Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Система аналізу контенту веб-сайту на базі формальних систем
Назва теми

КвРКІ 200241.20.02.18 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Виконав: студент IV курсу, група КІ2-20-2 _____ О.В. Попружний
Підпис — Ініціали, прізвище

Керівник _____ Є. Є. Федоров
Підпис, дата — Ініціали, прізвище

Нормоконтролер _____ С.М. Лисенко
Підпис, дата — Ініціали, прізвище

До захисту допускаю:
Зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Т.О. Говорущенко
Підпис — Ініціали, прізвище

«19» червня 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень бакалавр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О.Говорущенко

" 10 " 01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Попружному Олександру Васильовичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Система аналізу контенту веб-сайту на базі формальних систем

Керівник проекту (роботи) Федоров Є.Є. д.т.н., професор

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 15.02.2024 р. № 5

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2024 р.

3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Загальні відості про існуючі систми аналізу контенту веб сайтів на базі формальних систем

Проектування програмно-технічного засобу і методів реалізації системи.

Програмно-технічна реалізація програми аналізу контенту на веб-сайтах.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Архітектура контент-аналізу веб-ресурсів на основі формальних методів

Структура роботи процесів окремих компонентів API та Web-scraper

Система реалізації окремих частин програми аналізу контенту веб-сайтів на базі формальних систем, безпека даних, та процеси NLT, TexBlob

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Лисенко С.М., професор кафедри КІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІС		

7. Дата видачі завдання «_10_»_____01_____2024 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2024	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2024	виконано
3	Робота над розділом 1 – огляд, аналіз та порівняння серверної частини існуючих сервісів для роботи з відеоконтентом	01.03.2024	виконано
4	Робота над розділом 2 –опис архітектури та вибір програмних та апаратних засобів розробки	01.04.2024	виконано
5	Робота над розділом 3 – програмно-технічна реалізація серверної частини системи обробки, зберігання та передачі відеоконтенту	29.04.2024	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2024	виконано
7	Попередній захист ВКР	30.05.2024	виконано
8	Захист ВКР на засіданні ЕК	Червень 2024 року	

Студент _____ О. В. Попружний
Підпис Ініціали, прізвище

Керівник роботи _____ Є. Є. Федоров
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл.-листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 20041.20.02.18 ПЗ	Пояснювальна записка	62		
			Графічні матеріали			
2		КвРКІ 200241.20.02.18 Е8	Архітектура контент-аналізу веб-ресурсів на основі формальних методів	1		
3		КвРКІ 200241.20.02.18 Е8	Структура роботи процесів окремих компонентів API та Web-scraper	1		
4		КвРКІ 200241.20.02.18 Е8	Система реалізації окремих частин програми аналізу контенту веб-сайтів на базі формальних систем, безпека даних, та процеси NLT, TexBlob	1		

					КвРКІ 200241.20.02.18 ВП		
Зм	Арк	№ докум	Підпис	Дата			
Розробив		Пилуський			Відомість проекту	Літера: У	Аркуш: 1 / Аркушів: 1
Перевір.		Федорів					
						ХНУ, КІ2-20-2	
Н. контр.		Лисенко					
Затв.		Говорущенко		19.06			

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Система аналізу контенту веб-сайту на базі формальних систем».

Автор роботи: Попружний Олександр Васильович.

Керівник роботи: Федоров Євген Євгенович.

Пояснювальна записка: 62 с., 16 рис., 1 табл., 3 дод., 70 джерел.

Графічна частина: 3 креслення.

СИСТЕМА, ТЕКСТОВА АНАЛІТИКА, ЛЕКСИЧНИЙ АНАЛІЗ ТОКЕНІЗАЦІЯ, АЛГОРИТМИ, ФОРМАЛЬНІ, ТЕМАТИЧНЕ МОДЕЛЮВАННЯ.

Метою даної дипломної роботи є підвищення ефективності використання системи аналізу контенту на веб-сторінках на базі формальних систем

Об'єктом дослідження є процес аналізу тексту на веб-сторінках програмою на створеної на базі формальних систем

Предметом дослідження є система аналізу веб-контенту на базі формальних систем

Під час проведення даного дослідження був використаний метод систематичного огляду літератури для вивчення і аналізу предметної області даного дослідження з текстових джерел інформації.

Підпис студента

19.06.2024

Дата

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

HTML – HyperText Markup Language

GATE – General Architecture for Text Engineering

ШІ – штучний інтелект

XML – eXtensible Markup Language

NLP – Natural Language Processing

NLT – Natural Language Toolkit

CSS – Cascading Style Sheets

TF-IDF – Term Frequency-Inverse Document Frequency

BERT – Bidirectional Encoder Representations from Transformers

ПЗ – програмне забезпечення

CI – Continuous Integration

CD – Continuous Deployment

ВСТУП

У сучасному інтернет-просторі важливим аспектом стає аналіз контенту веб-сайтів для забезпечення ефективності та безпеки користувачів. Формальні системи аналізу контенту виявляються невід'ємною складовою в цьому процесі, забезпечуючи об'єктивну та структуровану оцінку інформації. В даному контексті важливо розглянути переваги використання таких систем, які сприяють підвищенню ефективності фільтрації та категоризації контенту, забезпечуючи безпеку та зручність для користувачів в онлайн-середовищі.

Ці формальні системи визначаються своєрідністю та точністю у виявленні небезпечних або небажаних вмістів, що включає в себе шкідливий або обманливий контент. Їх аналітичні можливості дозволяють ефективно виявляти порушення авторських прав, спам, та інші загрози для користувачів. Крім того, ці системи сприяють поліпшенню пошукових алгоритмів та рекомендацій, забезпечуючи точне та релевантне відображення інформації. Застосування формальних систем аналізу контенту веб-сайтів відкриває нові перспективи для покращення якості онлайн-взаємодії та захисту від негативного впливу віртуального середовища.

Об’єктом дослідження є процес аналізу тексту на веб-сторінках програмою на створеної на базі формальних систем

Предметом дослідження є система аналізу веб-контенту на базі формальних систем

Ці системи також допомагають в оптимізації веб-сайтів та підвищенні їхньої продуктивності, шляхом ідентифікації слабких місць у контенті та рекомендацій для його вдосконалення. Їхнє використання сприяє підтримці стандартів безпеки та відповідності, що робить їх важливим інструментом для власників веб-ресурсів та адміністраторів. За допомогою формальних систем аналізу контенту досягається баланс між відкритістю інформації та захистом від можливих негативних впливів, забезпечуючи користувачам якісний та безпечний веб-досвід.

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ІСНУЮЧІ СИСТЕМИ АНАЛІЗУ КОНТЕНТУ ВЕБ САЙТІВ НА БАЗІ ФОРМАЛЬНИХ СИСТЕМ

1.1 Основні принципи роботи

В сучасному інтернет-просторі важливим аспектом стає аналіз контенту веб-сайтів для забезпечення ефективності та безпеки користувачів [1]. Формальні системи аналізу контенту [2] виявляються невід'ємною складовою в цьому процесі, забезпечуючи об'єктивну та структуровану оцінку інформації. В даному контексті важливо розглянути переваги використання таких систем, які сприяють підвищенню ефективності фільтрації та категоризації контенту, забезпечуючи безпеку та зручність для користувачів в онлайн-середовищі.

Ці формальні системи визначаються своєрідністю та точністю у виявленні небезпечних або небажаних вмістів, що включає в себе шкідливий або обманливий контент. Їх аналітичні можливості дозволяють ефективно виявляти порушення авторських прав, спам, фішингові атаки та інші загрози для користувачів. Крім того, ці системи сприяють поліпшенню пошукових алгоритмів та рекомендацій, забезпечуючи точне та релевантне відображення інформації. Застосування формальних систем аналізу контенту веб-сайтів відкриває нові перспективи для покращення якості онлайн-взаємодії та захисту від негативного впливу віртуального середовища.

Системи аналізу контенту веб-сайту на базі формальних систем спираються на ретельно розроблені принципи роботи для ефективного виявлення та оцінки різноманітного контенту в онлайн-середовищі. Основні принципи роботи таких систем включають в себе етапи витягу даних з веб-сайтів, аналіз цих даних та подальше представлення результатів. На першому етапі система взаємодіє з веб-сайтом, витягуючи різноманітні дані, такі як текст, зображення, відео та інші елементи контенту. Цей процес може включати в себе сканування HTML-коду [3] сторінок, використання API [4] та інші методи для отримання інформації.

На другому етапі проводиться аналіз витягнутих даних. Система використовує формальні методи та алгоритми для ідентифікації ключових елементів, класифікації контенту та виявлення будь-яких недоречностей чи загроз. Це включає в себе визначення ключових слів, перевірку структури тексту, аналіз зображень та відео на предмет вмісту та безпеки.

Нарешті, на третьому етапі система представляє результати аналізу користувачеві або адміністратору в зручній формі. Це може бути у вигляді звітів, графіків, сумарних висновків чи рекомендацій для подальших дій. Через цей цикл витягу, аналізу та представлення інформації системи аналізу контенту веб-сайту забезпечують ефективну та надійну роботу у виявленні та управлінні різноманітним контентом в онлайн-середовищі.

1.2 Переваги використання формальних систем

Точність формальних систем є критичним аспектом при використанні формальних систем аналізу контенту веб-сайтів. Однією з основних переваг цих систем є їхня здатність надавати найвищу можливу точність у виявленні різноманітних видів загроз та небажаного контенту. Розроблені формальні методи та алгоритми націлені на уникання помилок та забезпечення надійного виявлення потенційних небезпек. Важливо підкреслити, що висока точність відіграє ключову роль у запобіганні неправильній ідентифікації чи пропуску потенційно небезпечного матеріалу, забезпечуючи тим самим ефективні та надійні заходи безпеки в онлайн-середовищі. Завдяки цій великій точності користувачі можуть довіряти системі, бути впевненими в її здатності ефективно захищати їх від потенційних загроз у великій мережі Інтернет.

Ефективність формальних систем виявляється не лише у їх високій точності, але й у здатності швидко та надійно аналізувати великі обсяги даних. Ці системи оптимізовані для роботи із значущою швидкістю, що є ключовим фактором у виявленні загроз в реальному часі. Забезпечуючи оперативну реакцію на нові

виклики, вони дозволяють ефективно адаптуватися до змін у сучасному інтернет-просторі. Ця висока ефективність робить їх невід'ємною складовою систем безпеки, спроможних негайно реагувати на емерджентні загрози та швидко апгрейдитися для вдосконалення відповіді на сучасні виклики в цифровому середовищі. Поєднання точності та швидкості роботи формальних систем визначає їх як ідеальний інструмент для ефективного реагування на постійно змінюючіся умови та загрози в онлайн просторі.

Розширюваність формальних систем також без сумніву є важливою характеристикою, яка надає їм здатність адаптуватися до різних потреб та обсягів веб-сайтів. Це досягається завдяки гнучкій конфігурації та можливості інтеграції з іншими інструментами без втрати ефективності. Розширюваність формальних систем передбачає можливість легко впроваджувати нові функції, модулі чи оновлення, що робить їх готовими відповідати зростаючим вимогам та змінам у сучасному інтернет-просторі.

Ця гнучкість дозволяє формальним системам інтегруватися в різноманітні архітектури веб-сайтів та інформаційних систем, що важливо при оптимізації їхнього функціоналу під конкретні умови та вимоги користувачів. Розширюваність робить формальні системи не лише потужними інструментами виявлення загроз та небажаного контенту, але й забезпечує їхню готовність до ефективної реакції на зміни в інтернет-середовищі та стрімкий розвиток технологій.

Пристосовуватися до зростання обсягу інформації є перевагою використання формальних систем аналізу контенту веб-сайтів. Здатність цих систем ефективно реагувати на постійне збільшення обсягу інформації в онлайн-середовищі визначається їхньою гнучкістю та адаптивністю. Вони автоматично адаптуються до змін у структурі веб-сайтів, сприймають нові тренди та здатні аналізувати великі обсяги даних, при цьому не зазнаючи значної втрати продуктивності. Крім того, використання формальних методів у цих системах гарантує створення консистентних та стабільних результатів аналізу. Це сприяє надійності у виявленні різноманітних аспектів контенту, включаючи важливість та актуальність

інформації. Завдяки цим характеристикам, формальні системи не лише вирішують завдання аналізу великих обсягів даних, але й забезпечують стабільність та ефективність у змінному та динамічному онлайн-середовищі, де інформаційний потік постійно розширюється.

Розширення функціональних можливостей представляє собою важливий аспект формальних систем аналізу контенту. Вони не тільки дозволяють користувачам впроваджувати нові функції, але й надають можливість розширювати свій функціонал шляхом додавання нових аналітичних модулів чи адаптації до конкретних потреб та вимог. Цей підхід надає гнучкість та масштабованість систем, що стає важливим умовою в умовах постійної зміни веб-простору. Можливість розширення функціональних можливостей дозволяє формальним системам адаптуватися до нових викликів та вимог, що виникають у зв'язку з еволюцією інтернет-середовища. Додавання нових модулів або налаштування систем під конкретні завдання допомагає забезпечити не лише ефективність у виявленні ризиків, але й високий рівень придатності до використання в різних контекстах.

У результаті, формальні системи аналізу контенту [2] не обмежуються статичними можливостями, але виявляються адаптивними та розвиваючимися інструментами. Це сприяє сталому вдосконаленню їх функціоналу у виявленні й обробці веб-контенту, відповідаючи сучасним та майбутнім вимогам безпеки та аналізу інформації в онлайн-середовищі.

1.3 Приклади формальних систем

1.3.1 GATE

GATE [5] визначається своєрідною загальною архітектурою для інженерії тексту, що забезпечує різноманітні функції для обробки текстових даних. Розглядаються основні аспекти, такі як токенізація [6], лематизація [7], розпізнавання частин мови та визначення іменованих сутностей [8]. Окрема увага

приділяється аналізу семантичних зв'язків, взаємодії з кореференцією та вилученню відносин між елементами тексту (рисунок 1.1).

