**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Чорноморський національний університет**
**імені Петра Могили**
**Факультет комп'ютерних наук**
**Кафедра інтелектуальних інформаційних систем**

**ДОПУЩЕНО ДО ЗАХИСТУ**
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2024 р.

**КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА**

# ІНТЕЛЕКТУАЛЬНА СИСТЕМА ОБРОБКИ ПРИРОДНОЇ МОВИ З ВИКОРИСТАННЯМ АЛГОРИТМІВ WEB-СКРЕЙПІНГ

Спеціальність 122 «Комп'ютерні науки»

**122 – КРМ – 601.21810314**

***Виконав студент 6-го курсу, групи 601***
***_____В. В. Котляренко***
«19» лютого 2024 р.

***Керівник*: *канд. техн. наук, доцент***
***_____І. О. Калініна***
«19» лютого 2024 р.

**Миколаїв – 2024 р.**

**Чорноморський національний університет ім. Петра Могили**
**Факультет комп'ютерних наук**
**Кафедра інтелектуальних інформаційних систем**

Освітньо-кваліфікаційний рівень **магістр**
Галузь знань **12 «Інформаційні технології»**
*(шифр і назва)*
Спеціальність **122 «Комп'ютерні науки»**
*(шифр і назва)*

**ЗАТВЕРДЖУЮ**
Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____» _____20___р.

**ЗАВДАННЯ**
**на виконання кваліфікаційної роботи**
**Котляренко Владиславу Валентиновичу**

1. Тема кваліфікаційної роботи магістра «Інтелектуальна система обробки природної мови з використанням алгоритмів web-скрейпінгу».
Керівник роботи Калініна Ірина Олександрівна, канд. техн. наук, доцент.
Затв. наказом Ректора ЧНУ ім. Петра Могили від «__»_____ 2024 р. № ___
2. Строк подання студентом роботи 19 лютого 2024 р.
3. Вхідні (початкові) дані до роботи: основні принципи та методи обробки природної мови, алгоритми вебскрапінгу, критерії відбору даних, програмні мови та фреймворки для NLP та вебскрапінгу, бібліотеки та інструменти для збору та обробки даних. Очікуваний результат: функціональна система, яка використовує алгоритми вебскрапінгу для збору даних та алгоритми обробки природної мови для їх аналізу, інтерфейс для користувачів.
4. Зміст пояснювальної записки (перелік питань, які потрібно розглянути):
– аналіз основних теоретичних концепцій та підходів у галузі обробки природної мови та вебскрапінгу;

– опис методів та алгоритмів обробки природної мови;

– розробка архітектури та дизайну системи;

– аналіз результатів тестування;

– основні висновки дослідження.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: аналіз вимог до умов праці, заходи з техніки безпеки та охорона праці під час роботи у випадку настання надзвичайних ситуацій.

7. Консультанти:

| **Розділ** | **Прізвище, ініціали та посада консультанта** | **Підпис** |
|---|---|---|
| Спеціальна частина з охорони праці | **д-р біол. наук, професор Григор'єва Л.І.** | |
| Методична частина | **канд. техн. наук, доцент Калініна І.О.** | |

Керівник роботи __канд. техн. наук, доцент Калініна І. О._____
*(наук. ступінь, вчене звання, прізвище та ініціали)*

_____
*(підпис)*

Завдання прийнято до виконання ___Котляренко В. В._____
*(прізвище та ініціали)*

_____
*(підпис)*

Дата видачі завдання «__31__» ____жовтня_____2023 р.

**КАЛЕНДАРНИЙ ПЛАН**
**Виконання кваліфікаційної роботи магістра**

Тема: Інформаційна система обробки природної мови з використанням алгоритмів web-скрейпінгу.

| № | Найменування роботи | Початок | Закінчення | Примітки |
|---|---|---|---|---|
| 1 | Подання заяви на затвердження теми КРМ | 01.09.2023 | 20.10.2023 | Виконано |
| 2 | Отримання завдання на виконання КРМ | 21.10.2023 | 08.11.2023 | Виконано |
| 3 | Проходження переддипломної практики, збір та аналіз матеріалів до КРМ | 04.12.2023 | 25.12.2023 | Виконано |
| 4 | Аналіз предметної області | 26.12.2023 | 28.12.2023 | Виконано |
| 5 | Розробка проєктних рішень | 02.01.2024 | 03.01.2024 | Виконано |
| 6 | Моделювання та конструювання ПЗ | 05.01.2024 | 07.01.2024 | Виконано |
| 7 | Кодування, тестування розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача | 08.01.2024 | 23.01.2024 | Виконано |
| 8 | Розробка спеціальної частини з охорони праці | 24.01.2024 | 27.01.2024 | Виконано |
| 9 | Перший попередній захист КРМ на засіданні комісії кафедри | 29.01.2024 | 29.01.2024 | Виконано |
| 10 | Корегування роботи за результатами попереднього захисту | 30.01.2024 | 30.01.2024 | Виконано |
| 11 | Рецензування КРМ | 09.02.2024 | 09.02.2024 | Виконано |
| 12 | Завершення оформлення КРМ та презентації | 10.02.2024 | 10.02.2024 | Виконано |
| 13 | Другий попередній захист КРМ на засіданні комісії кафедри | 12.02.2024 | 12.02.2024 | Виконано |
| 14 | Захист кваліфікаційної роботи | 26.02.2024 | 26.02.2024 | Виконано |

Розробив студент _____Котляренко В. В._____ _____
*(прізвище та ініціали)* (*підпис*)

Керівник роботи __канд. техн. наук, доцент Калініна І. О._ _____
*(наук. ступінь, вчене звання, прізвище та ініціали)* *(підпис)*

*«09» листопада 2023 р.*

# АНОТАЦІЯ

до кваліфікаційної роботи магістра
студента групи 601 ЧНУ ім. Петра Могили

**Котляренко Владислава Валентиновича**

на тему: **“ІНТЕЛЕКТУАЛЬНА СИСТЕМА ОБРОБКИ ПРИРОДНОЇ МОВИ З ВИКОРИСТАННЯМ АЛГОРИТМІВ WEB-СКРЕЙПІНГУ”**


**Актуальність** даної роботи полягає в необхідності розробки ефективних підходів до обробки великих обсягів текстових даних з Інтернету за допомогою методів природної обробки мови та вебскрапінгу. Такий підхід дозволить автоматизувати процеси збору та аналізу інформації, поліпшивши тим самим ефективність обробки даних.

**Об'єктом дослідження** є процес автоматизації збору текстових даних з вебсайтів та їх подальшого аналізу за допомогою методів обробки природної мови.

**Предметом дослідження** є методи вебскрапінгу та обробки природної мови, що застосовуються для автоматизованого збору та аналізу текстових даних.

**Метою дослідження** є підвищення ефективності обробки природної мови за допомогою алгоритмів вебскрапінгу.

Основна увага приділяється розробці алгоритмів, які забезпечують високу точність та ефективність обробки.

У процесі роботи використовувались сучасні методи машинного навчання, аналізу даних, та програмування для реалізації алгоритмів вебскрапінгу та обробки тексту. Розглядались популярні та ефективні методи NLP, такі як парсинг, семантичний аналіз, класифікація тексту. Спеціальна увага приділялась розробці ефективних алгоритмів для збору даних із різних вебресурсів, їх обробці та аналізу з метою виявлення інформативних характеристик та взаємозалежностей. Акцент зроблено на впровадженні машинного навчання та глибокого навчання для підвищення точності та надійності результатів обробки даних.

Для демонстрації ефективності розробленої системи було проведено низку експериментів на реальних даних. Результати показали значне покращення у точності та швидкості обробки даних порівняно з традиційними методами. Також було розроблено користувацький інтерфейс, що дозволяє з легкістю використовувати систему для специфічних потреб збору та аналізу даних.

Магістерська робота складається з шести розділів. Перший розділ присвячений аналізу сучасного стану проблеми та опису предметної сфери. У другому розділі описано методи та алгоритми, їх застосування та ефективність. Третій розділ присвячений детальному опису структури системи та її моделювання. Четвертий розділ містить опис програмної реалізації, результати тестування та аналізу системи. У шостому розділі наведено основні положення з охорони праці та цивільного захисту під час надзвичайних ситуацій. Загальний обсяг роботи – 80 сторінок. Кваліфікаційна робота магістра містить один додаток, 34 рисунків, 8 таблиць, посилання на 45 літературних джерел.

**Ключові слова**: інтелектуальна система, обробка природної мови, вебскрапінг, алгоритми NLP, машинне навчання, глибоке навчання, автоматизація збору даних, користувацький інтерфейс, аналіз даних.

# ABSTRACT

to the master's qualification work by the student of the group 601 of Petro Mohyla Black Sea National University

**Kotliarenko Vladyslav**

## “INTELLIGENT NATURAL LANGUAGE PROCESSING SYSTEM USING WEB SCRAPING ALGORITHMS”

**The relevance** of this work lies in the need to develop effective approaches for processing large volumes of textual data from the Internet using natural language processing methods and web scraping. This approach will allow for the automation of data collection and analysis processes, thereby improving the efficiency of data processing.

**The object of study** is the process of automating the collection of textual data from websites and their subsequent analysis using natural language processing methods.

**The subject of the study** is the web scraping algorithms and natural language processing methods that are used for automated collection and analysis of textual data.

**The purpose of the study** is increasing the efficiency of natural language processing using web scraping algorithms.

In the process of the work, modern methods of machine learning, data analysis, and programming were used to implement web scraping and text processing algorithms. Popular and effective NLP methods such as parsing, semantic analysis, and text classification were considered. Special attention was paid to developing effective algorithms for collecting data from various web resources, processing and analyzing them to identify informative characteristics and interdependencies. Emphasis was placed on the implementation of machine learning and deep learning to increase the accuracy and reliability of data processing results.

A series of experiments were conducted on real data to demonstrate the effectiveness of the developed system. The results showed a significant improvement in the accuracy and speed of data processing compared to traditional methods. A user interface was also

developed that allows the system to be easily used for specific data collection and analysis needs.

The master's thesis consists of six chapters. The first chapter is devoted to the analysis of the current state of the problem and a description of the subject area. The second chapter describes the methods and algorithms, their application, and effectiveness. The third chapter is dedicated to a detailed description of the system's structure and its modeling. The fourth chapter contains a description of the software implementation, testing results, and system analysis. The sixth chapter presents the main provisions of labor protection and civil defense during emergencies.

The total volume of work is 80 pages. The master's thesis contains one appendix, 34 figures, 8 tables, and references to 45 literary sources.

**Keywords**: intelligent system, natural language processing, web scraping, NLP algorithms, machine learning, deep learning, data collection automation, user interface, data analysis.

# ЗМІСТ

# ПЕРЕЛІК СКОРОЧЕНЬ

| | |
|---|---|
| ІС | – інтелектуальна система |
| ПЗ | – програмне забезпечення |
| BERT | – Bidirectional Encoder Representations from Transformers |
| GPT | – Generative Pretrained Transformer |
| NLP | – Natural Language Processing |
| UI | – User Interface |

# ВСТУП

У сучасному світі обробка природної мови (NLP) та вебскрапінг стають все більш значущими технологіями в різноманітних областях, від бізнес-аналітики до розробки інтелектуальних систем. Впровадження інтелектуальних систем, які ефективно використовують ці технології, може значно підвищити рівень автоматизації та оптимізації процесів обробки та аналізу великих обсягів даних. Саме така потреба і стала відправною точкою для цієї магістерської роботи.

Ця робота присвячена розробці інтелектуальної системи обробки природної мови, яка інтегрує алгоритми вебскрапінгу. Основна мета полягає у створенні системи, здатної збирати, обробляти, аналізувати та інтерпретувати великі обсяги текстової інформації з Інтернету, використовуючи передові методи NLP.

**Актуальність** даного дослідження полягає в необхідності розробки ефективних підходів до обробки великих обсягів текстових даних з Інтернету за допомогою методів природної обробки мови та вебскрапінгу. Такий підхід дозволить автоматизувати процеси збору та аналізу інформації, поліпшивши тим самим ефективність обробки даних.

**Об'єктом дослідження** є процес автоматизації збору текстових даних з вебсайтів та їх подальшого аналізу за допомогою методів обробки природної мови.

**Предметом дослідження** є алгоритми вебскрапінгу та обробки природної мови, що застосовуються для автоматизованого збору та аналізу текстових даних.

**Метою дослідження** є підвищення ефективності обробки природної мови за допомогою алгоритмів вебскрапінгу.

Основна увага приділяється розробці алгоритмів, які забезпечують високу точність та ефективність обробки.

Виходячи з мети, поставлено **наступні завдання**:

– літературний огляд та аналіз існуючих рішень;

– визначення вимог до системи;

– розробка алгоритмів вебскрапінгу;

– імплементація методів обробки природної мови;

– тестування та оцінка системи;

– аналіз отриманих результатів;

– розробка користувацького інтерфейсу;

– оформлення висновків та рекомендацій.

Для досягнення поставлених завдань, у магістерській роботі використовуються методи машинного навчання та глибокого навчання, а також різноманітні інструменти програмування. Це дозволить не тільки розширити можливості аналізу даних, але й підвищити точність та ефективність обробки.

Результатом цього дослідження має стати повністю функціональна система, здатна автоматизувати процеси збору та аналізу даних, що значно підвищить якість та швидкість обробки інформації, а також забезпечить нові можливості для її використання в різних галузях.

В контексті цієї роботи, особлива увага приділяється не лише технічному аспекту створення інтелектуальної системи, але й дослідженню її практичного застосування у різних сферах. Сьогодні, коли обсяги інформації в Інтернеті зростають лавиноподібно, важливість ефективного збору та аналізу текстових даних стає все більш вагомою. Ця система відкриває нові горизонти для компаній, наукових дослідників, маркетологів та інших фахівців, яким необхідно швидко обробляти великі обсяги текстової інформації та витягувати з неї корисні дані.

# 1 АНАЛІЗ ПРОБЛЕМИ ОБРОБКИ ПРИРОДНОЇ МОВИ ТА ВЕБСКРАПІНГУ

## 1.1 Характеристика сфери обробки природної мови

**Обробка природної мови (NLP)** – це галузь комп'ютерних наук, яка займається взаємодією між комп'ютерами та людською (природною) мовою. Основна мета NLP – це читання, розуміння та інтерпретація людської мови таким чином, щоби комп'ютери могли ефективно виконувати такі завдання, як переклад, розпізнавання мови, або автоматичне узагальнення текстів.

NLP є однією з найбільш динамічних та впливових областей у сучасній комп'ютерній науці та штучному інтелекті. Ця галузь зосереджена на розробці алгоритмів та систем, які дозволяють комп'ютерам розуміти, інтерпретувати та використовувати людську мову у її природній формі. Завдяки прогресу в NLP, машини сьогодні можуть виконувати широкий спектр мовних завдань, починаючи від автоматичного перекладу та закінчуючи генерацією тексту та автоматичним розпізнаванням мовлення.

Основними напрямками NLP є:

– синтаксичний аналіз;

– семантичний аналіз;

– прагматичний аналіз.

*Синтаксичний аналіз.* Полягає у розпізнаванні граматичної структури речень. Ключовим завданням тут є розподіл тексту на морфеми, слова, фрази та інші складові частини, а також визначення їх взаємних зв'язків. Наприклад, синтаксичний аналізатор може визначити, що слово "летить" у реченні "Птах летить" є дієсловом, яке вказує на дію, що виконується суб'єктом "птах".

*Семантичний аналіз.* Зосереджений на розумінні значення слів та фраз у контексті. Це включає визначення значень слів, виведення відносин між словами та фразами, а також визначення того, як змінюється значення в залежності від

контексту. Наприклад, семантичний аналіз допомагає розрізнити слово "банк" як фінансову установу від "банки варення".

*Прагматичний аналіз*. Вивчає в NLP, як контекст впливає на значення мови. Це включає аналіз умов використання мови та її вплив на сприйняття отриманої інформації. Прагматика досліджує, як люди використовують мову в соціальних контекстах, наприклад, які нюанси та підтексти вносяться в розмову залежно від ситуації.

Обробка природної мови має численні застосування, серед яких:

– *автоматичний переклад*: системи автоматичного перекладу, такі як Google Translate, використовують NLP для перекладу тексту та мовлення з однієї мови на іншу;

– *чат-боти та віртуальні асистенти*: сучасні чат-боти та віртуальні асистенти, такі як Siri та Alexa, використовують NLP для розуміння та відповідей на запити користувачів;

– *аналіз сентименту*: бізнес та маркетологи використовують NLP для аналізу сентименту в соціальних медіа, відгуках та інших текстових даних для зрозуміння ставлення споживачів до продуктів та брендів.

Сучасні досягнення в NLP забезпечуються завдяки використанню таких моделей, як трансформери, які лежать в основі систем типу BERT (Bidirectional Encoder Representations from Transformers) і GPT (Generative Pretrained Transformer). Ці моделі здатні вловлювати нюанси мови, не доступні більшості традиційних методів, забезпечуючи краще розуміння контексту та відтінків значення.

Незважаючи на значний прогрес, NLP все ще стикається з рядом технічних викликів:

– *обробка неструктурованих даних*: велика частина інформації в Інтернеті є неструктурованою, що ускладнює її обробку та аналіз;

– *мовна різноманітність*: існує велика кількість мов і діалектів, кожен з яких має свої унікальні особливості та виклики для обробки;

– *суб'єктивність та контекст*: важливо враховувати суб'єктивність та контекстуальні нюанси мови, особливо в задачах, пов'язаних з аналізом сентименту та емоцій.

Перспективи розвитку NLP обнадійливі. Завдяки швидкому розвитку технологій та збільшенню обчислювальних потужностей можна очікувати подальшого прогресу у здатності машин ефективно обробляти природну мову. Це відкриє нові можливості для автоматизації, покращення якості обслуговування та зростання продуктивності у різних секторах.

### 1.2 Опис застосування вебскрапінгу

**Вебскрапінг** – це процес автоматичного збору даних з вебсторінок. Ця техніка використовується для екстракції інформації з вебсайтів і перетворення її у структурований формат, зручний для аналізу та обробки. Вебскрапінг включає в себе завантаження вебсторінок, вилучення необхідної інформації з них, і, за потреби, її подальшу обробку. Основною перевагою вебскрапінгу є його здатність автоматизувати збір великих обсягів даних з різноманітних джерел у короткий термін. Важливо враховувати правові обмеження і етичні аспекти при використанні вебскрапінгу, оскільки не завжди дозволено автоматично збирати інформацію з веб-сайтів без відповідного дозволу. Збір даних за допомогою вебскрапінгу може порушувати правила авторських прав або політику конфіденційності. Перед використанням вебскрапінгу для будь-яких цілей важливо ознайомитися з правилами конкретного веб-сайту і дотримуватися їх.

Вебскрапінг включає в себе такі етапи:

– *визначення цільових даних*: вибір інформації, яку потрібно зібрати з вебсторінок;

– *збір даних*: використання інструментів для екстракції даних з вебсторінок;

– *обробка даних*: очищення та структурування зібраних даних для подальшого аналізу.

Для вебскрапінгу часто використовуються такі інструменти та мови програмування, як Python з бібліотеками BeautifulSoup та Scrapy, які дозволяють ефективно аналізувати та екстрактувати дані з HTML-коду сторінок. Процес може бути автоматизований за допомогою спеціальних скриптів, що значно підвищує швидкість та ефективність збору даних.

Одним з основних викликів вебскрапінгу є врахування юридичних обмежень та авторських прав. Не всі вебсайти дозволяють скрапінг своїх даних, і в деяких випадках це може порушувати політику використання сайту або авторські права. Тому при розробці та використанні інструментів вебскрапінгу важливо враховувати юридичні аспекти та використовувати дані відповідно до закону.

Таблиця 1.1 – Сфери застосування вебскрапінгу

| **Сфера застосування** | **Опис** |
|---|---|
| Моніторинг цін | Компанії використовують вебскрапінг для стеження за цінами конкурентів. Це дозволяє їм швидко реагувати на зміни ринку, коригуючи власні цінові стратегії. |
| Аналіз ринку | Вебскрапінг допомагає зібрати дані про ринкові тенденції, попит та пропозицію, дозволяючи компаніям ефективно планувати свою діяльність. |
| Дослідження в соціальних науках | Академічні дослідники використовують вебскрапінг для збору даних з соціальних мереж та інших платформ для проведення соціологічних та психологічних досліджень. |
| Збір наукових даних | Використання вебскрапінгу для екстракції наукових публікацій, статей та інших академічних ресурсів для досліджень. |
| Журналістика | Журналісти використовують вебскрапінг для збору інформації з різних джерел для написання статей та проведення розслідувань. |

Продовження таблиці 1.1

| | |
|---|---|
| Рекрутинг | HR-спеціалісти використовують вебскрапінг для збору даних про потенційних кандидатів з соціальних мереж та професійних платформ. |

В таблиці 1.1 наведені найпоширеніші сфери застосування вебскрапінгу, зокрема варто відзначити допомогу в області комерційного ринку, де від аналізу ситуації, а також продукції конкурентів, залежить власне успіхи та цінові стратегії компаній в цілому.

