3 Ранжування сторінок в мережі Інтернет

Більшість пошукових систем, використовують ботів для сканування сторінок в Інтернеті.

Пошуковий індекс – це, по суті, весь перелік веб-сайтів, з яких пошуковий сайт черпає дані, щоб надати користувачам результати пошуку. Хоча може здатися, що пошукові системи достатньо великі, щоб перевести вас на будь-який сайт в Інтернеті, це неправда. У результатах пошуку можуть відображатися лише проіндексовані сайти. Нові сайти завжди можна додавати до індексу, і саме це і є індексація - процес додавання веб-сайту до індексу. Індексація відбувається, коли веб-сканери, також звані павуками, сканують веб-сайти в Інтернеті [2].

Процес появи в результатах пошуку Google відбувається в три етапи – сканування, індексація та ранжування.

Перша взаємодія пошукового сайту з вашим веб-сайтом відбувається під час його сканування. Веб-сканер може виявити сайт багатьма способами – можливо, він переходить за посиланням з іншого сайту, або, можливо, ви надсилаєте свою карту сайту безпосередньо в пошуковий сайт. У будь-якому випадку, як тільки сканер знайде ваш сайт, він просканує його, щоб виявити, що на ньому розміщено. Він читає текст, оцінює макет і робить усе можливе, щоб читати зображення та відео

Після того, як пошуковик просканує ваш сайт, наступним кроком є індексація. Це критично — якщо ваш сайт не відповідає правильним вимогам. Пошуковик не проіндексує його, і сайт не матиме шансів на рейтинг. Кілька речей можуть призвести до того, що пошуковик не індексує сайт:

- Noindex: якщо сайт використовує тег «noindex» у своєму HTML, це повідомляє Google не індексувати цей сайт.

- Вміст: пошуковик не буде індексувати сторінку з вмістом, який не має цінності для користувачів.
- Дубльований вміст: сторінки, які повністю складаються з дубльованого вмісту, мають меншу ймовірність бути проіндексованими.
- Карти сайту (SiteMap): Створення та надсилання карти сайту дає змогу повідомляти пошуковику про ваш веб-сайт, підвищуючи ймовірність його сканування.
- Канонізація: якщо існує кілька версій сторінки, і ви позначаєте одну з них як неканонічну , тобто не «справжню» версію, пошуковик не буде індексувати цю версію.

Якщо нічого не попереджає пошуковик, сканер використає інформацію, знайдену на вашому сайті, щоб визначити, про що йдеться, а потім додасть її до свого пошукового індексу - величезної бази даних усього вмісту, який вони виявили та вважають достатньо якісним, щоб обслуговувати шукачів.

Далі алгоритми аналізують сторінки в індексі, беручи до уваги сотні факторів ранжирування або сигналів, щоб визначити порядок сторінок, які мають з’являтися в результатах пошуку за певним запитом. Це те, як пошукові роботи точно оцінюють, наскільки повно веб-сайт або веб-сторінка можуть надати шукачеві те, що він шукає.

Якщо чекати досить довго, існує велика ймовірність того, що пошуковик зрештою самостійно просканує та проіндексує ваш сайт.

Але чим швидше сайт буде проіндексовано, тим швидше можна почати збільшувати свій дохід. З цієї причини краще діяти активно. Можна зробити це, надіславши свою карту сайту безпосередньо в пошуковик.

Карта сайту XML (sitemap.xml) – це сторінки веб-сайту, створені у форматі XML. Вони містять інформацію для пошукових систем та доступні для сканування пошуковим роботом. XML-карта відрізняється від картки веб-сайту HTML.

- XML-карта дає такі переваги:
- вказується час, коли сторінки востаннє оновлювалися;
- можна визначити розташування сторінок конкретного веб-сайту;
- визначається пріоритет веб-сторінок у структурі;
- Ви можете визначити періодичність оновлення, а також важливість інших веб-сторінок.
- Крім того, коректне налаштування карти покращує індексацію сторінок сайту.

Є випадки коли карта сайту необхідна, а є випадки, коли можна обійтись без неї. Розберемо детальніше.

Файл необхідний у таких випадках:

- Якщо у вас багатосторінковий сайт і є ймовірність, що пошукові системи не помітять нові або змінені веб-сторінки.
- Веб-сайт новий та містить невелику кількість нових посилань.
- Нема відповідної структури сторінок.
- На сайті міститься багато мультимедійної інформації чи новин.

Немає необхідності в Sitemap у таких випадках:

- Веб-сайт малий.
- Немає великої кількості сторінок із новинами та мультимедійною інформацією.
- Введено докладну систему внутрішніх посилань.
- Тому спочатку слід оцінити доцільність Sitemap.

Окрім надсилання карти сайту, ви можете оптимізувати свій сайт декількома різними способами, щоб забезпечити йому найкращий шанс для індексування, зокрема:

- Забезпечення якості та оригінальності: переконайтеся, що всі ваші сторінки є цінними для користувачів. Практикуйте хорошу тактику веб-дизайну та уникайте дублювання вмісту.

- Перевірка ваших мета-тегів: подивіться на своєму сайті будь-які фальшиві теги noindex або канонічні теги — якщо вони помилково розміщені на сторінці, це означатиме, що сторінка не буде проіндексована. Звичайно, у випадках дублювання вмісту вам знадобляться мета-теги.
- Очищення навігації: переконайтеся, що у вас немає жодних «загублених» сторінок, тобто сторінок, на які немає посилань з іншого місця на сайті. Усі проіндексовані сторінки вашого сайту мають бути певним чином пов’язані одна з одною.

Якщо сайт оптимізовано для індексування та надіслано карту сайту, незабаром ваш сайт увійде в індекс пошуку.

1.4 Огляд додатків-аналогів для пошукової оптимізації

Інструменти SEO позбавлять вас від стомлюючого дослідження ключових слів і аналізу даних. За допомогою цих інструментів ви можете побачити, що працює та які частини вашої стратегії можна було б виграти від деяких налаштувань. Найкращі інструменти оптимізації пошукових систем також надають звіти про те, наскільки ви відповідаєте конкурентам і де є найбільші можливості. Більше того, вони дозволяють вимірювати ефективність пошуку в країнах, регіонах або мовах.

Якщо ви керуєте кількома веб-сайтами, інструменти оптимізації пошукових систем (SEO) допоможуть вам оцінити ефективність кожного сайту на ходу. Багато підприємців, які мають кілька веб-сайтів, зрештою поміщають багато даних в електронні таблиці та аналізують їх вручну. Але незабаром це стає надто великим і підвищує ризик того, що звіти будуть неточними. На щастя, ви можете використовувати програмне забезпечення SEO, щоб заощадити години зусиль і створити точні звіти.

Розглянемо близький аналог на який ми рівняємось – розширення для браузеру Chrome – SEO Minion. Він є одним з найбільш рейтингованих

додатків даного профілю та користується популярністю у SEO-спеціалістів та власників сайтів, які власноруч бажають покращити свою веб-сторінку.

Розширення браузера - це мікропрограма, яка розширює (доповнює) функціонал браузера, вона здатна через браузер вбудовуватися в хмарний софт, і розширити його функціонал. Їх називають browser extensions (розширення браузера), plug-in (плагін) або add-on (додаток), і зараз вони підтримуються практично всіма браузерами. Для створення розширення потрібно створити HTML, CSS, JavaScript складові проекту і продумати його функціональне навантаження. Готове розширення надається користувачеві у вигляді архіву, або завантажується через офіційний магазин розширень браузера (Chrome, Opera, Firefox та інші), і просто активується [4].

Переваги та функціонал додатку:

- Аналіз On-Page SEO: можливість проаналізувати SEO-проблеми для будь-якої веб-сторінки: мета-теги, заголовки, відкритий графік тощо.
- Перевірка непрацюючих посилань: перевірка всіх непрацюючих посилань на сторінці та їх класифікація на основі кодів станів.
- Перевірка перенаправлень (Redirects): додаток дозволяє перевірити усі переспрямування на сторінці (Server, MetaRefresh & Javascript)
- Інструмент попереднього перегляду SERP (Search Engine Result Page): допомагає перевірити позицію сторінки по відношенню до того чи іншого ключового запиту в пошуковик.
- Симуляція пошукової видачі з кількох місць: дозволяє перевірити рейтинг Google для свого веб-сайту в різних локаціях (країнах) без персоналізації.

- Перевірка Hreflang: дозволяє перевірити всі атрибути Hreflang на сторінці, а також на всіх сторінках, пов’язаних через теги з цим атрибутом.

Hreflang - це HTML-атрибут, який використовують для сайтів з контентом кількома мовами та для різних регіонів. Іншими словами, якщо головна сторінка вашого сайту, example.com, має текст українською та англійською мовами, ви можете використовувати цей атрибут для вказування мови та географічного регіону сторінки. Таким чином пошуковик пропонуватиме правильну версію сторінки користувачам, виходячи з їх місцезнаходження [3].

Гіперпосилання — це піктограма, графіка або текст, які посилаються на інший файл або об’єкт. Альтернативні назви - посилання або веб-посилання [6].

Перенаправлення (Redirect) – переспрямування браузера на інше посилання. Кожна сторінка в Інтернеті має адресу, URL-адресу, що означає «Уніфікований покажчик ресурсу». Іноді вміст переміщується з однієї URL-адреси на іншу URL-адресу. Саме тоді вам знадобиться перенаправлення. Переспрямування автоматично змушує браузер переходити з однієї URL-адреси на іншу URL-адресу.

Нижче представлено приклад роботи програми, а саме – перевірка посилань веб-сторінки.

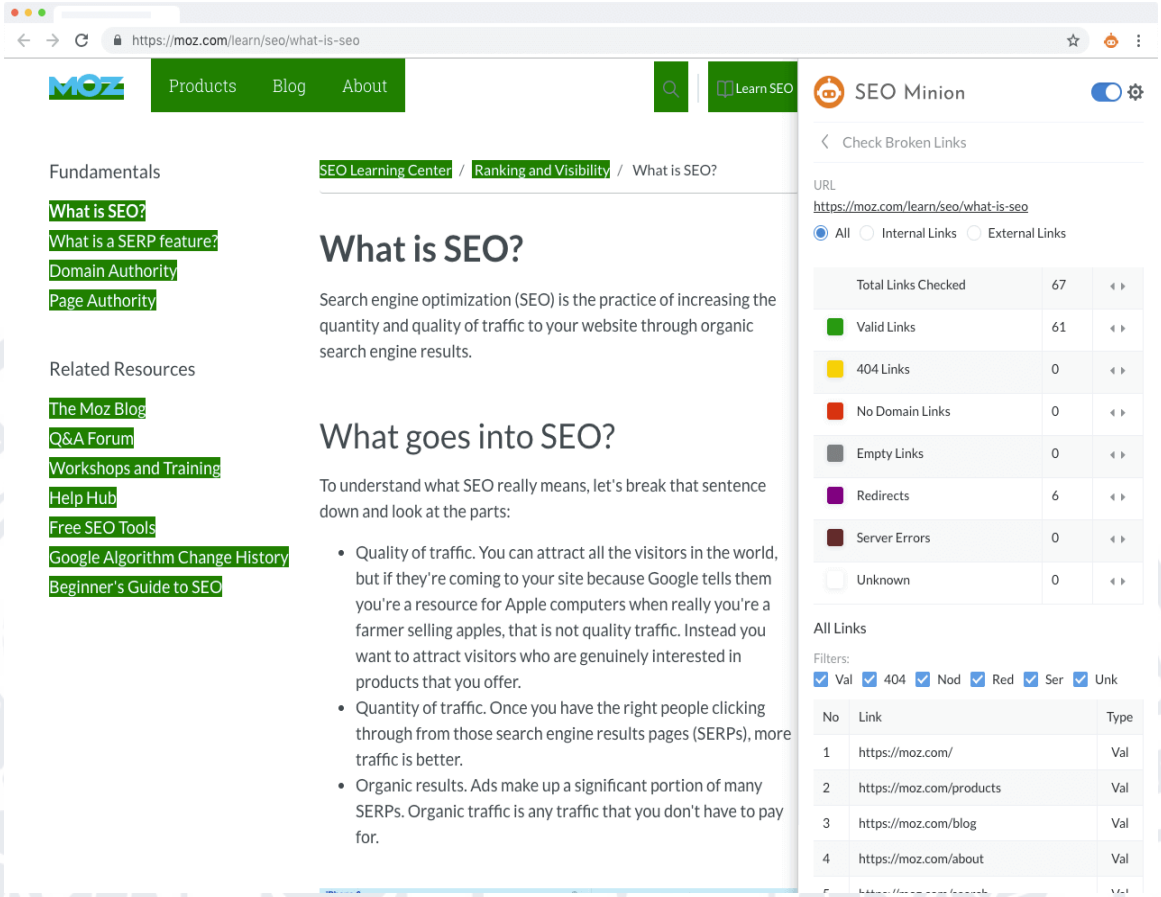

Рисунок 1.1 – Приклад роботи додатку SEO Minion

Ще одним відомим аналогом є Google Search Console. Додаток має багато функцій які надають аналітику та статистику по вашому веб-сайту. Призначений для користування як приватними особами так і корпоративними структурами. Гарно підходить для роботи веб-розробників, маркетологів, фахівців з оптимізації пошукових систем, адміністраторів, та власників бізнесу.