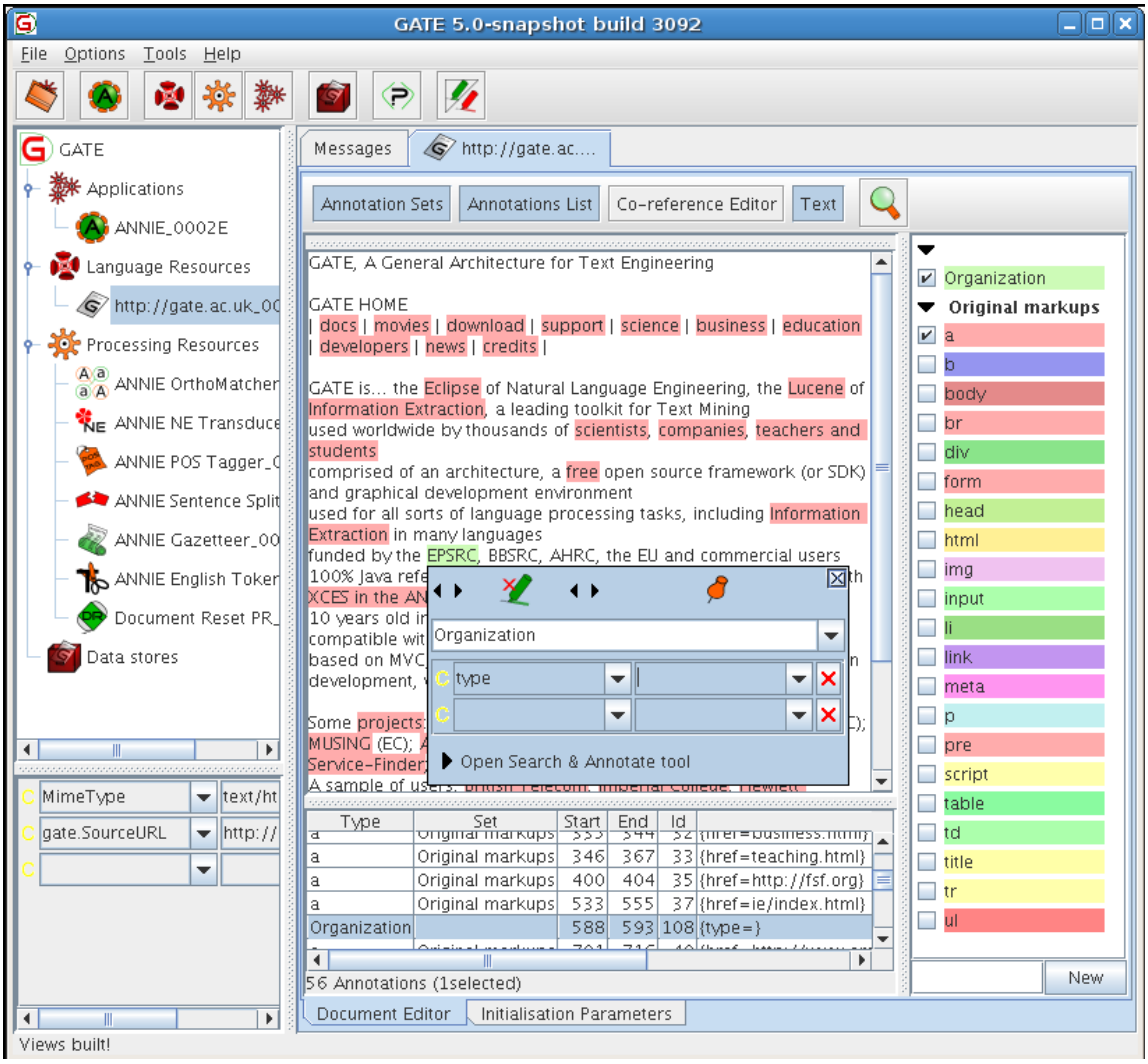


Рисунок 1.1 – Інтерфейс GATE

Під час розгляду GATE також вивчаються його можливості в різних сферах, таких як мовленнєвий аналіз, машинний переклад [9] і розробка чат-ботів [10]. Засвоєння особливостей цієї системи дозволяє отримати глибше розуміння процесів обробки природної мови та визначити її потенціал для вирішення конкретних завдань.

1.3.2 Hemingway editor

Практичне використання Hemingway Editor [11] охоплює широкий спектр користувачів. Блогери можуть використовувати його для створення більш читабельного контенту, що краще сприймається аудиторією. Студенти знаходять цей інструмент корисним для покращення академічних робіт, роблячи їх більш зрозумілими та чіткими. Письменники застосовують його для редагування своїх творів, підвищуючи їхню привабливість для читачів. Журналісти також користуються цим інструментом для спрощення своїх статей, забезпечуючи доступність інформації для широкої аудиторії (рисунок 1.2).

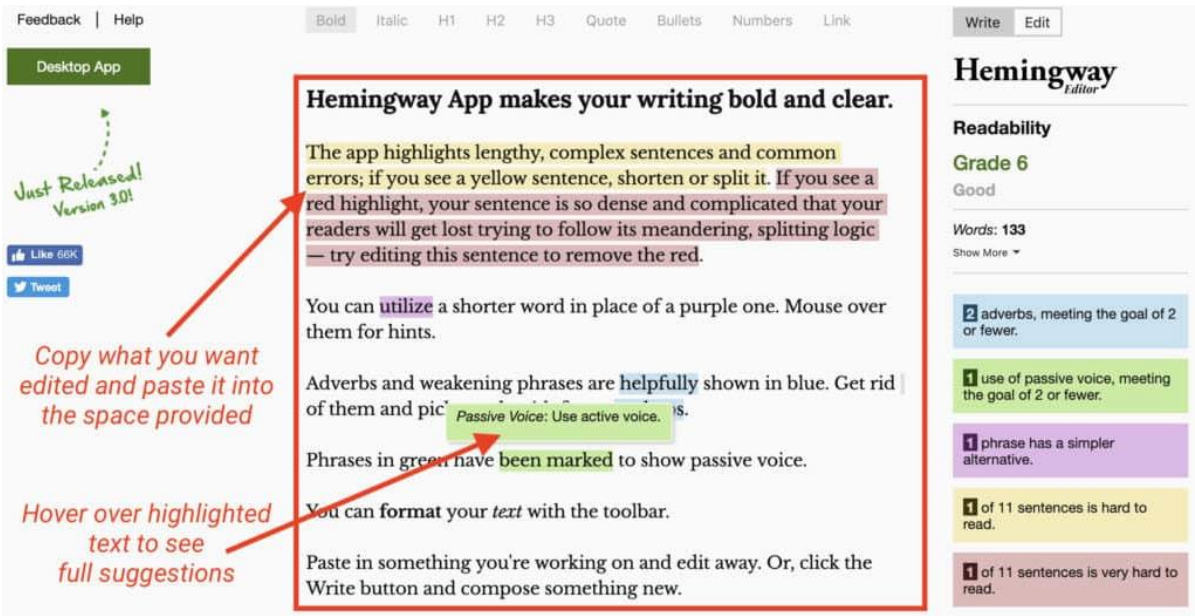


Рисунок 1.2 – Інтерфейс Hemingway editor

Однією з головних переваг Hemingway Editor є його зручний інтерфейс та легкість у використанні, що дозволяє швидко виявляти та виправляти помилки стилю та складні конструкції. Проте, він має деякі обмеження, такі як недостатня глибина аналізу для професійних редакторів і іноді надмірне спрощення тексту без врахування контексту. Незважаючи на ці недоліки, Hemingway Editor залишається корисним інструментом для всіх, хто прагне покращити якість свого письма та зробити його більш зрозумілим для читачів.

1.3.3 Stanford CoreNLP

Комплексний інструментарій для обробки природної мови NLP [12], розроблений на базі Стенфордського університету. Основною метою використання цього інструменту було проведення аналізу текстів з метою отримання різноманітної інформації, такої як токенізація [6], розпізнавання частин мови, лематизація [7], визначення іменованих сутностей, а також вивчення семантичних відносин та залежностей між словами у реченні (рисунок 1.2).

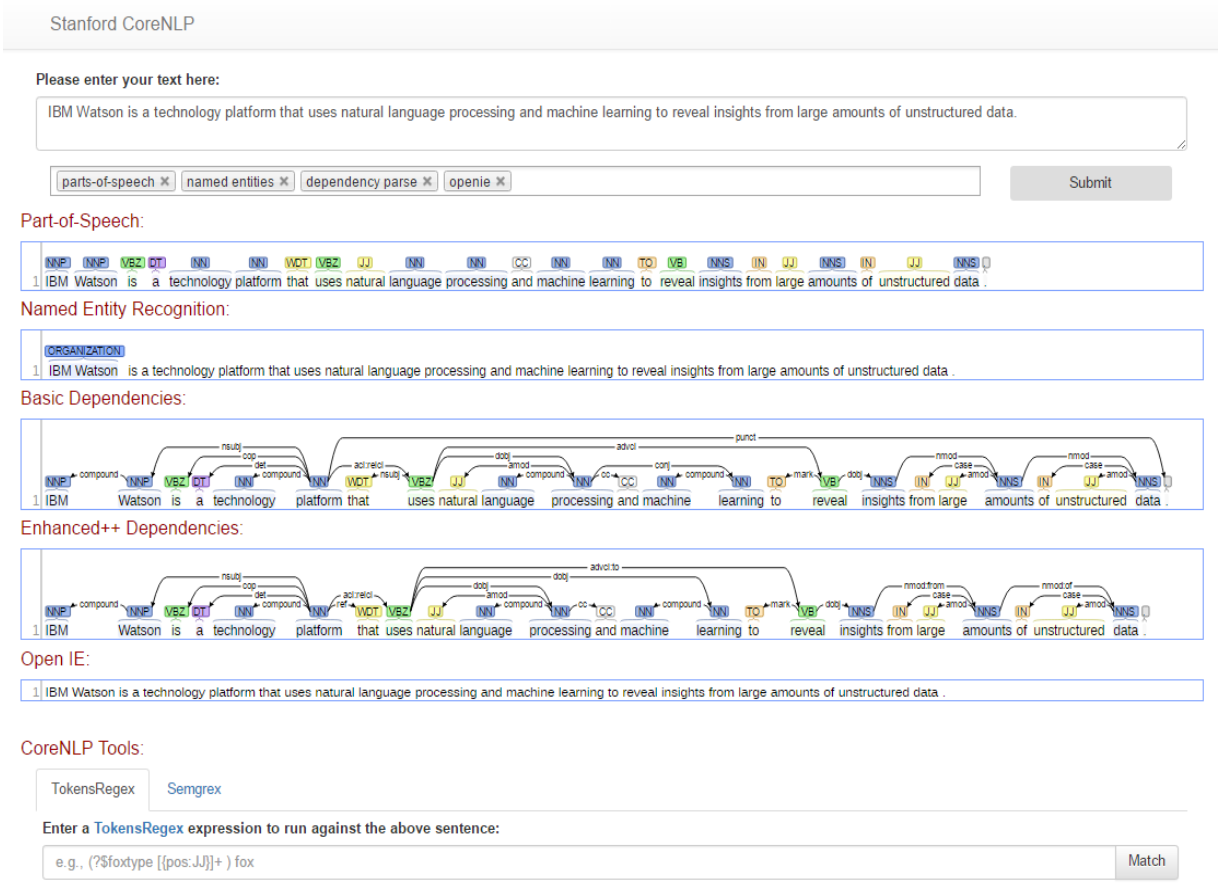


Рисунок 1.2 – Інтерфейс Stanford CoreNLP

Важливим аспектом використання Stanford CoreNLP [12] була можливість проведення аналізу сентименту текстів, що дозволило визначити емоційний тон висловленого контенту. Під час дослідження було звернено увагу на різноманітність функціоналу інструменту, який забезпечив достатній охоплення

різних аспектів обробки природної мови. Використання Stanford CoreNLP в дослідженні дозволило ефективно виконати завдання аналізу текстової інформації, а також виокремити ключові елементи для подальшого вивчення та використання у різних областях, пов'язаних з обробкою природної мови.

1.3.4 Google BERT

Google BERT (Bidirectional Encoder Representations from Transformers) [13] представляє собою важливий крок у розвитку алгоритмів обробки природної мови (NLP). Представлений Google в 2018 році, BERT відзначається тим, що він враховує контекст обох сторін слів у реченні, що робить його особливо ефективним для розуміння семантичних зв'язків та контексту в текстах. Цей алгоритм базується на трансформаторній архітектурі, яка дозволяє йому виявляти та враховувати взаємовідносини між словами у тексті, що поліпшує якість розуміння мови машинами. Однією з ключових переваг BERT є його здатність до контекстуального усвідомлення слів. У порівнянні з попередніми моделями, які розглядали слова ізольовано, BERT враховує слова, які оточують їх у реченні, роблячи його більш контекстуально усвідомленим. Це дозволяє алгоритму краще розрізняти семантичні аспекти та поліпшує результати у завданнях, таких як відповіді на запитання, аналіз тональності та вибірковий переклад.

Google BERT є частиною набору моделей в рамках Hugging Face Transformers, що дозволяє розробникам і дослідникам використовувати цей алгоритм для широкого спектру завдань в області NLP. Завдяки інтеграції в Hugging Face, користувачі мають доступ до попередньо навчених моделей BERT та можливості ефективно використовувати їх для вирішення різних завдань в галузі обробки природної мови (Рисунок 1.3).

Отже, GATE, Stanford CoreNLP і Google BERT представляють собою важливі інструменти в області формальних систем аналізу контенту, кожен з яких володіє

своїми особливостями та перевагами для вирішення завдань обробки та аналізу різноманітного контенту.