### 1.3 Огляд існуючих аналогів

Системи, які інтегрують можливості NLP з вебскрапінгом, використовуються у різних сферах, від маркетингового аналізу до академічних досліджень. Ці системи дозволяють автоматизувати процес збору текстових даних з Інтернету та їх подальшого аналізу за допомогою технологій обробки природної мови. Нижче наведено деякі існуючі аналоги таких систем:

1) Google Cloud Natural Language API;
2) Diffbot;
3) Import.io;
4) Mozenda;
5) Octoparse.

*Google Cloud Natural Language API* – це потужний інструмент, який дозволяє аналізувати текст та розуміти його структуру та значення. Він включає функції, такі як аналіз сентименту, класифікація тексту та виявлення сутностей. Цей інструмент може бути інтегрований із системами вебскрапінгу для автоматичного збору даних з вебсайтів та їх подальшого аналізу.

Основні функції Google Cloud NLP API включають в себе такі можливості:

– аналіз настрою тексту: API дозволяє визначити настрій тексту, тобто визначити, чи текст позитивний, негативний або нейтральний. Це корисно для аналізу відгуків, соціальних медіа, новин тощо;

– виявлення мови: система може автоматично визначити мову тексту. Це може бути корисно, коли у вас є велика кількість тексту з різних мов;

– виявлення іменованих сутностей: сервіс може визначити різні іменовані сутності, такі як імена осіб, місця, організації, дати тощо. Це може бути корисно для екстракції важливої інформації з тексту;

– відносини між сутностями: API може виявити відносини між різними іменованими сутностями в тексті. Наприклад, визначити, хто є власником певної компанії;

– синтаксичний аналіз: він надає можливість аналізувати синтаксис речень, включаючи визначення залежностей між словами та розпізнавання частин мови;

– класифікація тексту: за допомогою API можна класифікувати текст на основі заданого набору категорій, що допомагає в автоматизованому аналізі текстової інформації;

– екстракція ключових слів: користувачі можуть використовувати API для виділення ключових слів або фраз, які найбільше характеризують текст.

Розрахунок вартості використання Google Cloud NLP API зазвичай залежить від кількості запитів, кількості оброблених символів та обраного рівня підписки (наприклад, безкоштовний, платний план або корпоративний план). Вартість може варіюватися в залежності від регіону та обсягу використання.

*Diffbot* – це платформа, що використовує машинне навчання для перетворення вебсторінок у структуровані дані. Вона автоматично визначає та екстрагує дані з вебсайтів, такі як продукти, люди, статті та організації, і може бути використана для подальшого аналізу даних з використанням NLP.

Основні функції Diffbot включають в себе:

– автоматичний збір даних: Diffbot дозволяє збирати дані з вебсторінок автоматично, без необхідності написання власних скриптів для вебскрапінгу. Він може виділяти текст, зображення, таблиці, ціни товарів та багато іншої інформації з вебсайтів;

– екстракція структурованих даних (Structured Data Extraction): сервіс може розпізнавати та виділяти структуровані дані з вебсайтів, такі як заголовки, категорії, ціни, дати тощо. Це корисно для створення пошукових індексів, агрегаторів новин, електронних магазинів і багатьох інших застосувань;

– аналіз текстового контенту (Text Analysis): присутня можливість аналізувати текстовий контент вебсторінок, включаючи автоматичне визначення мови, витягування ключових слів, аналіз емоційного тону та інше. Це корисно для обробки новинних статей, блогів і коментарів;

– екстракція зображень і відео (Image and Video Extraction): Diffbot може витягувати зображення та відео з вебсайтів, що дозволяє створювати медіа-колекції, аналізувати зміст зображень і відео, створювати галереї та інше;

– аналіз сторінок соціальних медіа (Social Media Page Analysis): Diffbot може аналізувати профілі і сторінки в соціальних медіа, витягувати дані про користувачів, пости, коментарі, лайки тощо. Це корисно для аналізу впливу в соціальних мережах;

– моніторинг змін на вебсторінках (Web Page Monitoring): Diffbot може використовуватися для відстеження змін на вебсторінках і сповіщення про них. Це корисно для моніторингу цін, статей, новин тощо.

*Import.io* – це інструмент, який дозволяє перетворювати вебсторінки в структуровані дані. Це дозволяє користувачам легко збирати дані з вебсайтів без необхідності написання складного коду. Ці дані можуть бути інтегровані з NLP-системами для глибшого аналізу тексту.

Основні функції Import.io включають в себе:

– вебскрапінг (Web Scraping): Import.io надає інструменти для створення краулерів та екстракції даних з вебсторінок. Користувачі можуть створювати шаблони для збору даних зі сторінок, навіть якщо вони мають складну структуру;

– трансформація даних (Data Transformation): після вилучення даних, Import.io дозволяє вам перетворювати, фільтрувати та обробляти ці дані за допомогою різноманітних інструментів. Користувачі можуть виконувати операції, такі як конвертація форматів дат, об'єднання таблиць, видалення дублікатів тощо;

– автоматизація завдань (Task Automation): присутня можливість налаштовувати регулярну автоматизацію завдань для збору та обробки даних. Це корисно для постійного оновлення даних, що змінюються на вебсайтах;

– інтеграція з іншими сервісами (Integration): Import.io може інтегруватися з іншими сервісами та платформами, такими як Google Sheets, Excel, бази даних та інші. Це спрощує імпорт та експорт даних між різними застосунками;

– аналіз даних (Data Analysis): після вилучення та обробки даних, Import.io дозволяє проводити аналіз цих даних, створювати звіти та візуалізації для прийняття інформованих рішень;

– захист даних (Data Security): платформа забезпечує безпеку та конфіденційність ваших даних, включаючи можливість обмеження доступу та шифрування даних;

– підтримка для різних джерел даних (Data Source Support): Import.io може працювати з різними джерелами даних, включаючи вебсайти, API, бази даних, таблиці Excel та інші.

*Mozenda* – це ще один вебскрапінг-інструмент, який дозволяє компаніям збирати та структурувати великі обсяги даних з Інтернету. Mozenda може бути використана разом з NLP-рішеннями для розширеного аналізу текстових даних, отриманих з різних джерел.

Основні функції Mozenda включають в себе:

– створення агентів (Agents Creation): Mozenda дозволяє створювати "агентів" або правила для збору даних з вебсайтів. Користувачі можуть визначити, які дані потрібно видобути, як ці дані повинні бути витягнуті та як вони мають бути оброблені;

– витягнення структурованих даних (Structured Data Extraction): Mozenda витягує структуровані дані з вебсайтів, такі як таблиці, списки товарів, сторінки товарів, адреси, контакти тощо. Користувачі можуть налаштовувати агентів для видобування різних типів даних;

– підтримка для різних джерел даних (Data Source Support): платформа може використовуватися для збору даних з різних джерел, включаючи вебсайти, API, бази даних, файлові системи тощо;

– розкладні завдання (Scheduled Tasks): Mozenda дозволяє налаштовувати регулярні завдання для автоматичного оновлення даних на вебсайтах та інших джерелах. Це корисно для постійного отримання свіжих даних;

– трансформація даних (Data Transformation): користувачі можуть використовувати Mozenda для перетворення, обробки і очищення даних перед їхнім збереженням. Це включає в себе фільтрацію, об'єднання, конвертацію форматів, видалення дублікатів тощо;

– інтеграція з іншими системами (Integration): Mozenda може інтегруватися з іншими системами та платформами, такими як бази даних, електронні таблиці, CRM-системи та інші. Це дозволяє автоматизувати імпорт та експорт даних;

– аналіз та візуалізація даних (Data Analysis and Visualization): користувачі можуть аналізувати та візуалізувати дані, які були зібрані та оброблені за допомогою Mozenda, щоб робити інформовані рішення;

– захист та безпека даних (Data Security): Mozenda забезпечує захист даних та конфіденційність, включаючи можливість обмеження доступу до даних та шифрування.

*Octoparse* – це інструмент автоматизації вебскрапінгу, який дозволяє користувачам легко вилучати та обробляти дані з вебсторінок. Використовуючи Octoparse, можна збирати великі обсяги текстових даних, які потім можуть бути аналізовані за допомогою NLP для виявлення тенденцій, аналізу сентименту та інших завдань.

Користувачі можуть використовувати Octoparse для наступних функцій:

– вебскрапінг: збір даних з вебсайтів, включаючи текст, зображення, таблиці і інші типи даних;

– автоматизований процес створення: створення автоматизованих завдань для збору даних без необхідності програмування;

– екстракція структурованих даних: Витягнення структурованих даних зі сторінок, таких як списки товарів, таблиці цін, адреси тощо;

– збір даних з багатьох сторінок: можливість обходу багатьох сторінок та збору даних із кожної з них;

– паралельне виконання завдань: можливість запускати кілька завдань паралельно для швидкого збору даних;

– регулярне оновлення даних: автоматичне оновлення даних зазначеною періодичністю;

– автоматизована навігація: спрощена автоматична навігація по вебсайту для знаходження та збору даних;

– збереження даних у різних форматах: можливість зберігати зібрані дані у форматах, таких як Excel, CSV, бази даних тощо;

– підтримка JavaScript та AJAX: збір даних, які генеруються за допомогою JavaScript та AJAX-запитів;

– інтеграція з іншими застосунками: можливість інтегрувати Octoparse з іншими програмами та сервісами;

– відслідковування змін на сторінках: виявлення та сповіщення про зміни на вебсайтах;

– розкладні завдання: можливість налаштовувати регулярні завдання для автоматичного збору даних;

– аналіз та візуалізація даних: аналіз та візуалізація зібраних даних для прийняття інформованих рішень;

– захист та безпека даних: забезпечення захисту даних та конфіденційності користувача.

Переваги та недоліки кожної з систем наведені в таблиці 1.2.

Таблиця 1.2 – Перелік переваг та недоліків аналогічних систем.

| Система | Переваги | Недоліки |
|---|---|---|
| Google Cloud Natural Language API | Висока точність та надійність у аналізі тексту. | Вимагає деякого рівня технічної експертизи для налаштування. |
| | Широкий спектр мовних функцій, включаючи аналіз сентименту, виявлення сутностей та синтаксичний аналіз. | Вартість використання може бути високою при великому обсязі даних. |
| Diffbot | Автоматичне визначення та класифікація типів вебсторінок. | Обмежена функціональність у глибокому аналізі тексту порівняно з повноцінними NLP-системами. |
| | Забезпечує API для розширеної автоматизації. | Вартість може бути високою для масштабних проєктів. |
| Import.io | Підтримка масового збору та обробки даних. | Обмежені можливості аналітичної обробки даних на виході. |
| | Можливість перетворення вебсторінок на API без необхідності програмування. | Може вимагати додаткових інструментів для складного аналізу даних. |
| Mozenda | Велика гнучкість та масштабованість для збору даних. | Відсутність вбудованих NLP-функцій для глибокого аналізу тексту. |

Продовження таблиця 1.2

| | | |
|---|---|---|
| | Хороша підтримка та ресурси для навчання користувачів. | Вартість ліцензії може бути високою для деяких користувачів. |
| Octoparse | Можливість обробки складних вебсторінок з динамічним контентом. | Вимагає певного рівня технічних знань для складніших задач. |
| | Підтримка регулярних виразів та Xpath для точної екстракції даних. | Обмежені можливості для безпосереднього аналізу тексту або інтеграції з NLP без додаткового програмування. |

### 1.4 Постановка задачі

У сучасному інформаційному суспільстві, де обсяги цифрових даних стрімко зростають, виникає гостра потреба в ефективних інструментах для їх обробки та аналізу. В цьому контексті, обробка природної мови (NLP) та вебскрапінг відіграють ключову роль у вилученні та обробці текстових даних з веб-ресурсів. Завданням цієї магістерської роботи є створення інтелектуальної системи, яка інтегрує можливості NLP та вебскрапінгу для автоматизованого збору, обробки, аналізу, та інтерпретації великих обсягів текстової інформації з Інтернету. Для досягнення поставленої мети було виокремлено наступні **завдання**:

– оцінка існуючих методів та інструментів у галузі NLP та вебскрапінгу;

– аналіз сильних та слабких сторін існуючих систем, визначення можливостей для покращення;

– створення алгоритмів для ефективного збору текстових даних з різних вебресурсів;

– розробка методів для фільтрації та попередньої обробки зібраних даних;

– впровадження алгоритмів NLP для глибокого аналізу тексту, включаючи визначення сентименту, виявлення ключових словесних зворотів, класифікацію та семантичний аналіз;

– реалізація функцій автоматичного визначення тематики та змісту текстових матеріалів;

– планування та виконання ряду тестів для перевірки функціональності, продуктивності та точності системи;

– аналіз отриманих результатів та вдосконалення системи на основі зворотного зв'язку;

– створення зручного та інтуїтивно зрозумілого інтерфейсу для легкого доступу до функціоналу системи;

– інтеграція візуальних інструментів для демонстрації результатів аналізу;

– дослідження можливих областей застосування розробленої системи, включаючи бізнес-аналітику, медіа, академічні дослідження, тощо;

– оцінка впливу системи на оптимізацію робочих процесів у вказаних областях.

По завершенню виконання завдань очікуються наступні результати:

– розробка інноваційної системи, що відрізняється високою точністю аналізу даних та гнучкістю в застосуванні;

– поліпшення процесів збору та аналізу даних, значне зниження часу та ресурсів, необхідних для обробки великих обсягів інформації;

– створення інструменту, здатного надавати цінні інсайти та сприяти прийняттю обґрунтованих рішень у різних галузях.

В даній постановці задачі розглядаються комплексні цілі, що охоплюють технічні, етичні, юридичні та практичні аспекти розробки та впровадження інтелектуальної системи обробки природної мови з використанням алгоритмів вебскрапінгу, забезпечуючи тим самим глибокий та всебічний підхід до дослідження.

## Висновки до розділу 1

Обробка природної мови (NLP) виявляється динамічною та впливовою областю, що перетинається з багатьма аспектами сучасних технологій. Від синтаксичного аналізу до семантичного розуміння, NLP відіграє ключову роль у трансформації способу, який використовується для взаємодії з машинами, дозволяючи їм інтерпретувати та реагувати на людську мову більш ефективно. Втім, існують виклики, такі як обробка неструктурованих даних, різноманітність мов і мовна неоднозначність.

Вебскрапінг виступає як могутній інструмент для збору величезних обсягів інформації з Інтернету. Він має широкий спектр застосувань, від моніторингу цін до збору наукових даних. Однак, його використання пов'язане з питаннями щодо конфіденційності, авторських прав та етики.

Аналіз різних існуючих систем, що інтегрують NLP з вебскрапінгом, виявив різноманітність підходів та рішень у цій сфері. Платформи як Google Cloud Natural Language API, Diffbot, Import.io та інші, кожна пропонує унікальні функціональності, відкриваючи різні можливості для аналізу даних.

Постановка задачі демонструє прагнення розвинути систему, що ефективно поєднує NLP та вебскрапінг для глибшого та точнішого аналізу даних. Ця система має потенціал вирішити існуючі виклики в області обробки великих обсягів інформації, забезпечуючи більш глибоке розуміння зібраних даних.

Перший розділ підкреслює значення та потенціал інтеграції NLP та вебскрапінгу, виявляючи можливості та виклики, що стоять перед цією областю. Чітка постановка задачі та огляд існуючих рішень вказують на значний потенціал для розробки новаторської системи, яка могла б ефективно вирішити поточні виклики і відкрити нові горизонти у сфері обробки і аналізу даних.

# 2 МЕТОДИ ОБРОБКИ ПРИРОДНОЇ МОВИ ТА ВЕБСКРАПІНГУ

## 2.1 Методи обробки природної мови

В 1950-х роках, у розпал Холодної війни, виникла значна зацікавленість у машинному перекладі, особливо між англійською та російською мовами. Цей інтерес був частково викликаний політичною потребою у швидкому і точному перекладі між цими мовами.

Ранні системи NLP фокусувалися на прямому перекладі, використовуючи великі словники та набори граматичних та семантичних правил. Ці системи не володіли можливістю "розуміти" мову у сучасному розумінні цього слова; вони просто застосовували прямі правила для перекладу слів і фраз [1].

Нижче наведена табл. 2.1, яка зводить ключові моменти в історії розвитку обробки природної мови (NLP).

Таблиця 2.1 – Ключові моменти в історії розвитку обробки природної мов

| **Етап** | **Опис** | **Приклади** |
|---|---|---|
| Ранні системи NLP | Зосереджені на машинному перекладі, базуються на правилах. Використовують великі словники та набори граматичних правил. | Система перекладу між мовами (1950-ті). |
| Розвиток статистичних методів | Прогрес у 1980-х та 1990-х роках завдяки статистичним методам. Використання таких методів як статистичний машинний переклад і Hidden Markov Models. | Статистичний машинний переклад, Hidden Markov Models. |

Продовження таблиці 2.1

| | | |
|---|---|---|
| Вплив машинного навчання | Сучасний NLP значно змінився завдяки технікам машинного навчання. Методи глибинного навчання, такі як нейронні мережі, стали ключовими. | Моделі на основі глибинного навчання (наприклад, BERT, GPT). |

Ця таблиця показує, як з часом еволюціонували підходи до обробки природної мови, починаючи від базованих на правилах систем і закінчуючи сучасними техніками глибинного навчання.

**Перші практичні застосування**

Одним із перших значних зусиль у цій галузі був проєкт, що мав на меті розробку системи для автоматичного перекладу між англійською та російською мовами. Цей проєкт був мотивований не лише науковими, а й практичними цілями, оскільки існувала велика потреба у перекладі наукових текстів та технічної документації [1].

**Методологія**

Ці системи машинного перекладу зазвичай використовували двоступеневий процес:

1. Аналіз джерельного тексту: визначення граматичної структури та значення слів у реченні.

2. Генерація цільового тексту: використання правил для перекладу аналізованого тексту на цільову мову.

**Виклики та обмеження:**

– *буквальний переклад:* через відсутність глибокого розуміння контексту, ці системи часто здійснювали буквальний переклад, ігноруючи нюанси та ідіоматичні вирази;

– *обмежені словники:* словники та правила, які використовувалися у цих системах, були обмежені та не завжди відображали реальну мовну різноманітність;

– *висока помилковість:* через різні структури мов і обмеження технологій того часу, якість перекладу часто залишала бажати кращого.

Кругова діаграма, представлена на рис. 2.1, візуалізує відносну частку викликів та розвитку у трьох ключових етапах розвитку обробки природної мови (NLP).

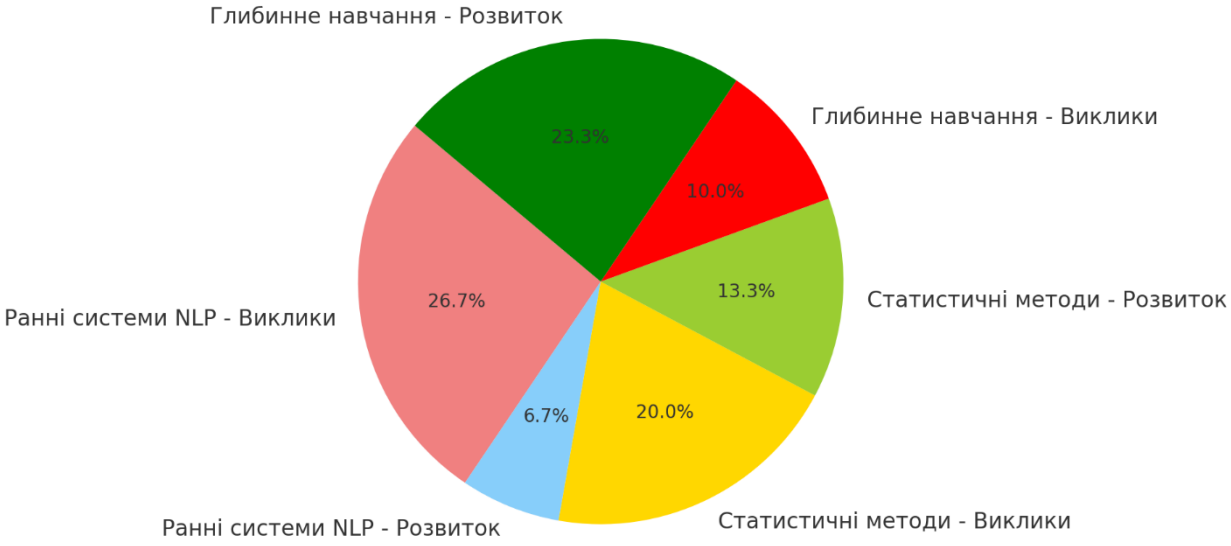


Рисунок 2.1 – Виклики та розвиток у методах обробки природної мови

1. **Ранні системи NLP**:

– *виклики (80%)*: велика частина цього сектора підкреслює значні виклики, які стояли перед ранніми системами NLP. Ці виклики включали обмеженість словників, буквальний переклад, та відсутність глибокого розуміння мовних нюансів;

– *розвиток (20%)*: невелика частина представляє розвиток, який був досягнутий на цьому етапі, зокрема розвиток базових алгоритмів перекладу та аналізу мови.