Google Search Console – це безкоштовна служба від Google, яка допомагає відстежувати, підтримувати та вирішувати проблеми з присутністю вашого сайту в результатах пошуку Google. Вам не обов’язково реєструватися в Search Console, щоб вас включили в результати пошуку Google, але Search Console допоможе вам зрозуміти та покращити те, як Google бачить ваш сайт.

Search Console пропонує такі інструменти та звіти для сайту:

- перевірка перевірте карту сайту;
- перевірка швидкості сканування та можливість переглянути статистику про те, коли Googlebot отримує доступ до Вашого сайту;
- перевірка конфігурацій файлу robots.txt , щоб допоможе виявити випадково заблоковані сторінки;
- перелік внутрішніх та зовнішніх сторінок, які посилаються на веб-сайт;
- список посилань, які Googlebot не зміг просканувати, включно з помилкою, отриманою Googlebot під час доступу до відповідних URL -адрес;
- список пошукових запитів за ключовими словами в Google, які призвели до того, що сайт було включено у видачу результатів пошуковика, а також загальну кількість кліків, і середній рейтинг кліків для таких списків;
- встановлення бажаного домену (наприклад, якщо власник сайту віддає перевагу домену example.com замість www.example.com або навпаки), який визначає, як URL-адреса сайту відображатиметься в результатах пошуку;
- можливість переглядати звіти про швидкість роботи сайту та звіти про взаємодію користувачем з сайтом;
- отримання сповіщення від Google;
- надання доступу до API інтеграцій;
- надання доступу до розділу для кращої взаємодії з мобільним пристроям.
- перевірка проблем з безпекою веб-сайту, якщо вони є. (Неполадки сайту або атаки зловмисного програмного забезпечення);
- можливість кооперативної роботи над сайтом;

- можливість проаналізувати схему структурованих даних на сайті на інформаційній панелі пошукової консолі Google. Також пропонується скористатися іншими інструментами Google признаеними для цього: Rich Results Test або Schema Markup Validator;
- Search Console надає інформацію про те, як Google сканує, індексує та обслуговує веб-сайти. Це може допомогти власникам веб-сайтів контролювати та оптимізувати ефективність пошуку на своєму веб-сайті;
- Google Search Console також має функцію під назвою Інструмент перевірки URL-адрес.

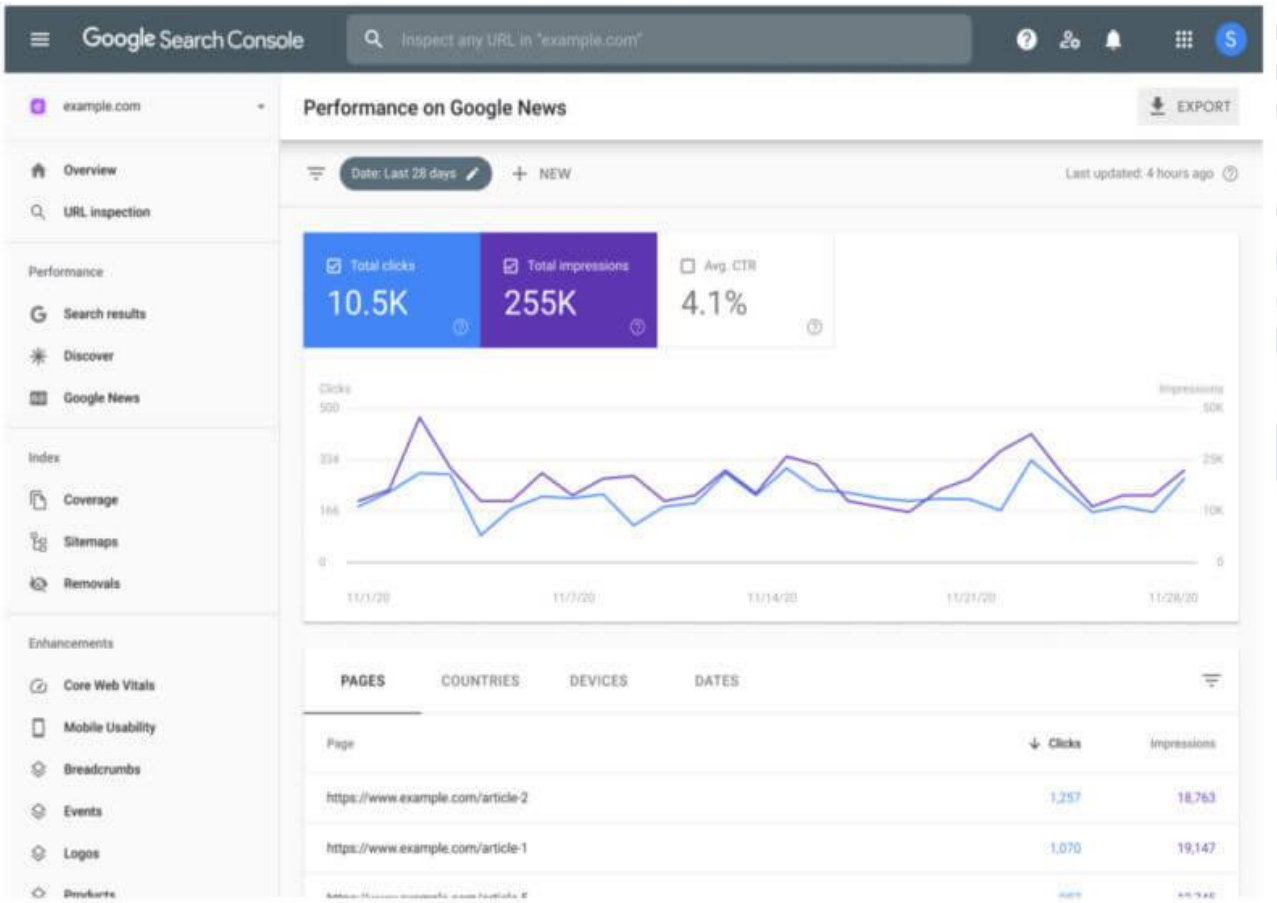

Рисунок 1.2 – Приклад роботи додатку Google Search Console

Наступний потужний аналог – це SEMRush. Маркетингові інструменти SEO, такі як SEMRush, як правило, є фаворитами шанувальників у спільноті SEO. Експертам подобається те, що вони дозволяють легко оцінювати свій рейтинг, а також визначати зміни та нові можливості рейтингу. Однією з найпопулярніших функцій цього інструмента для оптимізації пошукових систем є аналіз доменів проти доменів, який дозволяє легко порівняти ваш веб-сайт із веб-сайтами конкурентів. Якщо вам потрібні аналітичні звіти, які допоможуть вам краще зрозуміти пошукові дані вашого веб-сайту, трафік або навіть ваших конкурентів, ви зможете порівняти ключові слова та домени. Інструмент On-Page SEO Checker дозволяє легко відстежувати ваші рейтинги та знаходити рекомендації щодо покращення продуктивності вашого веб-сайту.

MozBar є доволі популярним Google Chrome розширенням серед спеціалістів пошукової оптимізації. Він дозволяє своїм користувачам перевіряти пошукову оптимізацію в браузері одним кліком миші. MozBar надає показники під час перегляду будь-якої веб-сторінки, а також дозволяє користувачам експортувати SERP у файл CSV і отримувати доступ до аналітики. Оновлення до MozBar Premium пропонує такі функції, як аналіз складності ключових слів, оптимізація сторінки та показники SERP.

Файл значень, розділених комами (CSV) — це текстовий файл із роздільниками, який використовує кому для відділення значень один від одного. Кожен рядок файлу є рядком даних. Кожен рядок складається з одного або кількох полів, розділених комами. Використання коми як роздільника полів є джерелом назви цього формату файлу. Файл CSV зазвичай зберігає дані таблиці (такі як числа чи текст) у вигляді тексту, в цьому випадку кожен рядок матиме однакову кількість полів. Формат файлу CSV не є повністю стандартизованим. Розділ полів комами є основою, але коми в даних або вбудовані розриви рядків повинні оброблятися спеціально. Деякі реалізації забороняють такий вміст, в той

час як інші оточують поле лапками, що ще раз створює необхідність екранувати їх, якщо вони присутні в даних [8].

Також дане розширення має відкрите API, що надає розробникам можливість інтегрувати сервіси в інші додатки.

Інтерфейс прикладного програмування (API) — це спосіб взаємодії двох чи більше комп'ютерних програм одна з одною. Це різновид програмного інтерфейсу, який пропонує послуги іншим частинам програмного забезпечення. Документ або стандарт, який описує, як створити чи використовувати таке з'єднання чи інтерфейс, називається специфікацією API. Кажуть, що комп'ютерна система, яка відповідає цьому стандарту, реалізує або надає API. Термін API може стосуватися або специфікації, або реалізації. На відміну від інтерфейсу користувача, який з'єднує комп'ютер з людиною, інтерфейс прикладного програмування з'єднує комп'ютери або частини програмного забезпечення один з одним. [9]

Нижче наведений приклад роботи програми:

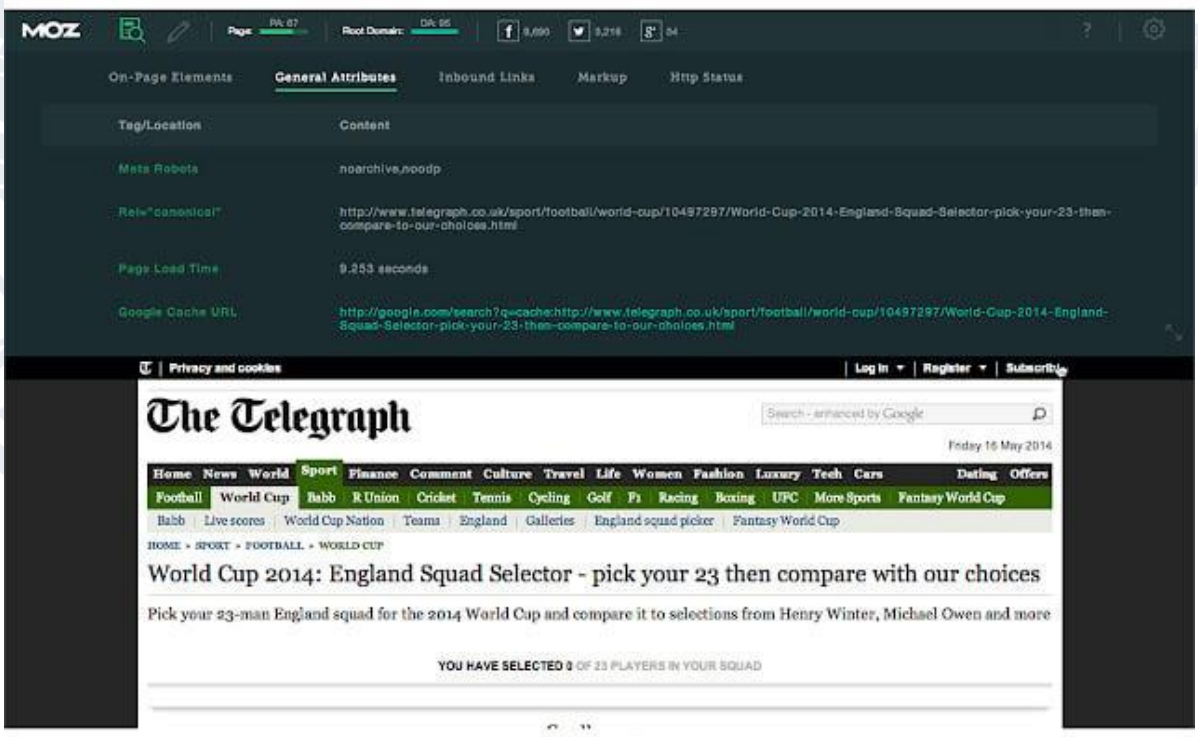

Рисунок.1.3 – Приклад роботи додатку MozBar

Висновки до розділу

У даному розділі було проведено ознайомлення з поняттям пошукової оптимізації та її видами. Визначено з якими даними працює SEO. Розглянуто етапи пошукової оптимізації, необхідні кроки для просування сторінок в мережі Інтернет та процес ранжування сторінок у пошукових системах. Виконано розбір та порівняння додатків які займаються аналізом недоліків веб-сторінок.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ДОДАТКУ ДЛЯ АНАЛІЗУ МЕТА-ДАНИХ САЙТУ