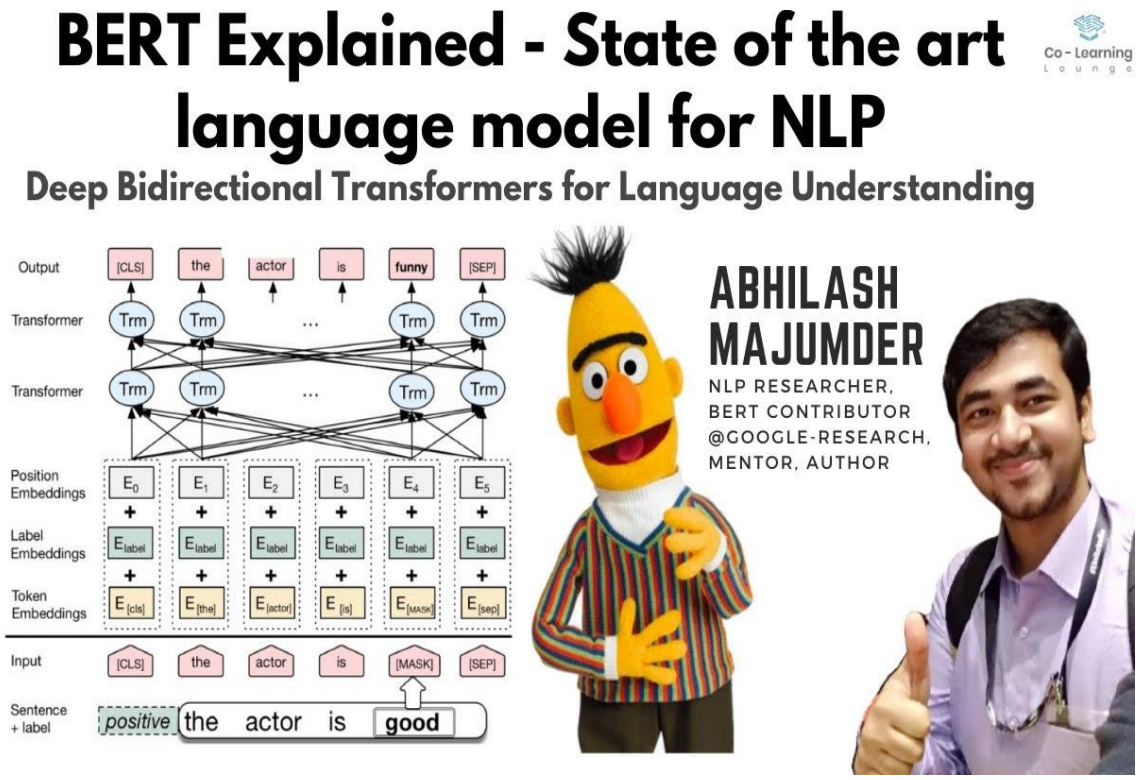


Рисунок 1.3 – Інтерфейс Google BERT

1.4 Обмеження систем

1.4.1 Недосконаліть резутатів

Системи аналізу контенту веб-сайту на базі формальних систем мають свої переваги, але також стикаються з обмеженнями, зокрема і недосконалістю результатів. Навіть найрозвинутіші алгоритми та методи не завжди можуть гарантувати абсолютну точність та ефективність в розпізнаванні та класифікації контенту. Обмеження, пов'язані з недосконалістю результатів, можуть виникати через ряд факторів. Складність мовлення, варіативність виразу та стрімке змінювання інформації можуть ускладнювати завдання аналізу для систем. Також

системи можуть виявляти труднощі у розпізнаванні саркастичного чи іронічного висловлення, або у визначенні нових, еволюційних форм небажаного контенту.

Важливо визнати, що в реальному світі існують ситуації, де системи можуть неправильно інтерпретувати або класифікувати контент. Це може бути особливо важливим у випадках, коли важливо уникати помилок, наприклад, у сфері безпеки або при виявленні небажаного вмісту. Для подолання цих обмежень, важливо вдосконалювати алгоритми, забезпечувати регулярне оновлення систем та впроваджувати нові методи аналізу. Можливості машинного навчання [14] та нейронних мереж [15] можуть допомогти підвищити адаптивність систем та покращити їхню точність у складних умовах. Важливо також враховувати етичні та прозорість у використанні цих систем для забезпечення довіри та визнання користувачами.

1.4.2 Еволюція мови та контенту

Ще одним із ключових обмежень є постійна зміна мовленнєвих та виразних форм в інтернет-просторі. Системи, які можуть бути досконало адаптовані до певного структурованого мовлення, можуть виявлятися менш ефективними у виявленні та розумінні нових, нестандартних виразів чи сленгу, що постійно з'являються в онлайн-спільнотах. Також, швидка еволюція контенту на веб-сайтах може викликати труднощі у розпізнаванні та класифікації нових форм небажаного контенту. Технологічний прогрес та креативність користувачів призводять до появи нових методів вираження та поширення інформації, що може бути викликом для систем аналізу контенту, які можуть бути побудовані на статичних моделях. Для подолання цих обмежень, важливо постійно оновлювати алгоритми та методи аналізу, враховуючи останні тенденції в онлайн-мовленні та контенті. Впровадження технологій машинного навчання та нейронних мереж може також сприяти адаптації систем до еволюції мови та контенту, роблячи їх більш гнучкими та ефективними в пошуку та розпізнаванні нових явищ у веб-просторі.

1.4.3 Етичність та прозорість

Системи аналізу контенту веб-сайту, які базуються на формальних системах, стикаються з важливими етичними та прозорістю обмеженнями, які необхідно ретельно враховувати. Одним з основних етичних аспектів є ризик упередженості та дискримінації в результатах роботи систем. Алгоритми, які використовуються для аналізу контенту, можуть виявити уразливість до певних форм впередженої поведінки, особливо якщо дані, на яких вони навчаються, містять у собі приховані виклики. Для забезпечення етичного використання систем аналізу контенту, важливо розробляти та вдосконалювати алгоритми так, щоб вони уникали створення чи посилення стереотипів та упереджень. Також, важливо проводити регулярні аудити та оцінки, щоб виявляти та виправляти можливі етичні ризики в роботі систем. Другим важливим аспектом є прозорість в роботі систем для користувачів. Користувачі повинні мати можливість розуміти, як система приймає рішення та класифікує контент. Забезпечення доступу до інформації про те, які критерії та алгоритми використовуються, допомагає користувачам розуміти процес та сприяє створенню довіри до системи.

Важливо враховувати, що вартість розробки та впровадження систем аналізу контенту також може включати витрати на підготовку персоналу, навчання та підтримку користувачів. Крім того, постійне оновлення технологічних рішень для відповіді на нові виклики та загрози може додатково підвищити фінансові затрати. Нерівність у доступі до захисних технологій може створити ризик для менших компаній та організацій, які можуть залишитися вразливими перед потенційними кіберзагрозами. Потрібно вдосконалювати доступність цих технологій та створювати ініціативи, спрямовані на забезпечення широкого кола організацій доступом до ефективних засобів захисту. Отже оптимізація процесів та впровадження ефективних стратегій управління витратами допоможе зменшити вартість та зробити технології безпеки більш доступними. Враховуючи ці аспекти,

можна сприяти розвитку більш узгодженої та врівноваженої кібербезпекової інфраструктури.

1.5 Допомога з системами

1.5.1 Вибір відповідної система

Вибір відповідної системи аналізу контенту є важливим етапом для досягнення успішних результатів в багатьох сферах діяльності. Перш за все, важливо провести оцінку своїх конкретних потреб в аналізі контенту. Це може бути фільтрація небажаного вмісту, виявлення загроз безпеці, або інші завдання, які визначаються специфікою вашої діяльності. Далі, необхідно провести дослідження ринку, щоб отримати повну картину про існуючі системи аналізу контенту. Споживач повинен порівняти доступні системи, оцінюючи їхні можливості та вартість. Важливо враховувати не лише функціонал, але й технічні характеристики, надійність та можливості розширення. Це дозволить забезпечити оптимальний вибір системи, яка відповідає конкретним потребам та можливостям бізнесу чи організації. Завершальний етап включає у себе вивчення відгуків користувачів та отримання рекомендацій від інших фахівців у галузі. Це допоможе підтвердити обрану систему та забезпечити її ефективне використання для досягнення поставлених завдань.

1.5.2 Налаштування системи

Рекомендується розпочати впровадження системи з пілотного запуску на обмеженій області або групі контенту. Це дозволить вам перевірити ефективність системи та виявити можливі проблеми, що можуть виникнути під час повної реалізації. Навчання персоналу: переконайтеся, що ваш персонал отримав достатній рівень підготовки та навчання з використання нової системи. Інструкції та семінари можуть бути корисними для того, щоб забезпечити ефективне

використання системи та мінімізувати можливі помилки. Моніторинг та оптимізація: після повного впровадження системи регулярно моніторте її роботу та здійснюйте оптимізації, якщо це необхідно. Враховуйте отримані результати та змінюйте налаштування або додаєте нові функції для забезпечення максимальної продуктивності та відповідності потребам вашої організації.

Важливою складовою ефективності будь-якої системи є проведення систематичного аналізу її результатів. Після завершення виконання завдань і обробки даних важливо ретельно оцінити отримані висновки. Аналіз результатів дозволяє виявити та висвітлити ключові показники та тренди у роботі системи. Для ефективного аналізу результатів необхідно використовувати інструменти моніторингу та звітності. Систематичне спостереження за роботою системи в реальному часі дозволяє оперативно виявляти можливі аномалії або проблеми. Звіти, у свою чергу, стають невід'ємною частиною процесу оцінки ефективності та надають деталізовану інформацію про роботу системи.

На основі отриманих результатів та звітів, важливо вжити заходів щодо оптимізації роботи системи. Виправлення та оптимізації можуть включати в себе внесення коригувань у алгоритми, удосконалення обчислювальних процесів чи впровадження нових методів для підвищення продуктивності системи в цілому. Процес оптимізації є невід'ємною частиною циклу вдосконалення та забезпечує найвищу ефективність системи на довгострокову перспективу.

1.5.3 Підтримка та оновлення

Ключовим аспектом ефективного функціонування будь-якої системи є належна служба підтримки. Важливо переконатися, що усі користувачі мають безперешкодний доступ до служби підтримки системи, щоб вчасно та ефективно вирішувати можливі труднощі, а також отримувати необхідну допомогу та консультації.

Регулярне оновлення системи є критично важливим елементом для забезпечення її адаптації до нових викликів та загроз, що можуть виникнути в веб-середовищі. Впровадження нових версій, патчів та оновлень дозволяє системі не тільки залишатися у найбільш актуальному стані, але й забезпечує виправлення можливих вразливостей, що можуть бути використані для атак та порушень безпеки.

1.6 Розвиток аналізу контенту веб-сайту на базі формальних систем

Штучний інтелект [15] в сучасному світі, тенденції розвитку систем аналізу контенту визначаються використанням штучного інтелекту. Це дозволяє вдосконалити процес обробки та розуміння інформації, що подається на веб-сайтах. Зокрема, системи аналізу контенту з ШІ здатні автоматично розпізнавати, класифікувати та інтерпретувати різні типи контенту, забезпечуючи ефективну фільтрацію та сортування інформації.

Ще однією важливою тенденцією є розвиток систем машинного навчання, які можуть адаптуватися до змін в контенті веб-сайтів. За допомогою нейронних мереж та алгоритмів машинного навчання, системи аналізу контенту можуть навчитися розпізнавати нові теми, тренди та мовні конструкції, щоб надавати більш точні та релевантні результати аналізу. Крім того, зростає увага до розвитку систем аналізу емоційного контенту за допомогою штучного інтелекту. Визначення та розуміння емоційного відгуку користувачів на веб-сайті дозволяє покращити якість обслуговування та персоналізувати взаємодію з відвідувачами.

Розширення можливостей машинного навчання для поліпшення результатів. Тенденції розвитку систем аналізу контенту визначаються також і розширенням можливостей машинного навчання. Ця технологія входить в новий етап свого розвитку, ставши важливим інструментом для поліпшення результатів аналізу контенту та забезпечення більш точного розуміння інформації. Машинне навчання дозволяє системам автоматично вчитися на основі зібраних даних, адаптуватися до

змін у контенті та надавати більш ефективні результати аналізу. Однією з ключових тенденцій є використання глибокого навчання для аналізу контенту веб-сайтів. Глибокі нейронні мережі дозволяють автоматично виявляти складні шаблони та взаємозв'язки в інформації, що допомагає у більш точному класифікуванні та розпізнаванні контенту. Завдяки цьому, системи аналізу можуть визначати не лише ключові слова, а й контекст, що робить аналіз більш обширним та відповідальним. Іншою важливою тенденцією є поєднання машинного навчання з обробкою природної мови, тому це дозволяє системам не лише розпізнавати тексти, але і розуміти їх сенс та емоційний відтінок.

1.7 Постановка завдання

Розробити систему аналізу контенту веб-сайту з використанням формальних систем аналізу тексту. Для виконання завдання потрібно виконати важливі кроки.

Вибір формальних систем для аналізу тексту: Обрати найбільш підходящі формальні системи для розробки системи аналізу контенту, наприклад, контент-аналіз, синтаксичний та семантичний аналіз, використання регулярних виразів та граматичних моделей.

Спроектувати архітектуру системи аналізу контенту, визначивши основні компоненти, їх функціональність та взаємодію.

Розробити програмне забезпечення, яке реалізує обрані формальні системи аналізу тексту. Це може включати розробку парсерів, алгоритмів класифікації тексту, методів екстракції інформації та інші необхідні компоненти.

Провести тестування розробленої системи на реальних даних, оптимізувати її продуктивність та точність аналізу.

Написати детальну документацію по розробленій системі, включаючи технічну документацію, інструкції для користувачів та опис методології.

Метою дипломної роботи є підвищення ефективності використання системи аналізу контенту на веб-сторінках на базі формальних систем, це значить, що

потрібна розробка більш точних та ефективних методів аналізу тексту, які можуть обробляти великі обсяги даних з веб-сторінок, застосувати передові формальні систем аналізу для підвищення якості та швидкості аналізу, та інтеграція нових підходів у існуючі системи аналізу контенту для поліпшення їх продуктивності.

Об’єкт дослідження є процес аналізу тексту на веб-сторінках програмою, створеною на базі формальних систем, це потребує веб-сторінки як джерела даних для аналізу, також процеси збору, обробки та аналізу текстової інформації, і програмне забезпечення, яке здійснює аналіз тексту.

Предметом дослідження є система аналізу веб-контенту на базі формальних систем, питання які потрібно розглянути формальні методи та алгоритми аналізу тексту, і архітектури системи аналізу контенту, ну і звісно методи інтеграції та застосування формальних систем для аналізу тексту на веб-сторінках.

Таким чином, робота охоплює широкий спектр завдань, від теоретичного аналізу існуючих методів до практичної реалізації та тестування розробленої системи, з метою підвищення ефективності та точності аналізу веб-контенту.

1.8 Висновки

У першому розділі було розглянуто загальні відомості про існуючі системи аналізу контенту веб-сайтів на базі формальних систем. Огляд показав, що сучасні системи використовують широкий спектр методів та технологій для ефективного аналізу текстових даних. Обробка натуральної мови NLP та машинне навчання ML є ключовими складовими цих систем, що дозволяють автоматизувати процеси класифікації тексту, семантичного аналізу та виявлення настроїв.

Таким чином, сучасні системи аналізу контенту веб-сайтів на базі формальних систем є інструментами, що забезпечують глибоке розуміння текстових даних, сприяють покращенню пошукової оптимізації.