2. **Статистичні методи**:

– *виклики (60%)*: ця частина діаграми вказує на виклики, з якими стикалися статистичні методи, такі як залежність від великих обсягів даних та обмежена можливість розуміння контексту;

– *розвиток (40%)*: показує значний прогрес, досягнутий завдяки впровадженню статистичних методів, включаючи більшу точність та надійність систем NLP.

3. **Глибинне навчання**:

– *виклики (30%)*: хоча сучасні методи, засновані на глибинному навчанні, мають менше викликів, такі як потреба в значних обчислювальних ресурсах та складність інтерпретації моделей, вони все ще існують;

– *розвиток (70%)*: найбільша частина діаграми, що відображає значний розвиток, досягнутий завдяки глибинному навчанню. Це включає в себе вдосконалення точності, здатності до розуміння контексту та гнучкості у застосуванні.

Ця діаграма підкреслює, як з часом NLP еволюціонував від простих, правилами керованих систем до складних моделей, заснованих на глибинному навчанні, причому кожен новий етап приніс з собою як нові можливості, так і нові виклики.

Цей період у розвитку NLP мав велике значення, оскільки поклав початок більш глибоким дослідженням у галузі мовних технологій та стимулював подальший розвиток в області комп'ютерних наук і лінгвістики [2].

**Додаткові аспекти раннього розвитку NLP:**

1. *програмування на основі правил:* ранні системи NLP в значній мірі покладалися на програмування на основі правил. Ці правила були розроблені лінгвістами та програмістами і часто включали величезні списки виключень і умов, що ускладнювало підтримку та розширення системи;

2. *перші конференції та робочі групи:* у цей період почали з'являтися перші конференції та наукові робочі групи, спрямовані на обмін знаннями та досвідом у галузі машинного перекладу, що свідчило про зростаючий інтерес до цієї області;

3. *співпраця між різними галузями:* розвиток NLP стимулював співпрацю між комп'ютерними науками, лінгвістикою, психологією та іншими дисциплінами, що

дозволило краще зрозуміти складність людської мови та способи її моделювання [2].

На рис. 2.2 зображені додаткові аспекти раннього розвитку обробки природної мови (NLP).

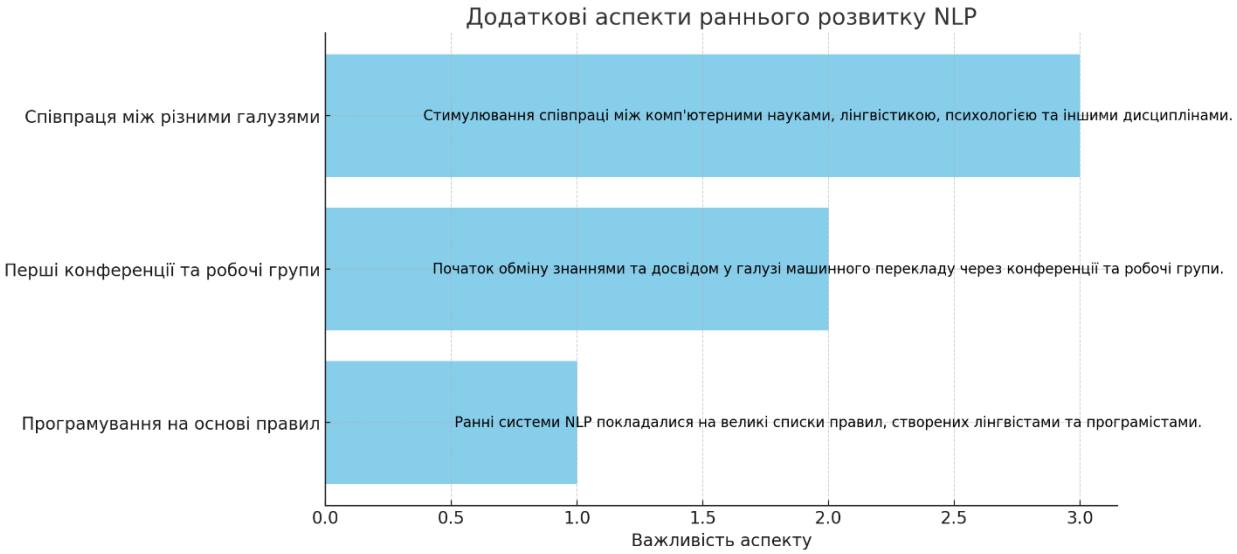


Рисунок 2.2 – Додаткові аспекти раннього розвитку обробки природної мови (NLP)

**1. Програмування на основі правил**.

В цьому аспекті висвітлено, що ранні системи NLP значною мірою покладалися на великі списки правил, які були створені лінгвістами та програмістами. Ці правила включали детальні граматичні інструкції та виключення, необхідні для обробки мови.

**2. Перші конференції та робочі групи**.

Цей аспект підкреслює початок активного обміну знаннями та досвідом у галузі машинного перекладу, що відбувався через організацію конференцій та створення робочих груп. Це сприяло подальшому розвитку NLP як наукової дисципліни.

**3. Співпраця між різними галузями**.

Важливість цього аспекту полягає в тому, що він відображає стимулювання співпраці між різними дисциплінами, такими як комп'ютерні науки, лінгвістика,

психологія та інші. Це міждисциплінарне взаємодія дозволила більш глибоко зрозуміти та моделювати складності людської мови.

Ця схема візуально підкреслює, як різні аспекти сприяли ранньому розвитку NLP, демонструючи еволюцію від простих правил до більш складних та міждисциплінарних підходів [3].

## 2.2 Методи та інструменти вебскрапінгу

**Вебскрапінг** – це процес автоматизованого збору даних з вебсторінок. Це включає витягування конкретної інформації, що може використовуватися для різних цілей, наприклад, аналізу ринку, моніторингу цін, збору контактної інформації тощо.

При використанні вебскрапінгу важливо враховувати юридичні та етичні аспекти. Це включає дотримання авторських прав, умов використання вебсайтів, а також захист приватності та персональних даних користувачів.

Знання HTML та CSS є критично важливим для ефективного скрапінгу, оскільки це основні мови, якими побудовані вебсторінки. XPath використовується для навігації по елементах сторінки та їх вибору.

1. Популярні бібліотеки для скрапінгу:

– *BeautifulSoup*: легка у використанні бібліотека для Python, яка дозволяє витягувати дані з HTML та XML файлів;

– *Scrapy*: інша бібліотека Python, яка використовується для створення масштабованих вебскраперів. Вона має можливості для обробки запитів, збору даних та їх обробки.

2. Робота з API та динамічним контентом.

Багато сучасних вебсайтів використовують API для надання даних у форматі JSON чи XML, що може бути більш ефективним для збору даних. Також важливо вміти працювати з динамічним контентом, який генерується за допомогою JavaScript.

Ефективний скрапінг вимагає розуміння та використання різноманітних технік та інструментів. Знання HTML та CSS є критично важливим, адже вони є основою будови вебсторінок. XPath використовується для точної навігації та вибору елементів на цих сторінках. Існує також ряд спеціалізованих бібліотек та фреймворків, розроблених спеціально для потреб вебскрапінгу, таких як BeautifulSoup та Scrapy, які значно полегшують цей процес. Крім того, важливим аспектом є вміння працювати з API та динамічним контентом, особливо у контексті сучасних вебзастосунків, що використовують JavaScript. Нижче наведено табл. 2.2, яка підсумовує основні аспекти та інструменти, необхідні для ефективного вебскрапінгу [11].

Таблиця 2.2 – Основні інструменти, які необхідні для ефективного вебскрапінгу

| **Інструмент** | **Опис** | **Використання** |
|---|---|---|
| HTML/CSS знання | Фундаментально для розуміння структури вебсторінок та витягування даних. | Базовий вебскрапінг та екстракція даних. |
| XPath | Використовується для навігації та вибору елементів на вебсторінці. | Точний вибір точок даних на вебсторінці. |
| BeautifulSoup | Python бібліотека для витягування даних з HTML та XML файлів. Зручна у використанні. | Прості до середньої складності завдання скрапінгу. |
| Scrapy | Python фреймворк для масштабного вебскрапінгу. Має можливості обробки даних та запитів. | Комплексні та масштабні операції вебскрапінгу. |

Продовження таблиці 2.2

| | | |
|---|---|---|
| API | Багато сучасних вебсайтів надають дані у форматі JSON або XML через API, що може бути більш ефективно для збору даних. | Доступ до структурованих даних безпосередньо з вебсайтів. |
| Динамічний Контент | Можливість обробки та скрапінгу контенту, який генерується за допомогою JavaScript, що є важливим для сучасних вебзастосунків. | Скрапінг вебсайтів, які значною мірою використовують JavaScript для генерації контенту. |

Використання цих інструментів та технік у вебскрапінгу дозволяє ефективно збирати, обробляти та аналізувати дані з вебсторінок. Від простого збору текстової інформації до складного аналізу динамічного контенту, ці інструменти та техніки є незамінними для дослідників, розробників та аналітиків у багатьох галузях. Завдяки постійному розвитку та вдосконаленню цих інструментів, можливості вебскрапінгу продовжують розширюватися, відкриваючи нові горизонти для збору та аналізу вебданих [7].

**Виклики та рішення у вебскрапінгу.**

**1. Обходження захисту від ботів**: багато вебсайтів мають механізми захисту від ботів, такі як CAPTCHA, які можуть ускладнити автоматизований збір даних. Існують різні методики та інструменти для обходження цих механізмів, але важливо використовувати їх відповідно до юридичних та етичних норм.

**2. Робота з великими об'ємами даних**: ефективна обробка та зберігання зібраних даних є ключовою для успішного вебскрапінгу. Це включає використання баз даних та хмарних технологій для зберігання та аналізу великих обсягів даних.

**3. Автоматизація та оптимізація процесу скрапінгу**: розробка ефективних скраперів, які можуть автоматично обробляти запити та адаптуватися до змін у

структурі вебсторінок, є важливою задачею. Використання планувальників завдань та оптимізація запитів може значно підвищити ефективність збору даних.

На рис. 2.3 наведена стовпчаста діаграма, яка візуалізує виклики та рішення у сфері вебскрапінгу.

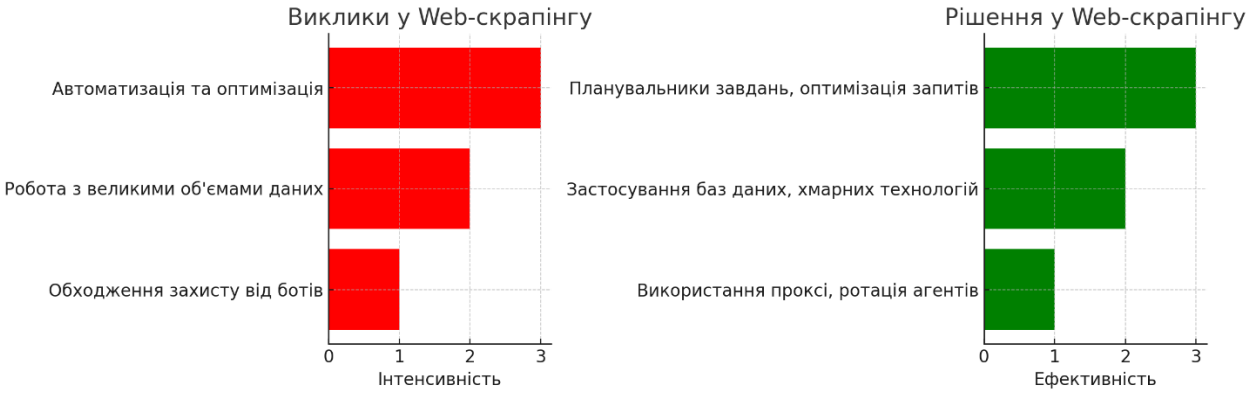


Рисунок 2.3 – Виклики та рішення у сфері вебскрапінгу

Ліва діаграма демонструє виклики.

**1. Обходження захисту від ботів**: висока інтенсивність виклику, вказує на складність обходження систем захисту вебсайтів.

**2. Робота з великими об'ємами даних**: помірна інтенсивність, підкреслює важливість ефективного збору та обробки великих даних.

**3. Автоматизація та оптимізація**: найменша інтенсивність серед викликів, але все ще важлива для ефективного скрапінгу.

Права діаграма відображає рішення.

**1. Використання проксі, ротація агентів**: ефективний спосіб обходження захисту від ботів.

**2. Застосування баз даних, хмарних технологій**: важливе рішення для роботи з великими об'ємами даних.

**3. Планувальники завдань, оптимізація запитів**: необхідні для підвищення ефективності скрапінгу.

Ця діаграма показує баланс між складнощами та можливими рішеннями в області вебскрапінгу, підкреслюючи важливість інноваційних підходів та технік для подолання цих викликів.

## 2.3 Sentiment-аналіз та quadruple extraction: методологія та застосування

Sentiment-аналіз, або аналіз емоційних забарвлень, — це процес визначення та категоризації емоційних виразів у тексті, зазвичай як позитивні, негативні або нейтральні. Цей аналіз має велике значення для розуміння сприйняття брендів, продуктів, політичних поглядів тощо.

Використання різноманітних технік, включаючи правила, машинне навчання, та глибинне навчання. Алгоритми, як-от Naive Bayes, Logistic Regression, та нейронні мережі, часто застосовуються в цій області (табл. 2.3).

Таблиця 2.3 – Основні методи та алгоритми, які використовуються у Sentiment аналізі

| Метод/Алгоритм | Опис | Застосування |
|---|---|---|
| Наївний Баєсівський класифікатор | Простий ймовірнісний класифікатор, базується на застосуванні теореми Баєса. Популярний для базового sentiment аналізу. | Прості завдання sentiment аналізу, аналіз соціальних медіа. |
| Логістична регресія | Статистична модель, що використовується для прогнозування ймовірності належності до певної категорії. | Розширений аналіз відгуків, моніторинг бренду. |
| Нейронні мережі | Глибинні навчальні моделі, які можуть виявляти складні закономірності в текстових даних. | Складний аналіз тексту, зокрема, із залученням контексту та іронії. |

Продовження таблиці 2.3

| | | |
|---|---|---|
| Support Vector Machines (SVM) | Алгоритми, які використовуються для класифікації та регресійного аналізу. Ефективні в текстових класифікаціях. | Текстова класифікація з великим набором функцій. |

Ця таблиця надає огляд найпопулярніших методів та алгоритмів, використовуваних у Sentiment аналізі, їх основні характеристики та сфери застосування. Від простих до більш складних технік, кожен метод має свої переваги та найкраще підходить для певних типів задач [10].

Quadruple extraction (витягування четвірок) є процесом ідентифікації та структурування інформації з текстів у формі четвірок, які складаються з суб'єкта, об'єкта, присудка, та додаткової інформації (наприклад, часу, місця).

Застосування технік з обробки природної мови для виявлення відносин між елементами у реченні. Використання синтаксичного аналізу, семантичного парсингу, та глибинного навчання для визначення структури та змісту інформації.

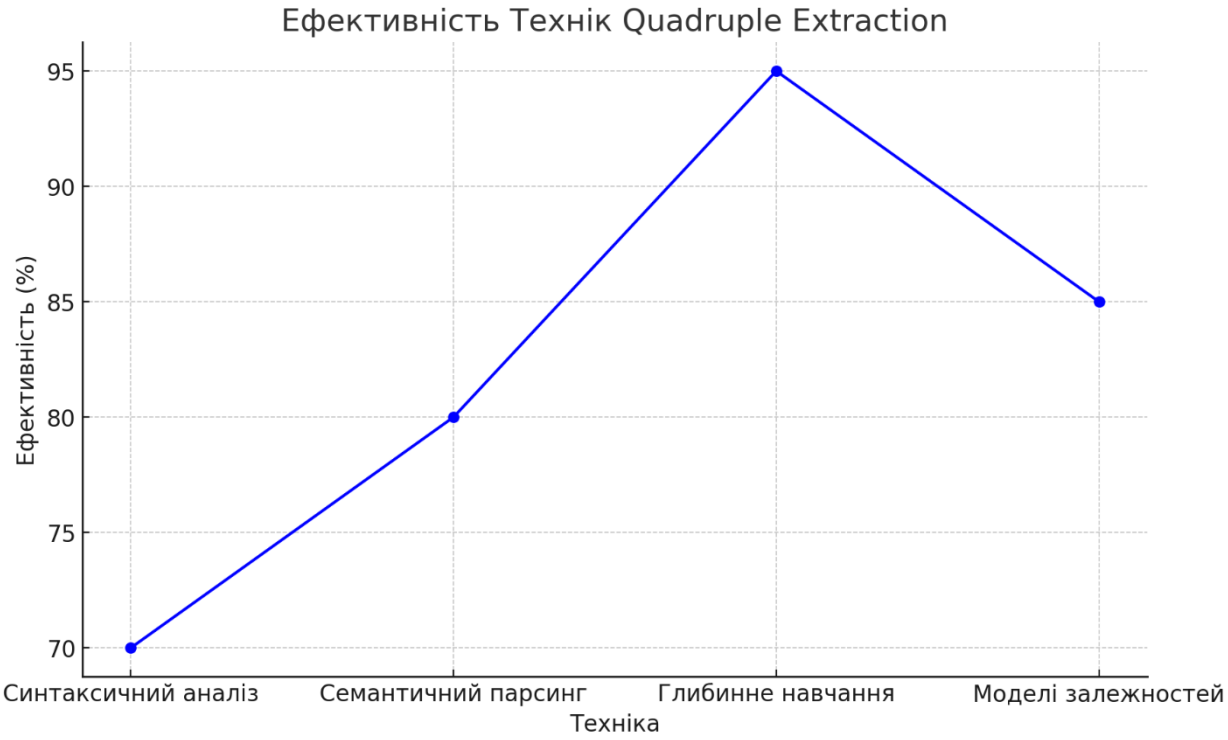


Рисунок 2.4 – Графік, який візуалізує прикладну ефективність різних технік та підходів в рамках Quadruple extraction

На рис. 2.4 представлено чотири основні техніки:

**1. синтаксичний аналіз**: має відносно високу ефективність (70%), що підкреслює його важливість у виявленні структури речення.

**2. семантичний парсинг**: показує ще вищу ефективність (80%), відображаючи його спроможність розуміти значення слів у контексті.

**3. глибинне навчання**: виділяється найвищою ефективністю (95%), що вказує на його здатність виявляти складні закономірності в тексті.

**4. моделі залежностей**: також демонструє високу ефективність (85%), підкреслюючи їх корисність у виявленні залежностей між елементами тексту.

Цей графік ілюструє, як різні техніки можуть використовуватися для ефективного Quadruple extraction, кожна з яких має свої унікальні переваги та застосування [9].

**Приклади застосування у різних сферах.**

Аналіз новинних статей для виявлення ключових подій, аналіз наукових текстів для виявлення зв'язків між різними концептами, використання у CRM системах для розуміння взаємовідносин між клієнтами та продуктами тощо.

Нижче наведено табл. 2.4, що демонструє приклади застосування Quadruple extraction у різних сферах.

Таблиця 2.4 – Приклади застосування Quadruple extraction

| Сфера | Застосування |
|---|---|
| Новинний аналіз | Виявлення ключових подій, фігур, місць та часу в новинах. |
| Наукові дослідження | Аналіз наукових текстів для виявлення зв'язків між концептами, дослідженнями, авторами та інституціями. |
| Бізнес-аналітика | Аналіз бізнес-документів для ідентифікації важливих бізнес-процесів, учасників та їх взаємодій. |
| CRM системи | Розуміння взаємовідносин між клієнтами, продуктами, взаємодіями та реакціями. |

Ця таблиця показує, як Quadruple extraction може бути використаний для різних цілей, від аналізу новин та наукових текстів до бізнес-аналітики та управління відносинами з клієнтами. Цей інструмент забезпечує глибоке розуміння тексту, дозволяючи виявити ключові елементи та їх взаємозв'язки, що є критично важливим для прийняття обґрунтованих рішень у цих сферах [13].

**Практичні приклади та дослідження.**

1. *Аналіз соціальних медіа:* використання Sentiment аналізу для визначення загального настрою користувачів щодо певних тем, брендів, або подій у соціальних медіа.

2. *Аналіз відгуків споживачів:* застосування Sentiment аналізу для аналізу відгуків клієнтів про продукти або послуги, допомагаючи компаніям розуміти сприйняття споживачами та виявляти потенційні проблеми.