2.1 Постановка задачі

Розроблене програмне забезпечення повинне аналізувати сайти та виявляти їх недоліки зі структурою, мета-даними, наповненням, задля подальшої пошукової оптимізації. Веб-додаток повинен надавати наступний функціонал:

- збір необроблених даних. Цей пункт означає можливість додатку надсилати GET-запити до веб-ресурсу та отримувати у відповідь розмітку сторінки поточного сайту. Розмітка сторінки в даному випадку є необробленими даними, які будуть структуруватись на подальших кроках роботи ПЗ;
- робота з картою сайту. Додаток повинен надавати користувачеві можливість завантажити файл Sitemap.xml. Якщо користувач надав такий файл - відбувається його зчитування і обробка. Після обробки карти сайту починається обхід сторінок згідно з заданими в файлі правилами. Якщо ж Sitemap.xml не був завантажений, тоді обхід сайту починається з заданої сторінки і подовжується по всім дочірнім;
- обробка зібраних даних. Після обходу сторінки сайту та збору її розмітки починається обробка даних. Із зібраного темплейту виділяються цільові елементи, такі як теги заголовків, мета-теги, теги картинок, атрибути, ключові слова сторінки та ін. Усі вони обробляються шляхом переборів, порівнянь регулярних виразів. Ключові слова обробляються алгоритмом Портера, що допомагає відсікти від слова суфікси та закінчення. Результатом даної обробки є список недоліків SEO знайденими на

обробленій сторінці. Саме цей список буде опрацьовуватись спеціалістами в області пошукової оптимізації;

- візуальне представлення результатів обробки користувачеві. З оброблених даних генерується темплейт, який видається користувачеві. Дані в ньому угруповані по сторінках та мають просту навігацію.

Додаток працює за схемою клієнт-серверного програмного забезпечення, тобто існує розділення на серверну частину, яка займатиметься обробкою даних та клієнтську частину, яка представлятиме користувачеві оброблені дані.

В даному програмному продукті планується збір даних для обробки «на льоту». Тобто, додаток в режимі реального часу створює запит до сторінки веб-сайту та займається збором (скрапінгом) необроблених даних для подальшого видалення потрібних елементів та їх аналізу.

Таким чином, після отримання розмітки сторінки додаток починає виділенням таких основних даних як:

- Мета-теги
- Теги розмітки та їх атрибути
- Посилання та редіректи
- Ключові слова сторінки

Аналіз мета-тегів є одним з найважливіших аспектів пошукової оптимізаці. Адже, теги <meta> є одними з перших, які сканує браузер, так як вони містять в собі дані, які вказують пошуковику, яким чином відображати сторінку, яка поточна мова сторінки, яка очікувана поведінка на різних девайсах та ін. Також мета-теги містять в собі заголовок сторінки та її опис, які повинні бути оформлені згідно певних правил, щоб пошукові роботи змогли їх правильно розпізнати та надати користувачеві при релевантному пошуковому запиті.

Аналіз тегів розмітки має на меті перевірити структурні теги сторінки та їх атрибути. Для пошукової оптимізації важливо, щоб сторінка

відповідала певній структурі, якщо всі правила дотримані, пошукові роботи браузера вважають цю сторінку кращою для сприйняття користувачем та підіймають її в пошуковому рейтингу. Прикладом «правильної» розмітки можна навести ієрархію заголовків на сторінці. Існує 6 заголовкових тегів (від <h1></h1> до <h6></h6>). Тег, що має більший номер не повинен бути нижче тега з меншим номером. Тобто розмітка представлена на лістингу 2.1 буде вважатись не коректною.

```
<H1> Python Requests Module</H1>
    <H3> METHODS</H3>
        <H2> COLOR PICKER</H2>
```

Лістинг 2.1 – Приклад неправильної ієрархії заголовків

Натомість, ієрархія заголовків повинна бути чітка та послідовна, наприклад, як у лістингу 2.2.

```
<H1> Python Requests Module</H1>
    <H2> Tutorials</H2>
        <H3> HTML and CSS</H3>
        <H3> Data Analytics</H3>
    <H2> References</H2>
        <H3> CSS</H3>
            <H4> Examples</H4>
        <H3> Programming</H3>
```

Лістинг 2.1 – Приклад правильної ієрархії заголовків

Також, серед правил оформлення заголовків і те, що тег <h1></h1> повинен бути лише один на сторінці.

В додатку аналізуватимуться також атрибути тегів розмітки. Для прикладу, у тегу <img> обов'язково повинен бути заповненим атрибут alt що відповідає за альтернативний текст, який буде показано на сторінці в разі неможливості завантажити зображення, що поміщено в тег <img>.

Для аналізу якості посилань виконується збір усіх значень атрибуту href тегу <a>. Таким чином, визначається статус, який віддає посилання та перевіряється можливість перенаправлення. За правилами пошукової оптимізації редіректів для одного посилання не повинно бути більше шести.

Ключові слова аналізуються шляхом збору їх на сторінці та обробки за допомогою алгоритму Портера. Це алгоритм спрямований на відсічення від слова суфіксів та закінчень. У результаті отримана статистика є більш чистою та правдивою.

2.3 Огляд підходів до реалізації поставленої задачі

Даний програмний продукт повинен знаходити та вказувати недоліки сайту з боку наповнення для подальшої пошукової оптимізації. Першим кроком у зборі інформації є написання парсера.

Парсинг - це процес автоматичного збору даних та їх структурування.

Спеціальні програми або сервіси-парсери "обходять" сайт і збирають дані, які відповідають заданій умові.

Простий приклад: припустимо, потрібно зібрати контакти потенційних партнерів із певної ніші. Ви можете зробити це вручну. Потрібно заходити на кожен сайт, шукати розділ «Контакти», копіювати в окрему таблицю телефон і т.д. Так на кожен майданчик у вас піде по п'ять-сім хвилин. Але цей процес можна автоматизувати. Задаєте в програмі для парсингу умови вибірки та через якийсь час отримуєте готову таблицю зі списком сайтів та телефонів.

Плюси парсингу очевидні — якщо порівнювати його з ручним збором та сортуванням даних:

- ви отримуєте дані дуже швидко;
- можна ставити десятки параметрів для складання вибірки;
- у звіті не буде помилок;

- парсинг можна налаштувати з певною періодичністю - наприклад, збирати дані щопонеділка;
- багато парсерів не тільки збирають дані, а й радять, як виправити помилки на сайті.

Область застосування парсингу можна звести до двох цілей:

- аналіз конкурентів, щоб краще розуміти, як вони працюють, та запозичувати у них якісь підходи;
- аналіз власного майданчика для усунення помилок, швидкого впровадження змін тощо.

Підходом до збору інформації на сторінці був обраний веб-скрапінг на основі бібліотеки мови Python – BeautifulSoup 4

Вебскрапінг (scraping) — перетворення у структуровані дані інформації з вебсторінок, які призначені для перегляду людиною за допомогою браузера.

Як правило, виконується за допомогою комп'ютерних програм, що імітують поведінку людини в інтернеті, або з'єднуючись з вебсервером напряму по протоколу HTTP, або управляючи повноцінним веббраузером. Але буває і скрапінг за допомогою копіювання даних людиною. Це форма копіювання, в якій конкретні дані збираються та копіюються з інтернету, як правило, в базу даних або електронну таблицю для подальшого пошуку чи аналізу.

Вебскрапінг включає в себе завантаження та вилучення. Спочатку завантажується сторінка (що робить браузер, коли ви переглядаєте сторінку), після цього можна добувати потрібну інформацію. Зміст сторінки може бути проаналізовано, переформатовано, його дані скопійовані в електронну таблицю тощо. Вебскрапери, як правило, беруть щось із сторінки, щоб використати це для інших цілей деінде. Прикладом може бути пошук і копіювання імен та телефонних номерів або компаній та їх URL-адрес до списку (контактне сканування).

Вебсторінки побудовані за допомогою текстових мов розмітки (HTML та XHTML) і часто містять велику кількість корисних даних у текстовій формі. Однак більшість вебсторінок призначені для кінцевих користувачів, а не для зручності автоматичного використання. Через це були створені набори інструментів, які «збирають» вебвміст. Вебскрапери — це прикладний програмний інтерфейс для вилучення даних з вебсайту.

Існують методи, які деякі вебсайти використовують для запобігання вебскрапінгу. Наприклад, виявлення та заборона ботів від сканування (перегляду) своїх сторінок. У відповідь на це існують вебскрапінгові системи, які спираються на використання методів аналізу об'єктної моделі документа, комп'ютерного бачення та обробку тексту природною мовою, щоб імітувати пошук людини, щоб дозволити збирати вміст вебсторінок для автономного синтаксичного аналізу.

Для отримання відповіді від сайту планується використати HTTP запит GET. В такому випадку веб-ресурс надасть потрібні дані для обробки.

HTTP визначає безліч методів запиту , які вказують, яке бажане дію виконається даного ресурсу. Незважаючи на те, що їхні назви можуть бути іменниками, ці методи запиту іноді називаються дієсловами HTTP . Кожен реалізує свою семантику, але кожна група команд поділяє загальні властивості: так, методи можуть бути безпечними, ідемпотентними або кешуються .

Метод GET просить подання ресурсу. Запити з цього методу можуть лише витягувати дані.

Для обробки строкових даних в багатьох випадках використовуватимуться регулярні вирази. Це дозволяє швидко та ефективно перевірити та обробити потрібну інформацію

Регулярний вираз — це рядок, який визначає шаблон пошуку підрядків у тексті. Одному шаблону може відповідати багато різних рядків. Термін «Регулярні вирази є перекладом англійського словосполучення «Regular expressions». Регулярний вираз, або коротко «регулярка»,

складається із звичайних символів та спеціальних командних послідовностей. Наприклад, \d задає будь-яку цифру, а \d+ задає будь-яку послідовність з однієї або більше цифр. Робота з регулярками реалізована у всіх сучасних мовах програмування. Однак існує кілька «діалектів», тому функціонал регулярних виразів може відрізнятись від мови до мови.

Зараз регулярні вирази використовуються багатьма текстовими редакторами та утилітами для пошуку та зміни тексту на основі вибраних правил. Багато мов програмування вже підтримують регулярні вирази для роботи з рядками. Наприклад, Perl і Tcl мають вбудований у їхній синтаксис механізм обробки регулярних виразів. Набір утиліт (включаючи редактор sed та фільтр grep), що постачаються в дистрибутивах Unix, одним із перших сприяв популяризації поняття регулярних виразів.

Регулярні вирази використовуються для стисненого опису деякої множини рядків за допомогою шаблонів, без необхідності перерахування всіх елементів цієї множини. При складанні шаблонів використовується спеціальний синтаксис, який зазвичай підтримує наступні операції:

1. Перелік Вертикальна характеристика поділяє допустимі варіанти. Наприклад, "gray|grey" відповідає gray або grey.

2. Угруповання Круглі дужки використовуються визначення області дії і пріоритету операторів. Наприклад, «gray|grey» і «gr(a|e)y» є різними зразками, але обидва описують безліч, що містить gray і grey.

3. Квантифікація. Квантифікатор після символу або групи визначає скільки разів попереднє вираз може зустрічатися.

a. {n, m}: загальний вираз повторень може бути від n до m включно.

b. {n,}: загальний вираз, n і більше повторень.

c. {,m}: загальне вираження, трохи більше m повторень.

d. {n}: Загальний вираз, рівно n повторень

e. ?: знак питання означає 0 або 1 раз, те саме, що і {0,1}. Наприклад, "colou?r" відповідає і color, і colour.

f. *: зірочка означає 0, 1 або будь-яку кількість разів ({0,}). Наприклад, «go*gle» відповідає gogle, google, goooogle, ggle та ін.

g. +: Плюс означає хоча б один раз ({1,}). Наприклад, go+gle відповідає gogle, google і т. д. (але не ggle)

Також планується, що даний додаток відслідковуватиме HTTP-статуси посилань. Перевірка посилань є невід'ємною і важливою частиною аналізу пошукової оптимізаці. Таким чином, ми можемо впевнитись, що немає пошкожених посилань, посилань на сторінки, які є невалідними, або не можуть бути відкритими. Далі розберем які види HTTP-статусів існують.

Код стану HTTP (HTTP status code) — частина першого рядка відповіді сервера при запитах за протоколом HTTP. Він є цілим число з трьох арабських цифр. Перша цифра вказує на клас стану. За кодом відповіді зазвичай йде відділена пробілом пояснювальна фраза англійською мовою, яка пояснює людині причину саме такої відповіді [5].

