

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДУ
ПОШУКОВОЇ ОПТИМІЗАЦІЇ ВЕБРЕСУРСІВ.
RESEARCH AND SOFTWARE IMPLEMENTATION OF THE METHOD FOR
SEARCH ENGINE OPTIMIZATION OF WEB RESOURCES

Виконав(ла): студент(ка) 2 курсу, групи КНТ-212м

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Системи штучного інтелекту

РАДЧЕНКО І.А.

(ПРИЗВИЩЕ та ініціали)

Керівник ЗЕЛЕНЬОВА І.Я.

(ПРИЗВИЩЕ та ініціали)

Рецензент ПОДКОВАЛІХІНА О.О.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

РАДЧЕНКА Івана Андрійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методу пошукової оптимізації вебресурсів.

Research and Software Implementation of the Method for Search Engine Optimization of Web Resources

керівник проєкту (роботи) к.т.н., доцент, ЗЕЛЕНЬОВА Ірина Яківна,

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 14 ” листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Дослідження методів пошукової оптимізації. 3. Програмна реалізація методу вебоптимізації. 4. Тестування методу пошукової оптимізації.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ЗЕЛЕНЬОВА І.Я., доцент		
Нормоконтролер	БЄЛОВА А.В., асистент		

7. Дата видачі завдання “ 08 ” вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2-3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів пошукової оптимізації.	4-5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7-8 тижні	Розділ 3
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	10-11 тижні	Додатки
8	Нормоконтроль та рецензування.	12 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка)

(підпис) Іван РАДЧЕНКО
(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Ірина ЗЕЛЕНЬОВА
(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
72 с., 12 табл., 32 рис., 2 дод., 25 джерел.

ПОШУКОВА ОПТИМІЗАЦІЯ, СЕМАНТИЧНИЙ АНАЛІЗ,
СТАТИСТИКА КЛЮЧОВИХ СЛІВ, МЕТОДИ ОПТИМІЗАЦІЇ, ПРОГРАМНА
РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ МЕТОДУ, АНАЛІЗ, PYTHON, SEO, NLP.

Об'єкт дослідження – аналіз пошукової оптимізації вебресурсу.

Предмет дослідження – методи та засоби автоматизованого складання семантичного ядра вебресурсу.

Мета роботи – підвищення ефективності пошукової оптимізації вебресурсів на основі удосконалення методів та створення програмного забезпечення підтримки процесу оптимізації.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, мови програмування Python та JavaScript, фреймворк Flask, бібліотеки NLTK, BeautifulSoup, PostgreSQL.

Практичні результати. Розроблено програмне забезпечення підтримки процесу пошукової оптимізації вебресурсів на основі методів семантичного аналізу та статистичного аналізу ключових слів.

Наукова новизна. Удосконалено метод семантичного аналізу вебресурсів для автоматичного формування семантичного ядра; розроблено методику оцінювання ефективності застосування ключових слів для пошукової оптимізації на основі статистичного аналізу.

Галузь використання – SEO оптимізація вебсайтів.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 72 pages, 12 tables, 32 figures, 2 appendices, 25 sources.

SEARCH OPTIMIZATION, SEMANTIC ANALYSIS, KEYWORD STATISTICS, OPTIMIZATION METHODS, SOFTWARE IMPLEMENTATION, METHOD TESTING, ANALYSIS, PYTHON, SEO, NLP.

Object of study is the analysis of search engine optimization of the web resource.

Subject of study is the methods and means of automated compilation of the semantic core of a web resource.

The purpose of the work is to increase the effectiveness of search optimization of web resources based on the improvement of methods and the creation of software to support the optimization process.

Materials, methods and technical means: structural and object-oriented programming, Python and JavaScript programming languages, Flask framework, NLTK, BeautifulSoup, PostgreSQL libraries.

Practical results. Software to support the process of search optimization of web resources based on methods of semantic analysis and statistical analysis of keywords has been developed.

Scientific novelty. The method of semantic analysis of web resources for automatic formation of the semantic core has been improved; a methodology for evaluating the effectiveness of the use of keywords for search engine optimization based on statistical analysis has been developed.

The field of use is SEO optimization of websites.

ЗМІСТ

С.

Перелік скорочень та умовних познач	7
Вступ	8
1 Аналіз предметної області	11
1.1 Визначення пошукової оптимізації та її застосування	11
1.2 Алгоритми роботи пошукових систем	14
1.3 Артефакти оптимізації	17
1.4 Оптимізація сайту	19
1.5 Висновки за розділом 1	20
2 Дослідження методів пошукової оптимізації	22
2.1 Методи семантичної обробки	22
2.2 Метод статистики ключових слів	24
2.3 Застосування методів в сучасних засобах пошукової оптимізації	27
2.4 Висновки за розділом 2	30
3 Програмна реалізація методу вебоптимізації	31
3.1 Функціональні можливості програмного забезпечення	31
3.2 Вибір технологій розробки	33
3.3 Проектування програмного забезпечення	41
3.4 Програмна реалізація застосунку	42
3.5 Висновки за розділом 3	43
4 Тестування методу пошукової оптимізації	44
4.1 Результати роботи програмного застосунку	44
4.2 Аналіз роботи методу складання семантичного ядра	46
4.3 Висновки за розділом 4	48
Висновки	50
Перелік джерел посилення	51
Додаток А Лістинги програмної реалізації методів	55
Додаток Б Слайди презентації	64

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
CSS	– Cascading Style Sheets;
HTML	– HyperText Markup Language;
LSI	– Latent Semantic Indexing;
MVC	– Model-View-Controller;
NLP	– Natural Language Processing;
ORM	– Object-Relational Mapping;
SE	– Search Engine;
SEO	– Search Engine Optimization;
TF-IDF	– Term Frequency–Inverse Document Frequency;
XML	– Extensible Markup Language;
БД	– база даних;
ПЗ	– програмне забезпечення.

ВСТУП

На даний момент мережа Інтернет є найбільш доступним способом комунікації, а також отримання та обміну інформацією між людьми. Доступ до інформації здійснюється через різноманітні вебсайти, які можуть складатися з тисяч або навіть мільйонів електронних сторінок на різноманітну тематику. Основною проблемою кінцевого користувача є навігація серед такого величезного потоку інформації [1].

Для допомоги в орієнтуванні серед цього «океану» інформації були спеціально розроблені пошукові системи. Величезні сервери зберігають у собі інформацію про мільярди проіндексованих сторінок. Саме завдяки пошуковим системам ми знаходимо саме ту інформацію, яку шукали. Але далеко не завжди перший знайдений варіант задовольнятиме потребам користувачів. Вебсторінка повинна бути заповнена релевантною інформацією, щоб при запиті до пошукової системи вона видала користувачеві саме цю вебсторінку в результатах пошуку. Основною метою пошукової оптимізації якраз і є привернення уваги до цієї сторінки кінцевого користувача [2].

В даний час велику актуальність набуває можливість підвищити рейтинг сайту в таблиці результатів пошукових систем. Чим вищий рейтинг, тим більше буде кількість показів вебсайту серед результатів відповідного запиту. Підвищення рейтингу вебсайту дозволить збільшити увагу користувачів до нього [3].

Інтернет-простір невпинно розширюється, а кількість вебсайтів зростає експоненційно, перетворюючи Всесвітню мережу на безмежний ресурс інформації та знань. У цьому контексті, значення ефективної пошукової оптимізації (SEO) стає все більш актуальним. Завдяки SEO можна значно підвищити видимість сайту в пошукових системах, що важливо не лише для бізнес-сторінок, а й для освітніх ресурсів, блогів, інформаційних порталів та інших вебсайтів [3].

Розвиток технологій автоматизації та використання штучного інтелекту в пошуковій оптимізації відкриває нові горизонти для розробників вебсайтів. З використанням передових методик, таких як машинне навчання та семантичний аналіз, можна значно підвищити ефективність SEO. Це дозволить не тільки оптимізувати контент під ключові запити, але й пропонувати користувачам більш релевантний та цікавий контент [4].

У розробці програмного забезпечення для SEO важливим аспектом є інтеграція з різноманітними аналітичними інструментами. Це дозволяє не тільки аналізувати поточний стан сайту, але й слідкувати за змінами в алгоритмах пошукових систем, адаптуючи стратегії оптимізації під актуальні тренди [1].

З огляду на зростаючий попит на ефективні рішення для SEO, стає очевидною потреба в розробці додатків, які б забезпечували комплексний підхід до пошукової оптимізації. Такі додатки повинні включати інструменти для роботи з ключовими словами, аналізу конкурентів, оцінки щільності та релевантності контенту, а також надавати рекомендації щодо покращення позицій сайту в пошукових системах [2].

На налаштування повноцінної пошукової оптимізації для вебсайту йде досить багато часу, тоді як методи пошукової обробки змінюються і вимагають до себе нові підходи [1].

Мета роботи це розробка програмного застосунку для автоматизації процесу пошукової оптимізації вебресурсів на основі методів семантичного аналізу та статистичної обробки ключових слів.

Задачі, які треба вирішити:

- дослідити існуючі методи та засоби пошукової оптимізації вебресурсів;
- розробити удосконалений метод семантичного аналізу тексту для автоматичного формування семантичного ядра сайту;
- розробити методику оцінювання ефективності ключових слів на основі їх статистичного аналізу;

- створити програмний застосунок реалізації даних методів;
- провести тестування розробленого програмного забезпечення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Визначення пошукової оптимізації та її застосування

Пошукова оптимізація (від англ. Search Engine Optimization, найпоширеніша аббревіатура «SEO») – це спосіб адаптації структури вебсайту, шляхом зовнішніх і внутрішніх змін, заради отримання вищого рейтингу в пошукових видачах [5].

Під зовнішніми змінами розуміється ряд заходів щодо збільшення кількості Інтернет-ресурсів, що мають посилання на даний вебсайт.

Внутрішня оптимізація – це оптимізація, спрямована на пошук та зміну структури вебсайту, що відповідає вимогам пошукових систем [6].

До внутрішньої оптимізації можна віднести:

- покращення дизайну для залучення потенційних клієнтів;
- написання якісного та унікального контенту;
- складання семантичного ядра;
- робота з meta-тегами.

Розглянемо, чим важлива пошукова оптимізація. Можна виділити три основні способи відвідування сайту [7]:

- прямий трафік – перехід на вебсайт користувачами, що самостійно вводять адресу сайту в рядку браузера або використовують закладку у своєму браузері. На прямий трафік припадає близько 17% всіх переходів [1];

- реферальний трафік – перехід на вебсайт користувачами, які знайшли згадку та посилання на вебсайт на якомусь іншому сайті і перейшли по ній. На реферальний трафік припадає близько 12% переходів;

- пошуковий трафік – перехід на вебсайт користувачами, які знайшли його через запит у пошуковій системі за певними ключовими словами. На пошуковий трафік припадає 71% усіх переходів.

Розподіл основних способів відвідування вебсайтів візуально представлено у вигляді діаграми на рисунку 1.1.



Рисунок 1.1 – Способи відвідування вебсайту [2]

Відразу стає зрозуміло, що пошукові системи є найпоширенішим способом відвідування вебсайту.

Згідно з дослідженнями Google, приблизний розподіл можливості відвідування вебсайту користувачем, залежно від місця в пошуковій видачі, виглядає так (табл. 1.1) [2].

Таблиця 1.1 – Розподіл можливості відвідування вебсайту

Перша сторінка пошукової видачі		Друга сторінка пошукової видачі	
Номер позиції в пошуковій видачі	Відсоток відвідувачів вебсайту	Номер позиції в пошуковій видачі	Відсоток відвідувачів вебсайту
1	2530%	1	97%
2	1520%	2	75%
3	1012%	3	64%
4	89%	4	32%
5	48%	510	1% і менше
6	45%		
7			
8	34%		
9	24%		
10	35%		

Проаналізувавши дані з таблиці 1.1, можна з упевненістю сказати, що найуспішнішими є перші три позиції.

Основною задачею зовнішньої оптимізації є збільшення кількості посилань на поточний вебсайт, розташованих на різних Інтернет-ресурсах. Додавання посилань на вебсайт можливо платним чи безоплатним способами [4].

Зовнішня оптимізація відіграє немало роль для отримання високої позиції від пошукових систем. Зовнішня оптимізація впливає на такі важливі показники, як Page Rank, тематичний індекс цитування і зважений індекс цитування [5].

Page Rank (з англ. «ранг сторінки», скорочено PR) – це метод, що розроблений та використовується компанією «Google», який спрямований на вимірювання важливості певної сторінки вебсайту. PR визначає популярність сторінок, які посилаються на той чи інший ресурс, та їх кількість. Кожній сторінці в Інтернеті надається деяке початкове числове значення, яке буде змінюватися в залежності від кількості посилань на неї.

Зважений індекс цитування (ЗІЦ) – сукупність параметрів, що враховує популярність сайтів, що посилаються на певний ресурс [9].