3. *Використання у бізнес-аналітиці:* застосування Quadruple extraction для аналізу бізнес-документів, таких як звіти, договори, щоб виявити важливі бізнес-процеси та відносини (рис. 2.5 – 2.6).

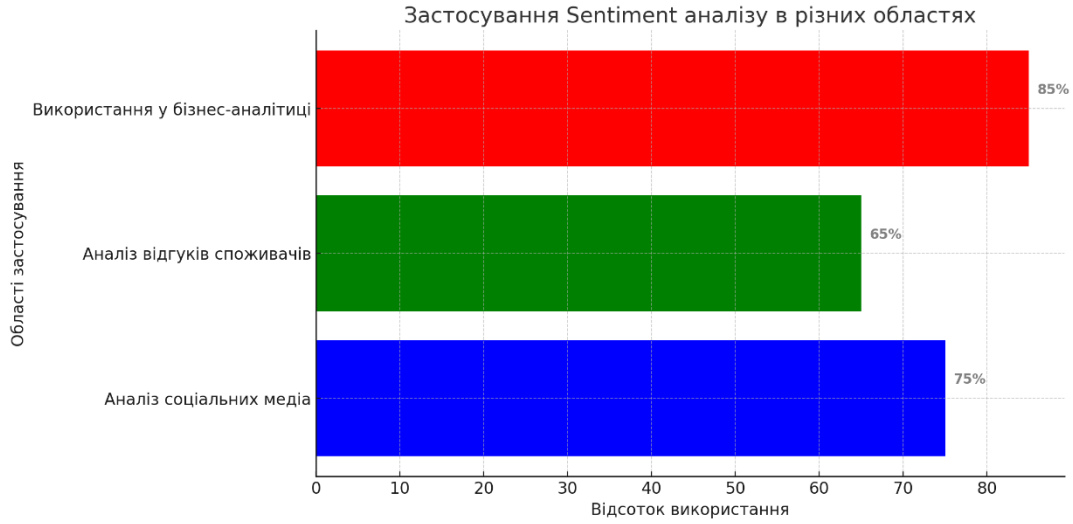


Рисунок 2.5 – Застосування Sentiment аналізу в трьох областях

Ця діаграма ефективно демонструє різні застосування Sentiment аналізу в трьох ключових областях: аналізі соціальних медіа, аналізі відгуків споживачів та в бізнес-аналітиці. З даних видно, що найбільше використання Sentiment аналізу спостерігається в області бізнес-аналітики (85%). Аналіз соціальних медіа також виявляє високий рівень застосування (75%). Аналіз відгуків споживачів, хоча й має трохи менший відсоток застосування (65%).

Ці дані підкреслюють універсальність та значущість Sentiment аналізу в сучасному світі, де зростаюча кількість даних з різних джерел може бути ефективно використана для забезпечення більшої інформативності та точності в прийнятті рішень у різних галузях.

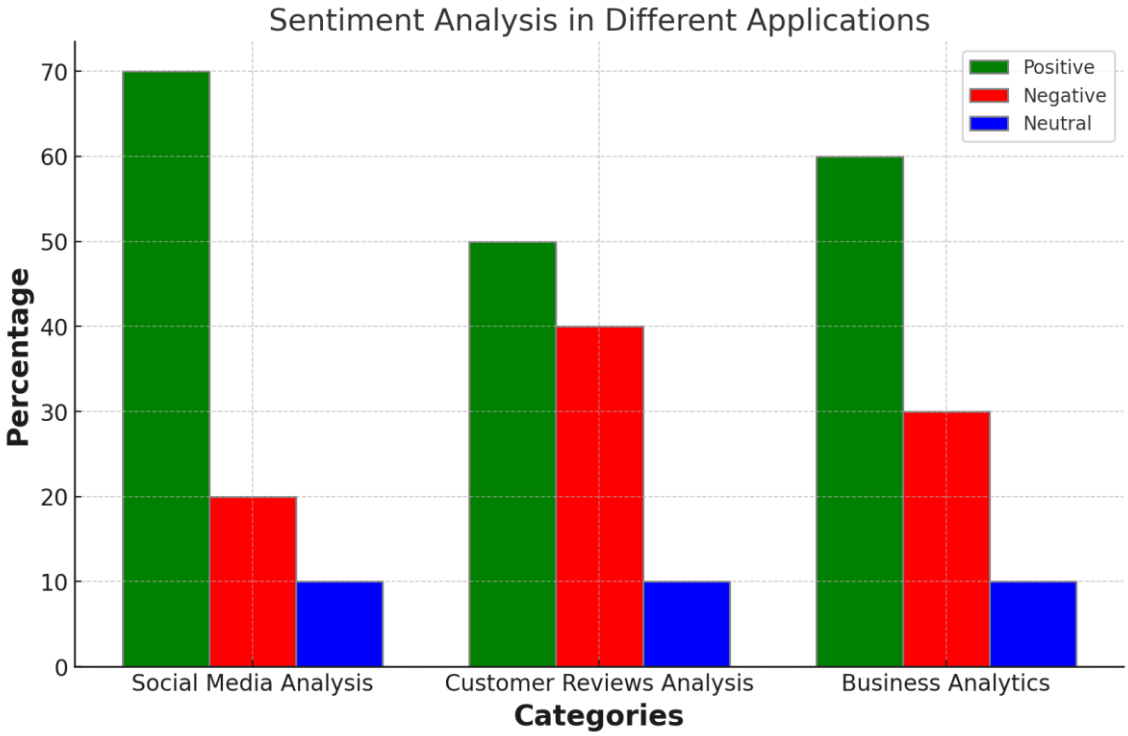


Рисунок 2.6 – Прикладний аналіз емоційних забарвлень (Sentiment Analysis) у трьох різних застосуваннях

На цій діаграмі представлено прикладний аналіз емоційних забарвлень (Sentiment Analysis) у трьох різних застосуваннях: аналіз соціальних медіа, аналіз відгуків споживачів та використання у бізнес-аналітиці. Кожна категорія включає відсоток позитивних, негативних, та нейтральних емоційних оцінок, що дає уявлення про загальний настрій у цих сферах.

**Висновки до розділу 2**

У цьому розділі розглядалися різноманітні методи та техніки обробки природної мови (NLP), які є фундаментальними для розуміння та інтерпретації людської мови машинами. Були проаналізовані такі ключові аспекти, як синтаксичний і семантичний аналіз, а також різні підходи до обробки текстових даних. Цей аналіз вказав на важливість вибору відповідних алгоритмів NLP, здатних ефективно вирішувати конкретні завдання аналізу даних.

Дослідження принципів та інструментів вебскрапінгу, що є ключовими для збору необхідних текстових даних з Інтернету. Було розглянуто різні техніки

вебскрапінгу. Також були висвітлені питання, пов'язані з автоматизацією збору даних, та важливість вибору ефективних інструментів для забезпечення надійності та точності збору даних.

Також у цьому розділі розглядалися sentiment аналіз та методики quadruple extraction як важливих елементів аналізу даних. Було висвітлено, як ці методи можуть використовуватися для глибокого розуміння контексту та значення тексту, а також для виявлення ключових аспектів і відносин у текстових даних. Особлива увага була приділена практичному застосуванню цих методів у різних сферах, від маркетингового аналізу до дослідження споживацької поведінки.

Розділ 2 дав глибоке розуміння необхідних моделей та методів, які будуть використовуватися для вирішення поставленої задачі. Від інструментів та технік вебскрапінгу до складних алгоритмів NLP і методів аналізу сентименту, цей розділ підкреслює багатогранність підходів та необхідність їх інтеграції для створення ефективної системи обробки даних. Він також вказує на необхідність глибокого розуміння кожного аспекту задачі та обрання оптимальних рішень для досягнення цілей дослідження.

# 3 ПРОЄКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ

## 3.1 Загальна архітектура

Інтелектуальна система обробки природної мови з використанням вебскрапінгу являє собою клієнт-серверну архітектуру(див. рис. 3.1), що має два серверні інстанси (Node.js та Python), а також підключення до бази даних.

Дана архітектура має ряд переваг:

– *гнучкість у виборі технологій*: використання двох різних серверних платформ (Node.js для асинхронної обробки запитів та Python для наукових обчислень та роботи з машинним навчанням) дозволяє ефективно розподіляти завдання між ними відповідно до їх сильних сторін;

– *масштабованість*: система легко масштабується завдяки можливості незалежного масштабування кожного компонента (клієнт, сервери, база даних), що дозволяє збільшувати обчислювальні ресурси відповідно до потреб;

– *висока доступність*: розподіленість системи на кілька серверів може забезпечити високу доступність та надійність, оскільки відмова одного компонента не обов'язково призведе до зупинки всієї системи;

– *спеціалізація компонентів*: кожен компонент системи може бути спеціалізований під конкретні завдання, наприклад, Node.js може обробляти веб-запити та взаємодію з користувачами, тоді як Python може займатися обробкою даних, машинним навчанням та аналітикою;

– *гнучкість* інтеграції: система може легко інтегруватися з іншими сервісами та базами даних завдяки використанню стандартних протоколів обміну даними і API, що забезпечує гнучкість у розширенні та інтеграції з іншими системами;

– *оптимізація продуктивності*: можливість використання спеціалізованих серверів для різних типів обробки (асинхронна обробка запитів у Node.js і важкі обчислення в Python) дозволяє оптимізувати продуктивність та ефективність системи;

– *безпека*: розділення компонентів системи також може підвищити її безпеку, оскільки можна ізолювати чутливі процеси та дані в окремих сервісах, мінімізуючи ризики для всієї системи.

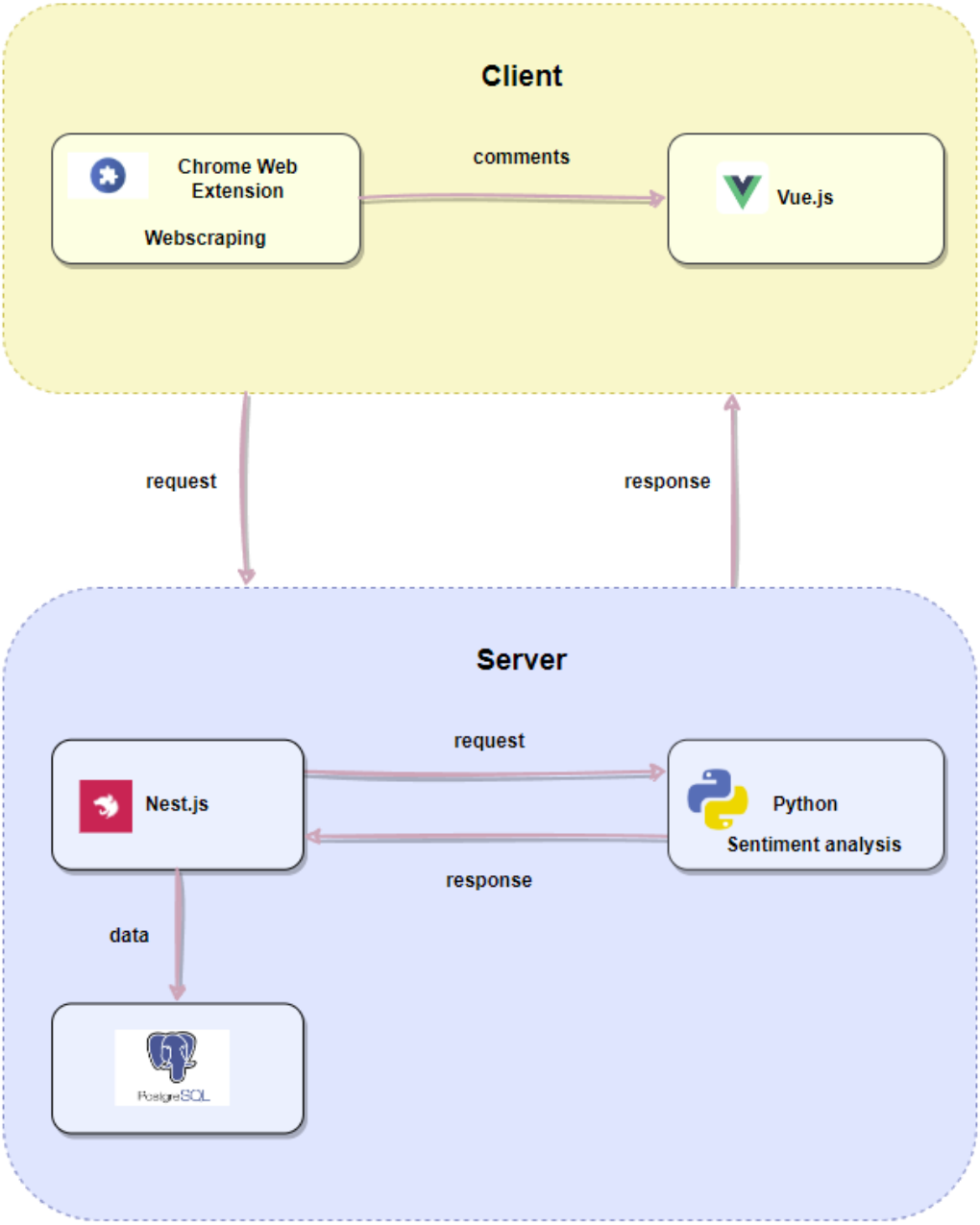


Рисунок 3.1 – Архітектура системи

Ця архітектура забезпечує гнучке та ефективне рішення для розробки складних інтелектуальних систем обробки природної мови, що вимагають високої продуктивності, надійності та масштабованості.

### 3.2 Проєктування бази даних

Для проєктування інтелектуальної системи був побудований план реалізації бази даних, адже необхідність розуміння основних сутностей є ключовою складовою побудови та масштабування програмного забезпечення. Таким чином першою таблицею було створено *migrations* (див. рис. 3.2).

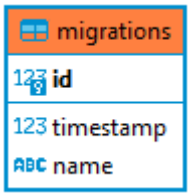


Рисунок 3.2 – Структура таблиці migrations

Дана таблиця включає в собі наступні поля:

– *id:* унікальний ідентифікатор;

– *timestamp:* часовий ідентифікатор;

– *name:* ім’я міграції, що використовується для порівняння з новоствореними та досі не застосованими міграціями.

Кожного разу, коли користувач за допомогою Nest.js CLI створює нову міграцію, вона автоматично отримує свій часовий ідентифікатор, до якого додається те ім’я, яке користувач додає при створені. Таким чином при виконанні міграційних файлів, спеціальний метод буде перепровіряти наявність тієї чи іншої міграції в базі даних, і за умови відсутності та помилки NotFound, міграція буде виконана.

Наступним кроком було створення таблиці користувачів (див. рис. 3.3), яка використовується для ідентифікації окремого пересічного користувача.

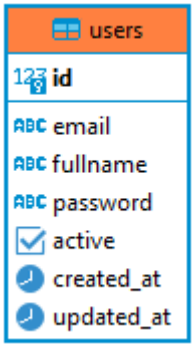


Рисунок 3.3 – Структура таблиці users

До таблиці користувачів були додані наступні поля:

– *id:* унікальний ідентифікатор;

– *email*: унікальна почтова адреса;

– *fullname*: повне ім’я;

– *password*: хешований пароль;

– *active*: поле, що визначає чи має право користувач на використання системи, приймає значення true або false.

– *created_at*: дата створення запису;

– *updated_at*: дата останнього оновлення запису.

Таблиця має основні як для програмування застосунків поля, яких буде цілком достатньо для процесу авторизації та менеджменту користувачів.

На цьому етапі проєктування інтелектуальної системи важливо врахувати, як сутності, що стосуються конкретного контексту, взаємодіють між собою та як вони пов'язані з основною сутністю, у нашому випадку — таблицею компаній (див. рис. 3.4). Створення цих сутностей та визначення зв'язків між ними допоможе у формуванні структурованої бази даних, яка буде ефективною для зберігання та обробки інформації в межах системи.

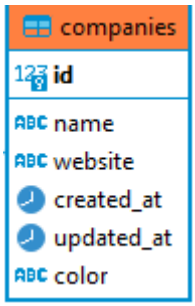


Рисунок 3.4 – Структура таблиці companies

До таблиці користувачів були додані наступні поля:

– *id:* унікальний ідентифікатор;

– *name*: повне ім’я;

– *website*: офіційний вебсайт компанії;

– *color*: унікальний колір компанії;

– *created_at*: дата створення запису;

– *updated_at*: дата останнього оновлення запису.

Наступним кроком було додано таблицю resources (див. рис. 3.5), яка, попри свої власні поля, має ключ до таблиці компаній, так як між даними таблицями є зв'язок One-To-Many, себто одна компанія може мати безліч ресурсів. Ресурси відповідають за перелік посилань на офіційні сторінки компаній в соціальних мережах, звідки скрапляться пости та коментарі до них.

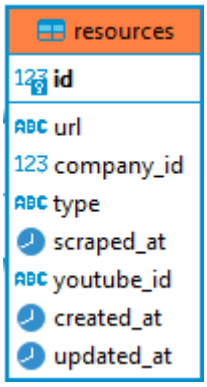


Рисунок 3.5 – Структура таблиці resources

Серед полів даної таблиці пропонуються наступні поля:

– *id:* унікальний ідентифікатор;

– *url*: посилання на ресурс;

– *company_id*: ідентифікатор компанії;

– *type*: тип ресурсу;

– *youtube_id*: ідентифікатор каналу в YouTube (якщо такий є);

– *scraped_at*: дата останнього скрапінгу ресурсу;

– *created_at*: дата створення запису;

– *updated_at*: дата останнього оновлення запису.

Ресурс є дуже важливою частиною застосунку, адже саме він зберігає посилання на соц. мережу, з якої будуть отримані дані для подальшого опрацювання. У системі ресурс може мати компанію, або може бути незалежним, але у такому випадку аналіз ресурсу не потрапляє до загального порівняння компаній на графіках. Типом ресурсу в даному випадку може виступати одне з трьох значень: facebook, linkedin, youtube. У разі створення ресурсу youtube, користувач також має додати ідентифікатор каналу, адже за його допомогою буде здійснюватися запит на офіційне API.

Переходячи до таблиць з даними, першою створюється posts (див. рис. 3.6), що відповідає за зберігання постів у системі.

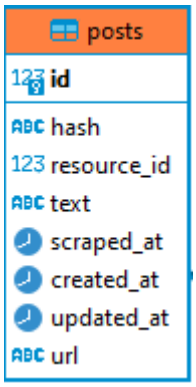


Рисунок 3.6 – Структура таблиці posts

Необхідними полями даної таблиці є:

– *id:* унікальний ідентифікатор;

– *hash*: унікальний хеш, який створюється на основі тексту посту та ідентифікатору компанії;

– *resource_id*: ідентифікатор ресурсу;

– *text*: контент посту;

– *url*: посилання на пост;

– *scraped_at*: дата останнього скрапингу посту;

– *created_at*: дата створення запису;

– *updated_at*: дата останнього оновлення запису.

Пост необхідний для правильної структуризації коментарів, та не приймає участі у аналізі тексту. Кожен ресурс може мати безліч постів, тому між двома сутностями встановлено зв'язок One-To-Many**.**

Таблиця коментарів (див. рис. 3.7) є найбільш задіяною серед усіх наявних сутностей в системі. Вона зберігає тексти коментарів, їх сентімент та емоційне забарвлення з аспектами.

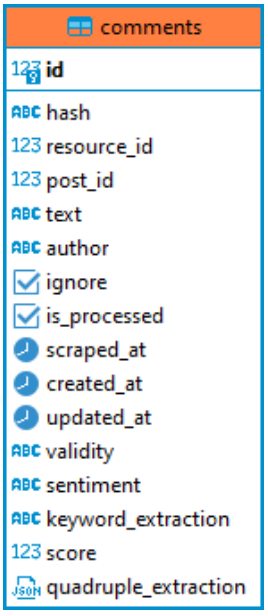


Рисунок 3.7 – Структура таблиці comments

Для повноцінного використання таблиці comments були додані наступні поля:

– *id:* унікальний ідентифікатор;

– *hash*: унікальний хеш, який створюється на основі тексту коментаря, ідентифікатору компанії та ідентифікатору посту;

– *resource_id*: ідентифікатор ресурсу;

– *post_id*: ідентифікатор посту;

– *text*: контент коментаря;

– *author*: автор коментаря;

– *ignore*: чи треба ігнорувати коментар при аналізі;

– *is_processed*: чи був коментар проаналізований раніше;

– *scraped_at*: дата останнього скрапингу коментаря;

– *created_at*: дата створення запису;

– *updated_at*: дата останнього оновлення запису.

– *validity*: статус валідності коментаря;

– *sentiment*: результат аналізу на сентімент;

– *keyword_extraction*: результат аналізу основних слів;

– *score*: результат аналізу на сентімент у числовому еквіваленті;

– *quadruple_extraction*: результат аналізу аспектів.