Виділяють 5 основних тіпив кодів стану HTTP:

- 1xx Інформаційні. Запит отримано, процес продовжується. Цей клас кодів стану вказує попередню відповідь, що складається лише з статусного рядку і опціональних хедерів, і закінчується пустим рядком. Оскільки стандарт HTTP/1.0 не описує ніяких кодів 1xx, сервер не обов'язково має висилати відповідь формату 1xx клієнту HTTP/1.0.
- 2xx Успішні операції. Цей клас статус кодів вказує на те, що клієнтський запит був отриманий, зрозумілий сервером, прийнятий і успішно оброблений.
- 3xx Перенаправлення. Клієнт повинен вжити додаткових заходів для виконання запиту. Цей вид кодів вказує, що клієнт має виконати додаткові дії для завершення виконання запиту. Клієнт може виконати конкретні дії без взаємодії з користувачем, у випадках коли метод, що використовується в другому запиті є GET або HEAD. Клієнтська програма не повинна перенаправляти запит

більш ніж у п'ять разів, так як такі переадресації зазвичай призводять до нескінченного циклу.

- 4xx Клієнтська помилка. Клас статус кодів 4xx призначений для випадків, в яких клієнт робить неправильні запити. За винятком відповіді на запит HEAD, сервер повинен включити у відповідь пояснення щодо помилкової ситуації, і чи є ця помилка тимчасовою або постійною. Ці коди стану застосовуються до будь-якого методу запиту.
- 5xx Серверна помилка. Серверу не вдалося виконати запит. Статус коди, що починаються з цифри «5» відносяться до випадків, в яких сервер ідентифікує, що сталася помилка або він по якійсь причині не в змозі виконати запит. Сервер повинен вислати інформацію, що містить пояснення помилкової ситуації, і вказати, чи є це тимчасовим або постійним, за винятком коли він отримував запит в типу HEAD. Зі свого боку, клієнтський браузер повинен повідомляти про будь-яку помилку користувача. Ці коди стану мають місце до будь-якого методу запиту [5].

Висновки до розділу

У другому розділі описано основні задачі які повинен вирішувати розроблений в рамках кваліфікаційної роботи додаток. Досліджено предметну область, виконано постановку задачі, описано методи та підходи до вирішення поставленої задачі.

РОЗДІЛ 3
РОЗРОБКА ДОДАТКУ ДЛЯ АНАЛІЗУ МЕТА-ДАНИХ САЙТУ

3.1 Опис засобів реалізації

Додаток було розроблено за допомогою мови програмування Python.

Python — інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Для частини додатку що реалізує скрапінг було використано потужну бібліотеку мови Пайтон – BeautifulSoup (bs4).

BeautifulSoup4 (bs4) - це бібліотека Python для отримання даних з файлів HTML і XML. Для природної навігації, пошуку та зміни дерева HTML модуль BeautifulSoup4 за замовчуванням використовує вбудований в Python парсер html.parser. BS4 також підтримує ряд сторонніх парсерів Python, таких як lxml, html5lib і xml (для аналізу XML-документів).

Бібліотека requests була використана для створення і надсилання HTTP запитів. Бібліотека requests є стандартом де-факто для створення HTTP-запитів у Python. Він абстрагує складність створення запитів за красивим простим API, щоб ви могли зосередитися на взаємодії зі службами та споживанні даних у своїй програмі.

Функціонал бібіліотеки:

- Keep-Alive та пул з'єднань

- Міжнародні домени та URL-адреси
- Сеанси із збереженням файлів cookie
- Перевірка TLS/SSL у стилі браузера
- Базова та коротка автентифікація
- Знайомі dictфайли cookie
- Автоматична декомпресія та декодування контенту
- Завантаження файлів із кількох частин
- Підтримка SOCKS-проксі
- Тайм-аути підключення
- Поточні завантаження
- Автоматичне дотримання.
- Фрагментовані запити HTTP

Бібліотека Certifi надає ретельно підібрану колекцію кореневих сертифікатів для перевірки надійності сертифікатів SSL під час перевірки ідентичності хостів TLS. Його було вилучено з проекту Requests .

Для створення виводу результатів програми використовується бібліотека Jinja2.

Jinja — це швидкий, виразний, розширюваний механізм створення шаблонів. Спеціальні заповнювачі в шаблоні дозволяють писати код, схожий на синтаксис Python. Потім шаблону передаються дані для візуалізації остаточного документа.

Це включає:

- Спадкування та включення шаблонів.
- Визначайте та імпортуйте макроси в шаблонах.
- Шаблони HTML можуть використовувати автоматичне екранування, щоб запобігти XSS від ненадійного введення користувача.
- Ізольоване середовище може безпечно відтворювати ненадійні шаблони.

- Асинхронна підтримка для створення шаблонів, які автоматично обробляють синхронізацію та асинхронні функції без додаткового синтаксису.
- Підтримка I18N з Babel.
- Шаблони компілюються в оптимізований код Python миттєво та кешуються або можуть бути скомпільовані заздалегідь.
- Винятки вказують на правильний рядок у шаблонах, щоб полегшити налагодження.
- Розширювані фільтри, тести, функції та навіть синтаксис.

Філософія Jinja полягає в тому, що хоча логіка програми належить до Python, якщо це можливо, вона не повинна ускладнювати роботу розробника шаблонів, надто сильно обмежуючи функціональність.

Для обробки XML і HTML в додаток імпортовано бібліотеку lxml.

lxml — це найбільш багатофункціональна та проста у використанні бібліотека для обробки XML і HTML мовою Python.

Існує безліч готових XML-парсерів, але для досягнення кращих результатів розробники іноді воліють писати свої власні XML- та HTML-парсери. Саме тоді в гру входить бібліотека lxml. Основні переваги цієї бібліотеки полягають у тому, що вона проста у використанні, надзвичайно швидка при розборі великих документів, дуже добре документована та забезпечує легке перетворення даних у типи даних Python, що призводить до більш легкого маніпулювання файлами

Набір інструментів XML lxml є прив’язкою Pythonic для бібліотек C libxml2 і libxslt . Він унікальний тим, що він поєднує швидкість і повноту функцій XML цих бібліотек із простотою рідного API Python, здебільшого сумісного, але кращого за добре відомий API ElementTree. Він значно розширює API ElementTree, забезпечуючи підтримку XPath, RelaxNG, XML Schema, XSLT, C14N та багато іншого.

urllib3 — потужний та зручний HTTP-клієнт для Python. Значна частина екосистеми Python вже використовує urllib3, і ви також повинні це зробити. urllib3 містить багато критичних функцій, яких немає в стандартних бібліотеках Python:

- Об'єднання з'єднань.
- Перевірка SSL/TLS на стороні клієнта.
- Завантаження файлів із багатокомпонентним кодуванням.
- Помічники для повторних запитів і роботи з перенаправленнями HTTP.
- Підтримка кодування gzip, deflate і brotli.
- Підтримка проксі для HTTP і SOCKS.
- 100% покриття тестом.
- urllib3 потужний і простий у використанні/

Бібліотека для роботи з регулярними виразами – re.

Регулярні вирази (Regular expressions) або RegEx – це послідовності символів, що задають шаблони для пошуку або заміни потрібного фрагмента тексту в рядку або файлі. Простіше кажучи, це свого роду крихітна мова програмування, що надає безліч інструментів для пошуку, заміни та отримання певних фрагментів тексту. Наприклад, можна швидко знайти в тексті адреси електронної пошти або телефонні номери. Регулярні висловлювання підтримуються більшістю сучасних мов програмування, в яких представлені різні за зручністю та функціоналом кошти. У цій статті ми розповімо про основні особливості застосування RegEx в мові Python.

- match- шукає послідовність на початку рядка
- search- шукає перший збіг із шаблоном
- findall- Шукає всі збіги з шаблоном. Повертає результуючі рядки у вигляді списку
- finditer- Шукає всі збіги з шаблоном. Повертає ітератор

- compile- компілює регулярне вираження. До цього об'єкта потім можна застосовувати всі перелічені функції
- fullmatch- весь рядок повинен відповідати описаному регулярному виразу

Крім функцій пошуку збігів, у модулі є такі функции:

- re.sub- для заміни у рядках
- re.split- для поділу рядка на частини

Також в додатку використана технологія віртуалізації Docker. Docker – це програмне забезпечення з відкритим кодом, найпопулярніша платформа для управління контейнерами.

Коли розробляється додаток, потрібно надати код разом з усіма його складовими, такими як бібліотеки, сервер, бази даних тощо. Та можна опинитися в ситуації, коли додаток працює на вашому комп'ютері, але відмовляється вмикатися та працювати на пристрої іншого користувача.

Ця проблема вирішується через створення незалежності програмного забезпечення від системи.

Docker розділяє ядро операційної системи на контейнери (Docker container), що працюють, як окремі процеси.

Він вирішує безліч завдань, пов'язаних зі створенням контейнерів, розміщенням в них додатків, управлінням процесами, а також тестуванням ПЗ і його окремих компонентів.

Він потрібен для більш ефективного використання системи і ресурсів, швидкого розгортання готових програмних продуктів, а також для їх масштабування і перенесення в інші середовища з гарантованим збереженням стабільної роботи.

Docker допомагає:

- мінімально використовувати ресурси;
- зручно приховати фонові процеси;
- просто масштабовувати;

- зменшити час між написанням і запуском коду;
- швидше тестувати;
- швидко розгортати;
- швидше створювати додатки.

Docker робить це за допомогою легкої платформи контейнерної віртуалізації.

У своєму ядрі docker дозволяє запускати практично будь-який додаток, безпечно ізольований в контейнері. Безпечна ізоляція дозволяє запускати на одному хості багато контейнерів одночасно.

Докер характеризується досить простим синтаксисом. Тому він досить простий для досвідчених IT-фахівців і для новачків. Програмне забезпечення сумісне з усіма версіями операційних систем Linux і Windows, тому сфера застосування Docker практично не обмежена.

Bootstrap - вільний набір інструментів для створення сайтів та веб-додатків . Включає HTML - і CSS шаблони оформлення для типографіки , веб форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсу, включаючи JavaScript розширення.

Bootstrap використовує сучасні напрацювання в області CSS та HTML , тому необхідно бути уважним за підтримки старих браузерів

Основні інструменти Bootstrap:

- Сітки - заздалегідь задані розміри колонок, які можна відразу використовувати, наприклад, ширина колонки 140 px відноситься до класу .span2 який можна використовувати в CSS описі документа.
- Шаблони – фіксований або гумовий шаблон документа.
- Типографіка - описи шрифтів , визначення деяких класів для шрифтів , таких як код, цитати і т.п.
- Медіа — надає певне керування зображеннями та відео.

- Таблиці — засоби оформлення таблиць, до додавання функціональності сортування.
- Форми - класи для оформлення форм та деяких подій, що відбуваються з ними.
- Навігація – класи оформлення для панелей, вкладок, переходу по сторінках, меню та панелі інструментів.
- Алерти - оформлення діалогових вікон, підказок і вікон, що спливають.

3.2 Опис програмної частини додатку

Розроблений програмний продукт має 2 модулі:

- UI
- Core

Розбиття на такі модулі добавляє гнучкості в розробці програмі та її подальшій підтримці програмістами. Поговоримо детальніше про кожен з модулів та що вони містять. Фактично це 2 окремі проекти, де модуль Core може бути використаним для створення інших інтерфейсів для даного аналізатора (наприклад консольний).

Core - модуль який містить в собі основну логіку додатку. Всі обчислення, фільтрація, групування, імпорт, генерація та виведення результатів програми. Також містить парсери сайтів.

UI - модуль для відображення даних, який має можливість малювати інтерфейс користувача та відображати дані, які прийшли з модуля Core.