2 ПРОЕКТУВАННЯ ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ І МЕТОДІВ РЕАЛІЗАЦІЇ СИСТЕМИ

2.1 Функціональні вимоги системи

Процес проектування програмного засобу [16] для аналізу контенту веб-сайтів на базі формальних систем, а також методи його реалізації визначаються вимогою до системи, потім розробляється архітектура та описуються методи реалізації. Функціональні вимоги визначають основні завдання і функції, які повинна виконувати система аналізу контенту веб-сайтів на базі формальних систем за допомогою збору данних при автоматичному скануванні веб-сторінок, розкладу завдань для підтримки актуальності збірних даних. Також слід зазначити що важливою є ще підтримка різних типів контенту таких як сканування статичних та динамічних веб-сторінок.

Використання технік веб-скрейпінгу [17] для автоматичного завантаження HTML-коду веб-сторінок. Почнемо з того що веб-скрейпінг — це метод автоматизованого збору даних з веб-сайтів. Використовуючи різні техніки та інструменти, можна завантажувати HTML-код веб-сторінок та витягати з нього потрібну інформацію. В веб-скрейпінгу існує бібліотеки та фреймворки, такі як BeautifulSoup [18], Scrapy [19], Puppeteer [20], Selenium [21] та інші. Ці інструменти дозволяють автоматизувати процес отримання HTML-коду та аналізу його структури. Під час витягування данних, вже після завантаження HTML-коду, веб-скрейпери використовують різні методи для витягування даних, такі як XPath [22], CSS-селектори [23], та регулярні вирази [24]. Ці методи дозволяють точно ідентифікувати, витягати потрібні елементи зі сторінки, такі як ось заголовки, текстові блоки, посилання, таблиці. Слід зазгначити що веб-скрейпінг може підпадати під дію законодавства про авторські права та умови використання веб-сайтів, тому важливо завжди перевіряти політики веб-сайтів щодо скрейпінгу та дотримуватися відповідних правових норм.

Налаштування частоти сканування, щодня чи щотижня, для підтримки актуальності зібраних даних, тому після налаштування веб-скрейпера важливо визначити частоту сканування веб-сторінок, щоб зібрані дані залишалися актуальними. У проекті використовуються інструменти які потрібні для планування завдань, такі як cron, для Unix-подібних систем [25], Windows Task Scheduler [26], а також спеціалізовані сервіси, такі як Apache Airflow [27], які дозволяють налаштовувати складні робочі процеси для збору та обробки даних.

Сучасні веб-сторінки можуть містити різноманітні типи контенту, і веб-скрейпери повинні бути здатні працювати з усіма цими типами. Для сканування статичних веб-сторінок достатньо завантажити HTML-код та витягти з нього необхідні дані. Це найпростіший випадок, оскільки контент сторінки не змінюється без перезавантаження. багато сучасних веб-сайтів використовують динамічні сторінки наприклад: JavaScript [28] для динамічного завантаження контенту. В таких випадках веб-скрейперам потрібно взаємодіяти з JavaScript-движком [29] сторінки. Інструменти, такі як Puppeteer або Selenium, дозволяють автоматизувати браузери та отримувати контент динамічних сторінок. Наступне AJAX [30], що дозволяє веб-сторінкам завантажувати дані асинхронно, без перезавантаження сторінки. Веб-скрейпери повинні мати можливість перехоплювати та обробляти ці запити, щоб отримати необхідні дані. Це може вимагати аналізу мережевих запитів у браузері для визначення потрібних ендпоінтів. Крім текстового контенту, веб-скрейпери можуть бути налаштовані для завантаження зображень, відео, аудіо та інших медіа-файлів. Це може включати завантаження файлів безпосередньо через URL або перехоплення потокових даних, в решті-решт забезпечення підтримки різних типів контенту дозволяє отримувати максимально повну та актуальну інформацію з веб-сторінок, що є критично важливим для багатьох застосувань, включаючи аналіз ринку, моніторинг новин, дослідження конкурентів та багато іншого.

2.1.1 Обробка даних

Обробка за допомогою парсинг HTML-коду для виділення текстового контенту та метаданих. Мета цього методу витягти текстовий контент та метадані з HTML-документу. Крім використання бібліотек для парсингу HTML, таких як BeautifulSoup або lxml [31] у Python, важливо також враховувати можливість обробки різних форматів HTML-коду, враховуючи те що деякі веб-сайти можуть використовувати JavaScript для динамічного завантаження контенту, тому це вимагає використання інших інструментів, які можуть взаємодіяти з JavaScript, таких як Selenium або Scrapy зі вбудованим JavaScript-двигуном. Після отримання HTML-коду можна використовувати різні методи для виділення текстового контенту та метаданих. Наприклад, для отримання текстового контенту можна використовувати не лише теги тексту, такі як <p>, , , а й класи або ідентифікатори, що відповідають текстовим блокам. Крім того, можна використовувати CSS-селектори для точного вибору елементів з текстовим контентом.

Щодо метаданих [32], важливо не лише отримати їх з відповідних тегів HTML, таких як <title>, <meta name="keywords">, <meta name="description">, а й врахувати можливість різних форматів внутрішнього представлення метаданих, таких як мікророзмітка, яка може бути використана для вбудованої семантики на веб-сторінці. Наступне є токенізація, що представляє собою розбиття тексту на окремі слова або токени. Це можна зробити за допомогою регулярних виразів у Python або використовуючи бібліотеки, такі як NLTK [33] або SpaCy [34]. Токенізація може враховувати різні аспекти мови, такі як розділові знаки, абревіатури та спеціальні символи. Після токенізації слідує нормалізація, процес приведення тексту до стандартизованої форми, приведення всіх літер до нижнього регістру для уникнення різниці між великими і малими літерами, видалення пунктуації та спеціальних символів, які не несуть смислового навантаження, та видалення стоп-слів, які не несуть значення, таких як "і", " це ", "але", "або", це

можна зробити за допомогою готових списків стоп-слів, які надають бібліотеки NLTK або SpaCy.

Лематизація являє собою процес приведення слова до його початкової форми, наприклад, "running" -> "run". Лематизацію можна здійснити за допомогою бібліотек NLTK або SpaCy. Якщо потрібно усічення слів до їх кореневих форм, наприклад, "running" -> "run", то використовується процес під назвою стемінг. Використовуються алгоритми, такі як PorterStemmer [35] або SnowballStemmer [36] з бібліотеки NLTK. Наступним потрібно фільтрувати дублікати контенту, використовуючи методи, такі як хешування тексту, алгоритми швидкого пошуку, алгоритм Rabin-Karp [37] або алгоритм Кнута-Морріса-Прата [38], для виявлення дублікатів. Алгоритми Rabin-Karp і КМП ефективно шукають підрядки в тексті, що дозволяє виявляти повторювані фрагменти. Також можна використовувати бібліотеки для обробки природної мови, які вміють розпізнавати семантичну схожість текстів і виявляти дублікати на основі цього.

Додаткові методи фільтрації дублікатів включають шинглинг, йдеться про розбиття тексту на перекриваючі підрядки, які потім хешуються для швидкого порівняння. Цей метод допомагає виявляти часткові дублікатів або сильно схожі тексти, алгоритми MinHash [39] та Locality-Sensitive Hashing [40] використовують шинглинг та спеціальні хеш-функції для ефективного пошуку схожих документів у великих колекціях.

Отже, обробка текстового контенту та метаданих з HTML-документів потребує використання спеціалізованих інструментів, таких як BeautifulSoup, lxml, Selenium та Scrapy. Важливими етапами цього процесу є токенізація, нормалізація, лематизація та стемінг, які забезпечують приведення тексту до стандартизованої форми. Для фільтрації дублікатів використовуються алгоритми хешування, Rabin-Karp, Кнута-Морріса-Прата, а також методи шинглинг та Locality-Sensitive Hashing. Це дозволяє ефективно виявляти та усувати дублікати, забезпечуючи чистоту та унікальність контенту. Таким чином, застосування цих методів і технологій автоматизує аналіз та структурування інформації з веб-сторінок.

2.1.2 Аналіз даних

Семантичний аналіз дозволяє автоматично визначати теми та ключові слова в текстових даних, використовуючи методи обробки природної мови (NLP). Це допомагає отримувати інсайти з великих обсягів тексту, автоматизувати класифікацію документів, та покращувати пошук інформації. Використання алгоритмів обробки природної мови (NLP) для розпізнавання та класифікації тем забезпечують аналіз тексту для визначення його змісту та контексту. В загальності Latent Dirichlet Allocation [41] це генеративна статистична модель, яка визначає теми в колекції документів, розглядає кожен документ як суміш декількох тем, а кожну тему, як суміш слів. Наприклад: у наборі новинних статей LDA може виявити теми, такі як "політика", "спорт", "технології", та показати, які слова найчастіше зустрічаються в кожній темі.

Моделі на основі трансформерів, нейронні мережі, такі як BERT, можуть розуміти контекст та залежності між словами в тексті. Це використовується для класифікації тексту та розпізнавання тем. Приклад - аналіз дописів у соціальних мережах для виявлення основних тем дискусій, таких як "зміна клімату", "здоров'я", "економіка". Крім того, ці моделі можуть бути застосовані для виявлення емоційного забарвлення тексту, визначення авторства або навіть для автоматичного генерування текстів. Висока точність та гнучкість трансформерів робить їх незамінними інструментами для обробки природної мови в різних сферах.

NLTK надає інструменти для попередньої обробки тексту (токенізація, стемінг, лематизація) та аналізу частоти слів. Приклад: Використання NLTK для видалення стоп-слів та лематизації тексту перед застосуванням LDA. Це дозволяє зменшити шум у тексті та покращити точність моделі для виявлення прихованих тем. Крім того, NLTK можна використовувати для розпізнавання частин мови, синтаксичного аналізу та створення словників і тезаурусів. Такий підхід сприяє більш глибокому аналізу тексту і покращенню результатів у різноманітних завданнях обробки природної мови.

SpaCy забезпечує швидку та ефективну обробку тексту, включаючи розпізнавання іменованих сутностей, частин мови та залежностей між словами. Приклад: Використання SpaCy для попередньої обробки тексту та побудови векторів документів для кластеризації або LDA. Завдяки своїй високій продуктивності та точності, SpaCy може ефективно обробляти великі обсяги тексту, забезпечуючи якісну підготовку даних для подальшого аналізу. Крім того, SpaCy підтримує інтеграцію з іншими інструментами та бібліотеками, що дозволяє створювати комплексні рішення для обробки природної мови. Такий підхід допомагає досягти кращих результатів у задачах класифікації, та тематифікації тексту.

Gensim [42] спеціалізується на побудові тематичних моделей, включаючи LDA та Word2Vec [43]. Надає інструменти для навчання моделей на великих наборах даних. Приклад: Використання Gensim для побудови LDA-моделі для великої колекції новинних статей. Цей підхід дозволяє виявляти основні теми в новинних потоках та аналізувати їх еволюцію з часом. Крім того, Gensim забезпечує ефективне оброблення тексту та створення векторних уявлень слів, що може бути використано для завдань кластеризації, пошуку подібних документів та побудови рекомендаційних систем. Завдяки своїй масштабованості та ефективності, Gensim є потужним інструментом для роботи з великими текстовими корпусами.

Використання TF-IDF [44] для визначення важливих ключових слів у тексті. TF-IDF - це статистичний метод для оцінки важливості слова в документі відносно до всієї колекції документів. Це один з основних методів для виявлення ключових слів. Кількість разів, коли слово зустрічається в документі відображає важливість слова в конкретному документі. Приклад: У статті про технології слово "інновація" може мати високе значення TF. Inverse Document Frequency (IDF) [45]. Чим рідше слово зустрічається в інших документах, тим вище його значення IDF. Приклад: Слово "інновація" може мати високе значення IDF, якщо воно зустрічається нечасто в інших статтях колекції.

2.1.3 Зберігання даних. Індексація та кешування даних

Використання реляційних або нереляційних баз даних для зберігання зібраних та оброблених даних є ключовим аспектом управління інформацією. Реляційні бази даних, такі як MySQL [46] або PostgreSQL [47], можуть використовуватися для структурованих даних, забезпечуючи надійне зберігання та ефективні запити за допомогою SQL. Вони добре підходять для даних з чітко визначеною схемою, таких як таблиці з конкретними атрибутами. Нереляційні бази даних, наприклад MongoDB [48] або Cassandra [49], можуть бути корисними для зберігання неструктурованих даних або документів, дозволяючи гнучке управління даними, які не вписуються в традиційні таблиці. Вони підтримують динамічні схеми та добре масштабуються для великих обсягів даних. Використання обох типів баз даних може забезпечити комплексний підхід до зберігання та обробки даних, що дозволяє оптимально використовувати переваги кожної технології відповідно до потреб проекту.

Індексація та кешування даних. Індексація полягає у створенні індексів, які значно покращують швидкість пошуку та доступу до даних. Завдяки індексації можна миттєво знаходити необхідну інформацію за допомогою різних запитів та фільтрів, що значно зменшує час обробки запитів і підвищує ефективність роботи з великими обсягами даних. Кешування полягає у тимчасовому збереженні результатів попередньо обчислених або запитаних даних. Це дозволяє значно підвищити швидкість доступу до інформації, особливо у випадках повторних запитів на ті самі дані. Кешування зменшує навантаження на основну базу даних та мережеві ресурси, покращуючи загальну продуктивність системи.

2.1.4 Інтерфейс користувача

Веб-інтерфейс надає можливість користувачам налаштовувати параметри системи, такі як параметри аналізу даних, налаштування моделей, окрім цього, інтерфейс забезпечує зручний і інтуїтивно зрозумілий доступ до цих налаштувань,

підтримуючи різні рівні доступу для різних категорій користувачів. Користувачі мають можливість зберігати і завантажувати свої конфігурації, а також отримувати підказки та допомогу щодо оптимального налаштування параметрів. Інтерфейс підтримує можливість попереднього перегляду змін та їх тестування перед застосуванням, щоб користувачі могли переконатися у правильності налаштувань без ризику вплинути на роботу системи.

При візуалізація даних важливо мати можливість візуалізувати дані за допомогою графіків, діаграм для кращого розуміння результатів аналізу. Панель адміністратора для управління користувачами, панель адміністратора повинна надавати зручні інструменти для керування користувачами системи, включаючи створення, редагування та видалення облікових записів, а також управління ролями та дозволами доступу.