Таблиця коментарів відіграє значущу роль в системі, так як всі аналізи та розрахунки відбуваються за допомогою контенту коментарів.

Для побудови окремих графіків та відслідковування результату скрапінгу коментарів була створена таблиця comments_history (див. рис. 3.8), що посилається на батьківський коментар, та має ті ж самі поля, що стосуються аналізу, як і батьківська таблиця.

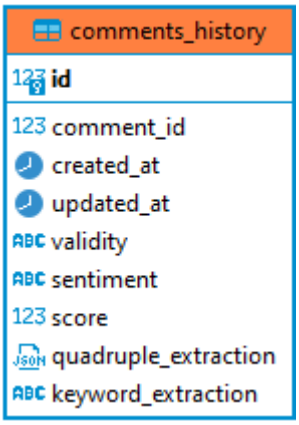


Рисунок 3.8 – Структура таблиці comments history

Таблиця історії коментарів стала рушієм для оформлення графіків, які можна фільтрувати по даті – такі графіки є масштабованими та інформативними, так як дозволяють користувачу одразу на декількох періодах оцінити результат обробки інформації.

Останньою таблицею у системі є історія підписників (див. рис. 3.9), яка відповідає за історію підписників кожного ресурсу. Дана інформація використовується у графіках для аналізу конкурентів.

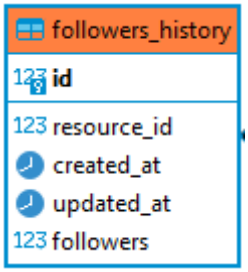


Рисунок 3.9 – Структура таблиці followers history

Полями даної таблиці є:

– *id:* унікальний ідентифікатор;

– *resource_id*: ідентифікатор ресурсу;

– *followers*: кількість підписників;

– *created_at*: дата створення запису;

– *updated_at*: дата останнього оновлення запису.

Загальна структура бази даних (див. рис. 3.10) була розроблена таким чином, щоб забезпечити гнучкість та масштабованість, а також легкість обслуговування та оновлення.

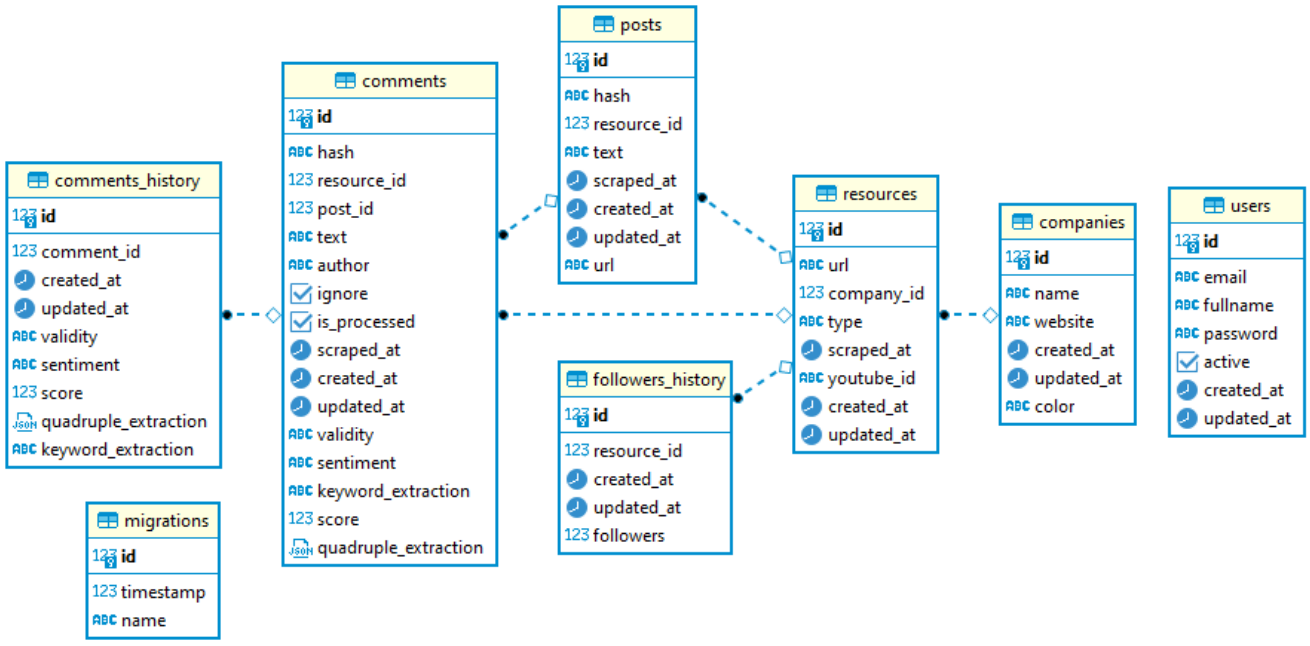


Рисунок 3.10 – ER-діаграма бази даних

Враховуючи необхідність високого рівня нормалізації для уникнення редундантності та забезпечення цілісності даних, були впроваджені чіткі зв'язки між таблицями, включаючи первинні та зовнішні ключі, що дозволяє точно відображати відносини між сутностями.

### 3.3 Розробка датасету

Для використання моделі ASQE (Aspect Sentiment Quad Prediction) з бібліотеки PyABSA, спочатку потрібно провести її тренування. PyABSA надає інструменти для аналізу сентименту в текстах, зокрема для виявлення аспектів і оцінки емоційного забарвлення тексту щодо цих аспектів. Для ефективної роботи моделі ASQE необхідно тренувати її на відповідному наборі даних, який містить приклади текстів з розміченими аспектами та оцінками емоційного забарвлення.

Процес тренування моделі ASQE включає наступні кроки:

– *підготовка даних*: необхідно підготувати набір даних, де кожен елемент містить текст, аспект, до якого він відноситься, та оцінку сентименту щодо цього аспекту. Дані повинні бути розмічені відповідно до вимог PyABSA.

– *конфігурація тренування*: необхідно визначити параметри тренування моделі, включаючи кількість епох, розмір партії (batch size), швидкість навчання (learning rate) та інші, що впливають на процес тренування та якість моделі.

– *тренування моделі*: необхідно запустити процес тренування, використовуючи підготовлені дані та налаштування. Під час тренування модель навчається розпізнавати аспекти у текстах та визначати емоційне забарвлення стосовно цих аспектів.

– *оцінка моделі*: після тренування моделі важливо оцінити її ефективність на тестовому наборі даних, щоб переконатися в її здатності точно аналізувати сентимент.

– *застосування моделі*: по завершенню тренування та оцінки моделі можна використовувати її для аналізу сентименту в нових текстах.

Для аналізу методів машинного навчання була обрана сфера автомобілів. Вибір даної сфери був зумовлений тим, що популярні автовиробники мають досить велику кількість підписників на своїх офіційних сторінках у соціальних мережах, та відповідно, більше коментарів та активностей користувачів під постами. Це допомагає в повну міру використовувати скрапінг та збирати велику кількість даних для обробки.

Попередньо проаналізувавши найпопулярніші марки авто та їх сторінки в соц. мережах, було обрано коментарі користувачів у розмірі 500 записів, та відформатовано їх до такого виду, якого потребує модель для навчання. Отриманий датасет (див. рис. 3.10) можна використовувати для тренування моделі.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Brand** | **Comment** | **Aspect** | **Sentiment** |
| 2 | Mercedes | Love the luxurious interior and advanced tech features. | DESIGN_FEATURES | positive |
| 3 | Mercedes | Service costs are higher than I expected, not happy about it. | PRICE | negative |
| 4 | Audi | Incredible quality for the price, far beyond my expectations. | DESIGN_FEATURES | positive |
| 5 | BMW | The design is stunning, but I wish it had more engine power. | DESIGN_FEATURES | positive |
| 6 | BMW | Great car for its price, but fuel consumption could be better. | COMPANY | negative |
| 7 | Audi | Company's reputation for reliability is well deserved. | COMPANY | positive |
| 8 | KIA | Customer service was helpful and responsive. | SERVICE | positive |
| 9 | Toyota | Quality of the materials used in the interior is top-notch. | SERVICE | positive |
| 10 | KIA | Design features like the LED headlights are both practical and stylish. | DESIGN_FEATURES | negative |
| 11 | BMW | Price is a bit steep, but you get what you pay for with this brand. | GENERAL | positive |
| 12 | Mercedes | General driving experience is smooth and comfortable. | GENERAL | positive |

Рисунок 3.10 – Створений з коментарів датасет

Для роботи з автомобільною тематикою датасету було додано поля про автомобілі, аспекти, пов'язані з автомобілями (комфорт, надійність, сервіс тощо), та сентименти, виражені у цих відгуках. Після збору такого набору даних, можна тренувати модель ASQE на цих даних за допомогою коду, що наведений у додатку А.

## Висновки до розділу 3

У третьому розділі було розглянуто ключові аспекти проєктування інтелектуальної системи, що охоплює загальну архітектуру системи, проєктування бази даних, а також розробку датасету, необхідного для тренування і валідації моделі.

Під час планування архітектури було визначено структуру та компоненти системи, зосередившись на її модульності та масштабованості. Було розглянуто архітектуру, яка дозволяє легко інтегрувати нові функціональні можливості та обробляти великі обсяги даних, використовуючи сучасні технології та підходи до розробки програмного забезпечення.

В одному з основних етапів планування було описано структуру бази даних, яка підтримує зберігання, пошук та аналіз даних, що використовуються в системі. Було розроблено схему бази даних, оптимізовану для швидкого доступу до даних та ефективності їх обробки з метою забезпечення високої продуктивності та надійності системи.

Під час розробки датасету було виконано процес збору та підготовки даних для тренування інтелектуальної системи. Особлива увага була приділена методам збору даних, їх очищенню та анотації з метою створення якісного датасету, що дозволяє ефективно тренувати моделі машинного навчання та глибокого навчання.

Загалом, даний розділ висвітлює комплексний підхід до проєктування інтелектуальної системи, починаючи від загальної архітектури та закінчуючи детальними аспектами розробки датасету. Ретельне планування та проєктування кожного аспекту системи є ключовим для створення ефективної, гнучкої та масштабованої інтелектуальної системи, здатної вирішувати складні завдання та адаптуватися до змінних вимог користувачів і технологічного середовища.

# 4 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

## 4.1 Програмні засоби для розробки системи

Для реалізації поставлених задач було зроблене дослідження стосовно найкращих інструментів розробки даної системи. В результаті були обрані наступні засоби розробки:

– Vue.js;

– Nest.js;

– Python;

– Postgres.

*Vue.js* – це прогресивний JavaScript-фреймворк, використовуваний для створення інтерфейсів користувача. Vue здобув популярність завдяки своїй гнучкості, простоті та легкій інтеграції. Vue підходить для створення як простих сторінок, так і динамічних вебзастосунків.

Основні Характеристики Vue.js:

– реактивність: Vue.js використовує систему реактивних даних, що дозволяє автоматично оновлювати відображення при зміні даних у моделях;

– компонентна архітектура: фреймворк пропонує структуру, засновану на компонентах, що сприяє перевикористанню коду та зручності управління;

– двосторонній прив'язок даних: забезпечує синхронізацію між моделлю та відображенням, що полегшує роботу з формами;

– легка інтеграція: Vue може бути легко інтегрований з існуючими проєктами, а також використаний у якості основи для розробки нових застосунків;

– інтуїтивність та легкість навчання: Vue відрізняється простотою у використанні, завдяки чому він доступний навіть для новачків у сфері програмування;

– детальна документація: фреймворк супроводжується докладною та зрозумілою документацією.

Варто відзначити популярність даного фреймворку, яка стрімко зростає протягом останніх п’яти років. Станом на зараз динаміку вподобань Vue.js можна оцінити за допомогою Github Stars (див. рис. 4.D.

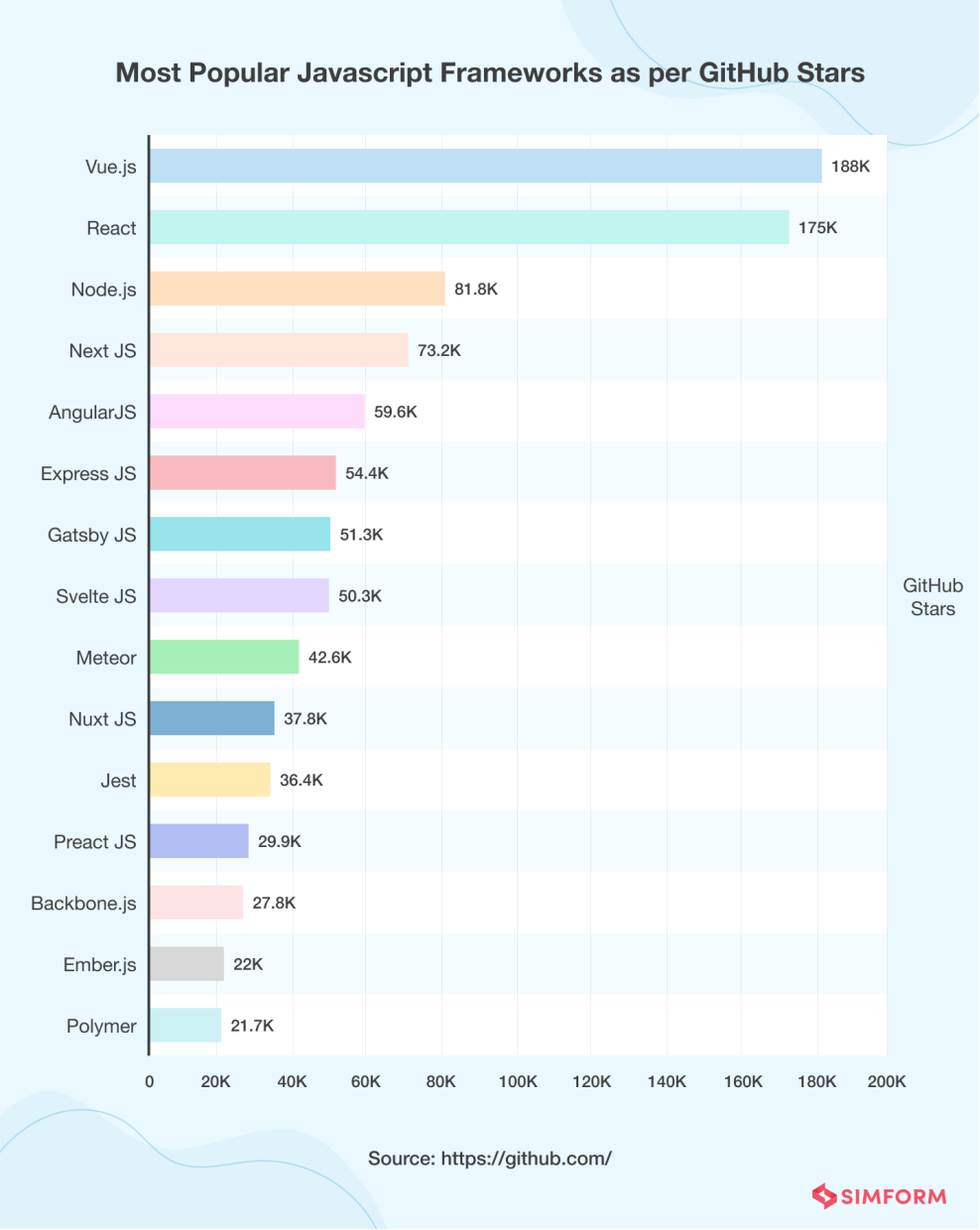


Рисунок 4.1 – Динаміка зірок в Github-репозиторіях фреймворків

Таким чином, Vue.js, як один з провідних JavaScript-фреймворків, заслужено вибудовує свою репутацію у світі веброзробки. Його головні переваги, такі як легкість у використанні, гнучка архітектура, та широкі можливості для розробки інтерактивних вебінтерфейсів, роблять його відмінним вибором для багатьох проєктів.

*Nest.js* – це потужний фреймворк для створення серверних застосунків на Node.js. Nest.js пропонує зручну модель розробки, що базується на TypeScript і використовує прогресивні концепції від Angular, що робить його особливо привабливим для розробників, які вже знайомі з Angular.

Основні характеристики Nest.js:

– архітектура, орієнтована на модулі: Nest.js використовує модульну структуру, що сприяє організації коду та його повторному використанні;

– використання TypeScript: забезпечує строгу типізацію та об'єктно-орієнтоване програмування, збільшуючи ефективність розробки та підтримки коду;

– залежності ін'єкції: система ін'єкції залежностей, запозичена з Angular, дозволяє легко управляти залежностями між різними частинами застосунка;

– мікросервісна архітектура: підтримка створення масштабованих мікросервісів з легкою інтеграцією різних транспортних протоколів;

– підтримка GraphQL та REST API: Nest.js дозволяє легко створювати як GraphQL, так і RESTful API;

– широка екосистема: велика кількість готових до використання пакетів та інтеграцій зі зовнішніми бібліотеками.

Nest.js є ідеальним для створення ефективних, добре структурованих серверних застосунків. Він підходить для великих, складних застосунків, де важлива чітка архітектура та легкість управління кодом. Nest.js також ефективний для створення мікросервісів, завдяки своїй підтримці різних комунікаційних механізмів та легкої інтеграції з іншими сервісами та базами даних.

*Python* – це високорівнева, інтерпретована, загального призначення мова програмування, яка була створена Гвідо ван Россумом у 1991 році. Python відомий своєю читабельністю та ефективністю, завдяки своєму чистому синтаксису та великій стандартній бібліотеці.

Основні характеристики Python.

1. Легкий синтаксис. Python має простий, легко читабельний синтаксис, що робить його відмінним вибором для новачків у програмуванні.

2. Багатоцільовість. Python використовується у різних областях, від веброзробки до наукових обчислень, даних та штучного інтелекту.

3. Велика стандартна бібліотека. Має широкий спектр модулів та бібліотек, що покривають багато потреб розробки.

4. Інтерпретована мова. Код Python виконується безпосередньо, що спрощує відлагодження та ітеративний розвиток.

5. Підтримка різних парадигм програмування. Підтримує об'єктно-орієнтоване, процедурне, та функціональне програмування.

6. Велика спільнота. Одна з найбільших та найактивніших спільнот розробників, що забезпечує підтримку та багато навчальних ресурсів.

Розглянемо основні переваги та недоліки мови Python (див. табл. 4.1).

Таблиця 4.1 – Основні переваги та недоліки мови Python

| **Переваги** | **Недоліки** |
|---|---|
| Легкий та читабельний синтаксис: Python дозволяє розробникам легко читати та писати код, що забезпечує швидке розуміння та розвиток проєктів. | Відносно повільніша швидкість виконання: Як інтерпретована мова, Python може бути повільнішим у порівнянні з компільованими мовами, як-от C++ чи Java. |
| Підтримка багатьох парадигм програмування: Підтримує різні стилі програмування, включаючи об'єктно-орієнтоване, процедурне та функціональне. | Управління пам'яттю: Не дає розробникам стільки ж контролю над управлінням пам'яттю, як деякі інші мови. |

Продовження таблиці 4.1

| | |
|---|---|
| Багата стандартна бібліотека та екосистема: Велика кількість бібліотек і фреймворків для різних завдань від веброзробки до наукових обчислень. | Мобільна та вбудована розробка: Не є ідеальним вибором для розробки мобільних застосунків або використання у вбудованих системах. |
| Широка спільнота та підтримка: Одна з найбільших спільнот розробників, що забезпечує чудову підтримку та ресурси для навчання. | Залежність від сторонніх бібліотек: Для певних завдань потрібно використовувати зовнішні бібліотеки, які можуть бути нестабільними або погано документованими. |
| Інтеграція з іншими мовами: Python можна легко інтегрувати з іншими мовами програмування, що робить його гнучким у багатомовних проєктах. | Динамічна типізація: Хоча це може бути перевагою, динамічна типізація може призвести до помилок, які важко виявити на ранніх етапах розробки. |
| Портативність: Код на Python легко переноситься між різними платформами і операційними системами без необхідності змін. | Використання пам'яті: Python може використовувати більше пам'яті, ніж деякі інші мови, що може бути критично для деяких застосунків. |

Python широко застосовується в багатьох галузях:

– веброзробка: завдяки фреймворкам, таким як Django та Flask;

– наукові та математичні обчислення: з використанням бібліотек, таких як NumPy, SciPy та Pandas;

– розробка програмного забезпечення: широке використання в скриптингу, автоматизації та розробці застосунків;

– машинне навчання та штучний інтелект: завдяки бібліотекам, як TensorFlow та PyTorch.

*PostgreSQL*, часто називаний Postgres, є потужною, відкритою, об'єктно-реляційною системою управління базами даних (СУБД). Вона розроблялася з

середини 1980-х років як частина проєкту POSTGRES на Університеті Каліфорнії в Берклі. З тих пір Postgres став одним із провідних рішень для управління базами даних у світі.