Рисунок 3.1 – Схема залежності модулів додатку

На даному етапі розробки результатами роботи програми є наступні екранні форми:

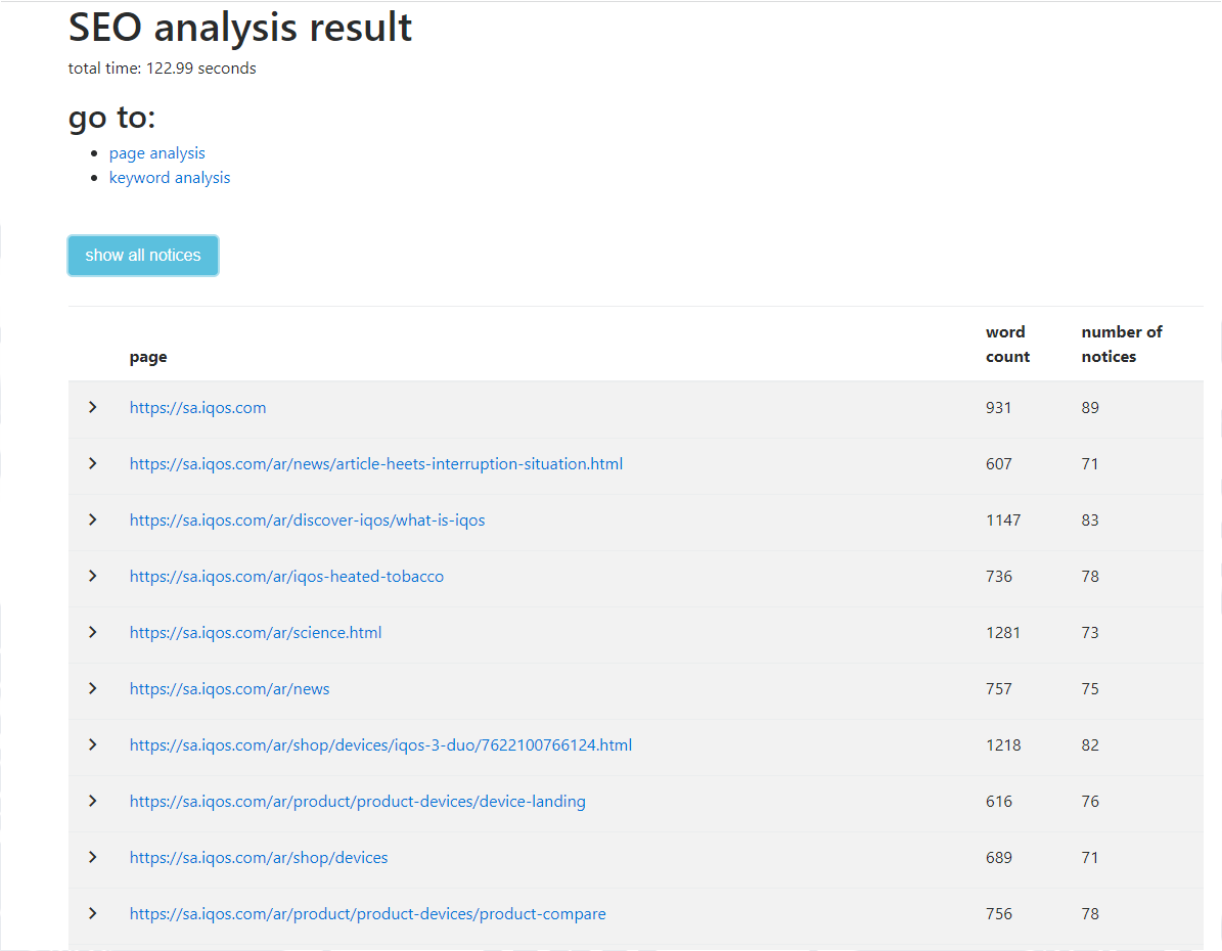

Рисунок 3.2 – Головний екран веб-додатку

На головному екрані додатку міститься акордеони з клікабельними посиланнями в якості заголовкові кожного з них. Посилання є обробленими додатком сторінками. Також виведене коротке меню навігації та кнопка за допомогою якою можна контролювати стан відразу усіх акордеонів (відчини або зачинити). Час обробки сторінок веб-сайту додатком склав 122.99 секунд. Всього оброблено 1 головна та 52 дочірні сторінки.

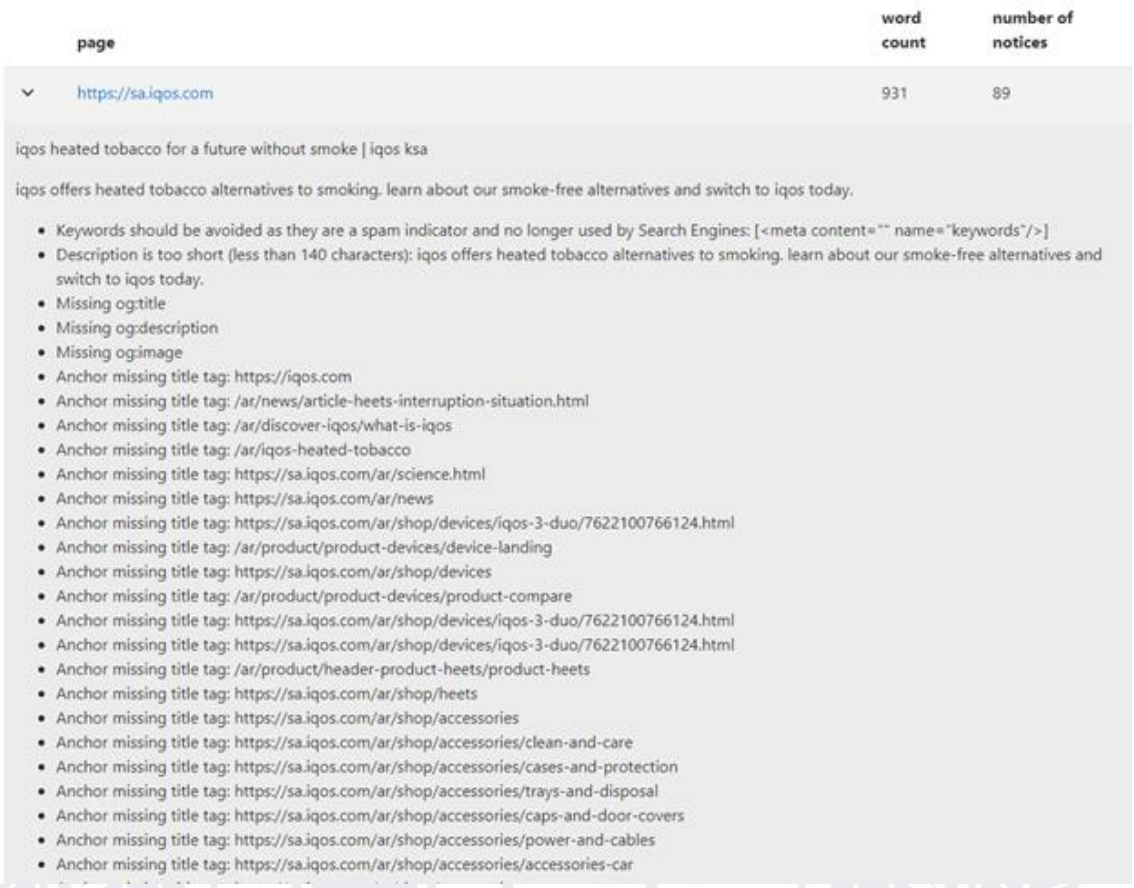

Рисунок 3.3 – Результати аналізу сторінки (1)

При розгортанні акордеону можна побачити результати аналізу даної сторінки. У цьому конкретному випадку ми бачимо, що у сторінки занадто короткий опис, пропущено деякі з мета-тегів та відсутні теги заголовків для дочірніх сторінок.

Поруч з посиланням на проаналізовану сторінку також виведено поле number of notice в якому підраховано кількість посилань та редіректів на дочірні сторінки та інші ресурси.

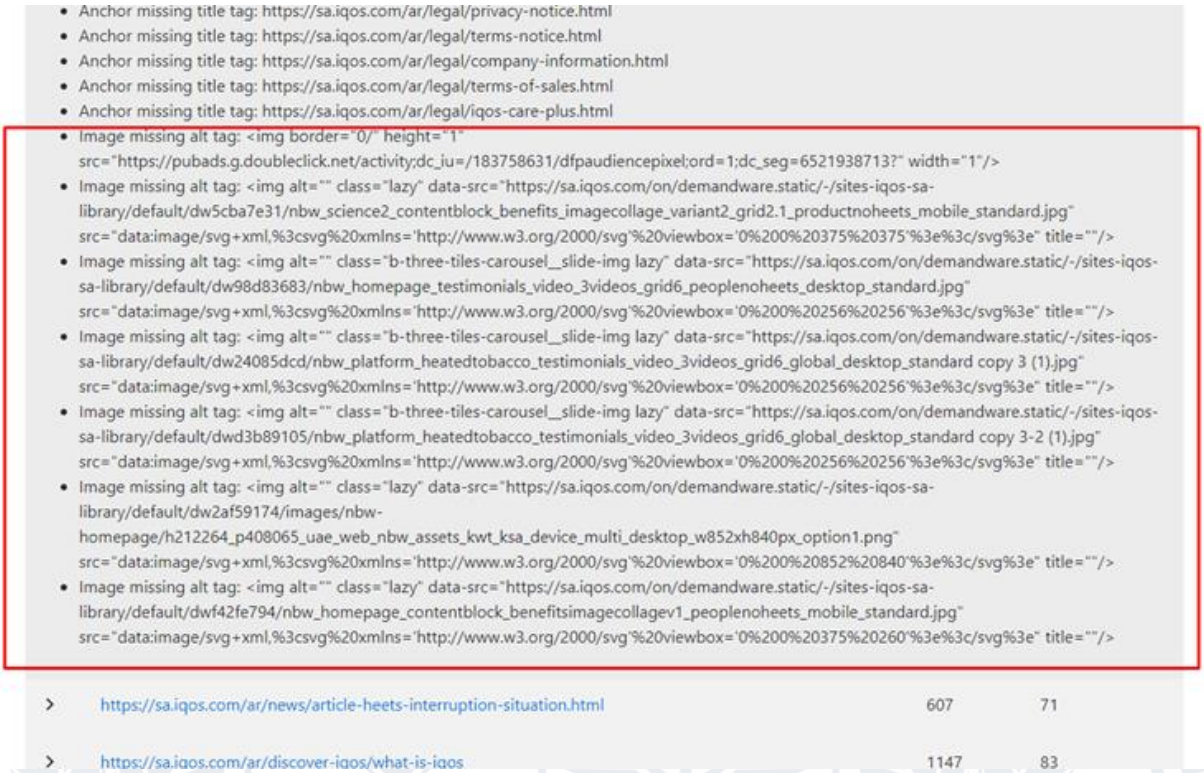

Рисунок 3.4 – Результати аналізу сторінки (2)

Також на цій сторінці знайдено ряд картинок в тегах яких є недоліки оформлення, а саме – пропущений альтернативний текст, який призначений для показу на місці картинки у разі, якщо сама картинка пошкоджена та не може завантажитись. Альтернативний текст є важливою складовою тегу img, так як у випадку, коли браузер не може завантажити картинку (наприклад, через її пошкодження, або слабке Інтернет-з’єднання користувача) повинене показуватись наповнення атрибуту alt, щоб користувач мав змогу зрозуміти яку інформацію несла картинка.

keyword analysis:

keywords	count
iqos	10559
من	3292
على	2940
heet	2053
us	1760
product	1624
device	1583
duo	1541
iqos duo	1541
care	1526
اكتشف	1470
التبغ	1366
inform	1215
العربية	1212

Рисунок 3.4 – Ключові слова веб-ресурсу

На рисунку вище виведено проаналізовані ключові слова сторінки. У стовбці keywords виведено у порядку спадання слова які найчастіше зустрічаються на сторінках веб-ресурсу. У стовбці count – частота зустрічання.

В основі додатку було створено такі класи - Page, Website та Http.

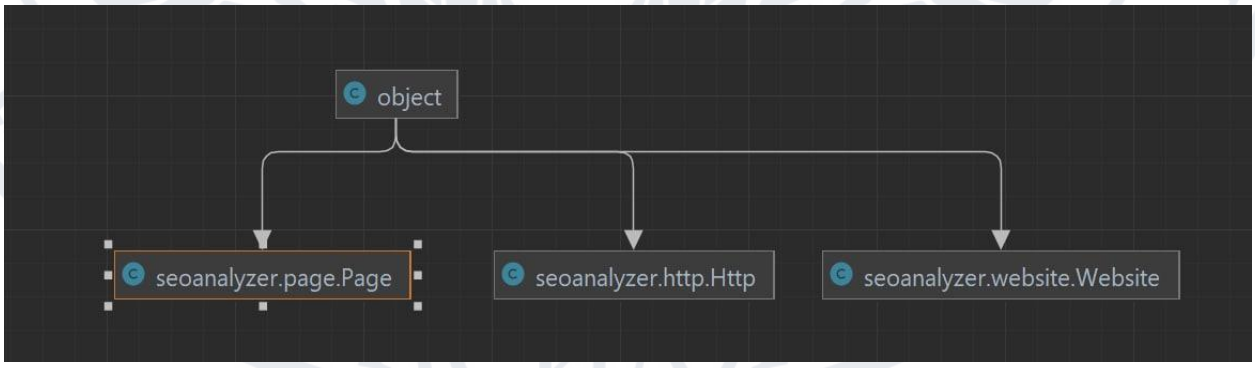

Рисунок 3.5 – Діаграма класів

Клас Http. В ньому відбувається надсилання GET запиту до веб-ресурсу з метою отримання у відповідь «сирих» необроблених даних, які будуть скрапитись та оброблятися в класі Page. Ознайомитись з реалізацією класу можна в додатку А.