Адміністратор буде мати можливість налаштовувати різні параметри системи, такі як з'єднання з базами даних, налаштування безпеки, параметри аналізу даних. Панель адміністратора також має включати інструменти для моніторингу та аналізу даних, щоб адміністратор мав можливість відслідковувати працездатність системи та виявляти можливі проблеми. Ще важливо, щоб панель адміністратора забезпечувала доступ до логів системи для детального аналізу подій та швидкого реагування на інциденти. Крім того, повинні бути можливості для створення та управління резервними копіями даних для забезпечення їхньої безпеки та відновлення у випадку збоїв. Панель адміністратора також може включати функціонал для налаштування повідомлень та сповіщень про критичні події, що дозволяє адміністратору оперативно реагувати на зміни у стані системи.

2.2 Нефункціональні вимоги

Нефункціональні вимоги визначають загальні характеристики системи, які впливають на її продуктивність, зручність використання та надійність. Одні з основних нефункціональних вимог, що важливо враховувати під час проектування

програмно-технічного засобу, включають в себе вимоги до продуктивності, якості, безпеки та масштабованості системи.

Продуктивність системи може визначатися швидкістю обробки даних, відгуком на запити користувачів та завантаженням ресурсів. Зручність використання передбачає забезпечення зручного та інтуїтивно зрозумілого інтерфейсу користувача, а також оптимального взаємодії з системою. Надійність включає в себе стійкість до відмов, відновлення після збоїв та забезпечення доступності сервісу для користувачів у будь-який час. Всі ці аспекти є ключовими при розробці та впровадженні програмно-технічних засобів у системі.

Висока продуктивність системи забезпечується здатністю швидко обробляти та аналізувати великі обсяги даних. Це включає оптимізацію алгоритмів обробки даних та ефективне використання апаратних ресурсів. Для досягнення високої швидкості обробки, використовуються методи паралельного обчислення, розподілені системи обробки даних та сучасні бази даних, здатні працювати з великими масивами інформації.

Швидкий аналіз даних дозволяє користувачам оперативно отримувати необхідну інформацію та приймати обґрунтовані рішення на основі актуальних даних, що особливо важливо в бізнес-середовищі, де швидкість реакції може мати вирішальне значення. Оптимізація завантаження веб-сторінок є також важливим аспектом продуктивності системи, який безпосередньо впливає на користувацький досвід.

Веб-сторінки повинні завантажуватися швидко, забезпечуючи миттєвий доступ до інформації та функціоналу системи. Для цього будуть використані методи мінімізації розміру файлів, асинхронне завантаження ресурсів, кешування даних та оптимізація серверного коду. Використання мереж доставки контенту CDN [50] також допомагає зменшити час завантаження, доставляючи контент з серверів, розташованих ближче до кінцевих користувачів.

2.2.1 Надійність

Надійність системи є критично важливою для забезпечення її стійкості до збоїв та безперервної роботи. Стійкість до збоїв означає здатність системи продовжувати функціонувати коректно навіть у разі відмови одного або кількох її компонентів. Це може бути досягнуто за рахунок використання кластеризації серверів, де навантаження розподіляється між декількома вузлами, що дозволяє системі залишатися функціональною, навіть якщо один з вузлів виходить з ладу. Звісно ще важливим є використання механізмів автоматичного перезапуску та самовідновлення для забезпечення мінімального часу простою та стабільної роботи. Механізми резервного копіювання та відновлення даних забезпечують збереження та відновлення критично важливої інформації у разі її втрати або пошкодження. Це включає регулярне створення резервних копій даних, зберігання їх у захищених місцях, а також наявність чітких процедур для швидкого відновлення даних у разі потреби.

2.2.2 Сумісність

Сумісність з основними веб-браузерами та операційними системами є важливим аспектом для забезпечення доступності та зручності використання системи. Це свідчить, що веб-інтерфейс системи повинен коректно відображатися та функціонувати у всіх популярних веб-браузерах, таких як Google Chrome, Mozilla Firefox, Microsoft Edge, Safari. Також система повинна працювати на різних операційних системах, включаючи Windows, macOS, Linux, таке досягається за рахунок використання стандартних веб-технологій та ретельного тестування інтерфейсу на різних платформах. Інтеграція з іншими системами через API дозволяє забезпечити взаємодію з різноманітними зовнішніми сервісами та додатками, що значно розширює функціональні можливості системи. Використання відкритих та стандартизованих API дозволяє легко підключати систему до інших програмних продуктів, таких як аналітичні платформи, чи

соціальні мережі, і забезпечує обмін даними в режимі реального часу, автоматизацію процесів та підвищення загальної ефективності роботи системи. Ще інтеграція через API також дозволяє забезпечити гнучкість та масштабованість, оскільки система може бути легко адаптована до змін та нових вимог бізнесу.

2.3 Вимоги до безпеки

2.3.1 Резервне копіювання, відновлення, автентифікація

Резервне копіювання та відновлення є важливим у забезпеченні безпеки та надійності системи. Вимоги до цього процесу визначають, як система повинна зберігати та відновлювати дані в разі втрати або пошкодження. Також важливо регулярність копіювання, коли визначається частота, з якою система створює резервні копії даних, це може бути щоденне, щотижневе, щомісячне або інше регулярне розкладування. Методи зберігання вимагають визначити, яким чином будуть зберігатися резервні копії, на локальних пристроях чи у хмарних сховищах.

Захист резервних копій від несанкціонованого доступу та втрати є важливим аспектом безпеки даних. Це може включати шифрування даних, встановлення контрольованого доступу та інші заходи безпеки. Важливо також визначити максимальний час, протягом якого система повинна бути відновлена до нормальної роботи після виникнення проблеми.

Автентифікація та авторизація є важливими аспектами забезпечення безпеки системи, які дозволяють контролювати доступ користувачів та захищати конфіденційні дані. Вимоги до цього процесу визначають, як система повинна перевіряти ідентичність користувачів та управляти їх доступом. Автентифікація вимагає, щоб система перевіряла ідентичність користувачів за допомогою унікальних ідентифікаторів, таких як імена користувачів та паролі.

2.3.2 Шифрування даних

Шифрування даних є забеспечує безпеку системи, що дозволяє захистити конфіденційні дані під час їх передачі та зберігання. Вимоги до цього процесу визначають, як система повинна захищати дані від несанкціонованого доступу. Шифрування при передачі даних по мережі, щоб уникнути перехоплення та читання інформації сторонніми особами. Це може бути досягнуто за допомогою протоколів шифрування, таких як SSL / TLS [51], і в загальному забезпечить захист даних між клієнтом та сервером.

Шифрування виконується не тільки при передачі через мережу а й при зберіганні даних. Потрібно захист даних на стороні сервера шляхом їх шифрування перед зберіганням у базі даних або файловій системі, і це свідчить, що навіть у разі несанкціонованого доступу до даних, вони будуть недоступні для читання без ключа розшифрування.

2.4 Архітектура системи

Загальна структура системи для аналізу контенту веб-сайту на базі формальних систем є добре структурованою та модульною, щоб забезпечити ефективну обробку та аналіз великої кількості даних. Модуль збору даних відповідає за автоматичне завантаження та збирання даних з веб-сторінок. Він може включати скрапінг HTML-коду сторінок, вилучення текстового контенту, метаданих та іншої важливої інформації для подальшого аналізу. Модуль аналізу контенту застосовує формальні системи для обробки зібраного контенту, використовуючи методи аналізу тексту, виявлення патернів та класифікації даних для отримання корисної інформації з веб-сайтів. Модуль зберігання даних відповідає за зберігання оброблених даних у базі даних або іншій системі зберігання, забезпечуючи доступ до даних для подальшого аналізу та використання. Модуль візуалізації результатів відповідає за створення графіків, діаграм та звітів, щоб користувачі могли легко інтерпретувати отримані дані.

Модульна архітектура дозволяє легко розширювати функціональність системи, змінюючи або додаючи нові компоненти без необхідності повного перероблення коду. Загально кожен модуль виконує чітко визначені завдання, що сприяє зручності управління та розвитку системи.

2.4.1 База даних

Для зберігання як структурованих, так і неструктурованих даних у системі аналізу контенту веб-сайту використовуються реляційні та NoSQL бази даних [52]. Реляційні бази даних, такі як PostgreSQL, забезпечують надійне зберігання структурованої інформації. PostgreSQL є надійною базою даних, яка підтримує широкий спектр функцій, включаючи складні SQL запити [53], транзакції та забезпечення цілісності даних. Це робить PostgreSQL ідеальним вибором для зберігання структурованих даних (рисунок 2.1).

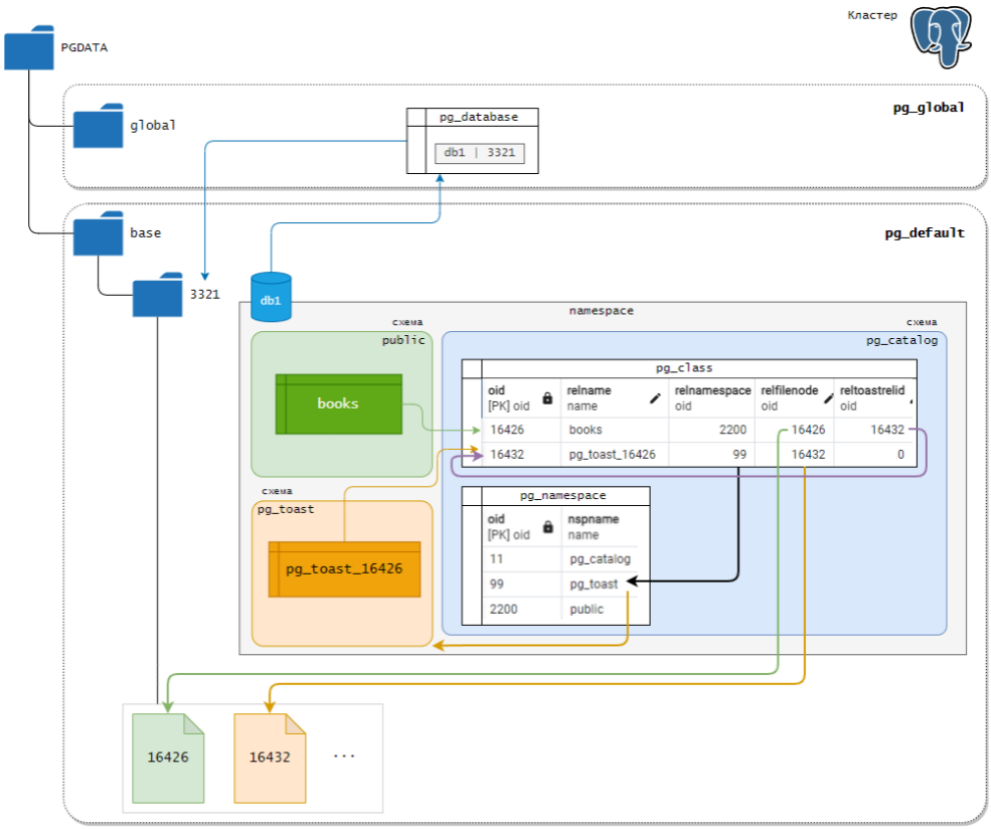


Рисунок 2.1 – Структура зберігання даних в PostgreSQL

PostgreSQL дозволяє ефективно взаємодіяти з даними через SQL запити, що забезпечує гнучкість у доступі та обробці інформації. Крім того, ця база даних підтримує складні операції та транзакції, що гарантує консистентність даних. Інтеграція з іншими системами та сервісами також легко реалізується завдяки підтримці стандартів SQL та численних інтерфейсів програмування. Це дозволяє зручно впроваджувати та підтримувати базу даних у різних проектах, забезпечуючи стабільну та надійну роботу системи.

NoSQL бази даних, такі як MongoDB, відмінно підходять для збереження неструктурованих даних. MongoDB надає гнучкий підхід до схеми даних, що дозволяє зберігати текстовий контент, зображення та інші типи неструктурованої інформації у форматі JSON. Це забезпечує зручність роботи з даними, які не підходять для зберігання у традиційних реляційних базах даних. MongoDB також відома своєю здатністю легко масштабуватися, що дозволяє обробляти великі обсяги даних без втрати продуктивності.

Крім того, MongoDB підтримує високу продуктивність при роботі з великими масивами даних та забезпечує швидкий доступ до інформації завдяки індексуванню та іншим оптимізаціям. Це робить MongoDB ідеальним вибором для зберігання динамічного контенту, який постійно змінюється та доповнюється. Взаємодія з даними у форматі JSON дозволяє розробникам легко маніпулювати неструктурованою інформацією, забезпечуючи гнучкість та ефективність роботи з різними типами контенту.

2.4.2 Веб-інтерфейс

Веб-інтерфейс є елементом системи аналізу контенту веб-сайту, який забезпечує зручний доступ до функціональності системи для користувачів. Він реалізується з використанням сучасних веб-фреймворків, таких як Django [54] або Flask [55], що забезпечує гнучкість, зручність у використанні та розширюваність. Потужним веб-фреймворком на базі Python є Django, який пропонує вбудовані

рішення для багатьох типових задач веб-розробки, таких як маршрутизація, обробка запитів, аутентифікація користувачів та робота з базами даних. Django також підтримує шаблонні системи, що дозволяє легко створювати динамічні веб-сторінки. Перевагами можна зазначити високу безпеку, масштабованість та швидкість розробки. Якщо Django є веб-фреймворком, то Flask є мікрофреймворком для Python, який надає більше гнучкості та простоти в порівнянні з Django. Flask підходить для створення менш складних додатків або сервісів, де потрібна висока кастомізація і забезпечує основні функції веб-фреймворків, такі як обробка запитів та маршрутизація, і дозволяє легко інтегрувати додаткові компоненти та бібліотеки відповідно до потреб проекту.

Отже в результаті йдеться що Django або Flask дуже корисний для розробки веб-інтерфейсу, бо забезпечує баланс між потужністю та гнучкістю, дозволяючи створювати як складні, так і прості веб-додатки, що відповідають потребам користувачів та системи.