Основні характеристики PostgreSQL:

1. Підтримка різноманітних типів даних: Postgres підтримує широкий спектр вбудованих типів даних, включаючи примітивні типи (як-от int і string), структуровані (json, xml), геопросторові (GIS) та інші.

2. Розширюваність: система може бути розширена за допомогою користувацьких функцій та типів даних, а також має підтримку різноманітних плагінів.

3. ACID-сумісність: гарантує надійність у транзакціях, що означає підтримку властивостей атомарності, послідовності, ізоляції та довговічності.

4. Підтримка складних запитів: PostgreSQL відомий своєю потужною підтримкою складних запитів, підзапитів, об'єднань та інших функцій SQL.

5. Висока надійність та стабільність: Використовується в великих та масштабних системах управління даними.

6. Мультиверсійний конкурентний контроль (MVCC): забезпечує ефективний контроль одночасного доступу до даних, що робить Postgres відмінним вибором для систем з великою кількістю транзакцій.

PostgreSQL використовується у різних галузях і сценаріях, включаючи електронну комерцію, вебзастосунки, фінансові системи, геопросторове відображення та багато інших. Він особливо популярний у складних застосунках, де потрібна висока надійність, масштабованість та гнучкість управління даними.

### 4.2 Опис програмної реалізації

Розробка інтелектуальної системи є складним процесом, що вимагає глибокого розуміння поставлених задач, а також вибору найбільш ефективних технологічних рішень та інструментів. Особлива увага була приділена аналізу підходів до розробки інтерфейсів користувача, засобів збору та обробки даних, а

також стратегіях тестування, що були застосовані для забезпечення надійності та стабільності програмного забезпечення.

Внаслідок планування проєкту були визначені основні необхідні компоненти для UI:

– авторизація;

– дашборд;

– список коментарів;

– список користувачів;

– список ресурсів;

– список компаній;

– аналітика.

Процес авторизації проходить за допомогою токенів доступу та оновлення, які називаються access token та, відповідно, refresh token. У системі при запуску міграцій створюється адміністратор (див. рис. 4.2), який згодом додає інших користувачів.

```
export class UsersSeed1694077551523 implements MigrationInterface {
  no usages
  public async up(queryRunner: QueryRunner): Promise<void> {
    const passwordHash = await bcrypt.hash(process.env.ADMIN_USER_PASSWORD, Number(process.env.BCRYPT_SALT_ROUNDS));
    const user = {
      fullname: process.env.ADMIN_USER_FULLNAME,
      email: process.env.ADMIN_USER_EMAIL,
      password: passwordHash,
      active: true,
    };
    await queryRunner.manager.save(Users, user);
  }

  no usages
  public async down(queryRunner: QueryRunner): Promise<void> {
    await queryRunner.manager.delete(Users, { email: process.env.ADMIN_USER_EMAIL });
  }
}
```

Рисунок 4.2 – Код міграції створення адміністратору

Всі користувачі в системі створюються з хешованим паролем, оскільки це поле є чутливим та виключно належить лише тому користувачу, який має доступ до акаунту. Пароль хешується за допомогою бібліотеки bcrypt.

Користувач у свою чергу починає свій шлях у застосунку зі сторінки авторизації (див. рис. 4.3), маючи можливість увійти в свій акаунт, просто заповнивши поля Email та Password.

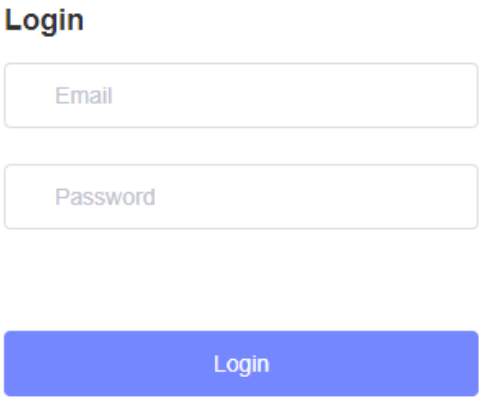


Рисунок 4.3 – Сторінка авторизації користувача

Після авторизації, користувач потрапляє на сторінку дашборди (див. рис. 4.4), яка відображає історію скрапінгу та середнє статистичне число, яке обраховується на основі глибокого аналізу тексту за певний період часу.

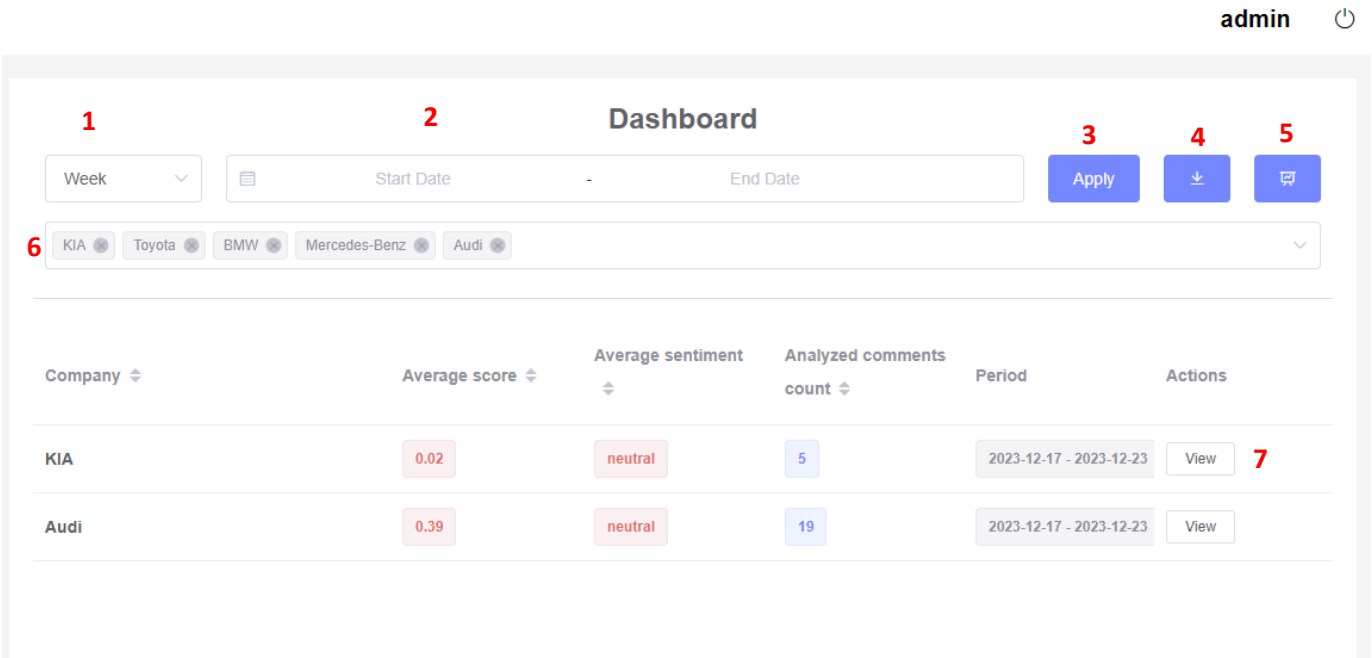


Рисунок 4.4 – Dashboard

Сторінка складається з двох основних компонентів – блок фільтрації та таблиця історії. До фільтрації відносяться наступні блоки:

– поле 1: вибір типу періоду, може бути або Month або Week;

– поле 2: вибір дати;

– поле 3: кнопка, що застосовує фільтрацію до даних;

– поле 4: кнопка, яка завантажує наявну таблицю як CSV файл;

– поле 5: кнопка, яка відкриває модальне вікно та порівняє компанії у вигляді стовпчикової діаграми (див. рис. 4.5);

– поле 6: селектор обраних компаній;

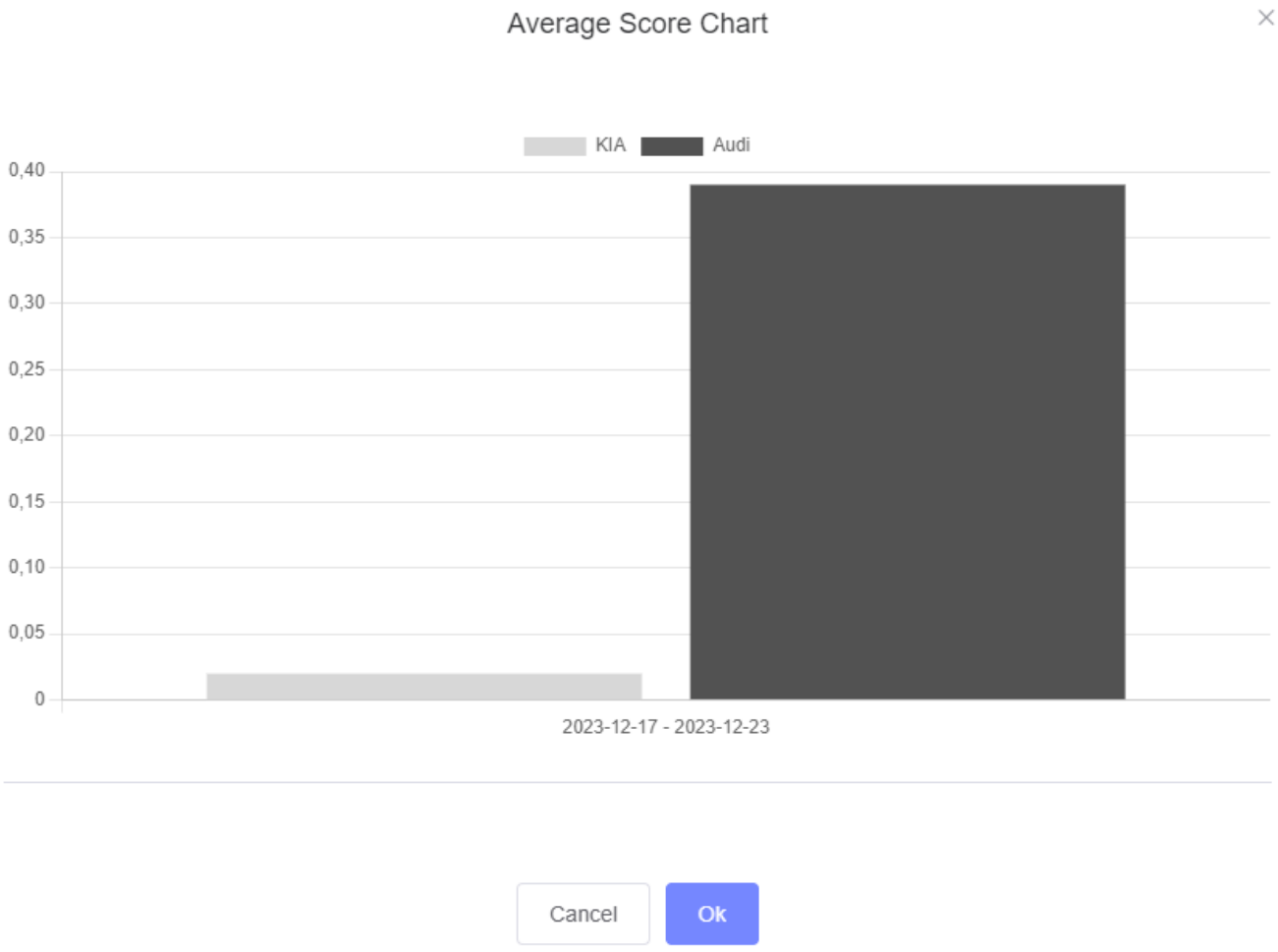


Рисунок 4.5 – Модальне вікно з діаграмою порівняння компаній за середнім показником сентіменту

Блок таблиці відображає дані компаній за допомогою таких полів, як назва компанії, середній показник сентіменту, кількість проаналізованих коментарів, а також період, протягом якого коментарі були проаналізовані. Навпроти кожного рядку є кнопка “View” (див. рис. 4.4), при натисканні на яку показується модальне вікно зі списком проаналізованих коментарів, що входять в обраний період (див. рис. 4.6). У відкритій модалці показано текст коментаря, його сентімент, а також дату аналізу.

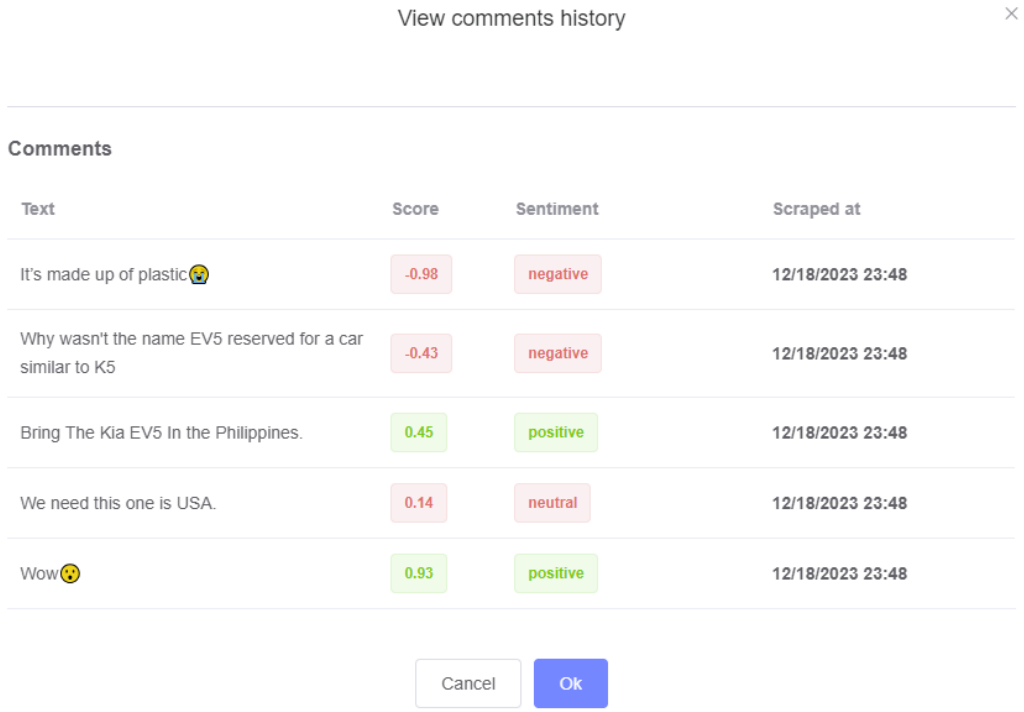


Рисунок 4.6 – список проаналізованих коментарів

Також однією з основних та водночас найскладніших є сторінка коментарів. Щоб перейти до цієї сторінки користувач може відкрити меню (див. рис. 4.7), що знаходиться з лівої сторони екрану, та обрати відповідну сторінку.

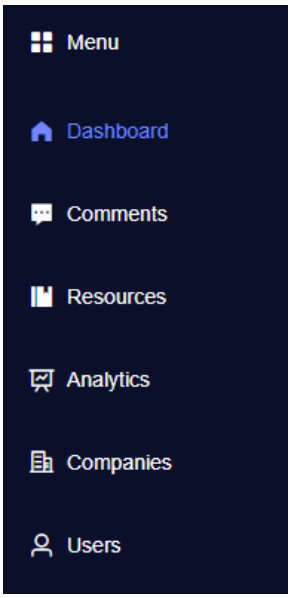


Рисунок 4.7 – меню

При редіректі на сторінку коментарів, користувач бачить перед собою також два блоки: фільтрацію та таблицю (див. рис. 4.8).

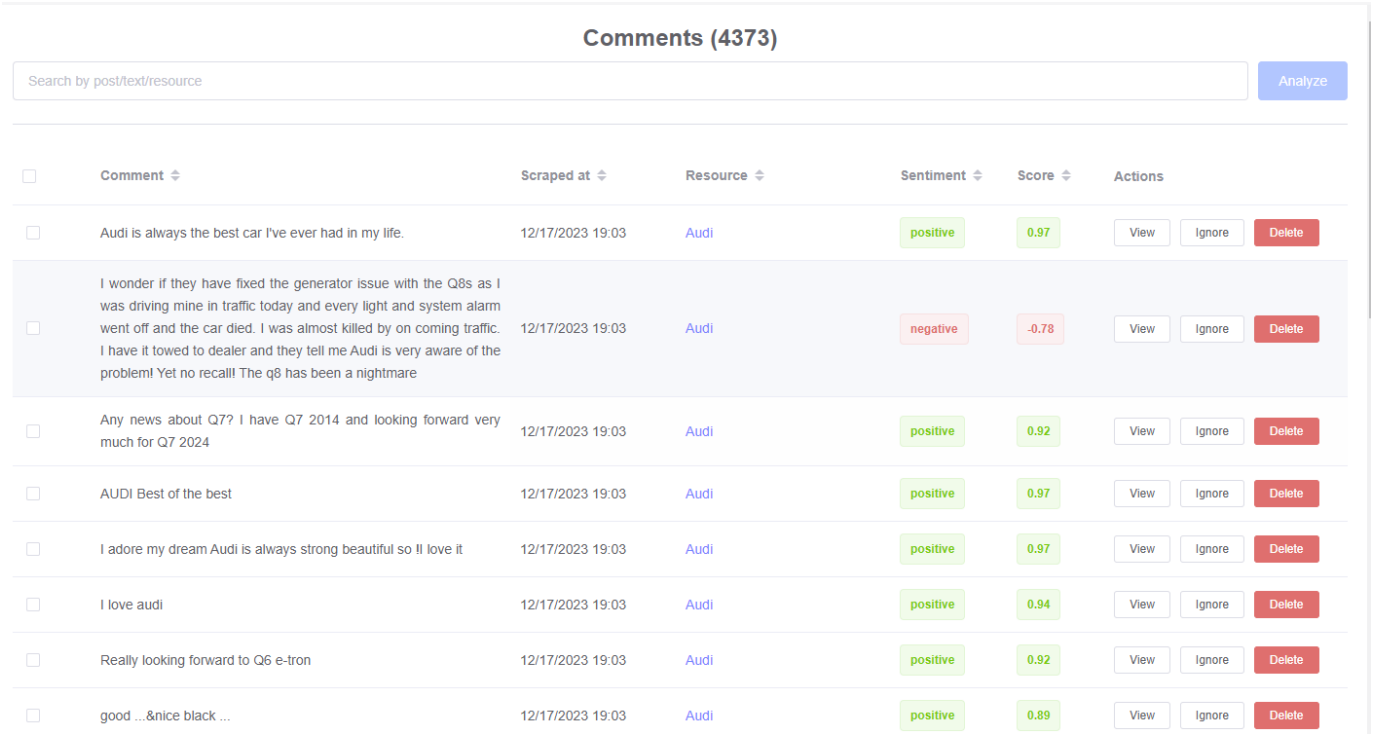


Рисунок 4.8 – Сторінка коментарів

До фільтрації відноситься поле пошуку коментаря, при використанні якого користувач може ввести будь-який текст, система у свою черга автоматично буде вести пошук введеного тексту по контенту коментаря, поста, а також ресурсу.

Таблиця коментарів має наступні поля:

– *Comment:* текст коментаря;

– *Scraped at:* дата скрапінгу коментаря;

– *Resource:* посилання на ресурс;

– *Sentiment:* результат сентімент аналізу;

– *Score:* результат сентімент аналізу в числовому еквіваленті;

– *Actions:* доступні дії стосовно рядку в таблиці.

Зліва в таблиці навпроти кожного рядку є пустий чекбокс, при натисканні на який кнопка Analyze змінюється на активну. Користувач в цьому випадку може натиснути на неї, після чого обраний коментар буде відправлений запитом до серверу, на якому за допомогою мови програмування Python буде здійснений аналіз тексту.

Серед доступних дій з рядком є три опції:

– *View*: при натисканні відкриваю модальне вікно з детальную інформацієї стосовно коментаря (див. рис. 4.9);

– *Ignore*: при натисканні виключає можливість коментаря бути проаналізованим;

– *Delete*: при натисканні видаляє коментар з системи.