Клас Website створено для обробки карти сайту (Sitemap.xml) та обходу дочірніх сторінок сайту. У випадку якщо користувач додатку завантажить Sitemap.xml функціонал цього класу обробить наданий файл, щоб отримати «правила» сканування основної та дочірніх сторінок. У випадку, якщо файл не був наданий, функціонал класу знайде усі дочірні сторінки та передасть на сканування класу Page. Ознайомитись з реалізацією класу можна в додатку Б.

Клас Page є основним класом обробки даних програми. В ньому відбувається збір та аналіз даних. За допомогою бібліотеки BeautifulSoup відбувається скрапінг даних з веб-ресурсу. В ньому обробляються такі дані як мета-теги, заголовки, картинки, відео, основні теги сторінки, атрибути тегів, посилання, ключові слова та ін. Ознайомитись з реалізацією класу можна в додатку В.

Методи та функції створених класів продемонстровано на класовій діаграмі нижче.

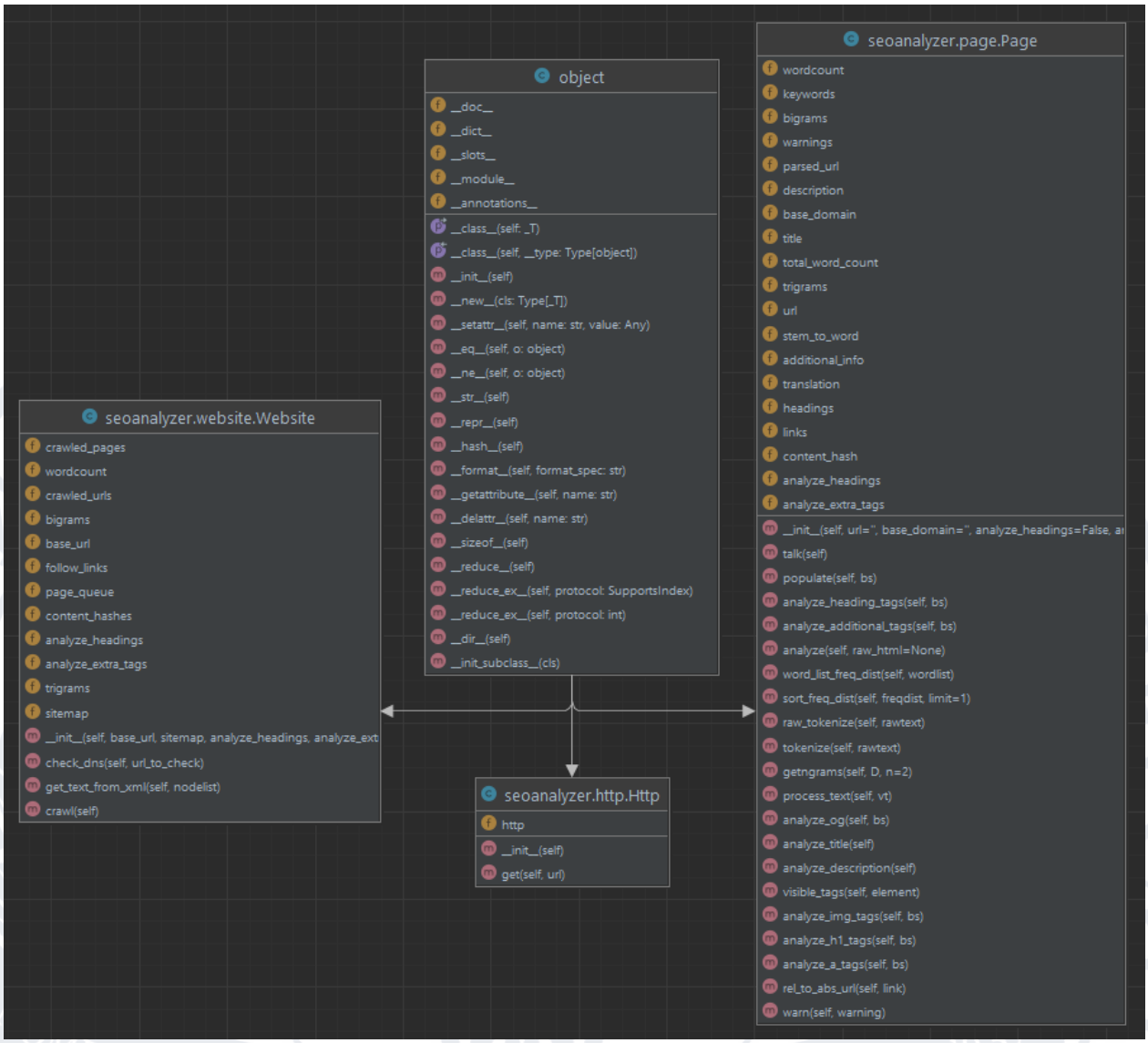

Рисунок 3.6 – Методи та функції класів

Основні функції та методи класу HTTP:

- __init__ - ініціалізація змінних класу;
- get – надсилання запиту GET до веб-ресурсу.

Основні функції та методи класу Website:

- __init__ - ініціалізація змінних класу;
- check_dns – робить запити до днс;
- get_text_from_xml – знаходить дочірні посилання від заданого;

- crawl – відпрацює у випадку отримання на вхід Sitemap.xml. Опрацює основне та дочірні посилання згідно правил заданих в карті сайту та передасть їх на сканування.

Основні функції та методи класу Page:

- __init__ - ініціалізація змінних класу;
- talk – повертає словник, який згодомо буде виведений в результатах роботи додатку. Словник містить усю основну оброблену інформацію. Посилання, ключові слова, попередження та ін.;
- populate – заповнення класових змінних даними зібрамими за допомогою BeautifulSoup;
- analyze_heading_tags – аналіз заголовків;
- analyze_additional_tags – аналіз інших тегів;
- analyze – основний клас аналізу де викликаються інші класові методи та заповнюють ворнінги;
- analyze_a_tags – аналіз посилань на сторінці;
- analyze_h1_tags – аналіз тегу заголовку;
- analyze_img_tags – аналіз тегів картинок;
- analyze_og – аналіз мета-тегів;
- rel_to_abs_url – перетворення відносних шляхів в абсолютні;
- stem – використовує стемінговий алгоритм Портера для того, щоб прибрати суфікси слів. Призначений для обробки ключових слів.

Функція main – збирає всі результати обробки класів та генерує html-файл.

Нижче наведена діаграма потуку даних у додатку. Як видно, з діаграми, першим кроком є скрапінг, за допомогоя кого ці дані збираються зі сторінки. Далі відбувається перевірка та обробка. Мета теги та інші теги сторінки перевіряються у більшості за допомогою регулярних виразів,

переборів та порівнянь. Ключові слова після збору обробляються алгоритмом Портера. За допомогою цього алгоритму і слів видаляються суфікси та закінчення, що дає змогу більш ефективно оцінити кількість входжень тих чи інших слів на сторінці. Посилання перевіряються шляхом отримання відповіді на запит. Основні відповіді, які моніторить програма:

- 200 – успішна операція;
- 300 – перенаправлення;
- 400 та 500 – помилка.

Рисунок 3.7 – Схема потоку даних

Додаток може бути запущений на будь-якій ОС сімейства лінукс, де встановлено python 3+. Мінімальні характеристики апаратного забезпечення, необхідного для запуску веб-додатку:

- Процесор – Intel Pentium та вище
- ОЗУ – 4Гб та більше
- Пам’ять – 10Гб та більше

Серед переваг розробленого додатку є те, що даний програмний продукт є проектом із відкритим кодом, тобто інші розробники можуть долучитись до його розвитку та удосконалення функціоналу. Також одним з плюсів застосунку є функціонал аналізу ключових слів зібраних зі сторінки сайту, це допомагає користувачу додатку отримати глибше розуміння фактору оцінки цінності змісту сторінки.

До недоліків додатку можна віднести мову програмування, якою він написаний, так як Python є повільнішим у своїй швидкодії у порівнянні з компільованими мовами програмування, таких як C# або C++.

3.3 Перспективи розвитку

Даний додаток має багато перспектив та горизонтів для розвитку. На поточному етапі повністю розроблено ядро додатку, яке виконує основні заплановані функції. Серед покращень та розширень планується написання серверу та прив’язка домену для вільного доступу користувачів мережі Інтернет.

До розширень функціоналу можна віднести додавання таких нових та важливих функції, як:

- Створення семантичного ядра сайту, що дозволить користувачу додатку легко зрозуміти над яким списком запитів варто провести роботи, щоб підняти сайт.
- Можливість управління окремими аспектами сайту безпосередньо з додатку.
- Візуалізація зібраних даних за допомою графіків та діаграм.
- Функціонал призначений для моніторингу трафіку клієнтів.
- SERP Location Checker

Висновки до розділу

Цей розділ присвячений реалізації програмного додатку для аналізу мета-даних сайту згідно з поставленою задачею, що описана в другому розіділі.

Показано розроблені можливості програми та особливості архітектури та функціонування, розглянуто створені класи та методи розробленого застосунку. Наведено екранні форми, на яких показано результати функціонування програмного засобу.

Також, наведені перспективи подальшого розвитку додатку.

ВИСНОВКИ

В результаті роботи був реалізований веб-додаток для SEO веб-сторінок. Дана програма є корисним помічником у сфері аналізу та оптимізації веб-ресурсів, виявлення недоліків сайту з боку наповнення та підняття сайту в рейтингу пошукових браузерів, дозволяючи розробникам, SEO-спеціалістам та власникам сайтів заощаджувати час та збільшувати потік клієнтів.

Для реалізації даного завдання були використані різні модулі та бібліотеки серед яких BeautifulSoup4, Bootstrap, urllib3, lxml, Jinja, Certifi, requests та ін, що допомогли реалізувати функціонал даного додатку.

Створені основні класи:

- Клас Http, в якому відбувається надсилання GET запиту до веб-ресурсу з метою отримання у відповідь «сирих» необроблених даних;
- Клас Website створено для обробки карти сайту (Sitemap.xml) та обходу дочірніх сторінок сайту;
- Клас Page для основним маніпуляцій та обробки даних програми, в якому відбувається збір та аналіз даних. В ньому обробляються такі дані як мета-теги, заголовки, картинки, відео, основні теги сторінки, атрибути тегів, посилання, ключові слова та ін.

У майбутньому планується розширити функціонал додатку, шляхом додавання та розширення існуючих функцій. Серед можливого нового функціоналу: додавання аналізу семантичного ядра сайту, SERP Location Checker додавання графіків, оптимізація та покращення клієнтського інтерфейсу та ін.

ЛІТЕРАТУРА

1. Пошукова оптимізація. URL: https://zz.te.ua/poshukova-optymizatsiia-shcho-tse-dlia-choho-potribno/ (дата звернення: 09.09.2022).
2. Як працює індексація. URL: https://www.webfx.com/blog/seo/what-is-google-indexing/ (дата звернення: 15.10.2022).
3. Hreflang. URL: https://seranking.com/ru/blog/atribut-hreflang/ (дата звернення: 11.09.2022).
4. Розширення для браузера. URL: https://evergreens.com.ua/ru/articles/browser-extensions.html (дата звернення: 05.11.2022).
5. Коди станів HTTP браузера. URL: https://restapitutorial.ru/httpstatuscodes.html (дата звернення: 18.10.2022).
6. Посилання. URL: https://www.computerhope.com/jargon/h/hyperlink.htm (дата звернення: 18.10.2022).
7. Redirect URL: https://yoast.com/what-is-a-redirect/ (дата звернення: 19.10.2022).
8. Wikipedia URL: https://en.wikipedia.org/wiki/Comma-separated_values (дата звернення: 21.10.2022).
9. D. Jones. Entity SEO: Moving from Strings to Things. DHJ Ventures 2021. 72 с.
10. E. Enge, S.n Spencer, J. C. Stricchiola. The Art Of SEO: Mastering Search Engine Optimization. O'Reilly Media; 3-й випуск 2015. 994 с.
11. M. Capala**.** The Psychology Of A Website: Mastering Cognitive Biases, Conversion Triggers And Modern SEO To Achieve Massive Results. 2021. 202 с.
12. S. DeGeyter The Best Damn Website & Ecommerce Marketing And Optimization Guide, Period. Independently published 2021. 141 с.