2.5 Вибір технологій та інструментів

2.5.1 Мова програмування Python, та її переваги

Python [56] є одним із найпопулярніших мов програмування для розробки систем аналізу контенту веб-сайтів завдяки своїм численним перевагам. Використання Python дозволяє реалізувати проект швидко, ефективно та з високою якістю. Мова має широкий спектр бібліотек, які спрощують розробку систем аналізу контенту: для веб-скрапінгу можна використовувати BeautifulSoup і Scrapy, для обробки тексту та NLP - NLTK, SpaCy та Gensim, а для побудови моделей машинного навчання - scikit-learn, TensorFlow та PyTorch [57]. Python надає зручні інструменти для роботи з текстом, що є критично важливим для аналізу контенту, включаючи регулярні вирази, потужні бібліотеки для NLP та інструменти для обробки великих обсягів тексту. Відомий своєю простотою та читабельністю коду, Python дозволяє швидко розробляти та підтримувати проект, його гнучкість

дозволяє легко інтегрувати різні компоненти системи, такі як бази даних, веб-фреймворки та інструменти машинного навчання. Python є популярним вибором для веб-розробки завдяки фреймворкам, таким як Django та Flask, які забезпечують просте створення та розгортання веб-додатків, а також інтеграцію з іншими компонентами системи. Мова має сильні інструменти для аналізу даних, такі як Pandas та NumPy, які дозволяють легко маніпулювати та аналізувати великі обсяги даних, що є важливим для побудови ефективних алгоритмів аналізу контенту. Python має велику та активну спільноту розробників, що означає наявність численних ресурсів, документації та підтримки, що робить розробку більш ефективною, оскільки можна легко знайти відповіді на питання та отримати допомогу. Python підтримує розробку масштабованих систем, що дозволяє обробляти великі обсяги даних та працювати з різними типами даних, що важливо для систем аналізу контенту, які можуть зростати з часом та вимагати обробки все більшої кількості інформації. Завдяки цим перевагам, Python є оптимальним вибором для реалізації системи аналізу контенту веб-сайтів на базі формальних систем, забезпечуючи ефективний та гнучкий інструментарій для збору, обробки, аналізу та візуалізації даних, що є критично важливим для успіху проекту.

2.5.2 Порівняння баз даних PostgreSQL та MongoDB

Переваги PostgreSQL. PostgreSQL є однією з найпопулярніших реляційних баз даних завдяки своїй надійності та функціональності. Основними перевагами є повна підтримка ACID-транзакцій [58], що забезпечує високу консистентність і надійність даних. Це робить PostgreSQL ідеальним вибором для застосувань, де критично важливо мати гарантії цілісності даних, таких як фінансові системи та системи управління даними.

Недоліки PostgreSQL. Незважаючи на численні переваги, PostgreSQL має й деякі недоліки. Одним з основних є складність у налаштуванні та управлінні, особливо коли мова йде про масштабування. Реляційна архітектура PostgreSQL

передбачає вертикальне масштабування, що може бути обмеженням у випадках, коли потрібно обробляти дуже великі обсяги даних.

Переваги MongoDB. MongoDB, як документо-орієнтована база даних NoSQL, пропонує значну гнучкість у роботі з даними. Вона дозволяє зберігати дані у вигляді документів JSON, що забезпечує динамічну схему. Це означає, що документи в одній колекції можуть мати різну структуру, що є великою перевагою для швидко змінюваних проектів або систем, де структура даних не є фіксованою.

Недоліки MongoDB. Основним недоліком MongoDB є менша консистентність даних порівняно з реляційними базами даних. Хоча MongoDB підтримує ACID-транзакції, ця підтримка є обмеженою для окремих документів і розширеною до кількох документів з версії.

Разом бази даних, PostgreSQL та MongoDB, мають свої сильні та слабкі сторони, що робить придатними для різних типів застосувань. PostgreSQL вирізняється своєю надійністю, підтримкою ACID-транзакцій та потужною системою індексації, що робить ідеальною для фінансових систем та проектів, де важлива консистентність даних і складні аналітичні запити. Однак, жорстка схема та складність у налаштуванні й масштабуванні можуть бути обмеженням для швидко змінюваних або великих проектів.

MongoDB, навпаки, пропонує високу гнучкість у роботі з даними завдяки динамічній схемі та легкому горизонтальному масштабуванню. Це робить базу даних добрим вибором для швидко розвиваючих веб-додатків та систем управління контентом. Проте, менша консистентність даних та обмежені можливості для складних аналітичних запитів можуть створювати труднощі для проектів, що вимагають високої точності та складної обробки даних.

Вибір між PostgreSQL та MongoDB залежить від специфічних потреб проекту: PostgreSQL краще підходить для систем, де критична надійність та складна аналітика, тоді як MongoDB є оптимальним вибором для гнучких та масштабованих систем з великими обсягами неструктурованих даних.

2.5.3 Фреймворки Scrapy та BeautifulSoup

Автоматизація веб-скрапінгу за допомогою Scrapy є чудовим фреймворком, який дозволяє автоматизувати процес збору даних з веб-сайтів, забезпечуючи можливість створення складних та структурованих скраперів для ефективного отримання інформації з різних джерел. Scrapy підтримує асинхронний скрапінг, що дає змогу швидше та ефективніше обробляти великі обсяги даних, що особливо корисно при скрапінгу великих веб-сайтів або багатьох сторінок. Гнучка архітектура Scrapy дозволяє легко розширювати його функціональність та адаптувати під конкретні потреби проекту, підтримуючи різні плагіни та розширення, що дозволяє інтегрувати його з іншими інструментами та сервісами.

BeautifulSoup, предстаавляється, як простий та інтуїтивно зрозумілий інструмент для парсингу HTML та XML документів, надає також зручний інтерфейс для вибору та витягування даних з розмітки веб-сторінок, це свідчить про ідеальний вибір для початківців та швидкого прототипування. BeautifulSoup дозволяє використовувати різні методи для пошуку та вибору елементів на веб-сторінці, такі як CSS селектори, XPath та регулярні вирази, та при цьому дозволяє легко адаптувати парсер під різні вимоги та формати даних. Він також має механізми обробки некоректної або неправильно структурованої розмітки, навіть якщо вони мають недоліки або помилки. Отже Scrapy забезпечує широкий функціонал для автоматизації скрапінгу та обробки великих обсягів даних, а BeautifulSoup є зручним та легким інструментом для простого парсингу HTML та XML документів.

2.6 Алгоритми та методи аналізу контенту

Опис основних алгоритмів аналізу тексту включає в себе семантичний аналіз, сентимент-аналіз, а також використання машинного навчання та обробки природної мови Natural Language Processing [59]. Семантичний аналіз зосереджується на алгоритмах класифікації та тематичного аналізу, які дозволяють

визначати контент за категоріями. Це може включати виявлення тем, ключових слів та категорій, до яких належить текст. Семантичний аналіз допомагає зрозуміти загальну ідею тексту та його контекст, що є важливим для більш глибокого розуміння контенту веб-сторінок. Завдяки алгоритмам тематичного аналізу можна автоматично групувати тексти за змістом, що спрощує обробку великих масивів інформації, таких як новинні статті, пости у соціальних мережах чи відгуки користувачів.

Сентимент-аналіз [60] спрямований на визначення емоційного тону тексту, використовуючи алгоритми для аналізу позитивного, негативного або нейтрального настрою. Це дозволяє оцінювати настрої користувачів та їхні емоційні реакції на різні теми або продукти. Сентимент-аналіз є важливим інструментом для бізнесу, маркетингу та соціальних досліджень, оскільки він допомагає зрозуміти, як сприймаються певні події чи продукти. Наприклад, аналіз відгуків на продукти може надати компаніям цінну інформацію про те, що подобається або не подобається їхнім клієнтам, що дозволяє краще задовольняти їхні потреби.

Рисунок 2.3 – Блок-схема роботи NLP

Обробка природної мови NLP включає використання інструментів та бібліотек, таких як NLTK і spaCy, для обробки та аналізу тексту. Ці інструменти надають широкий спектр функцій для роботи з текстовими даними, включаючи токенізацію, стемінг, лематизацію, частиномовний аналіз та розпізнавання іменованих сутностей. NLP дозволяє автоматизувати обробку тексту, що значно знижує витрати часу та ресурсів на ручний аналіз (рисунок 2.3).

2.7 Висновки

У другому розділі було розглянуто проектування програмно-технічного засобу та методи реалізації системи аналізу контенту веб-сайтів на базі формальних систем. В процесі аналізу були визначені ключові компоненти системи, включаючи збір даних, попередню обробку тексту, моделювання та візуалізацію результатів.

Для забезпечення ефективного збору даних розглянуто використання методів web scraping та API, що дозволяють автоматично отримувати необхідну інформацію з веб-сайтів. Попередня обробка тексту включає токенізацію, нормалізацію, видалення стоп-слів та інші методи, що підвищують якість даних для подальшого аналізу.

3 ПРОГРАМНО-ТЕХНІЧНА РЕАЛІЗАЦІЯ ПРОГРАМИ АНАЛІЗУ КОНТЕНТУ НА ВЕБ-САЙТАХ.

3.1 Вибір мови програмування та інструментів реалізіції програми

Python обраний для цього проекту через його численні переваги, які включають простоту синтаксису, багатий набір бібліотек для різних задач та велику спільноту розробників. Ці особливості роблять Python ідеальним вибором для збору даних, обробки тексту та застосування машинного навчання. Крім того, Python підтримує інтеграцію з іншими мовами програмування та інструментами, що дозволяє створювати гнучкі та масштабовані рішення, а його висока читабельність коду сприяє легшій підтримці та розвитку проектів, а активна спільнота розробників забезпечує швидкий обмін знаннями та підтримку. Завдяки своїй універсальності, Python використовується у багатьох галузях, від веб-розробки до наукових досліджень і фінансового аналізу, що робить його надзвичайно цінним інструментом у сучасному програмуванні

Одна з ключових переваг Python полягає в його потужному наборі бібліотек, що надають готові інструменти для різноманітних задач. Наприклад, бібліотеки BeautifulSoup і Scrapy значно спрощують процес веб-скрапінгу, дозволяючи ефективно збирати дані з веб-сторінок. Для обробки тексту і аналізу природної мови використовуються бібліотеки NLTK та spaCy, які забезпечують широкі можливості для лінгвістичного аналізу та роботи з текстовими даними. Крім того, бібліотеки машинного навчання, такі як TensorFlow і Scikit-learn, дозволяють впроваджувати складні алгоритми для глибокого аналізу та прогнозування на основі зібраних даних, що робить систему аналізу веб-контенту більш точною та ефективною (рисунок 3.1)

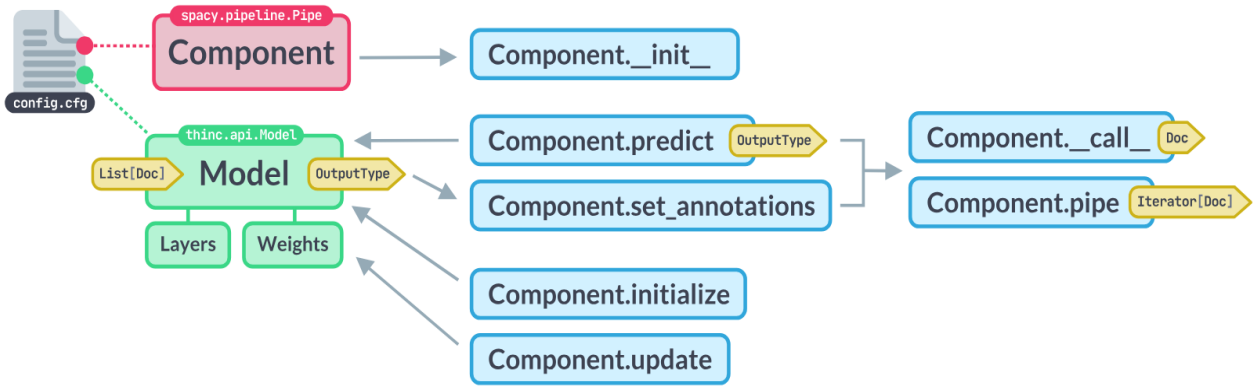


Рисунок 3.1 – Архітектура spaCy [61]

Інтеграція Python з іншими мовами програмування та інструментами є ще однією вагомою причиною для його вибору в цьому проекті. Python легко взаємодіє з базами даних, такими як MySQL, PostgreSQL або MongoDB, що дозволяє зберігати великі обсяги даних та швидко до них звертатися. Інтеграція з мовами програмування, такими як C++, Java, та інструментами для аналізу даних, наприклад, R, дозволяє використовувати Python у різноманітних середовищах та підвищує його продуктивність. Така гнучкість дозволяє створювати комплексні рішення, які можуть адаптуватися до змінних вимог проекту та легко масштабуватися відповідно до потреб.

Для написання коду використовується Visual Studio Code [61]. Цей вибір був зроблений завдяки численним перевагам, які пропонує цей редактор. Visual Studio Code є безкоштовним і відкритим програмним забезпеченням, яке підтримує багато мов програмування та має розширювану архітектуру через плагіни. Він пропонує інструменти для налагодження, вбудований термінал, інтеграцію з системами контролю версій, такими як Git, та багато функцій, які полегшують процес розробки. Завдяки інтуїтивно зрозумілому інтерфейсу та широким можливостям налаштування, Visual Studio Code забезпечує зручне та ефективне середовище для написання коду, що сприяє підвищенню продуктивності розробники.

Завдяки своїй універсальності, Python знайшов застосування у багатьох галузях, що робить його незамінним інструментом для розробки систем аналізу веб-контенту на базі формальних систем. Його використовують у веб-розробці для

створення динамічних веб-додатків, у наукових дослідженнях для обробки великих обсягів даних та проведення складних розрахунків, у фінансовому аналізі для моделювання ринкових тенденцій та оцінки ризиків. Така різноманітність застосувань підтверджує здатність Python ефективно вирішувати широкий спектр задач, забезпечуючи при цьому високу продуктивність та надійність. Це робить Python ідеальним вибором для розробки сучасних, високоефективних систем аналізу веб-контенту (рисунок 3.2).