Пошукову оптимізацію можна розділити на 3 основні типи:

- біла – оптимізація вебресурсу, при якій немає порушень правил пошукових систем;
- сіра – оптимізація вебресурсу, при якій допускаються незначні порушення правил пошукових систем, що не несе жодних наслідків (наприклад, використання неякісного контенту);
- чорна – оптимізація, основна мета якої обдурити пошукову систему заради вищої позиції в рейтингу. Однак алгоритми пошукових машин постійно покращуються, що допомагає швидше знаходити вебсайти, для яких було використано цей підхід. Наслідком використання чорної оптимізації може бути зниження вебсайту в загальному рейтингу пошукової системи або навіть виключення з нього.

1.2 Алгоритми роботи пошукових систем

На просторах мережі Інтернет знаходиться велика кількість сайтів з різною тематикою та різними адресами, запам'ятати та вивчити все просто неможливо. Ось тут і вступають у справу пошукові машини, а наскільки виявиться достовірний результат пошуків це вже залежатиме від правильної оптимізації кінцевого сайту [6].

Усі пошукові системи працюють на власних розроблених алгоритмах, реалізацію яких тримають у комерційній таємниці. Проте загальний принцип функціонування пошукових систем залишається приблизно однаковим (рис. 1.2).



Рисунок 1.2 – Алгоритм роботи пошукової системи [5]

Послідовність дій алгоритму приблизно така:

- спеціальні пошукові роботи, так звані боти, одержують з каталогів пошукової машини адреси сайтів [11];
- далі у справу вступає вже інший бот – краулер (від англ. crawler). Краулер знаходить усі посилання на сторінці, переглядає їх та заносить до бази даних. Для підтримки актуальності посилань краулер регулярно

перевіряє наявність нових посилань. Для пошуку посилань відводиться певний час, після якого краулер переходить до наступного вебсайту [12];

- кожна вебсторінка, яку знайшов краулер, буде оброблена вже іншим ботом – спайдером (англ. spider). Спайдер завантажить вихідний код сторінки та занесе його до бази;

- програма-індексатор (англ. indexer) обробляє зібрану інформацію та формує файли зворотнього індексу. Далі формується список всіх слів у тексті в алфавітному порядку та їх позиція у тексті. Алгоритм аналізу вмісту тесту залежить від конкретної системи пошуку [13].

Також зберігаються всі копії оригінальних сторінок – прямий індекс сторінок. З копій буде сформовано сніпети – блоки інформації про знайдений документ, які відображаються в результатах пошуку. До складу сніпету входять заголовок та опис документа. Також до складу сніпету може включатись додаткова інформація про сайт (рис. 1.3).

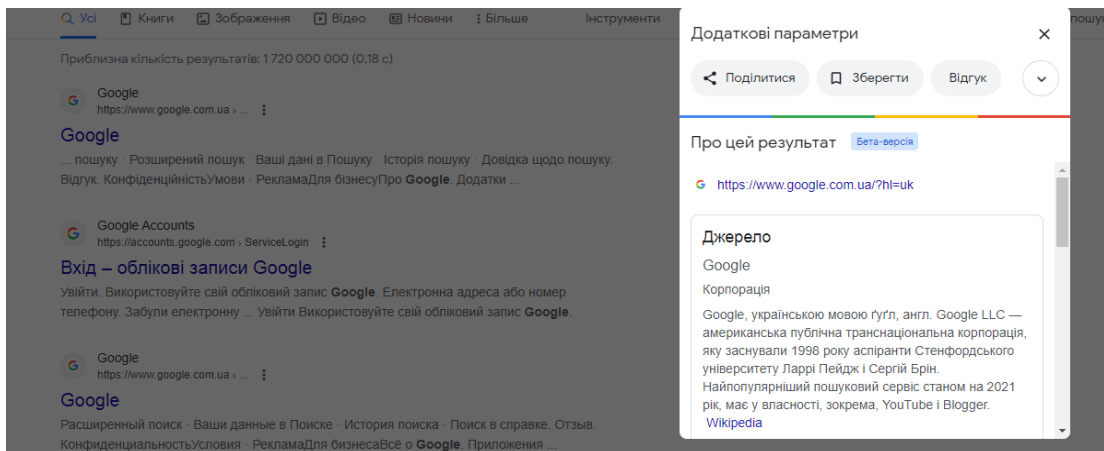


Рисунок 1.3 – Сніпет

Вся зібрана та оброблена інформація зберігається у базі даних, звідки вона виводиться користувачеві спеціальним програмним модулем, під назвою система видачі результатів (англ. search engine results engine) [2].

Прямий пошук із бази даних зворотніх індексів, що полягає у знаходження і виведення всіх наявних індексів, сам собою принесе малі результати, так як каталоги, що створені пошуковими машинами, дуже

великі за об'ємом. Для підвищення ефективності було прийнято рішення враховувати релевантність запиту списку сторінок, що відображаються в результатах пошуку. Далі відбувається ранжування (сортування) сторінки таким чином, що сторінка перебуватиме в пошуковій видачі тим вище (вага сторінки буде більшою), чим у ній буде більше відповідності запиту [4].

Якщо розглядати вагу сторінки векторно, потрібно перемножити частоту появи слова на сторінці і частоту появи слова в базі даних. Простіше кажучи, чим більше слово з пошукового запиту зустрічається на цій сторінці, і чим воно менше з'являється на інших проіндексованих сторінках, тим вага цієї сторінки буде більшою.

Взявши за основу цю формулу, пошукові машини складають власні алгоритми та виконують свою роботу в автоматичному режимі [5].

Поліпшення роботи алгоритму здійснюється з допомогою машинного навчання.

Суть машинного навчання зводиться до наступного. Команда спеціалістів-ассесорів (людей, задачею яких є оцінка релевантності результатів пошуку) роблять запити в пошуковій системі та потім здійснюють оцінку отриманих результатів. На основі цих результатів формуються аналітичні дані, що надходять до системи. Система аналізує ці дані та дані, що отримані самою машиною, та приймає рішення щодо вдосконалення алгоритму [3].

Однак, наскільки ідеально не були б написані ці алгоритми, все одно залишаються вебсайти, які не потрапляють до бази пошукових систем. До таких вебсайтів можна віднести:

- вебсайти, що ніяк не пов'язані з іншими вебсайтами (які не мають посилання на інші вебсайти і не мають вебсайти, що посилаються на нього);
- вебсайти, захищені паролями;
- вебсайти, що перебувають у розробці.

1.3 Артефакти оптимізації

Розглянемо основні елементи, які застосовуються для оптимізації вебсайту [5].

Теги HTML-контейнера <head>:

- тег <title> визначає заголовок документа. Текст, що міститься в цьому тегу, буде відображатися в заголовку вікна браузера, а також у пошуковій видачі [15];

- тег <meta> визначає метатеги, що використовуються для зберігання інформації, призначеної для браузерів та пошукових систем. За допомогою метатегів пошукові системи отримують опис сайтів, ключових слів та інші дані.

Тег <meta> має кілька атрибутів, але нас зараз цікавлять лише два [16]:

- ім'я метатега (атрибут «name»), що вказує на його призначення;
- атрибут «content» встановлює значення атрибута, заданого за допомогою «name».

Ми не розглядатимемо всі можливі значення атрибута «name», а зупинимося на трьох – «description», «keywords» та «robots».

Значення «description» є найпоширенішим та враховується всіма пошуковими машинами при індексації. Цей факт підвищує його важливість для просування та оптимізації. Атрибут «content» містить опис конкретної сторінки вебсайту. Розробники пошукових систем рекомендують обмежити довжину опису до 100 символів [3].

Значення «keywords», на відміну від «description», сприймають лише деякі пошукові системи, тоді як інші вважають його не обов'язковим. В атрибуті «content» вказується список ключових слів щодо теми певної сторінки вебсайту [4].

Значення «robots», як і «description», використовується більшістю пошукових машин та застосовується для інструкції з індексації. Атрибут «content» може приймати значення, наведені у таблиці 1.2.

Таблиця 1.2 – Значення атрибуту «content» для метатегу «robots»

Значення атрибуту	Пояснення
noindex	Забороняє індексування документа
nofollow	Забороняє прохід за посиланнями, що є у документі
index	Дозволяє індексування документа
follow	Дозволяє прохід за посиланнями
all	Рівносильно «index, follow»
none	Рівносильно «noindex, nofollow»

Якщо метатег відсутній, то це рівнозначно варіанту «all».

Визначення для оптимізації тексту на вебсторінці:

- положення тексту на сторінці – текст на сторінці має бути розміщений правильно і вписуватись у загальний дизайн вебсайту [4];
- основний текст – не менш важливим є складання грамотного тексту, у якому ключові слова розташовані ближче до початку тексту. Ключові слова повинні відповідати вмісту тексту і враховувати їх кількість і частоту появи в тексті [2];
- виділення тексту – пошукові системи вважають виділення тексту позначкою важливої інформації. До таких виділень можна віднести виділення напівжирним (тег), курсивом (тег) або просто підкреслення (тег <u>). Також необхідним буде виділення заголовків, тому що пошукові системи вважають наявність заголовків важливим і враховують це при ранжируванні. Мова HTML пропонує шість заголовків різного рівня <h1>, <h2>, <h3>, <h4>, <h5>, <h6>. Кожен тег вказує на відносну важливість

тексту після нього. Тег <h1> вказує на найважливіший заголовок, тоді як тег <h6> вказує на малозначущий заголовок [7].

Будь-яка стаття буде наочнішою, якщо вона буде підкріплена графічними матеріалами. Не винятком є й вебсайти. Зображення додаються до тексту за допомогою спеціального тегу . Описуючи даний тег, варто розглянути два його атрибути – «title» та «alt». Атрибут «title» – підказка, що з'являється при наведенні курсору на елемент і описує його вміст. Атрибут «alt» встановлює альтернативний текст для зображення, що дозволяє зрозуміти, що за зображення має бути. Незважаючи на те, що цей тег частіше використовувався в часи повільного інтернету, коли користувачі відключали завантаження зображень, все ж таки пошукові системи вважають застосування цих атрибутів важливими і враховують їх при індексації вебсторінок.

1.4 Оптимізація сайту

Вкрай важливо, щоб тематика вебсайту відповідала його змісту. При виявленні невідповідності або заміні вмісту методами «чорної» оптимізація пошукові системи досить швидко це помічають і накладають санкції (від зниження в рейтингу до повного виключення з нього) [1].

Фірмовий стиль вебсайту, якісне інформаційне наповнення та продумана навігація будуть більш запам'ятовуються для користувачів та привертати їхню увагу.

Оскільки більшість елементів форматування тексту HTML відповідно до рекомендації Консорціуму W3C не повинні використовуватися, рекомендується використовувати каскадні таблиці стилів (CSS) [7]. За допомогою цих таблиць можна не лише відформатувати текст статті, а й прикрасити загальну концепцію вебсайту. Пошукові системи також позитивно ставляться до застосування CSS [2].

Розташування сайту на постійній виділеній IP-адресі спрощує процес індексації пошуковими машинами, а також і викликає у них більше довіри до даного ресурсу.

SEF URL (англ. Friendly URL) – унікальна адреса вебсторінки, що відображає вкладеність директорій та вміст сторінки. Часто має такий вигляд – <http://site/catalog/tv/samsung/>. Компанія Google називає такі адреси «простими». Застосування таких адрес, зручних для сприйняття людиною, позитивно впливає ранжування вебсторінок.

Наявність додаткового меню внизу сторінки заохочується пошуковими системами при індексації та несе в собі функцію додаткової навігації.

Файл мапи сайту (англ. sitemap, sitemap.xml) – це файл з інформацією про всі сторінки сайту, які підлягають індексуванню. Отримавши доступ до цього файлу, пошукова система отримає список всіх сторінок вебсайту. Це особливо актуально, якщо вебсайт має динамічно створювані сторінки або сторінки з довгою адресою. Пошукові роботи можуть дістатися до них у процесі сканування, так як на сканування кожного вебсайту відведено певний час [11].

1.5 Висновки за розділом 1

У першому розділі було розглянуто основи пошукової оптимізації вебресурсів. Наведено визначення SEO та сфери її застосування, проаналізовано алгоритми роботи пошукових систем та артефакти оптимізації вебсторінок.

Основними результатами даного розділу є:

- визначено, що SEO являє собою комплекс заходів для адаптації вебресурсу до вимог пошукових систем. Найбільшу частку відвідувачів сайти отримують саме з пошукового трафіку;

- описано алгоритми функціонування пошукових систем, що включають процеси індексації, ранжування та виведення результатів пошуковими роботами;
- визначено основні артефакти оптимізації вебсторінки: мета-теги, розмітка тексту, графічні зображення.

Таким чином, у даному розділі закладено теоретичне підґрунтя для подальшого дослідження методів пошукової оптимізації.

2 ДОСЛІДЖЕННЯ МЕТОДІВ ПОШУКОВОЇ ОПТИМІЗАЦІЇ

2.1 Методи семантичної обробки

Семантична обробка інформації сайту для пошукової оптимізації – це багатогранний підхід, спрямований на покращення релевантності та видимості вебсайту в результатах пошуку. Цей процес передбачає аналіз і оптимізацію вмісту та структури вебсайту, щоб вони більш точно відповідали намірам пошуку користувача та алгоритмам пошукової системи [12].