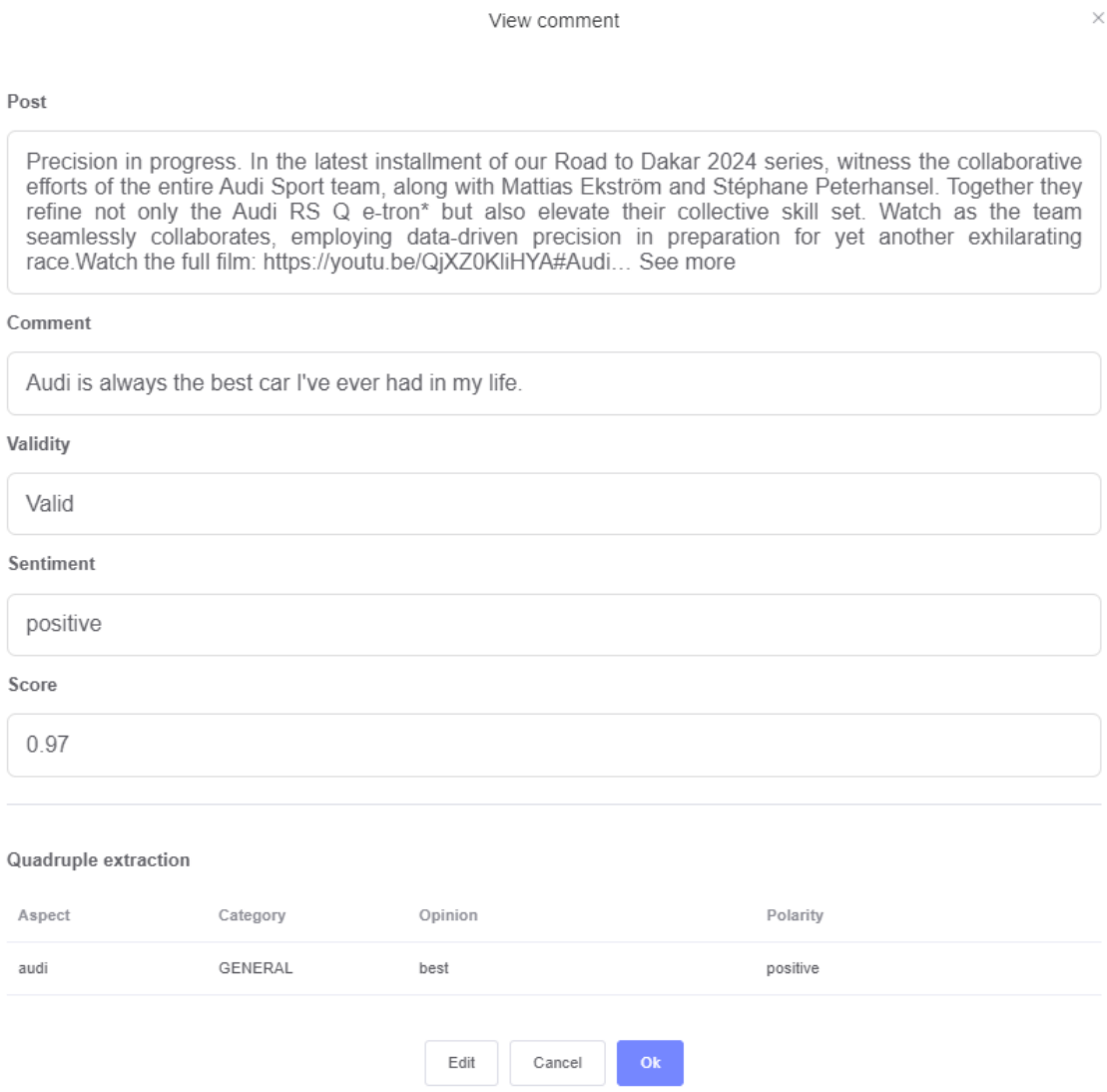


Рисунок 4.9 – Модальне вікно з детальною інформацієї про коментар

Наступною для використання є сторінка ресурсів (див. рис 4.10), яка відповідає за перегляд, створення, оновлення та видалення ресурсів. Таблиця на цій сторінці презентує дану сутність за допомогою наступних полів:

– *URL:* посилання на ресурс;

– *Company:* назва компанії;

– *Type:* тип ресурсу (facebook, youtube, linkedin);

– *Last scraped at:* дата останнього скрапінгу;

– *TSP:* загальне число постів, отриманих під час скрапінгу;

– *TSC:* загальне число коментарів, отриманих під час скрапінгу;

– *LSP:* число постів, отриманих під час останнього скрапінгу;

– *LSC:* число коментарів, отриманих під час останнього скрапінгу.

– *Actions:* доступні дії стосовно рядку в таблиці.

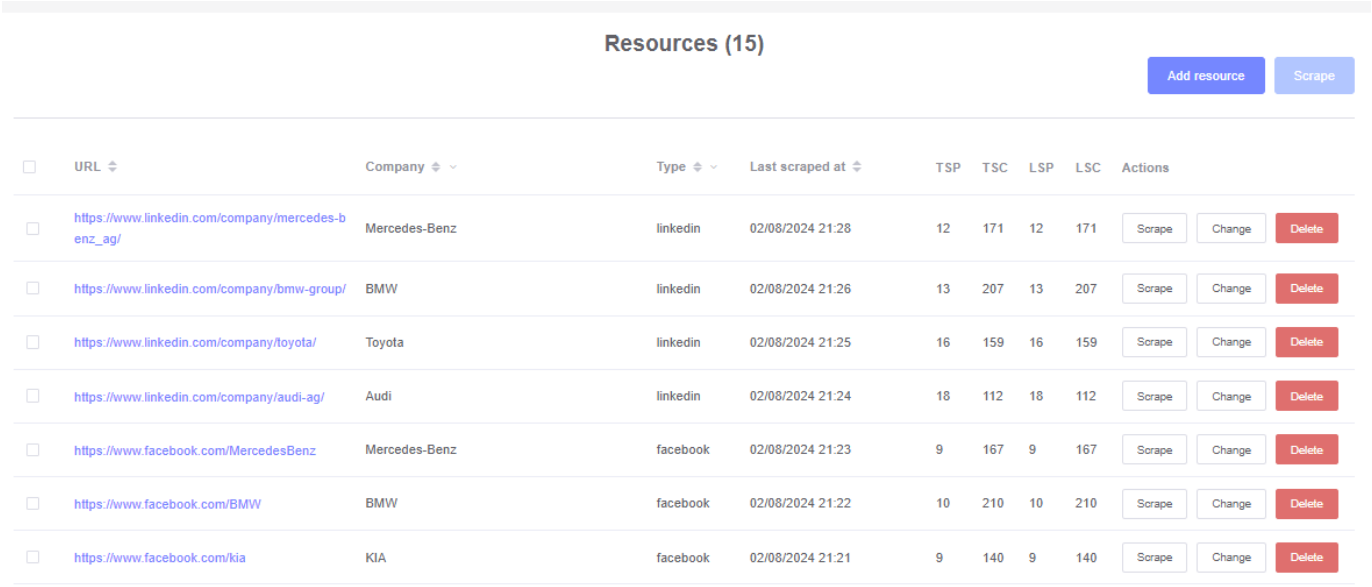


Рисунок 4.10 – Сторінка ресурсів

Користувач також має можливість додавати, змінювати існуючі, або видаляти ресурси. При натискані на кнопки Change / Add resource, буде з'являтись модальне вікно з необхідними полями для заповнення (див. рис. 4.11).

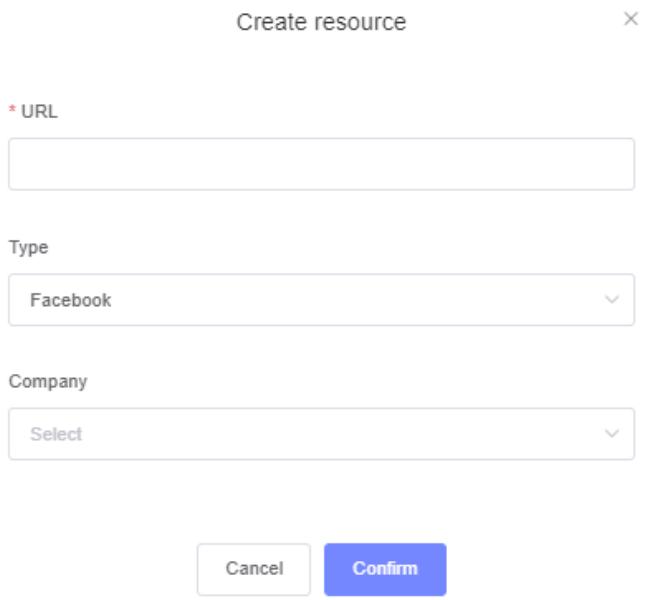


Рисунок 4.11 – Модальне вікно створення ресурсу

Задля видалення ресурсу достатньо просто натиснути на кнопку Delete.

Для скрапінгу необхідно обрати конкретний ресурс та натиснути на кнопку Scrape, після чого з'явиться вікно з відкритою соц. мережею, та почнеться процес зчитування даних.

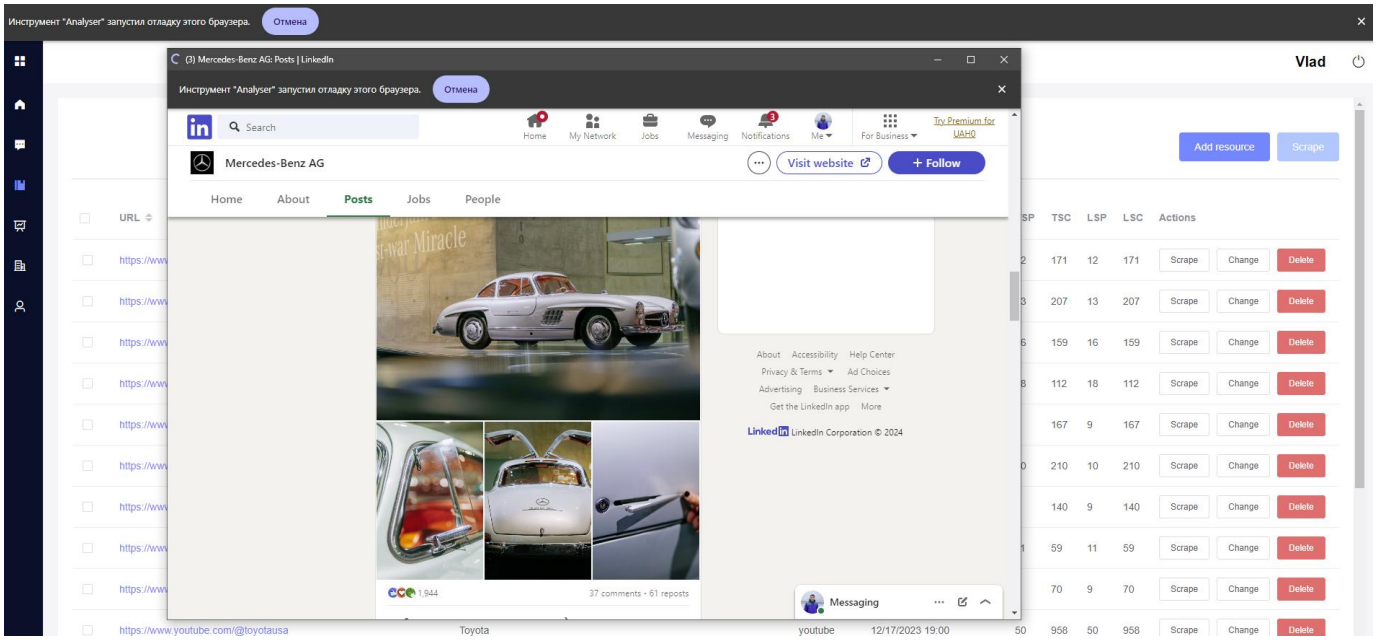


Рисунок 4.12 – процес скрапінгу ресурсу

Найцікавішою та водночас найінформативнішою для користувача є сторінка аналітики, яка включає в себе два графіки, що можуть надавати візуальну інформацію стосовно активностей компаній (див. рис. 4.13) та динаміку підписників (див. рис. 4.14).

Перший графік відображає топ компаній за показниками:

– кількість коментарів;

– кількість постів;

– кількість підписників.

Також є фільтр вибору типу ресурсу.

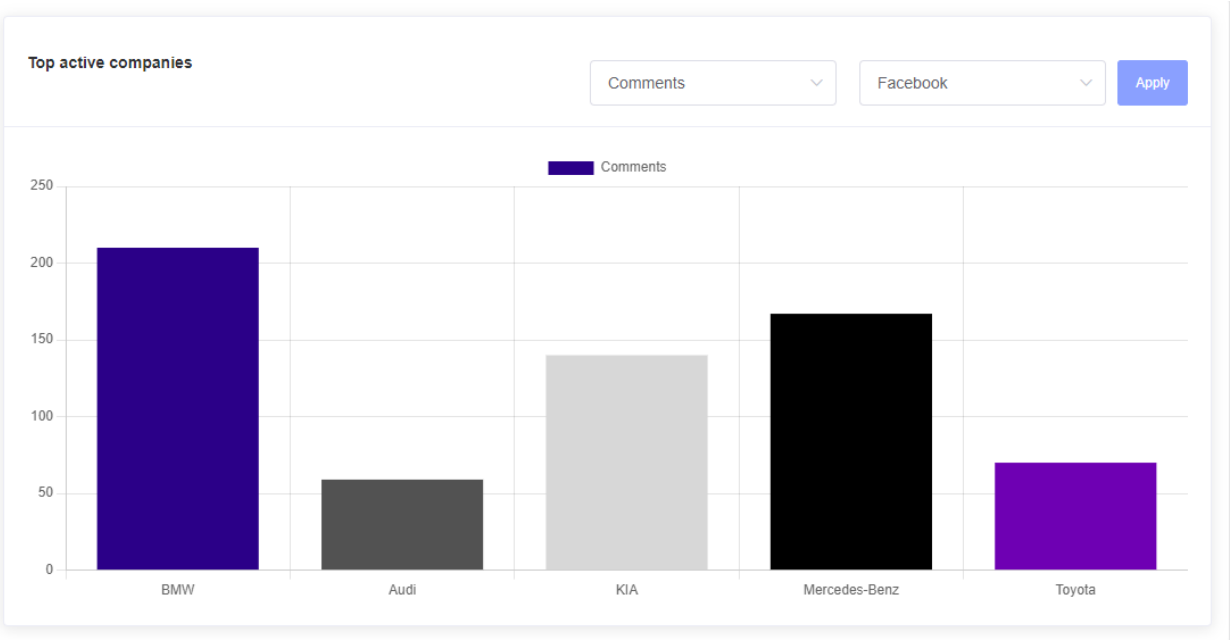


Рисунок 4.13 – Графік топ компаній в системі

Згідно з графіком, на офіційній сторінці BMW в Facebook написано найбільше коментарів, водночас для бренду Audi цей показник залишається найменшим у порівнянні з конкурентами.

Цей графік корисний для візуального порівняння взаємодії користувачів із різними автомобільними брендами в соціальних медіа, що може свідчити про популярність чи загальний інтерес до кожного бренду.

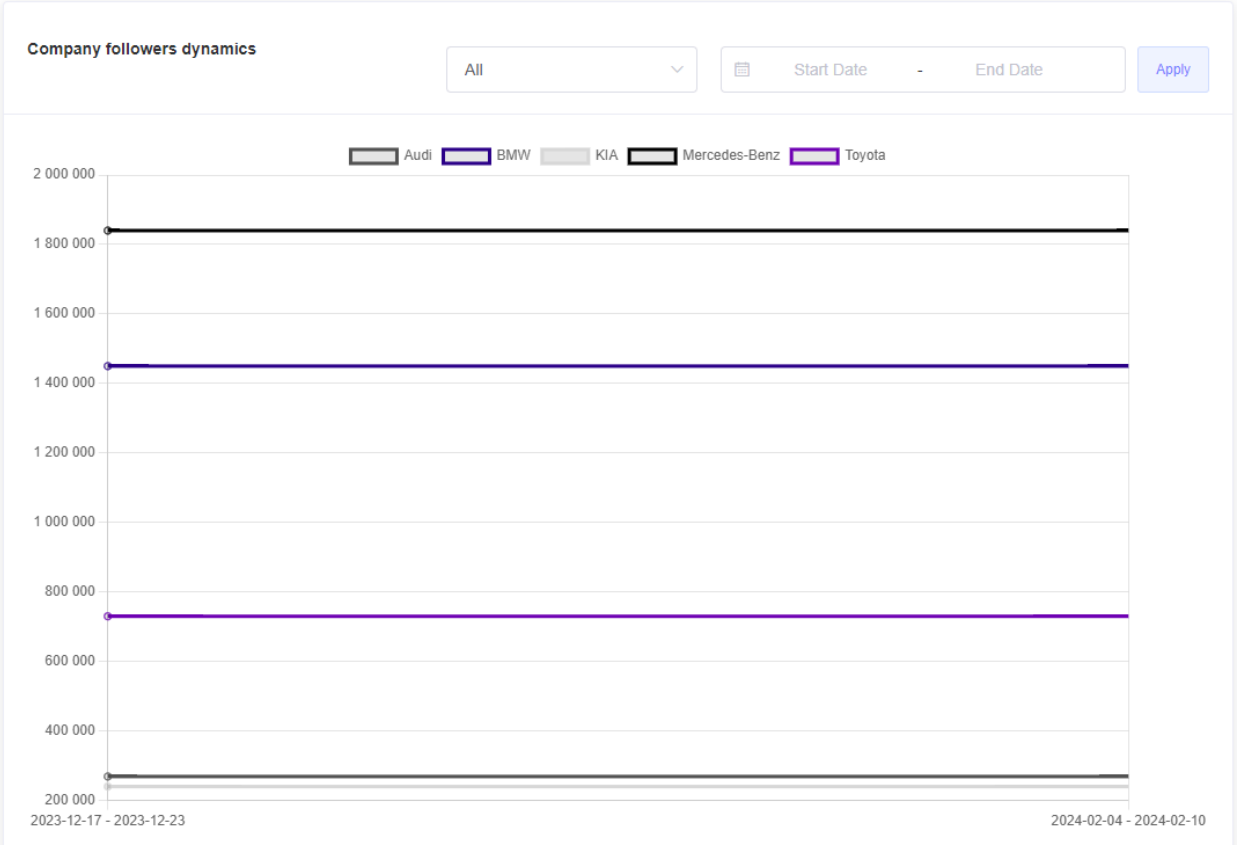


Рисунок 4.14 – Графік підписників у сторінках соціальних мереж компанії

На графіку зображена динаміка кількості послідовників (фоловерів) п'яти різних автомобільних компаній: Audi, BMW, KIA, Mercedes-Benz та Toyota. Це лінійний графік, де вертикальна вісь (вісь Y) вказує кількість підписників, а горизонтальна вісь (вісь X) представляє часовий проміжок.

Компанії Audi, BMW та Mercedes-Benz мають значно вищу кількість послідовників порівняно з KIA та Toyota, причому лінії для Audi, BMW та Mercedes-Benz виглядають майже горизонтально, що свідчить про стабільність або незначні зміни в кількості їхніх послідовників протягом вказаного часового періоду.

Для зручного менеджменту компаній була розроблена окрема сторінка з таблицею (див. рис. 4.15), де користувач має можливість переглядати, додавати, редагувати, або видаляти записи.

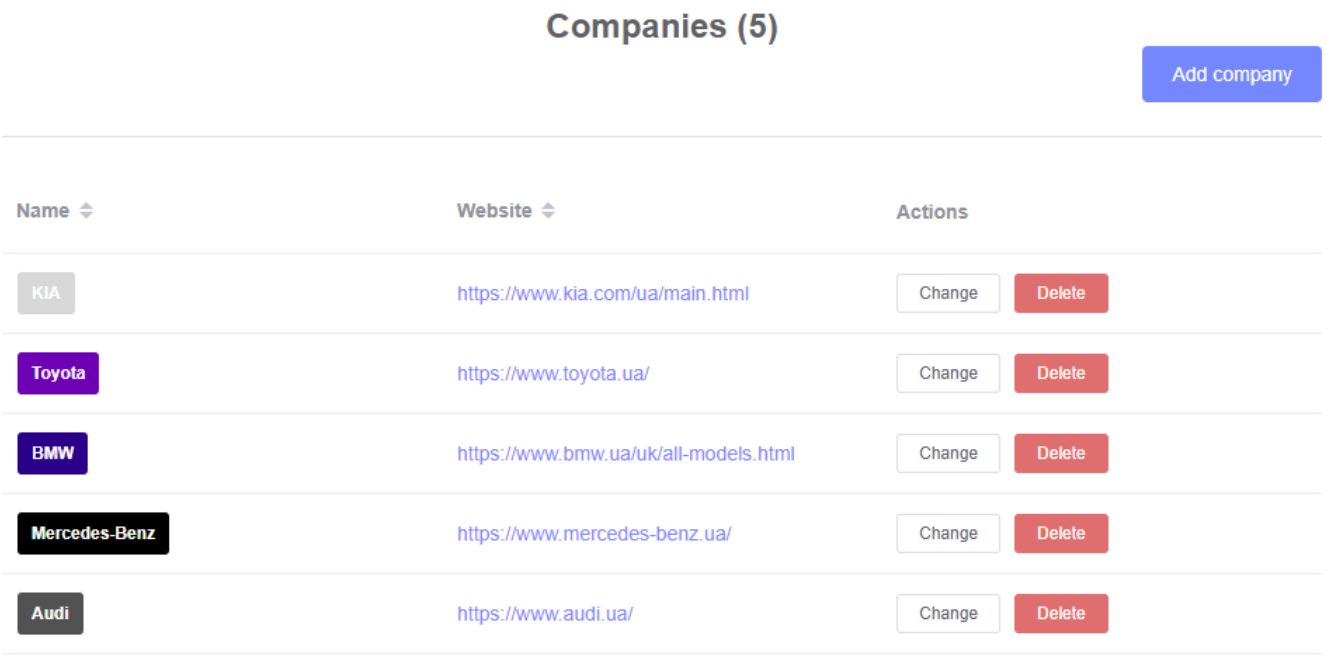


Рисунок 4.15 – Сторінка компаній

Таблиця компаній має наступні поля:

– Name: назва;

– *Website:* посилання на вебсайт;

– *Actions:* доступні дії стосовно рядку в таблиці.