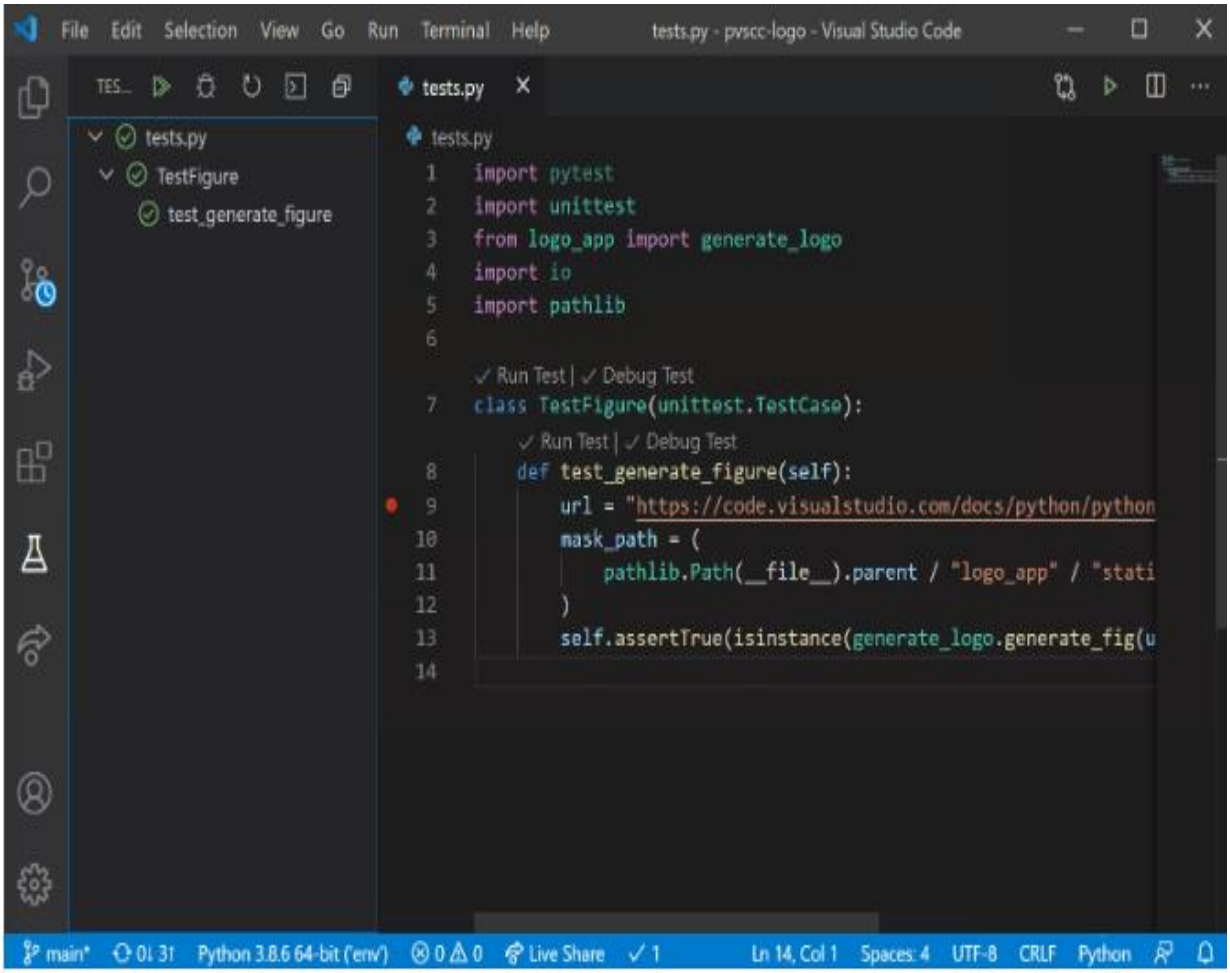


Рисунок 3.2 – Інтерфейс Visual Studio Code

Для реалізації системи аналізу контенту веб-сайту на базі формальних систем, для того щоб краще реалізувати проект, використано наступні бібліотеки Python (таблиця 3.1).

Таблиця 3.1 – Основні бібліотеки

BeautifulSoup	для парсингу HTML та XML документів, що дозволяє витягувати дані з веб-сторінок.
Requests	для здійснення HTTP-запитів, що дозволяє завантажувати веб-сторінки та інший контент.
Numpy	для роботи з числовими даними, що забезпечує ефективні обчислення.
Pandas	для маніпуляції та аналізу даних, що дозволяє обробляти та організовувати великі набори даних.
Scikit-learn	для застосування методів машинного навчання, що дозволяє створювати моделі для аналізу та прогнозування.
Natural Language Toolkit	для обробки та аналізу природної мови, що дозволяє працювати з текстовими даними.
Scrapy	для веб-скрейпінгу та вилучення даних з веб-сайтів, що дозволяє автоматизувати процес збору інформації.
spaCy	Для швидкої та ефективної обробки тексту.
TextBlob	Для простих операцій обробки тексту.

3.2 Реалізація модуля збору даних

Для реалізації збору даних використовуються інструменти: BeautifulSoup, Scrapy та Selenium. Кожен з яких має свої переваги та спеціалізацію.

BeautifulSoup обраний для парсингу статичних веб-сторінок завдяки своїй здатності легко і швидко витягати дані з HTML та XML-документів. Ця бібліотека надає простий інтерфейс для пошуку та навігації по дереву документів, що дозволяє ефективно витягувати потрібну інформацію. BeautifulSoup також автоматично виправляє некоректний HTML-код, що робить його ідеальним для роботи з реальними веб-сторінками, які можуть містити помилки.

Scrapy використаний для асинхронного веб-скрапінгу, забезпечуючи ефективний збір даних з великої кількості веб-сайтів. Цей фреймворк дозволяє керувати запитами, обробляти відповіді та зберігати зібрані дані у структурованому форматі, такому як JSON [62] або CSV [63]. Завдяки своїй асинхронній природі, Scrapy значно збільшує швидкість збору даних, що є критично важливим для масштабних проєктів. Крім того, Scrapy має вбудовані засоби для обробки складних задач, таких як авторизація та управління куками.

Selenium використовувався для автоматизації браузерів і взаємодії з динамічними веб-сторінками, що вимагають виконання JavaScript. Це дозволило збирати дані з веб-сайтів, які динамічно завантажують контент через AJAX-запити. Selenium автоматично керує браузером, імітуючи дії реального користувача, такі як натискання кнопок та заповнення форм. Поєднуючи Selenium з BeautifulSoup або іншими інструментами для парсингу, можна збирати дані з найскладніших веб-сторінок, які недоступні за допомогою простих HTTP-запитів.

Використання Scrapy для завантаження веб-сторінок. Web scraper — це програмний інструмент, який автоматично отримує дані з веб-сайтів (рисунок 3.2). Цей процес може бути виконаний автоматично без необхідності вручну відвідувати кожний веб-сайт та складати дані вручну.

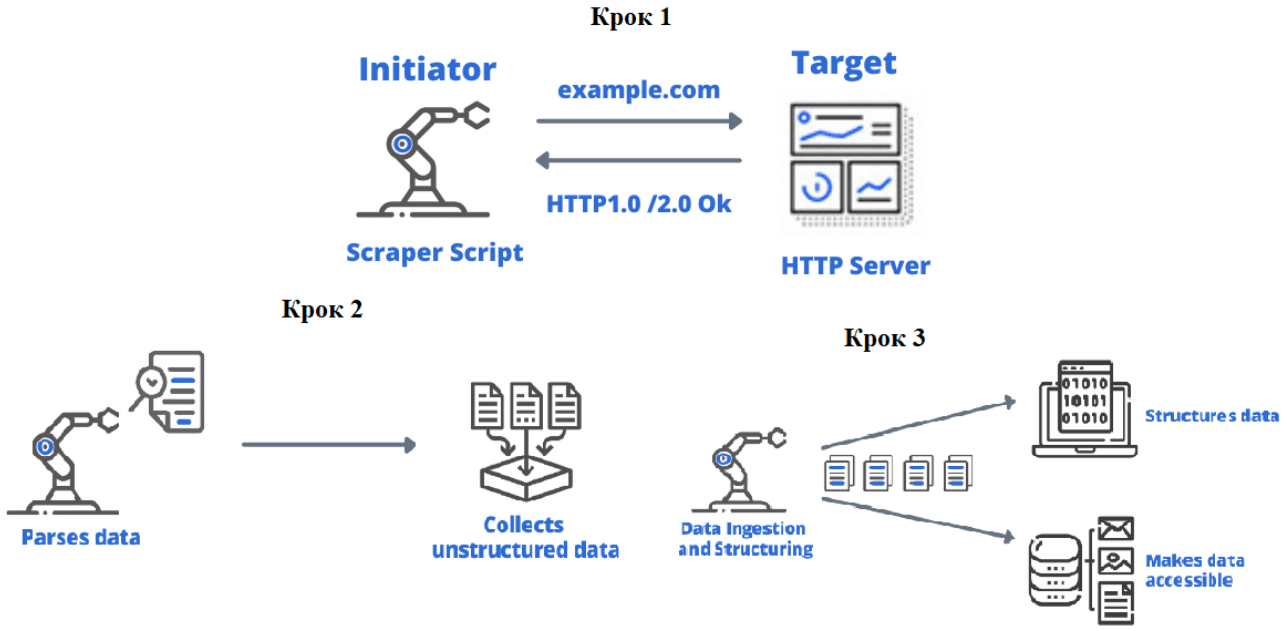


Рисунок 3.3 – Структура роботи Web scraper

Крок 1. Ініціатор використовує інструмент для скрейпінгу — програмне забезпечення хмарний сервіс або власний скрипт, щоб генерувати HTTP-запити для взаємодії з веб-сайтами та отримання даних. Це ПЗ може надсилати будь-які HTTP-запити: GET, POST, PUT, DELETE, HEAD або OPTIONS до цільового веб-сайту.

Крок 2. Якщо сторінка доступна, цільовий веб-сайт відповість на запит скрейпера статусом HTTP/1.0 200 OK. Після отримання HTML-відповіді наприклад, 200 OK скрейпер починає аналізувати документ та збирати неструктуровані дані.

Крок 3. Далі програма-скрейпер витягує сирі дані, зберігає їх та додає структуру, індекси відповідно до вказівок ініціатора. Структуровані дані доступні у читабельних форматах, таких як XLS, CSV, SQL.

3.3 Обробка та аналіз тексту

Процеси обробки та аналізу тексту забезпечують можливість перетворення неструктурованих текстових даних у структуровані та зрозумілі форми, що дозволяє отримати цінні інсайти та знання з великого обсягу інформації.

Використання сучасних технологій і інструментів, що дозволяє досягти високої точності і ефективності в обробці тексту. Для цього застосовується кілька інструментів, а саме NLTK, spaCy та TextBlob [64]. Кожен з цих інструментів має свої унікальні особливості та підходи до обробки природної мови.

Natural Language Toolkit один з найпопулярніших інструментів для роботи з текстовими даними. Він містить широкий набір функцій для обробки природної мови, що робить його надзвичайно корисним для різноманітних завдань. NLT надає наступні можливості у обробці та аналізу тексту (рисунок 3.4).

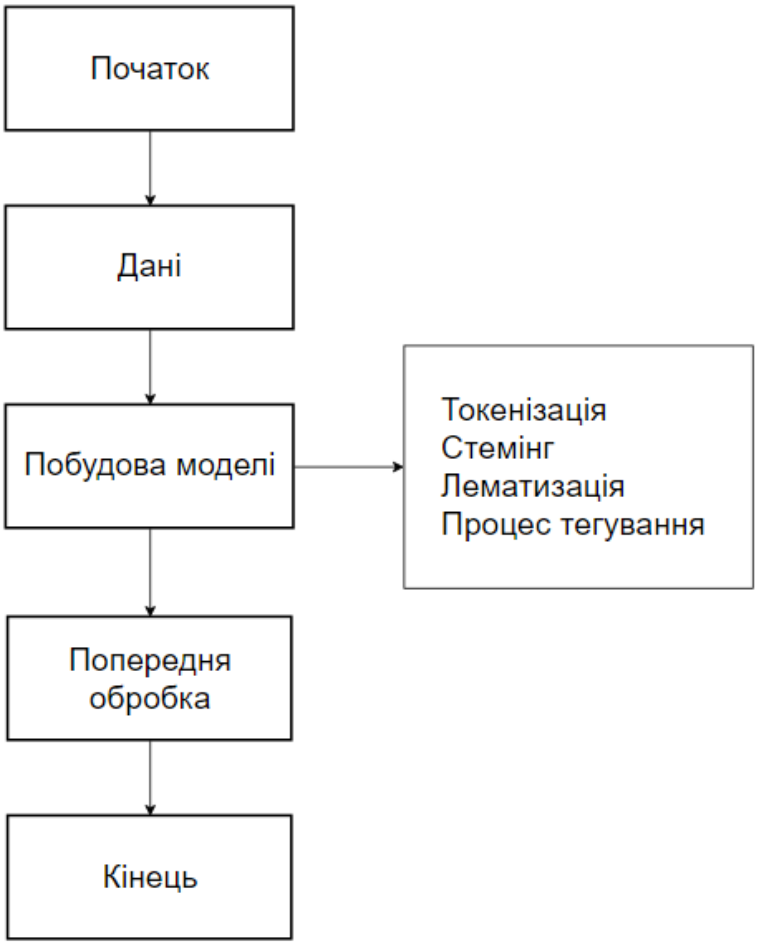


Рисунок 3.4 – Схема процесу NLT

Токенізація включає розбиття тексту на менші компоненти, такі як слова або речення і є базовим кроком у будь-якій обробці тексту, оскільки дозволяє працювати з текстом на більш деталізованому рівні.

Стемінг зводить слова до їх основної форми або кореня, що особливо корисно для нормалізації тексту, оскільки дозволяє об'єднувати різні форми одного і того ж слова.

Лематизація приводить слова до їх початкової форми, враховуючи контекст це свідчить про змогу досягти більшої точності порівняно зі стемінгом, особливо при обробці складних текстів.

Процес тегування частин мови визначає граматичну категорію кожного слова в тексті таких як: іменник, дієслово, прикметник. Це свідчить про важливість розуміння структури речення та його змісту.

Використання spaCy, як сучасна бібліотека для обробки природної мови, яка відзначається високою продуктивністю та підтримкою розширених функцій NLP. розроблена для швидкої і ефективної роботи з великими обсягами тексту і включає в себе всі необхідні інструменти для обробки природної мови.

Головні переваги spaCy мають високу продуктивність, готову до використання модель системи навчання на виробництві, також підтримку попередньо навчених векторів та вбудованих слів, вже навчені моделі доступні для різних мов та програм.

Підтримка розширення функціональності за допомогою спеціальних компонентів та атрибутів містить високу точність, простий процес упаковки та розгортання моделей та управління робочим процесом.