Одним із ключових аспектів семантичної обробки є розуміння наміру запитів користувача. Це означає вийти за межі простого зіставлення ключових слів, щоб зрозуміти контекст і нюанси пошукових термінів. Наприклад, пошуковий запит «apple» може стосуватися фруктів або технологічної компанії, а семантична обробка допомагає розрізняти ці контексти, щоб надати більш відповідний вміст [13].

Обробка природної мови (NLP) відіграє вирішальну роль у цьому процесі. Широко використовуються такі методи, як латентне семантичне індексування (LSI), яке визначає закономірності у зв'язках між термінами та поняттями в неструктурованих даних. LSI допомагає зрозуміти тематику вебсайту та його зв'язок із різними пошуковими запитами. Подібним чином термін частотно-інверсна частота документа (TF-IDF) використовується для оцінки важливості слова в документі, який є частиною більшої колекції [14].

Іншим методом є використання структурованих даних і розмітки схем, що полегшує пошуковим системам розуміння вмісту сайту. Це передбачає маркування або тегування інформації на вебсторінці (наприклад, оглядів, інформації про продукт або статей), щоб пошукові системи могли ефективніше індексувати та відображати цей вміст у результатах пошуку [11].

Семантична SEO також поширюється на взаємодію з користувачем. Це передбачає забезпечення того, щоб вебсайт був логічно структурований, а

вміст написаний у спосіб, який є інформативним і простим для користувачів. Це підвищує залучення користувачів, що є ключовим фактором, який пошукові системи враховують під час ранжування вебсайтів [11].

Порівнюючи ці методи, LSI та TF-IDF більше зосереджені на аналізі вмісту та аспекті релевантності, тоді як структуровані дані та розмітка схеми займаються ясністю та організацією інформації вебсайту. LSI та TF-IDF вимагають більш складних можливостей обробки мови, але забезпечують глибоке розуміння релевантності вмісту. З іншого боку, структуровані дані більш прості у впровадженні та безпосередньо допомагають пошуковим системам зрозуміти вміст сайту [15].

Що стосується залежності вебсайтів від семантичної оптимізації, існує чітка кореляція між добре реалізованими семантичними стратегіями SEO та покращеним рейтингом у пошукових системах (табл. 2.1). Вебсайти, які ефективно використовують семантичну обробку, мають тенденцію до вищого рейтингу за релевантними запитами, оскільки вони надають пошуковим системам чіткий контекстно релевантний вміст, який відповідає намірам користувачів. Це покращує взаємодію з користувачем, підвищує рівень залученості та, як наслідок, покращує ефективність SEO [14].

Таблиця 2.1 – Порівняння методів семантичної обробки тексту

Метод	Область застосування	Складність впровадження	Пряма вигода для SEO
LSI (Латентне семантичне індексування)	Аналіз контенту та його релевантності	Висока	Висока
TF-IDF (Частотність термінів-Зворотна частотність документів)	Аналіз важливості ключових слів	Середня	Середня
Структуровані дані та розмітка схеми	Організація контенту та ясність	Низька	Висока

На представленому рисунку 2.1 показано залежність між рівнем впровадження семантичного SEO і покращенням рейтингу вебсайтів у пошукових системах. З графіка видно, що із зростанням впровадження семантичних SEO-стратегій спостерігається позитивний вплив на позиції сайту в результатах пошуку. Це підкреслює важливість семантичної оптимізації у сучасних практиках SEO.

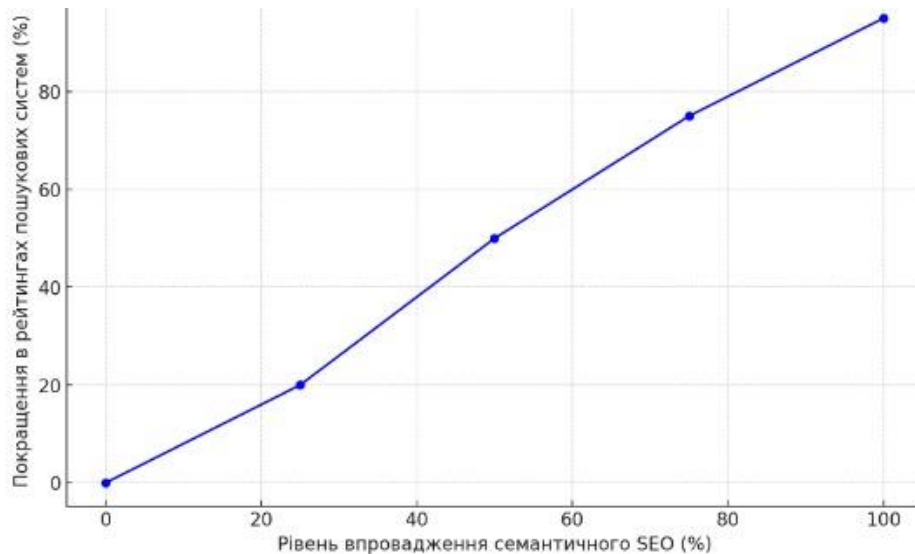


Рисунок 2.1 – Залежність рейтинга сайту від SEO

2.2 Метод статистики ключових слів

Аналіз і обробка ключових слів для пошукової оптимізації є критичним завданням для розробників вебсайтів і вебдодатків. Це передбачає використання статистичних методів для визначення та використання найефективніших ключових слів для покращення рейтингу сайту в пошуковій системі. Зазвичай цей процес починається з дослідження ключових слів для визначення релевантних термінів і фраз, пов'язаних із вмістом вебсайту та пошуковою поведінкою його аудиторії [16].

Одним із поширених підходів є використання частотного аналізу, який перевіряє, як часто певні ключові слова з'являються у вмісті вебсайту. Цей метод допомагає визначити, які терміни є найпоширенішими та

потенційно найбільш релевантними. Однак це може призвести до надмірної кількості ключових слів, коли вміст буде перевантажений ключовими словами, що може негативно вплинути на пошукову оптимізацію, якщо не обробляти її належним чином [17].

Інша техніка передбачає аналіз релевантності, який виходить за межі простої частоти та враховує контекст, у якому використовуються ключові слова. Цей метод оцінює, наскільки ключове слово релевантне загальному вмісту, допомагаючи переконатися, що ключові слова використовуються осмислено та відповідно до контексту [18].

Існує також аналіз конкуренції, де вивчається популярність і використання ключових слів серед конкуруючих вебсайтів. Це має вирішальне значення для розуміння того, які ключові слова є найбільш ефективними в певній ніші чи галузі. Ключові слова з високою конкуренцією, хоч і складніші для рейтингу, можуть бути дуже корисними, якщо їх використовувати ефективно [18].

Крім того, семантичний аналіз стає все більш важливим. Це передбачає розуміння намірів і контекстуальних значень пошукових запитів. Цей метод допомагає ідентифікувати пов'язані ключові слова та фрази, які можуть бути неочевидними, але мають відношення до намірів пошуку користувача [19].

Порівнюючи ці методи, частотний аналіз є простим, але може не дати глибокого розуміння ефективності ключових слів. Хоча аналіз релевантності є більш складним, він гарантує відповідність ключових слів контексту. Аналіз конкуренції забезпечує стратегічний погляд на використання ключових слів, необхідний для створення ніші на конкурентних ринках. Семантичний аналіз, найдосконаліший, тісно пов'язаний із сучасними алгоритмами пошукових систем, які визначають пріоритет користувачів (табл. 2.2).

Таблиця 2.2 – Порівняння методів статистичної обробки ключових слів

Метод	Фокус	Складність	Вплив на SEO
Аналіз частотності	Поширеність ключових слів	Низька	Середній
Аналіз релевантності	Контекстуальне використання	Середня	Високий
Аналіз конкуренції	Конкурентоспроможність ключових слів	Висока	Високий
Семантичний аналіз	Пошукові наміри та контекст	Висока	Дуже високий

Важливо зазначити, що хоча використання більшої кількості ключових слів може бути корисним, є точка зменшення прибутку, коли додавання більшої кількості ключових слів може не значно покращити SEO і навіть може бути контрпродуктивним, якщо це призведе до перенасичення ключовими словами.

Рисунок 2.2 демонструє залежність між кількістю ключових слів, які використовуються в SEO, та ефективністю цих зусиль. Спочатку, коли кількість ключових слів збільшується, ефективність SEO значно покращується. Однак після певного моменту швидкість підвищення ефективності сповільнюється, що вказує на зменшення прибутку від додавання більшої кількості ключових слів.

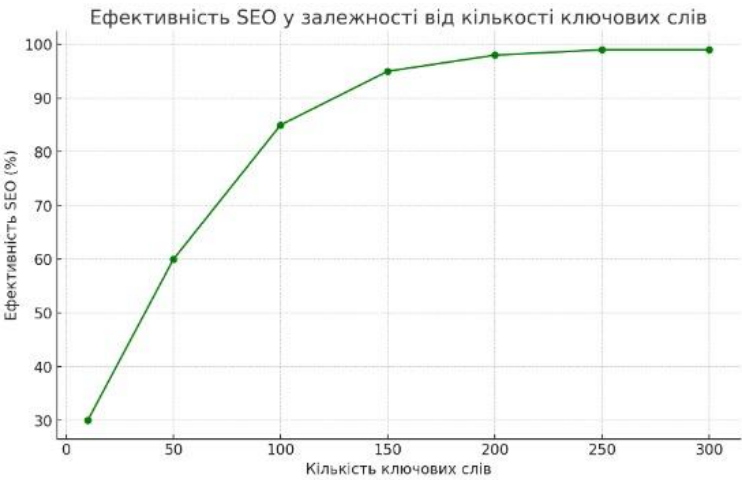


Рисунок 2.2 – Залежність ефективності SEO від кількості ключових слів

Ця тенденція підкреслює важливість збалансованого та стратегічного підходу до використання ключових слів у SEO. Хоча корисно включати різноманітні релевантні ключові слова, перевантаження вмісту надмірною кількістю ключових слів може бути контрпродуктивним. Ключовим є зосередження на релевантності, контексті та стратегічному використанні ключових слів відповідно до пошукових намірів користувача та конкурентного середовища.

2.3 Застосування методів в сучасних засобах пошукової оптимізації

Serpstat – це платформа для автоматизації аналізу конкурентів, ключових слів, посилань, аудиту сайту та його оптимізації (рис. 2.3). Нею користуються понад 200 тисяч користувачів, серед них такі компанії, як Uber, Shopify, Samsung, Philips, Deloitte. Інструмент був розроблений у 2013 році як внутрішній продукт, але згодом перетворився на багатофункціональну SEO-платформу. У базі даних Serpstat понад 5 мільярдів фраз [20].

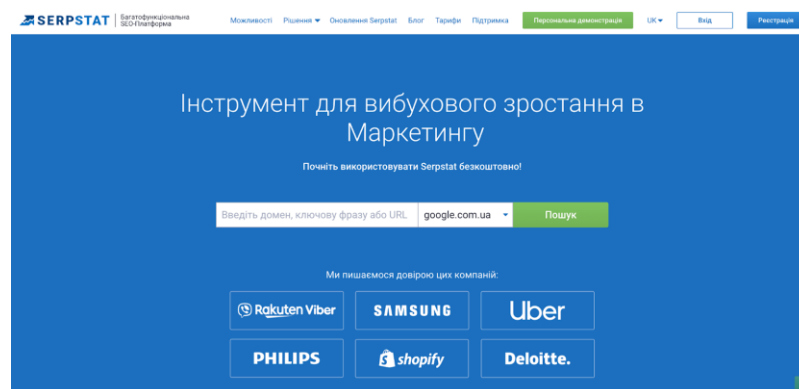


Рисунок 2.3 – Інтерфейс Serpstat [20]

Також автоматичне збирання семантичного ядра можна здійснити за допомогою KeywordShitter. Це безкоштовна програма. Щоб нею користуватися навіть не потрібно реєструватися (рис. 2.4) [21].