13. K. Azarenko**.** Ecommerce SEO Mastery: 10 Huge SEO Wins For Any Online Store .

14. E. Schwartz. Product-Led SEO: The Why Behind Building Your Organic Growth Strategy. Houndstooth Press 2021. 264 с.

15. R. Bryan. Local SEO Secrets: 20 Local SEO Strategies You Should be Using NOW. 2021. 276 с.

16. E. Schmidt, J. Rosenberg. How Google Works. Grand Central Publishing 2017. 320 с.

17. J. Jantsch, P. Singleton. SEO for Growth: The Ultimate Guide for Marketers, Web Designers & Entrepreneurs. SEO for Growth; 1st edition 2016. 238 с.

18. B. Clay. Search Engine Optimization All-in-One For Dummies 3rd Edition. For Dummies 2015. 800 с.

19. What Is SEO. URL: https://searchengineland.com/guide/what-is-seo (дата звернення: 10.10.2022).

20. Search Engine Optimization URL: https://mailchimp.com/marketing-glossary/seo/ (дата звернення: 16.10.2022).

21. Learn Search Optimization Best Practices – Moz. URL: https://moz.com/learn/seo/what-is-seo (дата звернення: 10.10.2022).

22. Beginner's Guide to SEO. URL: https://moz.com/beginners-guide-to-seo (дата звернення: 16.10.2022).

23. Do you need an SEO? URL: https://developers.google.com/search/docs/fundamentals/do-i-need-seo (дата звернення: 17.10.2022).

24. SEO Services. URL: https://develux.com/seo-services (дата звернення: 18.10.2022).

25. The 23 Best Google Chrome Extensions for SEO. URL: The 23 Best Google Chrome Extensions for SEO (дата звернення: 10.10.2022).

26. What Is Google Indexing and How Does It Work? URL: https://www.webfx.com/blog/seo/what-is-google-indexing/ (дата звернення: 12.11.2022).

27. Curated SEO Tools Best SEO Tools For Marketers. URL: https://curatedseotools.com/ (дата звернення: 12.11.2022).

28. A smarter way to do SEO. URL: https://moz.com/ (дата звернення: 12.11.2022).

29. Search Console. https://delante.co/definitions/search-console/ (дата звернення: 08.11.2022).

30. 18 BEST SEO TOOLS THAT SEO EXPERTS ACTUALLY USE IN 2022. URL: https://www.oberlo.com/blog/seo-tools (дата звернення: 13.10.2022).

31. Wellcome to PYTHON. URL: https://www.python.org/ (дата звернення: 13.10.2022).

32. Beautiful Soup Documentation. URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (дата звернення: 13.10.2022).

33. Requests. URL: https://pypi.org/project/requests/ (дата звернення: 14.10.2022).

34. Python Requests Module. URL: w3schools.com/python/module_requests.asp (дата звернення: 14.10.2022).

35. Regular expression operations. URL: https://docs.python.org/3/library/re.html (дата звернення: 16.10.2022).

36. Regular expression. URL: https://regex101.com/ (дата звернення: 14.10.2022).

37. urllib3. URL: https://pypi.org/project/urllib3/ (дата звернення: 15.10.2022).

38. lxml - XML and HTML with Python. URL: https://lxml.de/ (дата звернення: 15.10.2022).

39. Jinja. URL: https://uk.wikipedia.org/wiki/Jinja (дата звернення: 16.10.2022).

40. Jinja URL: https://jinja.palletsprojects.com/en/3.1.x/ (дата звернення: 16.10.2022).

41. Certifi. URL: https://github.com/certifi/python-certifi (дата звернення: 16.10.2022).

42. Certifi. URL: https://pypi.org/project/certifi/ (дата звернення: 16.10.2022).

43. API. URL: https://habr.com/ru/post/464261/ (дата звернення: 14.10.2022).

44. Bootstrap. URL: https://getbootstrap.com/docs/4.6/getting-started/introduction/ (дата звернення: 17.10.2022).

45. Bootstrap. URL: https://getbootstrap.com/docs/5.1/getting-started/introduction/ (дата звернення: 17.10.2022).

46. Список кодів стану HTTP. URL: https://uk.wikipedia.org/wiki/HTTP (дата звернення: 17.10.2022).

47. REST API. URL: https://itvdn.com/ru/blog/article/rest-api-18 (дата звернення: 18.10.2022).

48. Using HTTP Methods for RESTful Services. URL: https://www.restapitutorial.com/lessons/httpmethods.html (дата звернення: 19.10.2022).

49. L. Odden. Optimize: How to Attract and Engage More Customers by Integrating SEO, Social Media, and Content Marketing. Gildan Audio and Blackstone Publishing 2021

50. B. Bailyn, E. Bailyn. SEO Made Easy: Everything You Need to Know About SEO and Nothing More 1st Edition. Que Publishing 2013. 256 с.

51. Січко Т. В. Метод ранжування на інформаційно-довідкових сайтах. Вісник Хмельницького національного університету. Серія: Технічні науки, том 2. С. 45-51

ДОДАТОК А

Клас HTTP для надсилання GET запиту до веб-ресурсу

```
class Http():
    def __init__(self):
        user_agent = {'User-Agent': 'Mozilla/5.0'}
        self.http = PoolManager(
            timeout=Timeout(connect=1.0, read=2.0),
            cert_reqs='CERT_REQUIRED',
            ca_certs=certifi.where(),
            headers=user_agent
        )

    def get(self, url):
        return self.http.request('GET', url)

http = Http()
```

ДОДАТОК Б

Клас Website для Sitemap.xml та обходу дочірніх сторінок сайту

```
class Website():
    def __init__(self, base_url, sitemap, analyze_headings, analyze_extra_tags,
follow_links):
        self.base_url = base_url
        self.sitemap = sitemap
        self.analyze_headings = analyze_headings
        self.analyze_extra_tags = analyze_extra_tags
        self.follow_links = follow_links
        self.crawled_pages = []
        self.crawled_urls = set([])
        self.page_queue = []
        self.wordcount = Counter()
        self.bigrams = Counter()
        self.trigrams = Counter()
        self.content_hashes = defaultdict(set)

    def check_dns(self, url_to_check):
        try:
            o = urlsplit(url_to_check)
            socket.gethostbyname(o.hostname)
            return True
        except:
            pass

        return False

    def get_text_from_xml(self, nodelist):
        """
        Stolen from the minidom documentation
        """
        rc = []

        for node in nodelist:
            if node.nodeType == node.TEXT_NODE:
                rc.append(node.data)

        return ''.join(rc)

    def crawl(self):
        if self.sitemap:
            page = http.get(self.sitemap)
            if self.sitemap.endswith('xml'):
                xmldoc = minidom.parseString(page.data.decode('utf-8'))
                sitemap_urls = xmldoc.getElementsByTagName('loc')
                for url in sitemap_urls:
```

```
self.page_queue.append(self.get_text_from_xml(url.childNodes))
            elif self.sitemap.endswith('txt'):
                sitemap_urls = page.data.decode('utf-8').split('\n')
                for url in sitemap_urls:
                    self.page_queue.append(url)

        self.page_queue.append(self.base_url)

        for url in self.page_queue:
            if url in self.crawled_urls:
                continue

            page = Page(url=url, base_domain=self.base_url,
                        analyze_headings=self.analyze_headings,
                        analyze_extra_tags=self.analyze_extra_tags)

            if page.parsed_url.netloc != page.base_domain.netloc:
                continue

            page.analyze()

            self.content_hashes[page.content_hash].add(page.url)

            for w in page.wordcount:
                self.wordcount[w] += page.wordcount[w]

            for b in page.bigrams:
                self.bigrams[b] += page.bigrams[b]

            for t in page.trigrams:
                self.trigrams[t] += page.trigrams[t]

            self.page_queue.extend(page.links)

            self.crawled_pages.append(page)
            self.crawled_urls.add(page.url)

            if not self.follow_links:
                break
```