Для кожної компанії також доданий свій унікальний колір, його користувач обирає самотужки при створенні. Колір відповідає за відображення компанії на графіках.

При натискані на кнопки Change / Add company, буде з'являтись модальне вікно з необхідними полями для заповнення (див. рис. 4.14), яке буде змінюватись в залежності від мети користувача.

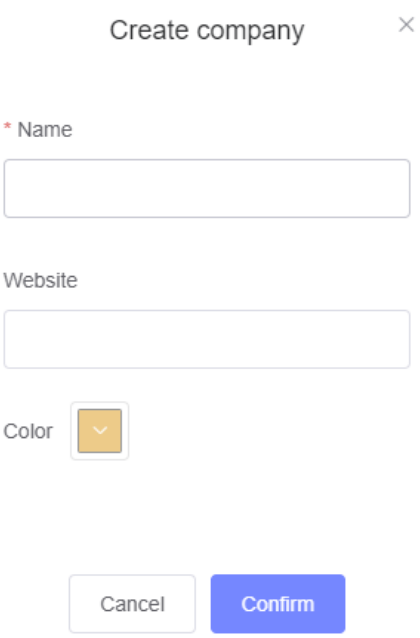


Рисунок 4.16 – Модальне вікно створення компанії

Сторінка користувачів системи має дещо схожий дизайн, та також слугує інструментом для менеджменту, де є можливість створювати, оновлювати, видаляти юзерів системи.

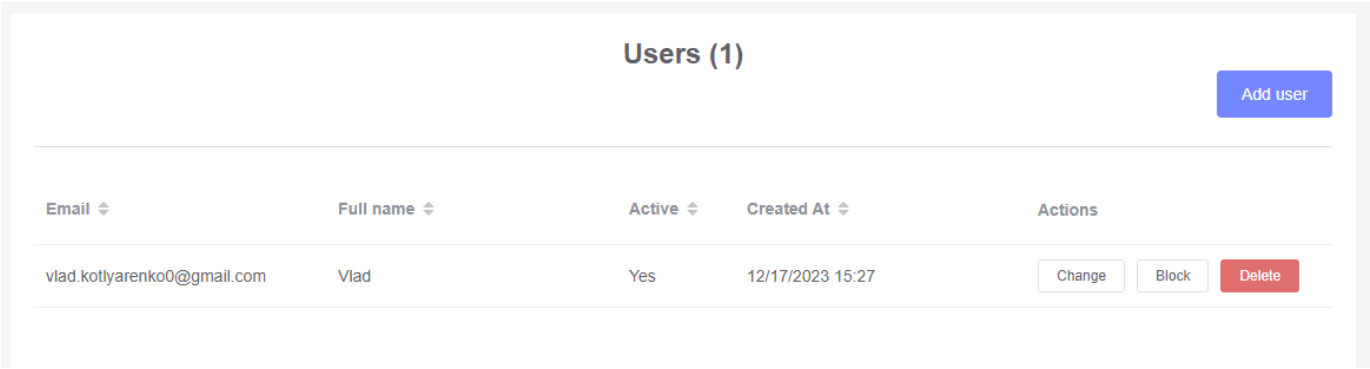


Рисунок 4.17 – Сторінка користувачів

Ця сторінка є типовою для систем управління користувачами та може бути використана адміністратором для моніторингу та керування правами доступу користувачів до ресурсів платформи.

### 4.3 Тестування методу аналізу тексту

Тестування методів аналізу тексту є критично важливим етапом у процесі розробки інтелектуальних систем, що включають обробку природної мови (NLP). Це дозволяє переконатися, що алгоритми та моделі коректно інтерпретують текстові дані, ефективно виявляють задані патерни та відповідають на завдання класифікації, розпізнавання сутностей, визначення сентименту чи інші завдання, на які вони були спрямовані.

Тестування починається з підготовки тестових випадків, які містять репрезентативний набір даних, включаючи еталонні текстові приклади з відомими результатами. Тестові випадки повинні охоплювати як типові сценарії, так і крайові випадки, щоб максимально широко перевірити роботу системи. Саме тому було виокремлено 10 коментарів реальних користувачів з різним емоційним забарвленням. Даним коментарям була надана суб’єктивна оцінка та сентімент.

Перелік обраних коментарів наведено прямими цитатами:

1) *Audi is always the best car I've ever had in my life*;

2) *I adore my dream Audi is always strong beautiful so !I love it*;

3) *Really looking forward to Q6 e-tron*;

4) *It’s made up of plastic*;

5) *Q8*;

6) *Winter is only for the rich*;

7) *Black Panther*;

8) *Any news about Q7? I have Q7 2014 and looking forward very much for Q7 2024*;

9) *Love*;

10) *Audi ...sorry to tell you that we hate electric cars*.

Далі варто надати власну оцінку обраним коментарям, після чого проаналізувати їх та порівняти результати. Для виконання поставлених задач реалізовано таблицю порівняння отриманих даних з репрезентативними.

Таблиця 4.1 – Порівняння аналізу тексту

| **Номер коментаря зі списку** | **Репрезентативна оцінка** | **Репрезентативний сентімент** | **Отримана оцінка після аналізу** | **Отриманий сентімент після аналізу** |
|---|---|---|---|---|
| 1 | 0.7 – 1 | positive | 0.97 | positive |
| 2 | 0.7 – 1 | positive | 0.97 | positive |
| 3 | 0.7 – 1 | positive | 0.92 | positive |
| 4 | -1 – -0.8 | negative | -0.98 | negative |
| 5 | -0.2 – 0.2 | neutral | -0.23 | neutral |
| 6 | -0.2 – 0.2 | neutral | -0.36 | neutral |
| 7 | -0.2 – 0.2 | neutral | 0 | neutral |
| 8 | 0.7 – 1 | positive | 0.92 | positive |
| 9 | 0.5 – 1 | positive | 0.86 | positive |
| 10 | -0.8 – -0.5 | negative | -0.62 | negative |

Результати в таблиці показують, що метод аналізу тексту здатен точно ідентифікувати сентимент, оскільки отримані оцінки та сентименти після аналізу добре узгоджуються з репрезентативними даними. Наприклад, коментарі з репрезентативними позитивними оцінками мають високі позитивні оцінки після аналізу, тоді як негативні та нейтральні коментарі також мають відповідні оцінки.

Первісна оцінка та сентимент були призначені на основі ретельного ручного аналізу, що забезпечило базову лінію для порівняння. Після застосування алгоритму аналізу тексту ми отримали оцінки, які відображають ступінь позитивності або негативності коментарів, та визначили відповідний сентимент. Загалом, спостерігається висока кореляція між репрезентативними та отриманими даними, що свідчить про високу точність методу.

Зокрема, позитивні коментарі були вірно ідентифіковані з високою впевненістю, оскільки отримані оцінки після аналізу знаходяться у верхньому діапазоні позитивного спектру. Негативні коментарі також були точно класифіковані, з оцінками, які відповідають негативному діапазону. Коментарі, що

мали нейтральний сентимент, демонструють низьку інтенсивність емоційного забарвлення, що коректно відображено алгоритмом як нейтральний сентимент Ці результати підтверджують ефективність використаного підходу до аналізу тексту і дають підстави вважати, що алгоритм може бути застосований для більш широкого діапазону даних, надаючи точну та надійну автоматизовану оцінку сентименту.

## Висновки до розділу 4

У даному розділі детально розглянуто ключові аспекти розробки інтелектуальної системи, включаючи вибір програмних інструментів, специфіку реалізації функціональних компонентів та результати тестування розроблених методів.

У результаті дослідження актуального ринку розробки було охарактеризовано технологічний стек, який був вибраний для реалізації проекту. Було продемонстровано, що вибір сучасних мов програмування, фреймворків, баз даних та інших інструментів розробки дозволив створити надійну, ефективну та масштабовану систему, яка відповідає сучасним вимогам до обробки великих обсягів даних.

Загалом, реалізація інтелектуальної системи відповідає всім сучасним стандартам розробки програмного забезпечення. Система продемонструвала здатність ефективно обробляти та аналізувати великі обсяги текстових даних. Тестування методів аналізу тексту підтвердило, що система готова до впровадження моделей обробки даних.

# ВИСНОВКИ

У даній кваліфікаційній роботі магістра було досліджено та розроблено інтелектуальну систему для обробки природної мови, яка інтегрує алгоритми вебскрапінгу для збору даних з Інтернету. Робота включала аналіз проблематики обробки природної мови та застосування вебскрапінгу, огляд існуючих аналогів та сучасних методів у цих сферах. Особливу увагу було приділено методам sentiment-аналізу та quadruple extraction як ключовим інструментам для інтерпретації емоційного забарвлення тексту.

В рамках проєктування системи було розроблено її загальну архітектуру, структуру бази даних та методики розробки датасету, що відіграли вирішальну роль у забезпеченні ефективності та гнучкості системи. Програмна реалізація системи була здійснена з використанням сучасних програмних засобів, що дозволило створити надійне та швидкодіюче рішення, здатне до масштабування та адаптації під змінювані умови використання.

Тестування методів аналізу тексту підтвердило високу точність та надійність розробленої системи, що відкриває шлях для її застосування у різноманітних областях, від маркетингових досліджень до моніторингу соціальних медіа.

Результати роботи демонструють значний потенціал застосування комбінованих методів обробки природної мови та вебскрапінгу для розширення можливостей збору та аналізу великих обсягів даних.

Загалом, розроблена система виявляється ефективним інструментом для обробки природної мови та вебскрапінгу, здатним вирішувати широкий спектр завдань у цих областях. Результати дослідження можуть бути корисними для подальшого розвитку інтелектуальних систем, спрямованих на обробку текстової інформації з веб-ресурсів.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Manning, C.D., Schütze, H. "Foundations of Statistical Natural Language Processing". MIT Press. 1999.

2. Jurafsky, D., Martin, J.H. "Speech and Language Processing". 3rd edition. 2019.

3. Bird, S., Klein, E., Loper, E. "Natural Language Processing with Python". O'Reilly Media. 2009.

4. Bengfort, B., Bilbro, R., Ojeda, T. "Applied Text Analysis with Python". O'Reilly Media. 2018.

5. Mitchell, R., Frank, E. "Web Scraping with Python: Collecting More Data from the Modern Web". 2nd edition. O'Reilly Media. 2018.

6. Grus, J. "Data Science from Scratch: First Principles with Python". O'Reilly Media. 2015.

7. Buczak, A. L., Guven, E. "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection". IEEE Communications Surveys & Tutorials. 2016.

8. Liu, B. "Sentiment Analysis: Mining Opinions, Sentiments, and Emotions". Cambridge University Press. 2015.

9. Taddy, M. "Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining". ACM Books. 2016.

10. Pang, B., Lee, L. "Opinion Mining and Sentiment Analysis". Foundations and Trends in Information Retrieval. 2008.

11. Mitchell, R. "Web Scraping with Python: Collecting Data from the Modern Web". O'Reilly Media, 2018.

12. Lawson, R. "Web Scraping with Python: An Introduction to Scraping the Web Using Python Libraries". Apress, 2020.

13. Tjaden, B. "Web Scraping with Python: Successfully scrape data from any website with the power of Python". Packt Publishing, 2019.

14. Sweigart, A. "Automate the Boring Stuff with Python: Practical Programming for Total Beginners". No Starch Press, 2020.

15. Simon Munzert, Christian Rubba, Peter Meißner, Dominic Nyhuis. "Automated Data Collection with R: A Practical Guide to Web Scraping and Text Mining". John Wiley & Sons, 2015.

16. Vargiu, E., Urru, M. "Web scraping and indexing: A new approach for rapidly processing customer reviews". In 2012 21st International Conference on Pattern Recognition (ICPR 2012).

17. Grinberg, M. "Flask Web Development: Developing Web Applications with Python". O'Reilly Media, 2018.

18. Kopf, G. "Pro Web Scraping for Data Science with Python". Apress, 2021.

19. Richardson, L., Amundsen, M., Ruby, S. "RESTful Web APIs: Services for a Changing World". O'Reilly Media, 2013.

20. Morresi, O., Nicoletti, M. C., Bruno, G. "Data quality assessment from web sources: a method based on web scraping". Journal of Data and Information Quality (JDIQ), 2016.

21. Comer, D. "Internetworking with TCP/IP Vol.1: Principles, Protocols, and Architecture". Pearson, 2013.

22. Kurose, J.F., Ross, K.W. "Computer Networking: A Top-Down Approach". Pearson, 2016.

23. Gollmann, D. "Computer Security". Wiley, 2011.

24. Tannenbaum, A.S., van Steen, M. "Distributed Systems: Principles and Paradigms". Prentice Hall, 2007.

25. Coulouris, G., Dollimore, J., Kindberg, T., Blair, G. "Distributed Systems: Concepts and Design". Pearson, 2011.

26. Majer, B. "Vue.js: Up and Running: Building Accessible and Performant Web Apps". O'Reilly Media, 2018.

27. Yerofeyev, D. "Full-Stack Vue.js 2 and Laravel 5: Bring the frontend and backend together with Vue, Vuex, and Laravel". Packt Publishing, 2017.

28. Gaonkar, S. "Building Modern Node.js Applications with Nest.js: Leveraging the power of JavaScript with Nest.js". Packt Publishing, 2020.

29. Owen, K. "Nest.js: A Progressive Node.js Framework for Building Efficient and Scalable Server-Side Applications". Amazon Digital Services LLC, 2019.

30. Lutz, M. "Learning Python". O'Reilly Media, 2013.

31. Matthes, E. "Python Crash Course: A Hands-On, Project-Based Introduction to Programming". No Starch Press, 2019.

32. Riggs, S., Krosing, H. "PostgreSQL: Up and Running". O'Reilly Media, 2017.

33. Douglas, B., Douglas, C. "PostgreSQL Developer's Guide". Packt Publishing, 2015.

34. Silberschatz, A., Korth, H.F., & Sudarshan, S. (2020). "Database System Concepts" (7th ed.). McGraw-Hill Education.

35. Elmasri, R., & Navathe, S.B. (2016). "Fundamentals of Database Systems" (7th ed.). Pearson.

36. Garcia-Molina, H., Ullman, J.D., & Widom, J. (2020). "Database Systems: The Complete Book" (2nd ed.). Pearson.

37. Date, C.J. (2020). "SQL and Relational Theory: How to Write Accurate SQL Code" (3rd ed.). O'Reilly Media.

38. Cairo, A. (2019). "How Charts Lie: Getting Smarter about Visual Information". W.W. Norton & Company.

39. Yau, N. (2013). "Data Points: Visualization That Means Something". Wiley.

40. Few, S. (2012). "Show Me the Numbers: Designing Tables and Graphs to Enlighten" (2nd ed.). Analytics Press.

41. Tufte, E.R. (2001). "The Visual Display of Quantitative Information". Graphics Press.

42. Garrett, J.J. (2010). "The Elements of User Experience: User-Centered Design for the Web and Beyond" (2nd ed.). New Riders.

43. Saffer, D. (2009). "Designing for Interaction: Creating Innovative Applications and Devices" (2nd ed.). New Riders.

44. Cooper, A., Reimann, R., Cronin, D., & Noessel, C. (2014). "About Face: The Essentials of Interaction Design" (4th ed.). Wiley.

45. Tidwell, J. (2010). "Designing Interfaces: Patterns for Effective Interaction Design" (2nd ed.). O'Reilly Media.

# ДОДАТОК А

## Лістинг програмного коду

***main.py***

```
from data_preprocessing import comment_preproc
from transformers import AutoModelForSequenceClassification, AutoTokenizer
from pyabsa import ABSAInstruction
sa_model = AutoModelForSequenceClassification.from_pretrained("gyesibiney/Sentiment-review-analysis-roberta-3")
sa_tokenizer = AutoTokenizer.from_pretrained("gyesibiney/Sentiment-review-analysis-roberta-3")
qe_model = ABSAInstruction.ABSAGenerator("multilingual")


def sentiment_analysis(text_comment):
    sa_inputs = sa_tokenizer(text_comment, return_tensors="pt")
    sa_outputs = sa_model(**sa_inputs)
    sa_outputs = sa_outputs.logits.softmax(dim=-1).tolist()
    score = round((sa_outputs[0][1] - 0.5)*2, 2)
    sa_result = {"sentiment": "negative" if -1 <= score <= -0.4 else "neutral" if -0.4 < score <= 0.4 else "positive",
                 "score": score}
    return sa_result


def quadruple_extraction(text_comment):
    acos_result = qe_model.predict(text_comment)
    quadruples = acos_result['Quadruples']
    quadruples_filt = []
    for quad in quadruples:
```

```
        if quad['aspect'] == 'NULL' and quad['opinion'] == 'NULL':
            continue
        categories = quad['category'].split('#')
        if len(categories) != 2:
            quad['category'] = 'GENERAL'
        elif categories[0] in ['COMPANY', 'SERVICE']:
            quad['category'] = categories[0]
        elif categories[1] in ['QUALITY', 'DESIGN_FEATURES', 'PRICE']:
            quad['category'] = categories[1]
        else:
            quad['category'] = 'GENERAL'
        quadruples_filt.append(quad)
    qe_result = quadruples_filt
    return qe_result


text_comment = "i would be happy to help delivery any all or all your units to your clients!"
text_comment = comment_preproc(text_comment)
sa_result = sentiment_analysis(text_comment)
qe_result = quadruple_extraction(text_comment)
print(sa_result)
print(qe_result)
```

***data_processing.py***

```
import json
import re
```

```
import string
def remove_punctuation(text_comment):
    remove = string.punctuation
    remove = remove.replace("!", "").replace(".", "").replace("?", "")
    text_comment = re.sub('[%s]' % re.escape(remove), ' ', text_comment)
    return text_comment

def expand_contractions(text_comment):
    with open('contractions.json', 'r') as fp:
        contractions_dict = json.load(fp)
    contractions_re = re.compile('(%s)' % '|'.join(contractions_dict.keys()))
    def replace(match):
        return contractions_dict[match.group(0)]
    return contractions_re.sub(replace, text_comment)

def remove_digits(text_comment):
    return re.sub('\w*\d\w*','', text_comment)

def remove_emojis(data):
    emoji = re.compile("["
        u"\U0001F600-\U0001F64F"
        u"\U0001F300-\U0001F5FF"
        u"\U0001F680-\U0001F6FF"
        u"\U0001F1E0-\U0001F1FF"
        u"\U00002500-\U00002BEF"
        u"\U00002702-\U000027B0"
        u"\U00002702-\U000027B0"
```

```
        u"\U000024C2-\U0001F251"
        u"\U0001f926-\U0001f937"
        u"\U00010000-\U0010ffff"
        u"\u2640-\u2642"
        u"\u2600-\u2B55"
        u"\u200d"
        u"\u23cf"
        u"\u23e9"
        u"\u231a"
        u"\ufe0f"
        u"\u3030"
                "]+", re.UNICODE)
    return re.sub(emoji, '', data)

def remove_links(text_comment):
    text_comment = re.sub(r'http\S+', '', text_comment)
    text_comment = re.sub(r'www\S+', '', text_comment)
    text_comment = ' '.join([item for item in text_comment.split() if '@' not in item])
    return text_comment

def remove_stop_words(text_comment):
    with open('stop_words.json', 'r') as fp:
        stop_words = json.load(fp)
    for stop_word in stop_words:
        if stop_word in text_comment:
            text_comment = text_comment.replace(stop_word, "")
    return text_comment
```

```
def remove_singles(text_comment):
    return re.sub('\\b[^(i)]\\b', ' ', text_comment)

def remove_repetitive_char(text_comment):
    return re.sub(r'(\W)(?=\1)', '', text_comment)

def comment_preproc(text_comment):
    text_comment = text_comment.lower()
    text_comment = remove_stop_words(text_comment)
    text_comment = expand_contractions(text_comment.replace("’", "'"))
    text_comment = remove_emojis(text_comment)
    text_comment = remove_links(text_comment)
    text_comment = remove_digits(text_comment)
    text_comment = remove_punctuation(text_comment.replace("…", " "))
    text_comment = remove_singles(text_comment)
    text_comment = remove_repetitive_char(text_comment)
    return text_comment
```