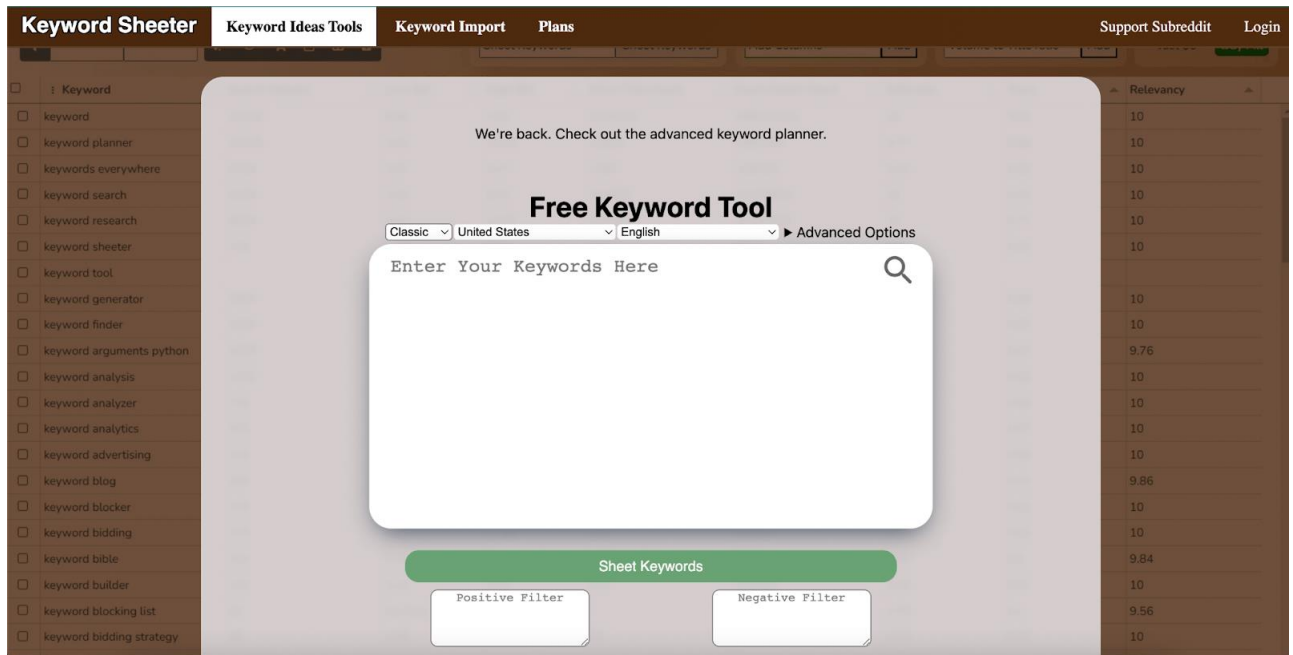


Рисунок 2.4 – Інтерфейс KeywordShitter [21]

Keyword Tool – ще один цінний ресурс, особливо для тих, хто хоче глибше зануритися в дослідження ключових слів (рис. 2.5). Він використовує автозаповнення Google для генерації пропозицій ключових слів, надаючи інформацію про фактичні пошукові запити, які використовують люди. Це може бути особливо корисним для розуміння пошукових інтенцій.

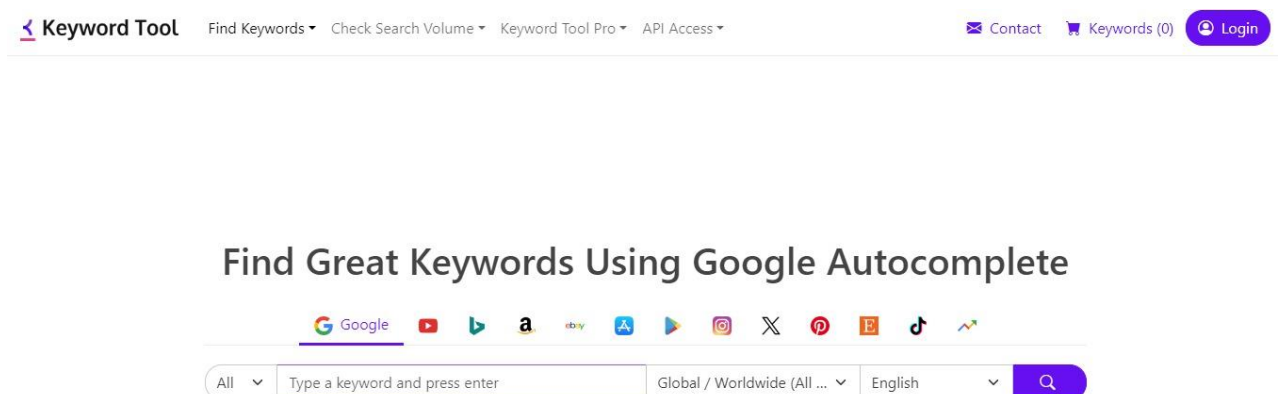


Рисунок 2.5 – Інтерфейс Keyword Tool [22]

У сфері пошукової оптимізації інструменти на кшталт Serpstat, KeywordShitter та Keyword Tool відіграють ключову роль у підвищенні видимості та рейтингу вебсайтів. Кожен з цих інструментів пропонує унікальні функції та можливості, роблячи їх цінними для різних аспектів SEO.

Serpstat відомий своїм комплексним підходом. Це не просто інструмент для дослідження ключових слів; він охоплює аналіз вебсайтів, аналіз конкурентів, аналіз зворотніх посилань та аудит сайтів. Це робить його універсальним інструментом для широкого спектру завдань SEO. Serpstat особливо корисний для тих, хто хоче отримати всебічне уявлення про свою SEO-перспективу, включаючи інформацію про дії конкурентів.

KeywordShitter, навпаки, має більш вузький підхід. Як впливає з назви, це переважно інструмент для генерації ключових слів. Він відомий своєю здатністю швидко створювати велику кількість ключових слів. Хоча йому бракує аналітичної глибини Serpstat, KeywordShitter ідеально підходить для пошуку ідей довгих ключових слів. Він простий і безпосередній, що робить його хорошою відправною точкою для пошуку ключових слів.

Keyword Tool більш складний, ніж KeywordShitter з точки зору аналізу, але не надає такого комплексного набору SEO-інструментів, як Serpstat.

Порівнюючи ці інструменти, Serpstat виділяється своїми всебічними можливостями, що робить його підходящим як для початківців, так і для досвідчених користувачів, які потребують глибокого аналізу своєї SEO-діяльності. KeywordShitter, хоча і обмежений у функціональності, відмінно підходить для швидкої генерації ключових слів, особливо на початкових етапах планування SEO. Keyword Tool знаходить баланс, пропонуючи більш детальні інсайти ключових слів, які можуть бути вирішальними для адаптації контенту для привернення уваги користувача (табл. 2.3).

Таблиця 2.3 – Порівняння SEO-інструментів

Особливість/Інструмент	Serpstat	KeywordShitter	Keyword Tool
Всебічний аналіз SEO	Так	Ні	Ні
Детальний аналіз ключових слів	Так	Основний	Так
Аналіз конкурентів	Так	Ні	Ні
Аналіз зворотніх посилань	Так	Ні	Ні
Аудит сайту	Так	Ні	Ні
Генерація ключових слів	Так	Так (висока кількість)	Так
Інсайти інтенцій користувачів	Помірний	Ні	Високий

2.4 Висновки за розділом 2

У другому розділі було проведено аналіз основних методів пошукової оптимізації вебресурсів та огляд існуючих засобів їх підтримки.

Основні результати полягають в наступному:

- розглянуто методи семантичного аналізу тексту, такі як LSI та TF-IDF, для підвищення релевантності контенту. Визначено ключову роль цих методів в SEO;
- проаналізовано метод статистичного аналізу ключових слів як способу оцінки ефективності їх використання на вебресурсі;
- здійснено огляд сучасних засобів пошукової оптимізації, таких як Serpstat, Keyword Shitter, SEMrush. Визначено їхні переваги та недоліки.

Отримані результати дозволили сформулювати вимоги до програмного забезпечення, розробка якого передбачена в наступних розділах роботи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДУ ВЕБОПТИМІЗАЦІЇ

3.1 Функціональні можливості програмного забезпечення

У процесі дослідження специфіки теми, розглянутої в даній роботі, були ідентифіковані наступні труднощі, що виникають під час оптимізації вебсайтів:

- більшість програмних продуктів на ринку вимагають регулярних фінансових інвестицій;
- змінні уподобання користувачів вимагають неперервного оновлення контенту вебсайту;
- змінювана політика ранжування в пошукових системах вимагає безперервного моніторингу та аналізу;
- часові витрати на аналіз ключових слів для оптимізації сайту, що знижують ефективність роботи SEO-фахівця.

На основі аналізу зазначених вище проблем була запропонована розробка програмного забезпечення, що включатиме наступні можливості (табл. 3.1).

Таблиця 3.1 – Функції програмного забезпечення

Функція	Параметри
Семантичне ядро вебсторінки	Загальна кількість символів
	Кількість символів без урахування пробілів
	Загальна кількість слів
	Кількість унікальних слів
	Кількість загальноновживаних слів
Збір статистики по ключових словах	Введення користувачем пошукових слів з пошукових систем.

Аналізуючи важливість кожної з цих функцій, можна виділити особливу увагу на семантичний аналіз тексту. Він дозволяє не тільки визначити основні характеристики тексту, але й дає змогу оцінити його якість та релевантність для певних ключових слів. Це надзвичайно важливо

для створення ефективного контенту, який би відповідав запитам користувачів та критеріям пошукових систем.

Збір статистики по ключових словах, які вводяться користувачами, також є ключовим моментом. Він дозволяє аналізувати тенденції та переваги цільової аудиторії, адаптувати контент під запити пошукових систем та підвищити відвідуваність сайту. Такий підхід забезпечує більш цілеспрямовану оптимізацію вебсайтів.

Інтеграція цих двох ключових функцій у одну програму значно спрощує процес оптимізації вебсайтів. Це не тільки економить час та ресурси спеціалістів, але й підвищує ефективність їхньої роботи, забезпечуючи більш якісний та відповідний контент.

Макет застосунку повинен бути простим та інтуїтивно зрозумілим. Із основних елементів користувацького інтерфейсу повинні бути присутні поле введення адреси вебсторінки для аналізу, кнопка для активації функції аналізу, поле виводу тексту з вебсторінки та поле з переліком ключових слів (рис. 3.1).

The wireframe shows a web application interface with the following elements:

- Top right corner: Two buttons labeled "Log IN" and "Sing UP".
- Below the buttons: A horizontal input field labeled "Введіть URL вебсторінки" (Enter website URL) and a button labeled "Кнопка «Аналіз»" (Analysis button).
- Below the input field: A large rectangular area labeled "Текст із вебсторінки" (Text from website).
- Below the text area: A large rectangular area labeled "Перелік ключових слів" (List of keywords).

Рисунок 3.1 – Макет вебзастосунку

3.2 Вибір технологій розробки

Node.js і Django – це дві широко використовувані технології у веброзробці, кожна з яких має свої переваги та сценарії найкращого використання.

Node.js – середовище виконання JavaScript, побудоване на механізмі JavaScript V8 Chrome, дозволяє розробникам використовувати JavaScript на стороні сервера. Його неблокуюча, керована подіями архітектура робить його особливо придатним для створення масштабованих і високопродуктивних програм, особливо вебпрограм у реальному часі, таких як сервери чату, онлайн-ігри та інструменти для співпраці. Node.js добре себе показує в обробці кількох одночасних з'єднань із високою пропускнуою здатністю, що робить його найкращим вибором для програм, які потребують обробки даних у реальному часі [21].

Django – це вебфреймворк високого рівня, написаний на мові програмування Python, який заохочує швидку розробку та чистий, прагматичний дизайн. Він відомий своїм підходом, що забезпечує вбудовану панель адміністратора та багато готових до використання функцій. Django користується особливою перевагою через свою надійність, масштабованість і безпеку. Це чудовий вибір для розробників, яким потрібно швидко створювати складні вебсайти з динамічними даними. Підтримка Django ORM (Object-Relational Mapping) полегшує роботу з базами даних, а його архітектура MVC забезпечує чітке розділення завдань [22].

Порівнюючи Node.js і Django, Node.js більше стосується гнучкості використання JavaScript у всьому стеку та його високої продуктивності в обробці одночасних запитів, тоді як Django має на меті швидку розробку із багатьма вбудованими функціями та сильним акцентом на безпеці та зручності підтримки коду.

Неблокуючий ввід-вивід і керована подіями модель Node.js роблять його більш ефективним для додатків реального часу, тоді як Django з його

повною стандартною бібліотекою більше підходить для додатків, де швидкість розробки та структура додатка є пріоритетом (табл. 3.2).

Таблиця 3.2 – Порівняння Node.js і Django

Особливість/Аспект	Node.js	Django
Мова	JavaScript	Python
Архітектура	Подієво-орієнтована, неблокуючий ввід-вивід	Модель-Вид-Контролер (MVC)
Продуктивність	Висока для реального часу, паралельних додатків	Висока для структурованих, орієнтованих на дані додатків
Швидкість розробки	Гнучка, але може бути повільніша через відсутність готових рішень	Швидка завдяки підходу «все включено»
Сфера застосування	Вебдодатки у реальному часі, односторінкові додатки	Вебсайти, орієнтовані на дані, додатки, які потребують швидкої розробки

Наведений рисунок 3.2 ілюструє тенденції популярності Node.js і Django з 2015 по 2021 рік. Дані для створення графіку були взяті зі статистичної інформації на GitHub. Обидві технології демонструють тенденцію до зростання популярності з часом. Однак Node.js демонструє дещо вищу позицію, що свідчить про швидше зростання його бази користувачів і залучення спільноти порівняно з Django [23].