ДОДАТОК В

Клас Page для збору даних та їх аналізу

```
TOKEN_REGEX = re.compile(r'(?u)\b\w\w+\b')

HEADING_TAGS_XPATHS = {
    'h1': '//h1',
    'h2': '//h2',
    'h3': '//h3',
    'h4': '//h4',
    'h5': '//h5',
    'h6': '//h6',
}

ADDITIONAL_TAGS_XPATHS = {
    'title': '//title/text()',
    'meta_desc': '//meta[@name="description"]/@content',
    'viewport': '//meta[@name="viewport"]/@content',
    'charset': '//meta[@charset]/@charset',
    'canonical': '//link[@rel="canonical"]/@href',
    'alt_href': '//link[@rel="alternate"]/@href',
    'alt_hreflang': '//link[@rel="alternate"]/@hreflang',
    'og_title': '//meta[@property="og:title"]/@content',
    'og_desc': '//meta[@property="og:description"]/@content',
    'og_url': '//meta[@property="og:url"]/@content',
    'og_image': '//meta[@property="og:image"]/@content'
}

IMAGE_EXTENSIONS = set(['.img', '.png', '.jpg', '.jpeg', '.gif', '.bmp', '.svg',
'.webp', '.avif',])


class Page():
    """
    Container for each page and the core analyzer.
    """

    def __init__(self, url='', base_domain='', analyze_headings=False,
analyze_extra_tags=False):
        """
        Variables go here, *not* outside of __init__
        """

        self.base_domain = urlsplit(base_domain)
        self.parsed_url = urlsplit(url)
        self.url = url
        self.analyze_headings = analyze_headings
        self.analyze_extra_tags = analyze_extra_tags
```

```
        self.title = ''
        self.description = ''
        self.keywords = {}
        self.warnings = []
        self.translation = bytes.maketrans(punctuation.encode('utf-8'), str(' ' *
len(punctuation)).encode('utf-8'))
        self.links = []
        self.total_word_count = 0
        self.wordcount = Counter()
        self.bigrams = Counter()
        self.trigrams = Counter()
        self.stem_to_word = {}
        self.content_hash = None

        if analyze_headings:
            self.headings = {}
        if analyze_extra_tags:
            self.additional_info = {}

    def talk(self):
        """
        Returns a dictionary that can be printed
        """

        context = {
            'url': self.url,
            'title': self.title,
            'description': self.description,
            'word_count': self.total_word_count,
            'keywords': self.sort_freq_dist(self.keywords, limit=5),
            'bigrams': self.bigrams,
            'trigrams': self.trigrams,
            'warnings': self.warnings,
            'content_hash': self.content_hash
        }

        if self.analyze_headings:
            context['headings'] = self.headings
        if self.analyze_extra_tags:
            context['additional_info'] = self.additional_info

        return context

    def populate(self, bs):
        """
        Populates the instance variables from BeautifulSoup
        """

        try:
            self.title = bs.title.text
```

```
            except AttributeError:
                self.title = 'No Title'

            descr = bs.findAll('meta', attrs={'name': 'description'})

            if len(descr) > 0:
                self.description = descr[0].get('content')

            keywords = bs.findAll('meta', attrs={'name': 'keywords'})

            if len(keywords) > 0:
                self.warn(f'Keywords should be avoided as they are a spam indicator
and no longer used by Search Engines: {keywords}')

        def analyze_heading_tags(self, bs):
            """
            Analyze the heading tags and populate the headings
            """

            try:
                dom = lh.fromstring(str(bs))
            except ValueError as _:
                dom = lh.fromstring(bs.encode('utf-8'))
            for tag, xpath in HEADING_TAGS_XPATHS.items():
                value = [heading.text_content() for heading in dom.xpath(xpath)]
                if value:
                    self.headings.update({tag: value})

        def analyze_additional_tags(self, bs):
            """
            Analyze additional tags and populate the additional info
            """

            try:
                dom = lh.fromstring(str(bs))
            except ValueError as _:
                dom = lh.fromstring(bs.encode('utf-8'))
            for tag, xpath in ADDITIONAL_TAGS_XPATHS.items():
                value = dom.xpath(xpath)
                if value:
                    self.additional_info.update({tag: value})

        def analyze(self, raw_html=None):
            """
            Analyze the page and populate the warnings list
            """

            if not raw_html:
                valid_prefixes = []
```

```
            # only allow http:// https:// and //
            for s in ['http://', 'https://', '//',]:
                valid_prefixes.append(self.url.startswith(s))

            if True not in valid_prefixes:
                self.warn(f'{self.url} does not appear to have a valid
protocol.')
                return

            if self.url.startswith('//'):
                self.url = f'{self.base_domain.scheme}:{self.url}'

            if self.parsed_url.netloc != self.base_domain.netloc:
                self.warn(f'{self.url} is not part of
{self.base_domain.netloc}.')
                return

            try:
                page = http.get(self.url)
            except HTTPError as e:
                self.warn(f'Returned {e}')
                return

            encoding = 'ascii'

            if 'content-type' in page.headers:
                encoding = page.headers['content-type'].split('charset=')[-1]

            if encoding.lower() not in ('text/html', 'text/plain', 'utf-8'):
                # there is no unicode function in Python3
                # try:
                #     raw_html = unicode(page.read(), encoding)
                # except:
                self.warn(f'Can not read {encoding}')
                return
            else:
                raw_html = page.data.decode('utf-8')

        self.content_hash = hashlib.sha1(raw_html.encode('utf-8')).hexdigest()

        # remove comments, they screw with BeautifulSoup
        clean_html = re.sub(r'<!--.*?-->', r'', raw_html, flags=re.DOTALL)

        soup_lower = BeautifulSoup(clean_html.lower(), 'html.parser')
#.encode('utf-8')
        soup_unmodified = BeautifulSoup(clean_html, 'html.parser') #.encode('utf-
8')

        texts = soup_lower.findAll(text=True)
        visible_text = [w for w in filter(self.visible_tags, texts)]
```

```
        self.process_text(visible_text)

        self.populate(soup_lower)

        self.analyze_title()
        self.analyze_description()
        self.analyze_og(soup_lower)
        self.analyze_a_tags(soup_unmodified)
        self.analyze_img_tags(soup_lower)
        self.analyze_h1_tags(soup_lower)

        if self.analyze_headings:
            self.analyze_heading_tags(soup_unmodified)
        if self.analyze_extra_tags:
            self.analyze_additional_tags(soup_unmodified)

        return True

    def word_list_freq_dist(self, wordlist):
        freq = [wordlist.count(w) for w in wordlist]
        return dict(zip(wordlist, freq))

    def sort_freq_dist(self, freqdist, limit=1):
        aux = [(freqdist[key], self.stem_to_word[key]) for key in freqdist if
freqdist[key] >= limit]
        aux.sort()
        aux.reverse()
        return aux

    def raw_tokenize(self, rawtext):
        return TOKEN_REGEX.findall(rawtext.lower())

    def tokenize(self, rawtext):
        return [word for word in TOKEN_REGEX.findall(rawtext.lower()) if word not
in ENGLISH_STOP_WORDS]

    def getngrams(self, D, n=2):
        return zip(*[D[i:] for i in range(n)])

    def process_text(self, vt):
        page_text = ''

        for element in vt:
            if element.strip():
                page_text += element.strip().lower() + u' '

        tokens = self.tokenize(page_text)
        raw_tokens = self.raw_tokenize(page_text)
        self.total_word_count = len(raw_tokens)
```

```
        bigrams = self.getngrams(raw_tokens, 2)

        for ng in bigrams:
            vt = ' '.join(ng)
            self.bigrams[vt] += 1

        trigrams = self.getngrams(raw_tokens, 3)

        for ng in trigrams:
            vt = ' '.join(ng)
            self.trigrams[vt] += 1

        freq_dist = self.word_list_freq_dist(tokens)

        for word in freq_dist:
            root = stem(word)
            cnt = freq_dist[word]

            if root not in self.stem_to_word:
                self.stem_to_word[root] = word

            if root in self.wordcount:
                self.wordcount[root] += cnt
            else:
                self.wordcount[root] = cnt

            if root in self.keywords:
                self.keywords[root] += cnt
            else:
                self.keywords[root] = cnt

    def analyze_og(self, bs):
        """
        Validate open graph tags
        """
        og_title = bs.findAll('meta', attrs={'property': 'og:title'})
        og_description = bs.findAll('meta', attrs={'property': 'og:description'})
        og_image = bs.findAll('meta', attrs={'property': 'og:image'})

        if len(og_title) == 0:
            self.warn(u'Missing og:title')

        if len(og_description) == 0:
            self.warn(u'Missing og:description')

        if len(og_image) == 0:
            self.warn(u'Missing og:image')

    def analyze_title(self):
```

```
        """
        Validate the title
        """

        # getting lazy, create a local variable so save having to
        # type self.x a billion times
        t = self.title

        # calculate the length of the title once
        length = len(t)

        if length == 0:
            self.warn(u'Missing title tag')
            return
        elif length < 10:
            self.warn(u'Title tag is too short (less than 10 characters):
{0}'.format(t))
        elif length > 70:
            self.warn(u'Title tag is too long (more than 70 characters):
{0}'.format(t))

    def analyze_description(self):
        """
        Validate the description
        """

        # getting lazy, create a local variable so save having to
        # type self.x a billion times
        d = self.description

        # calculate the length of the description once
        length = len(d)

        if length == 0:
            self.warn(u'Missing description')
            return
        elif length < 140:
            self.warn(u'Description is too short (less than 140 characters):
{0}'.format(d))
        elif length > 255:
            self.warn(u'Description is too long (more than 255 characters):
{0}'.format(d))

    def visible_tags(self, element):
        if element.parent.name in ['style', 'script', '[document]']:
            return False

        return True

    def analyze_img_tags(self, bs):
```

```
        """
        Verifies that each img has an alt and title
        """
        images = bs.find_all('img')

        for image in images:
            src = ''
            if 'src' in image:
                src = image['src']
            elif 'data-src' in image:
                src = image['data-src']
            else:
                src = image

            if len(image.get('alt', '')) == 0:
                self.warn('Image missing alt tag: {0}'.format(src))

    def analyze_h1_tags(self, bs):
        """
        Make sure each page has at least one H1 tag
        """
        htags = bs.find_all('h1')

        if len(htags) == 0:
            self.warn('Each page should have at least one h1 tag')

    def analyze_a_tags(self, bs):
        """
        Add any new links (that we didn't find in the sitemap)
        """
        anchors = bs.find_all('a', href=True)

        for tag in anchors:
            tag_href = tag['href']
            tag_text = tag.text.lower().strip()

            if len(tag.get('title', '')) == 0:
                self.warn('Anchor missing title tag: {0}'.format(tag_href))

            if tag_text in ['click here', 'page', 'article']:
                self.warn('Anchor text contains generic text:
{0}'.format(tag_text))

            if self.base_domain.netloc not in tag_href and ':' in tag_href:
                continue

            modified_url = self.rel_to_abs_url(tag_href)

            url_filename, url_file_extension = os.path.splitext(modified_url)
```

```
            # ignore links to images
            if url_file_extension in IMAGE_EXTENSIONS:
                continue

            # remove hash links to all urls
            if '#' in modified_url:
                modified_url = modified_url[:modified_url.rindex('#')]

            self.links.append(modified_url)

    def rel_to_abs_url(self, link):
        if ':' in link:
            return link

        relative_path = link
        domain = self.base_domain.netloc

        if domain[-1] == '/':
            domain = domain[:-1]

        if len(relative_path) > 0 and relative_path[0] == '?':
            if '?' in self.url:
                return f'{self.url[:self.url.index("?")]}{relative_path}'

            return f'{self.url}{relative_path}'

        if len(relative_path) > 0 and relative_path[0] != '/':
            relative_path = f'/{relative_path}'

        return f'{self.base_domain.scheme}://{domain}{relative_path}'

    def warn(self, warning):
        self.warnings.append(warning)
```

Додаток Г

Алгоритм Портера:

```
_step2list = {
              "ational": "ate",
              "tional": "tion",
              "enci": "ence",
              "anci": "ance",
              "izer": "ize",
              "bli": "ble",
              "alli": "al",
              "entli": "ent",
              "eli": "e",
              "ousli": "ous",
              "ization": "ize",
              "ation": "ate",
              "ator": "ate",
              "alism": "al",
              "iveness": "ive",
              "fulness": "ful",
              "ousness": "ous",
              "aliti": "al",
              "iviti": "ive",
              "biliti": "ble",
              "logi": "log",
              }

_step3list = {
              "icate": "ic",
              "ative": "",
              "alize": "al",
              "iciti": "ic",
              "ical": "ic",
              "ful": "",
              "ness": "",
              }


_cons = "[^aeiou]"
_vowel = "[aeiouy]"
_cons_seq = "[^aeiouy]+"
_vowel_seq = "[aeiou]+"

# m > 0
_mgr0 = re.compile("^(" + _cons_seq + ")?" + _vowel_seq + _cons_seq)
# m == 0
_meq1 = re.compile("^(" + _cons_seq + ")?" + _vowel_seq + _cons_seq + "(" +
_vowel_seq + ")?$")
# m > 1
```

```
_mgr1 = re.compile("^(" + _cons_seq + ")?" + _vowel_seq + _cons_seq + _vowel_seq
+ _cons_seq)
# vowel in stem
_s_v = re.compile("^(" + _cons_seq + ")?" + _vowel)
# ???
_c_v = re.compile("^" + _cons_seq + _vowel + "[^aeiouwxy]$")

# Patterns used in the rules

_ed_ing = re.compile("^(.*)(ed|ing)$")
_at_bl_iz = re.compile("(at|bl|iz)$")
_step1b = re.compile("([^aeiouylsz])\\1$")
_step2 =
re.compile("^(.+?)(ational|tional|enci|anci|izer|bli|alli|entli|eli|ousli|ization
|ation|ator|alism|iveness|fulness|ousness|aliti|iviti|biliti|logi)$")
_step3 = re.compile("^(.+?)(icate|ative|alize|iciti|ical|ful|ness)$")
_step4_1 =
re.compile("^(.+?)(al|ance|ence|er|ic|able|ible|ant|ement|ment|ent|ou|ism|ate|iti
|ous|ive|ize)$")
_step4_2 = re.compile("^(.+?)(s|t)(ion)$")
_step5 = re.compile("^(.+?)e$")

# Stemming function

def stem(w):
    """Uses the Porter stemming algorithm to remove suffixes from English
    words.

    >>> stem("fundamentally")
    "fundament"
    """

    if len(w) < 3: return w

    first_is_y = w[0] == "y"
    if first_is_y:
        w = "Y" + w[1:]

    # Step 1a
    if w.endswith("s"):
        if w.endswith("sses"):
            w = w[:-2]
        elif w.endswith("ies"):
            w = w[:-2]
        elif w[-2] != "s":
            w = w[:-1]

    # Step 1b

    if w.endswith("eed"):
```

```
    s = w[:-3]
    if _mgr0.match(s):
        w = w[:-1]
else:
    m = _ed_ing.match(w)
    if m:
        stem = m.group(1)
        if _s_v.match(stem):
            w = stem
            if _at_bl_iz.match(w):
                w += "e"
            elif _step1b.match(w):
                w = w[:-1]
            elif _c_v.match(w):
                w += "e"

# Step 1c

if w.endswith("y"):
    stem = w[:-1]
    if _s_v.match(stem):
        w = stem + "i"

# Step 2

m = _step2.match(w)
if m:
    stem = m.group(1)
    suffix = m.group(2)
    if _mgr0.match(stem):
        w = stem + _step2list[suffix]

# Step 3

m = _step3.match(w)
if m:
    stem = m.group(1)
    suffix = m.group(2)
    if _mgr0.match(stem):
        w = stem + _step3list[suffix]

# Step 4

m = _step4_1.match(w)
if m:
    stem = m.group(1)
    if _mgr1.match(stem):
        w = stem
else:
    m = _step4_2.match(w)
```

```
    if m:
        stem = m.group(1) + m.group(2)
        if _mgr1.match(stem):
            w = stem

# Step 5

m = _step5.match(w)
if m:
    stem = m.group(1)
    if _mgr1.match(stem) or (_meq1.match(stem) and not _c_v.match(stem)):
        w = stem

if w.endswith("ll") and _mgr1.match(w):
    w = w[:-1]

if first_is_y:
    w = "y" + w[1:]

return w
```

Лук’янчук Олександра В’ячеславівна
Прізвище, ім’я по батькові

Інформаційних і прикладних технологій
Факультет

122 Комп’ютерні науки
Шифр і назва спеціальності

Технології обробки даних (Data Science)
Освітня програма

ДЕКЛАРАЦІЯ АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ

Усвідомлюючи свою відповідальність за надання неправдивої інформації, стверджую, що подана кваліфікаційна (магістерська) робота на тему «Розробка веб-додатку для аналізу мета-даних сайту»
є написаною мною особисто.

Одночасно заявляю, що ця робота:

- не передавалась іншим особам і подається до захисту вперше;
- не порушує авторських та суміжних прав, закріплених статтями 21-25 Закону України «Про авторське право та суміжні права»;
- не отримувалась іншими особами, а також дані та інформація не отримувалась у недозволений спосіб.

Я усвідомлюю, що у разі порушення цього порядку моя кваліфікаційна робота буде відхилена без права її захисту, або під час захисту за неї буде поставлена оцінка «незадовільно».

(дата) (підпис здобувача освіти)