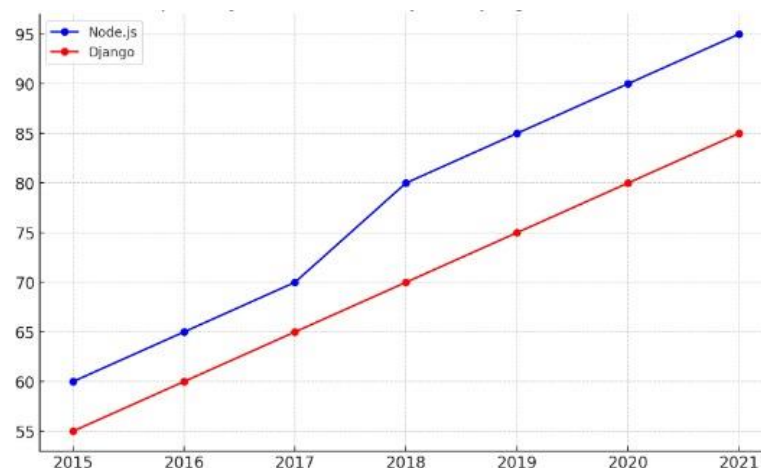


Рисунок 3.2 – Тенденції популярності Node.js і Django

Цю тенденцію можна пояснити зростанням популярності JavaScript як повноцінної мови розробки та зростаючим попитом на додатки реального часу. Зростання Django, незважаючи на стабільність, є більш поступовим, що відображає його постійний попит у сфері розробки застосунків з динамічними даними та структурованих вебдодатків.

Вилучення тексту з вебсайтів є критичним завданням для багатьох програм, включаючи збирання, аналіз даних і оптимізацію пошукових систем. BeautifulSoup є однією з найпопулярніших бібліотек для цієї мети, особливо в спільноті Python, але це не єдиний доступний варіант [24].

BeautifulSoup чудово аналізує документи HTML і XML, дозволяючи розробникам легко переміщатися, шукати та змінювати дерево аналізу. Він добре працює з різними аналізаторами, такими як lxml і html5lib, що робить його гнучким і потужним. Бібліотека відома своєю простотою та можливістю обробки неправильної розмітки.

Однак існують альтернативи, такі як Scrapy та lxml. Scrapy – це повноцінна фреймворк для парсингу контенту. Він не тільки витягує дані з вебсайтів, але й обробляє їх і зберігає в потрібному форматі. Scrapy є асинхронним, тобто він швидше, ніж BeautifulSoup, обробляє запити та дані. Він більше підходить для великомасштабних завдань зчитування вебсторінок [25].

Бібліотека Python lxml надає простий API для аналізу XML- і HTML-файлів. Незважаючи на те, що цей інструмент швидший за BeautifulSoup, він не такий гнучкий, коли справа доходить до обробки погано відформатованого HTML [1].

Порівнюючи ці інструменти, BeautifulSoup виділяється простотою використання та гнучкістю, особливо для початківців або невеликих проєктів. Scrapy більше підходить для масштабних, складних завдань для збирання та обробки контенту. lxml є найкращим вибором для критично важливих до продуктивності програм (табл. 3.3).

Таблиця 3.3 – Порівняння BeautifulSoup, Scrapy, lxml

Особливість/Інструмент	BeautifulSoup	Scrapy	lxml
Тип	Бібліотека	Фреймворк для вебскрепінгу	Бібліотека
Легкість використання	Висока	Середня	Середня
Швидкість	Середня	Висока	Висока
Гнучкість	Висока	Висока	Середня
Масштабованість	Низька	Висока	Середня
Найкраще використання	Прості завдання парсингу, обробка поганого HTML	Великомасштабний парсинг вебсайтів, обробка даних	Високопродуктивний парсинг, добре структурований HTML/XML

На графіку наведено гіпотетичний огляд тенденцій популярності інструментів вилучення тексту BeautifulSoup, Scrapy та lxml з 2015 по 2021 рік (рис. 3.3). На цій ілюстрації BeautifulSoup демонструє послідовне зростання популярності, ймовірно, завдяки простоті використання та гнучкості, що робить його кращим вибором для багатьох розробників, особливо тих, хто має невеликий досвід парсингу вебсайтів або працює над невеликими проєктами [2].

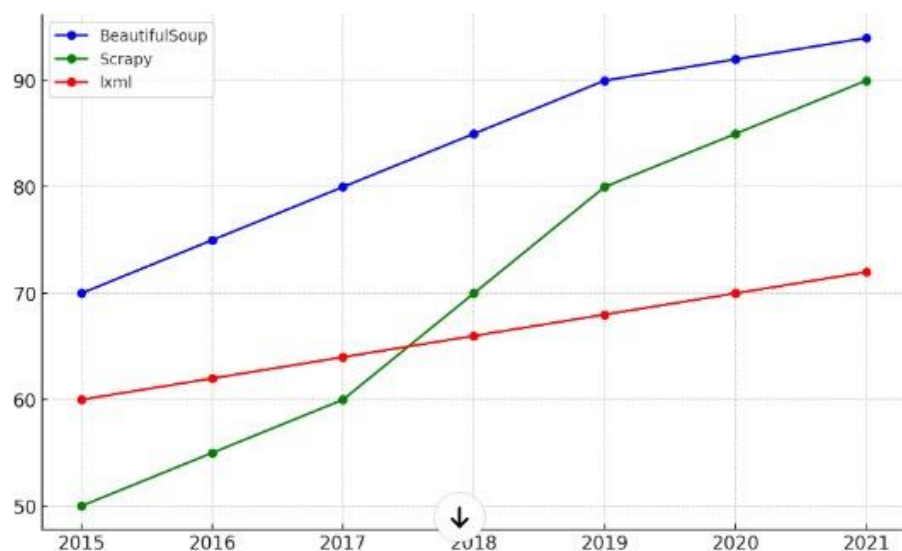


Рисунок 3.3 – Тенденції популярності інструментів вилучення тексту BeautifulSoup, Scrapy та lxml

Популярність Scrapy також демонструє значну тенденцію до зростання, що відображає його все більшу популярність для більш складних і масштабних завдань зі збору інформації на вебсайтах. Його комплексна структура та можливості асинхронної обробки роблять його сильним вибором для таких додатків.

Популярність lxml залишається відносно стабільною з невеликим зростанням з роками. Його зосередженість на продуктивності та ефективності робить його кращим інструментом для критично важливих додатків, які потребують швидкого та ефективного аналізу добре структурованого HTML або XML [7].

Ці тенденції висвітлюють різноманітні потреби в збиранні та вилученні тексту, причому кожен інструмент знаходить свою нішу на основі своїх сильних сторін і основних випадків використання. Вибір між BeautifulSoup, Scrapy та lxml значною мірою залежить від конкретних вимог проєкту, зокрема від масштабу текстової інформації, складності завдань, а також потреби у швидкості й ефективності.

Розуміння можливостей і відмінностей між бібліотеками обробки природної мови (NLP), такими як NLTK і spaCy, має вирішальне значення для проєктів, пов'язаних з аналізом тексту, обробкою мови або машинним навчанням [9].

NLTK (Natural Language Toolkit) є однією з найбільш широко використовуваних бібліотек для академічних і освітніх цілей. Вона відома своєю великою колекцією інструментів для символічної та статистичної обробки природної мови. NLTK чудово підходить для створення прототипів і навчання завдяки обширній документації та простоті використання. Вона охоплює широкий спектр завдань у NLP, від базових до складних, що робить її універсальною для різноманітних дослідницьких та освітніх проєктів [20].

spaCy, з іншого боку, призначена для використання у виробництві. Вона відома своєю швидкістю та ефективністю. spaCy високо оптимізована і надає спеціальні функції для поширених випадків використання в NLP, таких

як токенізація, тегування частини мови, розпізнавання іменованих об'єктів і аналіз залежностей. Її API є простим і розробленим, щоб бути простим у використанні, що робить цю бібліотеку кращим вибором для розробників, які створюють реальні програми [10].

Порівнюючи NLTK і spaCy, важливо відзначити, що перевага NLTK полягає в її наборі функцій і придатності для академічних цілей, тоді як spaCy зосереджується на продуктивності та простоті використання у виробничих середовищах. NLTK пропонує більш широкий спектр алгоритмів для різних завдань NLP, що є корисним для експериментів і навчання. spaCy, зосереджена на конкретних завданнях, забезпечує високу ефективність і більше підходить для розгортання програм NLP у виробничих умовах (табл. 3.4).

Таблиця 3.4 – Порівняння NLTK і spaCy

Особливість/Інструмент	NLTK	spaCy
Призначення	Навчальні та академічні цілі	Виробниче використання
Швидкість	Помірна	Висока
Легкість використання	Висока (завдяки документації)	Висока (завдяки API)
Оптимізація	Не оптимізована для великих масштабів	Сильно оптимізована для великих масштабів
Функціональність	Широкий спектр алгоритмів і функцій	Сфокусована на конкретних завданнях NLP
Використання в реальних проєктах	Обмежене	Широко розповсюджене

Для візуалізації популярності кожного інструменту, можна створити графік на основі статистичних даних з порталу Github, який відображатиме тренди, наприклад, засновані на кількості завантажень, кількості згадок у форумах або статтях, або на даних з опитувань розробників. Це допоможе оцінити, як розвивається інтерес до цих бібліотек у спільноті (рис. 3.4).

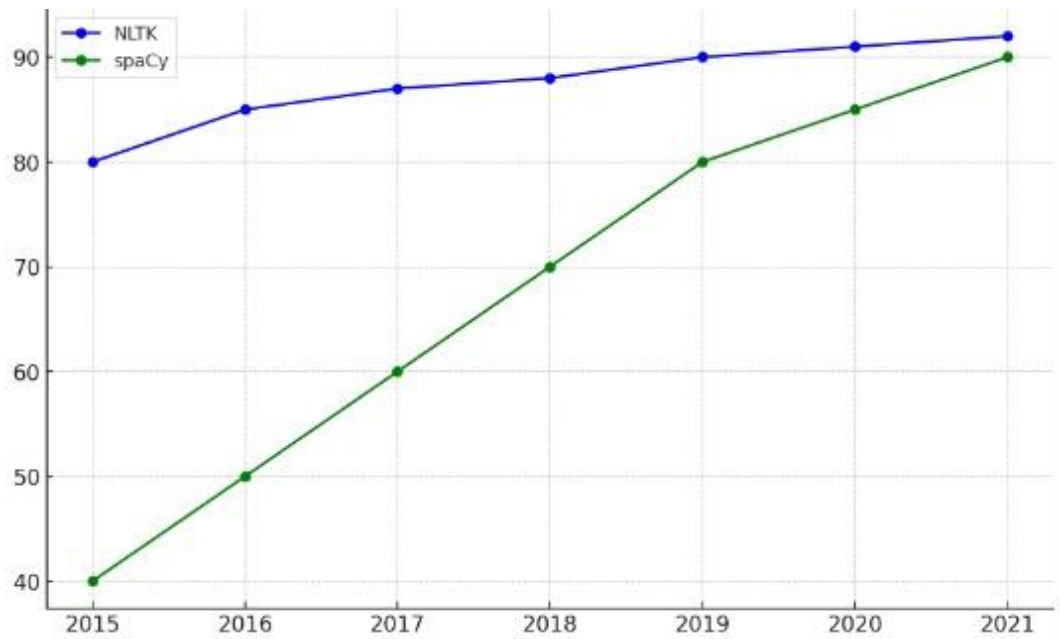


Рисунок 3.4 – Порівняння NLTK і spaCy

Згідно з даними на рисунку 3.4, NLTK демонструє стабільно високий рівень популярності, що може бути пов'язано з її широким використанням у академічних цілях та великою кількістю навчальних ресурсів.

spaCy, у свою чергу, показує стрімке зростання популярності, особливо в останні роки. Це може відображати зростаючий інтерес до використання spaCy в продуктивному середовищі, завдяки її високій продуктивності та зручному API.

Такі інструменти, як Moz і SEMrush, є доволі важливими, коли мова йде про аналіз показників пошукової оптимізації. Обидва пропонують набір функцій для покращення видимості в Інтернеті, але вони мають відмінні характеристики та сильні сторони.

Moz відомий широкими можливостями SEO, зокрема показником авторитетності домену, який широко використовується для оцінки потенційної позиції вебсайту. Він пропонує інструменти для дослідження ключових слів, аналізу посилань і аудиту сайтів, що робить його комплексним рішенням для професіоналів із SEO. Користувачський інтерфейс Moz інтуїтивно зрозумілий, і він відомий своєю потужною спільнотою та

освітніми ресурсами, що робить його чудовим вибором для тих, хто новачок у SEO або хоче поглибити своє розуміння.

Тим часом SEMrush пропонує більш різноманітний спектр інструментів цифрового маркетингу, що виходить за межі SEO та включає функції для соціальних мереж, контент-маркетингу та маркетингових досліджень. Його інструмент дослідження ключових слів є особливо надійним, надає детальну інформацію про стратегії ключових слів конкурентів. SEMrush також пропонує широку аналітику для зворотніх посилань і кампаній PPC («оплата за клік»), що робить його універсальним інструментом для ширшої стратегії цифрового маркетингу [22].

Порівнюючи Moz і SEMrush, Moz більше зосереджений на поглибленому SEO-аналізі та особливо корисний завдяки аналізу повноважень домену та стратегій створення посилань. SEMrush, з іншого боку, є всеохоплюючим інструментом цифрового маркетингу, який пропонує широкий спектр функцій, які виходять за межі SEO та включають інші аспекти онлайн-маркетингу (табл. 3.5).

Таблиця 3.5 – Порівняння Moz і SEMrush

Особливість/Інструмент	Moz	SEMrush
Фокус	SEO	Цифровий маркетинг
Особливості SEO	Поглиблений аналіз домену та посилань	Розширений аналіз ключових слів та конкурентів
Інструменти для соціальних медіа	Обмежені	Розширені
Аналітика PPC	Обмежена	Розширена
Легкість використання	Висока	Висока
Ресурси для Навчання	Багаті	Обмежені

Обидва інструменти мають тенденцію до зростання популярності, але SEMrush демонструє більш стрімке зростання в порівнянні з Moz.

Цей ріст популярності SEMrush може бути пов'язаний з його універсальністю як інструменту цифрового маркетингу, що включає не тільки SEO, але й інструменти для соціальних медіа, контент-маркетингу та дослідження ринку. Moz, з іншого боку, залишається стабільно популярним серед тих, хто спеціалізується на SEO, завдяки своїм глибоким аналітичним можливостям і сильній спільноті (рис. 3.5).



Рисунок 3.5 – Порівняння Moz і SEMrush

3.3 Проєктування програмного забезпечення

Розробка вебдодатку для пошукової оптимізації, а саме для автоматичного складання семантичного ядра на основі тексту сторінки вебсайту, включає кілька ключових етапів. Алгоритм, який описує цей процес разом із рекомендованими технологіями та очікуваними результатами на кожному кроці, наведено у таблиці 3.6.

Таблиця 3.6 – Етапи розробки програмного застосунку

Номер етапу	Технологія	Результат
Початкове налаштування	Node.js, Python	Базова структура вебдодатку, налаштування сервера.
Модуль вилучення тексту	BeautifulSoup, щоб отримати текст із заданих URL-адрес вебсайтів	Необроблені текстові дані з вебсторінок
Обробка та нормалізація тексту	NLTK	Чистий і нормалізований текст, готовий для подальшого аналізу
Вилучення ключових слів	TF-IDF, LDA	Список основних і додаткових ключових слів, релевантних вмісту вебсайту
Компіляція семантичного ядра	SEMrush	Семантичне ядро, що складається з найбільш релевантних і ефективних ключових слів
Інтерфейс користувача для відображення результатів	React.js	Інтерактивний інтерфейс, де користувачі можуть переглядати згенероване семантичне ядро та взаємодіяти з ним
Розгортання	Heroku	Активна, доступна вебпрограма для автоматичної оптимізації SEO.

3.4 Програмна реалізація застосунку

Для розробки вебдодатку для SEO та автоматичного складання семантичного ядра сайту потрібно використати кілька технологій. Реалізацію кожного етапу розробки можна описати таким чином:

- початкове налаштування клієнтської частини (інтерфейсу за стосунку) за допомогою бібліотеки React.js. Команди налаштування у консолі наведено в лістингу A.1, базова структура компоненту App наведена в лістингу A.2;
- створення модуля вилучення тексту за допомогою бібліотеки BeautifulSoup (лістинг A.3);

- створення модуля обробки та нормалізації тексту за допомогою бібліотеки NLTK (лістинг A.4);
- створення модуля вилучення ключових слів (лістинг A.5);
- створення таблиць у базі даних для зберігання інформації про вебсайти та ключові слова на прикладі реляційної БД PostgreSQL (лістинг A.6);
- інтеграція бази даних з сервеною частиною (лістинг A.7);
- після обробки та зберігання ключових слів у базі даних, можна автоматично генерувати файл із семантичним ядром (лістинг A.8). Цей код включає зберігання даних у базі даних та генерацію файлу CSV із семантичним ядром.

3.5 Висновки за розділом 3

У третьому розділі описується проектування та програмна реалізація застосунку підтримки пошукової оптимізації вебресурсів.

Основні результати даного розділа:

- сформовано функціональні вимоги до програмного забезпечення на основі аналізу предметної області. Визначено головні функції: семантичний аналіз тексту та збір статистики по ключовим словам;
- обрано технології для реалізації системи: мова Python, фреймворк Flask, бібліотеки NTLK, BeautifulSoup, PostgreSQL;
- спроектовано архітектуру та базу даних програмного застосунку відповідно до поставлених вимог;
- виконано програмну реалізацію основних модулів: аналізатор тексту, збирач статистики, вебінтерфейс користувача.

Розроблене програмне забезпечення буде використано в наступному розділі для тестування запропонованого методу пошукової оптимізації.

4 ТЕСТУВАННЯ МЕТОДУ ПОШУКОВОЇ ОПТИМІЗАЦІЇ

4.1 Результати роботи програмного застосунку

Основний екран програми, який часто називають домашньою сторінкою, служить відправною точкою для всіх операцій (рис. 4.1). Тут користувачів можуть побачити інтуїтивно зрозумілий макет. Однією з ключових особливостей цього інтерфейсу є спеціальне поле введення, де користувачі можуть вводити URL-адресу вебсторінки, яку вони бажають проаналізувати. Цей зручний дизайн гарантує, що навіть ті, хто має мінімальний технічний досвід, зможуть легко орієнтуватися та використовувати програму.

Поруч із полем введення URL-адреси розташована кнопка «Аналізувати» – важливий компонент програми. Користувачі можуть активувати аналіз, натиснувши цю кнопку після введення URL-адреси вебсторінки. Дизайн і розташування цієї кнопки розроблені для забезпечення максимальної видимості та доступності.

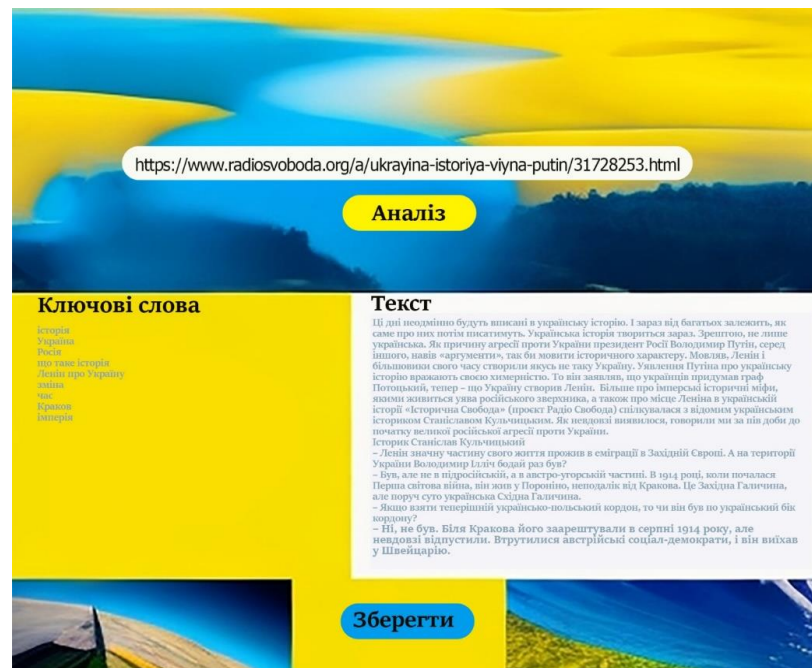


Рисунок 4.1 – Стартова сторінка

Після початку аналізу програма обробляє введену URL-адресу та витягує текст із вебсторінки. Потім отриманий текст відображається у визначеній області в інтерфейсі, розташованій під полем введення URL-адреси (рис. 4.2). Ця область відображення тексту дозволяє користувачам візуально підтверджувати текст, що аналізується, додаючи додатковий рівень взаємодії з користувачем і прозорість процесу.



Рисунок 4.2 – Сторінка аналізу

Аналіз завершується виділенням і відображенням ключових слів. Після завершення аналізу тексту програма демонструє список вилучених ключових слів. Цей список відображається в розділі під областю відображення тексту (рис. 4.3). Ключові слова представлені впорядковано, що дозволяє користувачам легко зрозуміти й оцінити ключові компоненти та теми, присутні в тексті вебсторінки. Ця функція особливо корисна для спеціалістів із SEO та розробників контенту, які прагнуть зрозуміти цільність ключових слів вебсторінок.



Рисунок 4.3 – Перелік ключових слів

Інтерфейс користувача програми розроблено з урахуванням чіткості та функціональності. Кожен розділ, від введення URL-адреси до відображення ключових слів, створено для забезпечення бездоганної та ефективної роботи. Інтерфейс, як показано на малюнках програми, демонструє продуманий підхід до взаємодії з користувачем, гарантуючи, що процес аналізу вебсторінки є простим і ефективним.

Загалом, ця вебпрограма є цінним інструментом для аналізу вебсторінок, пропонуючи функції, які доступні для початківців і корисні для професіоналів. Поєднання зручного інтерфейсу, ефективного вилучення тексту та детального аналізу ключових слів робить його незамінним інструментом в арсеналі кожного, хто хоче оптимізувати вміст або зрозуміти структуру вебсторінок.

4.2 Аналіз роботи методу складання семантичного ядра

Метод складання семантичного ядра для вебсторінки або документа передбачає систематичну ідентифікацію та організацію ключових слів і фраз, які є центральними для змісту.

Розглянемо такі параметри (табл. 4.1):

- загальна кількість ідентифікованих ключових слів – загальна кількість унікальних ключових слів, виділених із вмісту;
- високорелевантні ключові слова – ключові слова, які найбільше пов’язані з основною темою;
- середньорелевантні ключові слова – ключові слова, які певною мірою пов’язані з темою, але не настільки центральні;
- низькорелевантні ключові слова – ключові слова з незначною релевантністю до теми;
- щільність ключових слів – відсоток появи ключового слова порівняно із загальною кількістю слів;
- різноманітність ключових слів – співвідношення унікальних ключових слів до загальної кількості ключових слів.

Таблиця 4.1 – Параметри аналізу

Параметр	Кількість слів вхідного тексту					
	150	300	500	1000	2000	5000
Ключові слова високої релевантності	50	90	230	250	245	215
Ключові слова середньої релевантності	70	72	80	106	204	210
Ключові слова низької релевантності	10	50	60	70	100	20
Щільність ключових слів	50%	37%	23%	20%	20%	19%
Різнорманітність ключових слів	60%	60%	76%	81%	85%	83%

На графіку зображено залежність різних параметрів від кількості слів у тексті (рис. 4.4). За даними, можна побачити, як кількість ключових слів високої, середньої та низької релевантності змінюється зі збільшенням обсягу тексту.

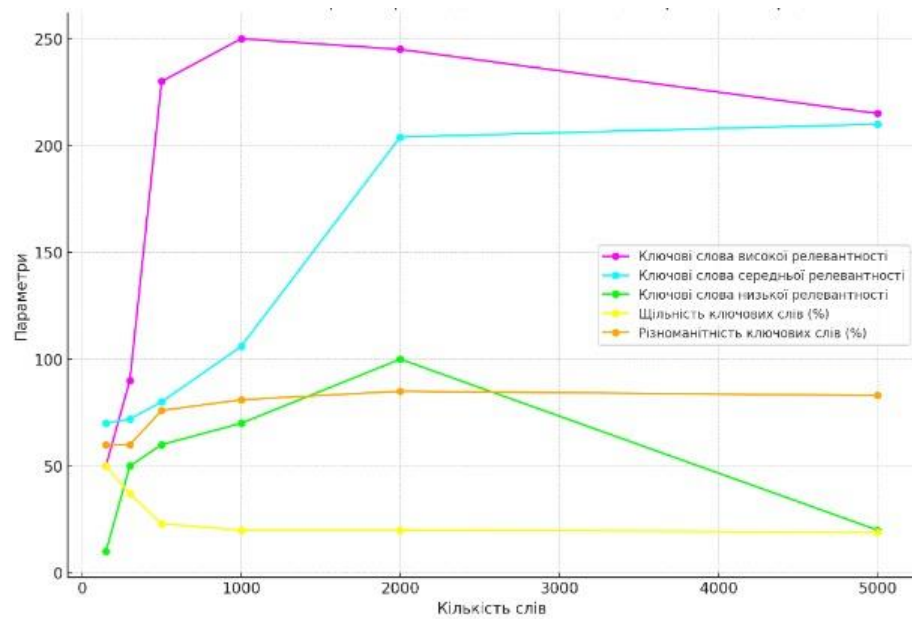


Рисунок 4.4 – Залежність параметрів від кількості слів у тексті

Цікаво, що кількість ключових слів високої релевантності спочатку різко зростає, а потім злегка знижується при великих обсягах тексту. Ключові слова середньої релевантності демонструють поступове зростання, в той час як кількість ключових слів низької релевантності має менш очевидну тенденцію зростання і навіть знижується для найбільшого обсягу тексту.

Що стосується щільності ключових слів, то цей показник знижується зі збільшенням кількості слів, що свідчить про меншу концентрацію ключових слів у більших текстах. Різноманітність ключових слів, навпаки, зростає з підвищенням обсягу тексту, що вказує на більшу різноманітність слів у більш об'ємних текстах.

4.3 Висновки за розділом 4

У четвертому розділі було проведено тестування розробленого програмного застосунку для пошукової оптимізації та проаналізовано ефективність методу складання семантичного ядра.

Основні результати полягають в наступному:

- продемонстровано результати роботи програми, включаючи інтерфейс користувача, процес аналізу тексту та виведення ключових слів;
- експериментально досліджено залежності показників якості семантичного ядра від обсягу вхідного тексту. Встановлено раціональні межі застосування методу;
- підтверджено можливість автоматизації процесу SEO за допомогою розробленого програмного застосунку на основі запропонованих методів.

Загалом проведене тестування підтвердило ефективність реалізованого підходу до пошукової оптимізації вебресурсів та доцільність подальшого його вдосконалення.

ВИСНОВКИ

У результаті дослідження, виконаного в рамках дипломної кваліфікаційної роботи магістра, було розроблено програмний застосунок для автоматизації процесу пошукової оптимізації вебресурсів на основі методів семантичного аналізу та статистичного аналізу ключових слів.

Розроблений метод дозволяє автоматично формувати семантичне ядро сайту та оцінювати ефективність SEO за обраними критеріями. Програмна реалізація методу у вигляді вебдодатку дає змогу оптимізувати контент вебресурсів відповідно до уподобань цільової аудиторії.

Таким чином, мета дослідження, яка полягала в підвищенні ефективності SEO на основі удосконалення методів та створення підтримуючого ПЗ, може вважатися досягнутою.

Наукова новизна отриманих результатів підтверджується удосконаленням методу семантичного аналізу та розробкою методики оцінювання ефективності ключових слів для SEO.

Практична значущість полягає у спрощенні та автоматизації процесу пошукової оптимізації вебсайтів шляхом застосування розробленого програмного додатку. Перспективами подальшої роботи є розширення функціоналу програми та впровадження у виробництво.

Висновки щодо вирішених задач:

- проаналізовано сучасні методи та програмні засоби, визначено їх недоліки та напрями вдосконалення;
- запропоновано методику автоматичної оцінки ефективності ключових слів за статистичними критеріями релевантності та щільності;
- створено програмний застосунок, що реалізує дані методи та надає користувацький вебінтерфейс;
- експериментально підтверджено ефективність запропонованих методів оптимізації та доцільність застосування розробленого ПЗ.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Глушок Д.Р. Аналітичне дослідження інтерфейсів та пошукова оптимізація веб-сайтів об'єднаних територіальних громад для покращення їх інтерактивних характеристик [Електронний ресурс] / Д.Р. Глушок. – Режим доступу: <http://elartu.tntu.edu.ua/handle/lib/37924>.
2. Зозуляк Б.М. Пошукова оптимізація веб-сайту факультету комп'ютерних інформаційних систем і програмної інженерії [Електронний ресурс] / Б.М. Зозуляк. – Режим доступу: <http://elartu.tntu.edu.ua/handle/123456789/19506>.
3. Sinicki A. Site Architecture [Electronic resource] / A. Sinicki // The Complete Beginner's Guide to Search Engine Optimization. – Berkeley, CA, 2018. – Access mode: https://doi.org/10.1007/978-1-4842-3635-2_5.
4. Ucyigitler S. Optimization of Transferable Site–Site Potentials Using a Combination of Stochastic and Gradient Search Algorithms [Electronic resource] / Sinan Ucyigitler, Mehmet C. Camurdan, J. Richard Elliott // Industrial & Engineering Chemistry Research. – 2012. – Vol. 51, no. 17. – P. 6219–6231. – Access mode: <https://doi.org/10.1021/ie201186q>.
5. Yagup V. G. Research Optimization of the Mode of Full Reactive Power Compensation in the Electrical Network [Electronic resource] / V. G. Yagup, K. V. Yagup // Tekhnichna Elektrodynamika. – 2023. – Vol. 2023, no. 5. – P. 55–63. – Access mode: <https://doi.org/10.15407/techned2023.05.055>.
6. Yang X.-S. Cuckoo search [Electronic resource] / Xin-She Yang // Nature-Inspired Optimization Algorithms. – [S. l.], 2021. – P. 141–155. – Access mode: <https://doi.org/10.1016/b978-0-12-821986-7.00017-2>.
7. Seo H.-J. Development and Validation of Web-based Q-sort System [Electronic resource] / Hee-Jung Seo // Taegu Science University Defense Security Institute. – 2021. – Vol. 5, no. 1. – P. 145–158. – Access mode: <https://doi.org/10.37181/jscs.2021.5.1.145>.
8. Automatic Knowledge Recommending System Using E-Mail

[Electronic resource] / [DooHyun Kim et al.] // E-Commerce and Web Technologies. – Berlin, Heidelberg, 2005. – P. 118–127. – Access mode: https://doi.org/10.1007/11545163_12.

9. Koo H.-S. A Development Tool for Web Applications based on Cocoon Framework [Electronic resource] / Heung-Seo Koo // Journal of Korean Institute of Intelligent Systems. – 2005. – Vol. 15, no. 6. – P. 742–747. – Access mode: <https://doi.org/10.5391/jkiis.2005.15.6.742>.

10. Jeong D. Development of web-based simulator for supply chain management [Electronic resource] / Donghwa Jeong, Minseok Seo, Yoonho Seo // 2009 Winter Simulation Conference - (WSC 2009), Austin, TX, USA, 13–16 December 2009. – [S. l.], 2009. – Access mode: <https://doi.org/10.1109/wsc.2009.5429199>.

11. Seo Y. Web-based Product Information Visualization through VRML Code Generation [Electronic resource] / Yoonho Seo // Integrated Design and Manufacturing in Mechanical Engineering '98. – Dordrecht, 1999. – P. 577–584. – Access mode: https://doi.org/10.1007/978-94-015-9198-0_71.

12. Seo D. Intelligent Web Crawler for Supporting Big Data Analysis Services [Electronic resource] / Dongmin Seo, Hanmin Jung // The Journal of the Korea Contents Association. – 2013. – Vol. 13, no. 12. – P. 575–584. – Access mode: <https://doi.org/10.5392/jkca.2013.13.12.575>.

13. Seo J. S. Web based System for Supporting Medical Treatment in Korean Medicine based on Korean Medicine Ontology [Electronic resource] / Jin Soon Seo // Korean Journal of Oriental Physiology & Pathology. – 2014. – Vol. 28, no. 1. – Access mode: <https://doi.org/10.15188/kjopp.2014.02.28.1.113>.

14. Seo E. A Study on the Preferred Factors of Audio Content Based on Web Novels [Electronic resource] / Euna Seo, Youngjae Kim // The Korean Society of Culture and Convergence. – 2021. – Vol. 43, no. 11. – P. 89–108. – Access mode: <https://doi.org/10.33645/cnc.2021.11.43.11.89>.

15. Yonghyun Hwang. Structure-aware web transcoding for mobile devices [Electronic resource] / Yonghyun Hwang, Jihong Kim, Eunkyong Seo //

IEEE Internet Computing. – 2003. – Vol. 7, no. 5. – P. 14–21. – Access mode: <https://doi.org/10.1109/mic.2003.1232513>.

16. Edgar M. Page Experience: Core Web Vitals and More [Electronic resource] / Matthew Edgar // Tech SEO Guide. – Berkeley, CA, 2023. – P. 95–106. – Access mode: https://doi.org/10.1007/978-1-4842-9054-5_6.

17. Seo E.-G. Comparative Analysis of Web Accessibility in National Libraries [Electronic resource] / Eun-Gyoung Seo, Hee-Jung Kim // Journal of the Korean Society for Library and Information Science. – 2008. – Vol. 42, no. 3. – P. 345–364. – Access mode: <https://doi.org/10.4275/kslis.2008.42.3.345>.

18. Seo Y. Development of Web-based CAM system [Electronic resource] / Yoonho Seo, Dae-Young Kim, Suk-Hwan Suh // The International Journal of Advanced Manufacturing Technology. – 2005. – Vol. 28, no. 1-2. – P. 101–108. – Access mode: <https://doi.org/10.1007/s00170-004-2422-y>.

19. Seo B.-K. Cooperation and Enforcement of Chinese Platforms and App Providers: Focusing on web Novel Contents App [Electronic resource] / Bong-Kyo Seo, Hwajin Lee // The Journal of Modern China Studies. – 2022. – Vol. 23, no. 4. – P. 167–203. – Access mode: <https://doi.org/10.35820/jmcs.23.4.6>.

20. Uchang Park. An Implementation of XML Documents Search System based on Similarity in Structure and Semantics [Electronic resource] / Uchang Park, Yeojin Seo // International Workshop on Challenges in Web Information Retrieval and Integration, Tokyo, Japan. – [S. l.]. – Access mode: <https://doi.org/10.1109/wiri.2005.8>.

21. Белз О. Seo-оптимізація сайтів вишів України / О. Белз // Формування ринкової економіки в Україні. – 2018. – Вип. 39. – С. 3–8.

22. Астістова Т. І. Seo-оптимізація в системі моніторингу web-ресурсів [Електронний ресурс] / Т. І. Астістова // Technologies and Engineering. – 2023. – № 1. – С. 9–17. – Режим доступу: <https://doi.org/10.30857/2786-5371.2023.1.1>.

23. Ivanechko N. SEO Optimization: the Semantic Core [Electronic resource] / Nelia Ivanechko, Rostyslav Okrepkyi, Valentyn Pavelko // Problems of

Systemic Approach in the Economy. – 2022. – No. 1(87). – Access mode: <https://doi.org/10.32782/2520-2200/2022-1-16>.

24. Solodka N. SEO-Optimization of Website on Google Search Engine [Electronic resource] / N. Solodka, D. Sorokin, O. Liashenko // Transactions of Kremenchuk Mykhailo Ostrohradskyi National University. – 2020. – Vol. 1. – P. 107–112. – Access mode: <https://doi.org/10.30929/1995-0519.2020.1.107-112>.

25. SMM i SEO оптимізація, просування у соціальних мережах як способи підвищення популярності музичної освіти [Електронний ресурс] / [Летуновська Наталія Євгенівна та ін.]. – [Б. м.], 2017. – Режим доступу: <http://essuir.sumdu.edu.ua/handle/123456789/64627>.

ДОДАТОК А

Лістинги програмної реалізації методів

A.1 Команди створення нового React-застосунку

```
npx create-react-app seo-app  
cd seo-app  
npm start
```

A.2 Базова структура файлу «App.js»

```
import React from 'react';  
  
function App() {  
  return (  
    <div className="App">  
      <header>  
        <button>Login</button>  
        <button>Register</button>  
      </header>  
      <main>  
        <input type="text" placeholder="Enter website URL" />  
        <button>Load Semantic Core</button>  
      </main>  
    </div>  
  );  
}  
  
export default App;
```

A.3 Код модуля вилучення (парсингу) тексту

```

from bs4 import BeautifulSoup
import requests

def extract_text_from_url(url):
    """
    Функція для витягнення тексту з вебсторінки.
    :param url: URL вебсторінки для скрапінгу.
    :return: Витягнутий текст.
    """
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')
    text = soup.get_text()
    return text

```

A.4 Код модуля обробки та нормалізації тексту

```

import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# Завантаження необхідних ресурсів NLTK
nltk.download('punkt')
nltk.download('stopwords')

def clean_and_normalize_text(text):
    """
    Функція для очищення та нормалізації тексту.
    :param text: Вхідний текст.

```

```
:return: Очищений та нормалізований текст.
```

```
"""
```

```
tokens = word_tokenize(text)
```

```
tokens = [word.lower() for word in tokens if word.isalpha()]
```

```
stop_words = set(stopwords.words('english'))
```

```
filtered_tokens = [word for word in tokens if not word in stop_words]
```

```
return ' '.join(filtered_tokens)
```

A.5 Код модуля вилучення ключових слів

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
from gensim import corpora, models
```

```
def extract_keywords_tfidf(text):
```

```
    """
```

```
    Функція для визначення ключових слів за допомогою TF-IDF.
```

```
:param text: Вхідний текст.
```

```
:return: Список ключових слів.
```

```
    """
```

```
    vectorizer = TfidfVectorizer(max_df=0.8, max_features=10000)
```

```
    tfidf_matrix = vectorizer.fit_transform([text])
```

```
    feature_names = vectorizer.get_feature_names_out()
```

```
    dense = tfidf_matrix.todense()
```

```
    text_tfidf = dense[0].tolist()[0]
```

```
    keyword_scores = {feature_names[i]: text_tfidf[i] for i in
range(len(feature_names))}
```

```
    return sorted(keyword_scores, key=keyword_scores.get, reverse=True)
```

```
def extract_keywords_lda(text):
```

```
    """
```

Функція для визначення ключових слів за допомогою LDA.

```
from nltk.tokenize import word_tokenize
from gensim import corpora, models

def extract_keywords_lda(prKeyWord_text):
    """
    Функція для визначення ключових слів за допомогою LDA.
    :param prKeyWord_text: Вхідний текст.
    :return: Список ключових слів.
    """
    prKeyWord_tokens = [word_tokenize(prKeyWord_text)]
    prKeyWord_dictionary = corpora.Dictionary(prKeyWord_tokens)
    prKeyWord_corpus = [prKeyWord_dictionary.doc2bow(token) for token
in prKeyWord_tokens]
    prKeyWord_lda_model =
models.ldamodel.LdaModel(prKeyWord_corpus, num_topics=1,
id2word=prKeyWord_dictionary, passes=15)
    prKeyWord_topics =
prKeyWord_lda_model.print_topics(num_words=5)
    prKeyWord_keywords = [word for topic in prKeyWord_topics for word
in topic[1].split('+')]
    return prKeyWord_keywords
```

A.6 SQL-запити для створення таблиць у БД

```
CREATE TABLE websites (
    id SERIAL PRIMARY KEY,
    url VARCHAR(255) NOT NULL,
    last_scraped TIMESTAMP
);
```

```
CREATE TABLE keywords (
    id SERIAL PRIMARY KEY,
    website_id INT REFERENCES websites(id),
    keyword VARCHAR(255) NOT NULL,
    frequency INT NOT NULL
);
```

A.7 Код інтеграція бази даних із Python

```
import psycopg2

def connect_db():
    return psycopg2.connect("dbname=seo_db      user=yourusername
password=yourpassword")

def save_keywords(website_id, keywords):
    conn = connect_db()
    cursor = conn.cursor()
    for keyword, frequency in keywords.items():
        cursor.execute("INSERT INTO keywords (website_id, keyword,
frequency) VALUES (%s, %s, %s)", (website_id, keyword, frequency))
    conn.commit()
    cursor.close()
```

A.8 Код автоматичного створення файлу із семантичним ядром

```
import csv

def generate_semantic_core_file(website_id):
    conn = connect_db()
    cursor = conn.cursor()
    cursor.execute("SELECT keyword, frequency FROM keywords
WHERE website_id = %s", (website_id,))
    keywords = cursor.fetchall()
    cursor.close()
    conn.close()

    with open(f'semantic_core_{website_id}.csv', 'w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(['Keyword', 'Frequency'])
        for keyword, frequency in keywords:
            writer.writerow([keyword, frequency])

# Приклад використання
generate_semantic_core_file(1)
```

A.9 Код запуск сервера

```
from flask import Flask, request, jsonify
import threading
import json

app = Flask(__name__)
```

```

# Запуск сервера Flask
def run_server():
    app.run(host='0.0.0.0', port=5000)

# Основна точка входу для вебзастосунку
@app.route('/process', methods=['POST'])
def process_request():
    # Тут буде обробка запиту
    return jsonify({"message": "Запит оброблено"})

# Запуск сервера у окремому потоці
threading.Thread(target=run_server).start()

```

A.10 Код форми вводу посилання на вебсторінку

```

import React, { useState } from 'react';

const URLInputForm = ({ onAnalyze }) => {
    const [url, setUrl] = useState("");

    const handleSubmit = (e) => {
        e.preventDefault();
        onAnalyze(url);
    };

    return (
        <form onSubmit={handleSubmit}>
            <label>
                URL:
                <input

```

```
        type="text"
        value={url}
        onChange={(e) => setUrl(e.target.value)}
      />
    </label>
    <button type="submit">Аналіз</button>
  </form>
);
};

export default URLInputForm;
```

ДОДАТОК Б
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА
РЕАЛІЗАЦІЯ МЕТОДУ ПОШУКОВОЇ
ОПТИМІЗАЦІЇ ВЕБРЕСУРСІВ

Виконав
ст. гр. КНТ-212м Іван РАДЧЕНКО

Керівник
к.т.н., доцент Ірина ЗЕЛЕНЬОВА

Рисунок Б.1 – Слайд 1

МЕТА ТА ЗАДАЧІ ПРОЄКТУ

Мета роботи це розробка програмного застосунку для автоматизації процесу пошукової оптимізації веб-ресурсів на основі методів семантичного аналізу та статистичної обробки ключових слів.

Задачі, які треба вирішити:

- дослідити існуючі методи та засоби пошукової оптимізації веб-ресурсів;
- розробити удосконалений метод семантичного аналізу тексту для автоматичного формування семантичного ядра сайту;
- розробити методику оцінювання ефективності ключових слів на основі їх статистичного аналізу;
- створити програмний застосунок реалізації даних методів;
- провести тестування розробленого програмного забезпечення.

Рисунок Б.2 – Слайд 2

СПОСОБИ ВІДВІДУВАННЯ ВЕБСАЙТУ



3

Рисунок Б.3 – Слайд 3

РОЗПОДІЛ МОЖЛИВОСТІ ВІДВІДУВАННЯ ВЕБСАЙТУ

Перша сторінка пошукової видачі		Друга сторінка пошукової видачі	
Номер позиції в пошуковій видачі	Відсоток відвідувачів вебсайту	Номер позиції в пошуковій видачі	Відсоток відвідувачів вебсайту
1	2530%	1	97%
2	1520%	2	75%
3	1012%	3	64%
4	89%	4	32%
5	48%	510	1% і менше
6	45%		
7			
8			
9			
10			

4

Рисунок Б.4 – Слайд 4

РОБОТА ПОШУКОВОЇ СИСТЕМИ

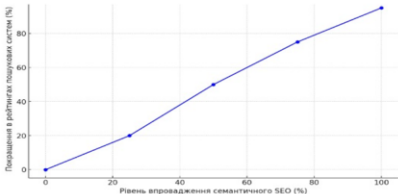


5

Рисунок Б.5 – Слайд 5

МЕТОДИ СЕМАНТИЧНОЇ ОБРОБКИ

Метод	Область застосування	Складність впровадження	Пряма вигода для SEO
LSI(Латентне семантичне індексування)	Аналіз контенту та його релевантності	Висока	Висока
TF-IDF (Частотність термінів-Зворотна частотність документів)	Аналіз важливості ключових слів	Середня	Середня
Структуровані дані та розмітка схеми	Організація контенту та ясність	Низька	Висока

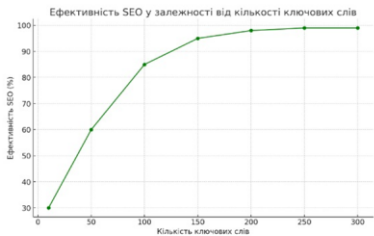


6

Рисунок Б.6 – Слайд 6

МЕТОД СТАТИСТИКИ КЛЮЧОВИХ СЛІВ

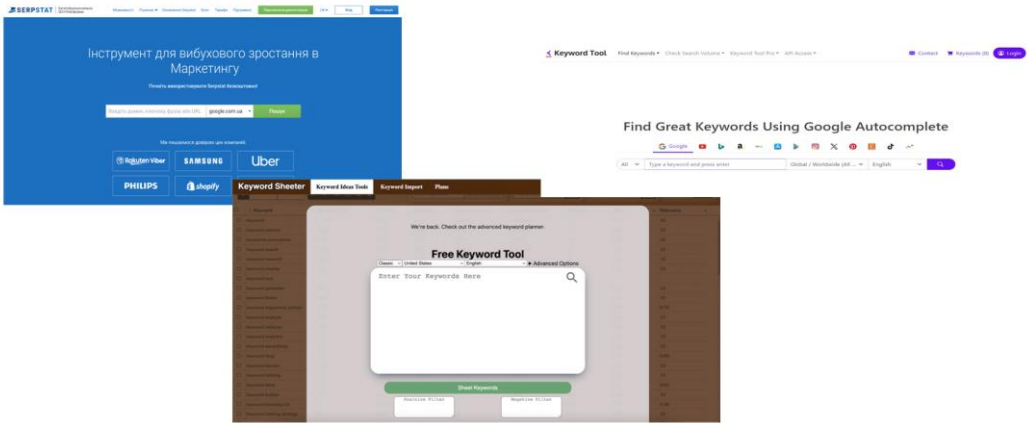
Метод	Фокус	Складність	Вплив на SEO
Аналіз частотності	Поширеність ключових слів	Низька	Середній
Аналіз релевантності	Контекстуальне використання	Середня	Високий
Аналіз конкуренції	Конкурентоспроможність ключових слів	Висока	Високий
Семантичний аналіз	Пошукові наміри та контекст	Висока	Дуже високий



7

Рисунок Б.7 – Слайд 7

ЗАСТОСУВАННЯ МЕТОДІВ В СУЧАСНИХ ЗАСОБАХ ПОШУКОВОЇ ОПТИМІЗАЦІЇ



8

Рисунок Б.8 – Слайд 8

ТАБЛИЦЯ ПОРІВНЯННЯ SEO-ІНСТРУМЕНТІВ

Особливість/Інструмент	Serpstat	KeywordShitter	Keyword Tool
Всебічний аналіз SEO	Так	Ні	Ні
Детальний аналіз ключових слів	Так	Основний	Так
Аналіз конкурентів	Так	Ні	Ні
Аналіз зворотних посилань	Так	Ні	Ні
Аудит сайту	Так	Ні	Ні
Генерація ключових слів	Так	Так (висока кількість)	Так
Інсайти інтенцій користувачів	Помірний	Ні	Високий

9

Рисунок Б.9 – Слайд 9

ФУНКЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функція	Параметри
Семантичне ядро вебсторінки	Загальна кількість символів
	Кількість символів без урахування пробілів
	Загальна кількість слів
	Кількість унікальних слів
	Кількість загальноновживаних слів
Збір статистики по ключових словах	Введення користувачем пошукових слів, з пошукових систем

10

Рисунок Б.10 – Слайд 10

ВИБІР ТЕХНОЛОГІЙ РОЗРОБКИ

Особливість/Аспект	Node.js	Django
Мова	JavaScript	Python
Архітектура	Подієво-орієнтована, неблокуючий ввід-вивід	Модель-Вид-Контролер (MVC)
Продуктивність	Висока для реального часу, паралельних додатків	Висока для структурованих, орієнтованих на дані додатків
Швидкість розробки	Гнучка, але може бути повільніша через відсутність готових рішень	Швидка завдяки підходу "все включено"
Сфера застосування	Веб-додатки у реальному часі, односторічкові додатки	Веб-сайти, орієнтовані на дані, додатки, які потребують швидкої розробки

Особливість /Інструмент	BeautifulSoup	Scrapy	lxml
Тип	Бібліотека	Фреймворк для вебскрепінгу	Бібліотека
Легкість використання	Висока	Середня	Середня
Швидкість	Середня	Висока	Висока
Гнучкість	Висока	Висока	Середня
Масштабованість	Низька	Висока	Середня
Найкраще використання	Прості завдання парсингу, обробка поганого HTML	Великомасштабний парсинг вебсайтів, обробка даних	Високопродуктивний парсинг, добре структурований HTML/XML

Особливість/Інструмент	NLTK	spaCy
Призначення	Навчальні та академічні цілі	Виробничі використання
Швидкість	Помірна	Висока
Легкість використання	Висока (завдяки документації)	Висока (завдяки API)
Оптимізація	Не оптимізована для великих масштабів	Сильно оптимізована для великих масштабів
Функціональність	Широкий спектр алгоритмів і функцій	Сфокусована на конкретних завданнях NLP
Використання в реальних проектах	Обмежене	Широко розповсюджене

11

Рисунок Б.11 – Слайд 11

ТЕХНОЛОГІЧНИЙ СТЕК ВЕБДОДАТКУ

Номер етапу	Технологія	Результат
Початкове налаштування	Node.js, Python	Базова структура вебдодатку, налаштування сервера
Модуль вилучення тексту	BeautifulSoup, щоб отримати текст із заданих URL-адрес вебсайтів	Необроблені текстові дані з вебсторінок
Обробка та нормалізація тексту	NLTK	Чистий і нормалізований текст, готовий для подальшого аналізу
Вилучення ключових слів	TF-IDF, LDA	Список основних і додаткових ключових слів, релевантних вмісту вебсайту
Компіляція семантичного ядра	SEMrush	Семантичне ядро, що складається з найбільш релевантних і ефективних ключових слів
Інтерфейс користувача для відображення результатів	React.js	Інтерактивний інтерфейс, де користувачі можуть переглядати згенероване семантичне ядро та взаємодіяти з ним
Розгортання	Heroku	Активна, доступна вебпрограма для автоматичної оптимізації SEO

12

Рисунок Б.12 – Слайд 12

ВИСНОВКИ

Наукова новизна отриманих результатів підтверджується удосконаленням методу семантичного аналізу та розробкою методики оцінювання ефективності ключових слів для SEO.

Практична значущість полягає у спрощенні та автоматизації процесу пошукової оптимізації вебсайтів шляхом застосування розробленого програмного додатку. Перспективами подальшої роботи є розширення функціоналу програми та впровадження у виробництво.

Висновки щодо вирішених задач:

- проаналізовано сучасні методи та програмні засоби, визначено їх недоліки та напрями вдосконалення;
- запропоновано методику автоматичної оцінки ефективності ключових слів за статистичними критеріями релевантності та щільності;
- створено програмний застосунок, що реалізує дані методи та надає користувацький вебінтерфейс;
- експериментально підтверджено ефективність запропонованих методів оптимізації та доцільність застосування розробленого ПЗ.