Основні можливості spaCy мають розпізнавання іменованих сутностей NER, що виділяє і класифікує назви власні імена, організації, місця, тому це дозволяє автоматично витягувати важливі імена з тексту, що може бути корисним для аналізу новин, соціальних медіа та джерел. Визначення граматичні категорії кожного слова частини мови за допомогою POS tagging. Це допомагає краще розуміти структуру речення і контекст, в якому вживається слово.

Процес лематизація приводить слова до їх початкової форми, і дозволяє уникнути проблем з різними формами одного і того ж слова. Визначення граматичних зв'язки між словами в реченні, є важливим для глибшого розуміння

тексту, за допомогою аналізу залежностей. Фрагмент використання наведено нижче:

```
import spacy
nlp = spacy.load("en_core_web_sm")
text = "Apple is looking at buying U.K. startup for $1 billion."
doc = nlp(text)
entities = [(ent.text, ent.label_) for ent in doc.ents]
tokens = [(token.text, token.lemma_, token.pos_) for token in doc]
print("Entities:", entities)
print("Tokens:", tokens)
```

TextBlob – це бібліотека, яка спрощує виконання різноманітних операцій з текстом, забезпечуючи зручний інтерфейс для аналізу настроїв, перекладу тексту. TextBlob відомий своєю простотою у використанні та інтуїтивно зрозумілим інтерфейсом. Основні можливості TextBlob:

Аналіз настроїв, що визначається полярністю, а саме позитивна, негативна або нейтральна також суб'єктивність, а саме об'єктивна чи суб'єктивна. Це особливо корисно для аналізу відгуків, коментарів у інших текстів, де важливо розуміти емоційний фон.

Дозволяє перекладати текст між різними мовами, що є корисним для аналізу контенту на багатомовних веб-сайтах і забезпечує простий інтерфейс для роботи з текстом а також легке виконання основних операцій з текстом, таких як токенізація, лематизація та визначення частини мови (рисунок 3.4). Фрагмент реалізації TextBlob наведенно нижче:

```
from textblob import TextBlob
text = "I like programming in Python. It is very exciting and fulfilling."
blob = TextBlob(text)
sentiment = blob.sentiment
```

```
translated_blob = blob.translate(to='es')
print("Sentiment:", sentiment)
print("Translated Text:", translated_blob)
```

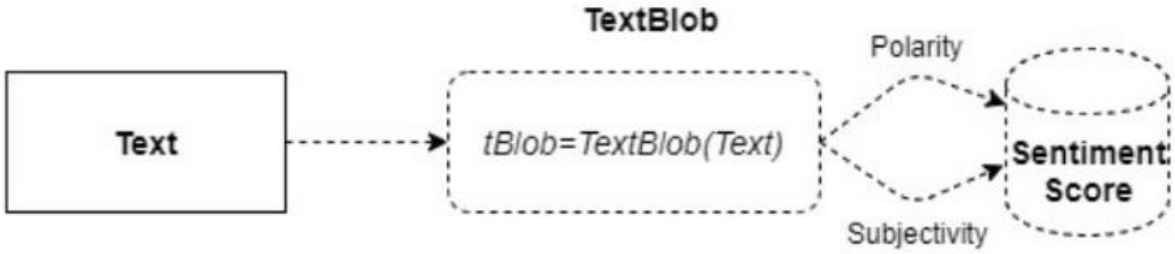


Рисунок 3.5 – Схема TextBlob

3.4 Застосування машинного навчання системи

Щоб реалізувати данну задачу потрібно використовувати машинне навчання для аналізу контенту на веб-сайтах. Для цього потрибно скористуватися популярними бібліотекою, а саме: scikit-learn [65].

Бібліотека Scikit-learn є бібліотекою машинного навчання, написана на Python, надаю широкий вибір алгоритмів навчання і таким чином, одна з основних переваг цієї бібліотеки полягає в тому, що вона працює на основі декількох поширених математичних бібліотек і легко інтегрує їх одна з одною. Ще однією важливою перевагою є широка спільнота користувачів та докладна документація, що дозволяє швидко знаходити відповіді на питання. Це свідчить, що scikit-learn є дуже корисним інструментом для вирішення задач навчання з учителем, таких як класифікація та регресія, а також для завдань навчання без учителя, таких як кластеризація, зменшення розмірності та детектування аномалій.

Одразу до прикладу. Щоб побудувати модель «мішка слів» на основі частоти слів у відповідних оголошеннях, можна скористатися класом векторизації кількостей CountVectorizer, реалізованим у бібліотеці scikit-learn. Фрагмент реалізації наведено нижче:

```
    from sklearn.feature_extraction.text import
CountVectorizer
    from sklearn.model_selection import train_test_split
    from sklearn.naive_bayes import MultinomialNB
    from sklearn.pipeline import make_pipeline
    from sklearn import metrics
    texts = ["This is a positive example.", "This is a
negative example.", "Another positive example."]
    labels = [1, 0, 1]  # 1 - позитивний, 0 - негативний
    X_train, X_test, y_train, y_test =
train_test_split(texts, labels, test_size=0.2,
random_state=42)
    model = make_pipeline(CountVectorizer(),
MultinomialNB())
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = metrics.accuracy_score(y_test, y_pred)
    print(f"Accuracy: {accuracy:.2f}")
```

3.5 Інтеграція компонентів системи та взаємодія між модулями

Інтеграція компонентів системи є важливим аспектом розробки програми аналізу контенту на веб-сайтах. Вона передбачає створення ефективної взаємодії між різними модулями системи та забезпечення можливості взаємодії з зовнішніми сервісами через API. Для успішної інтеграції компонентів системи необхідно забезпечити правильну взаємодію між різними модулями. Кожен модуль виконує свою функцію і передає результати своєї роботи іншим модулям.

Модуль збору даних відповідає за збирання контенту з веб-сайтів. Він використовує веб-краулери та парсери для отримання необхідної інформації.

Модуль попередньої обробки даних відповідає за очищення та підготовку зібраних даних для подальшого аналізу. Це може включати видалення HTML-тегів, нормалізацію тексту, та видалення стоп-слів.

Модуль аналізу даних використовує алгоритми машинного навчання та обробки природної мови для аналізу підготовлених даних. Модуль може включати підмодулі для класифікації чи кластеризації.

Модуль візуалізації результатів відповідає за представлення результатів аналізу в зручному для користувача форматі, наприклад, через графіки, або діаграми (рисунок 3.6).

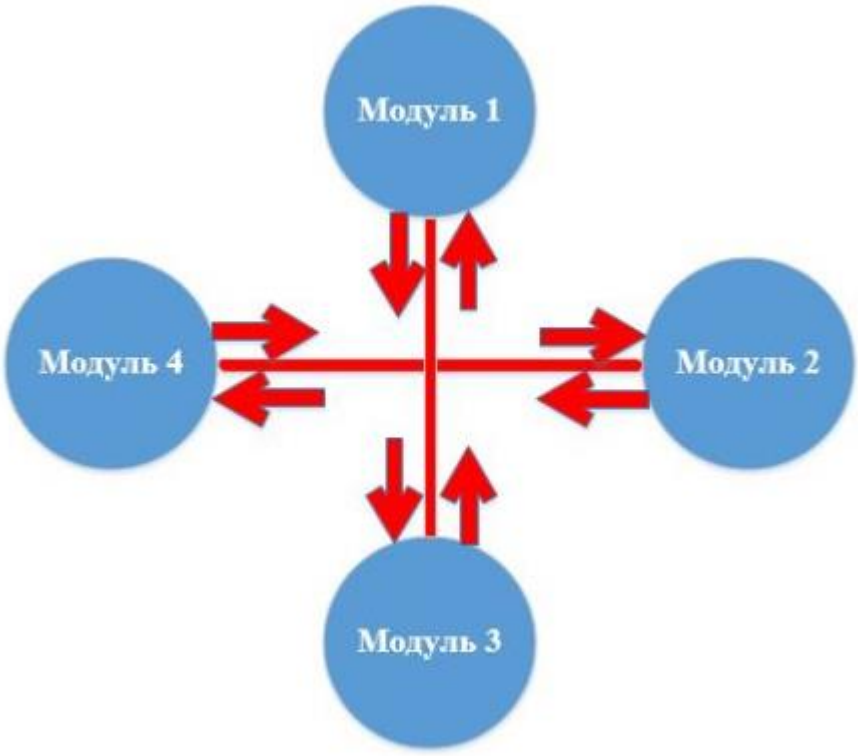


Рисунок 3.6 – Взаємодія між модулями

Для забезпечення можливості взаємодії з зовнішніми сервісами та програмами необхідно розробити API, яке дозволить здійснювати зовнішні запити до системи аналізу контенту. Таким чином, API стає інтерфейсом між системою та зовнішніми користувачами або іншими системами, надаючи доступ до різних функціональних можливостей. Потрібно забезпечити можливість отримання

контенту з веб-сайтів через зовнішні запити. Наприклад, POST-запити для додавання нових URL для аналізу. Отже користувачі зможуть надсилати запити для додавання нових джерел даних, які система має аналізувати. Фрагмент реалізації нижче:

```
@app.route('/collect', methods=['POST'])
def collect():
    url = request.json.get('url')
    if not url:
        return jsonify({'error': 'URL is required'}),
400
    data = collect_data(url)
    return jsonify({'data': data})
```

Отже інтеграція компонентів системи та створення API для зовнішніх запитів є критичними елементами для забезпечення ефективного функціонування програми аналізу контенту на веб-сайтах, адже правильна організація взаємодії між модулями та зовнішніми системами забезпечує надійність та гнучкість у роботі всієї системи.

Блок-схема ілюструє, як різні компоненти системи взаємодіють один з одним через API, забезпечуючи процес збору, обробки, аналізу та управління даними. Користувачі можуть надсилати запити через API для додавання нових URL, отримання результатів аналізу або управління системою, а кожен модуль виконує свою специфічну функцію в цьому процесі. Основні етапи включають збір даних, попередню обробку, аналіз, та управління системою.

Таким чином, маршрут дозволяє надсилати нові URL для збору даних, Система отримує URL, збирає контент і повертає зібрані дані у форматі JSON. Надавати доступ до результатів аналізу через GET-запити, для отримання результатів аналізу. Це включає отримання тематичних кластерів, результатів класифікації, виявлених аномалій (рисунок 3.7).

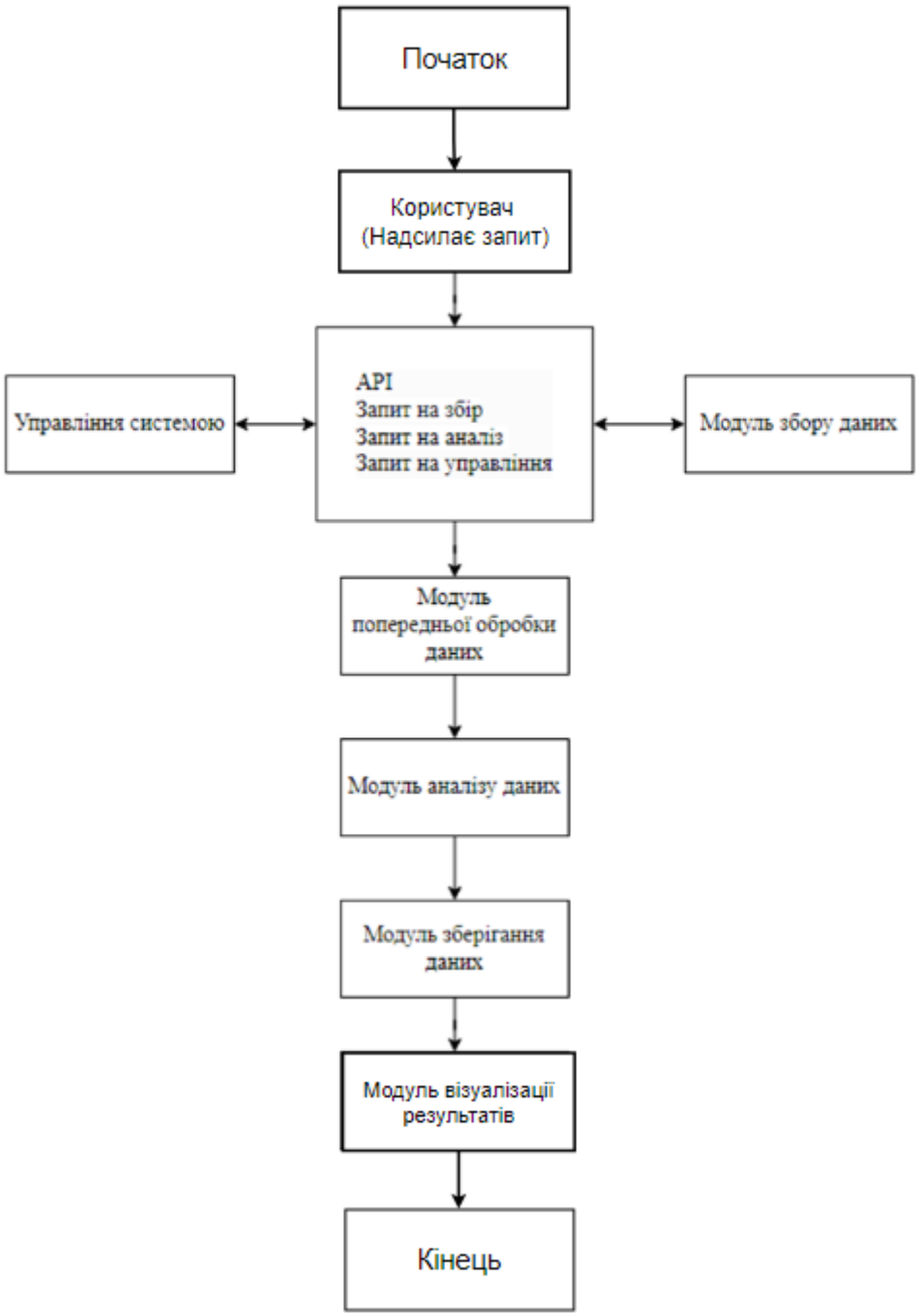


Рисунок 3.7 – Схема взаємодії компонентів через API

Це свідчить, що користувачі можуть отримувати результати роботи системи в реальному часі або за запитом. Таким чином, користувачі можуть надсилати текстові дані для аналізу і отримувати результати роботи алгоритмів машинного навчання. Щоб забезпечити можливість адміністрування системи через API.

Наприклад, контроль за станом модулів, чи перегляд логів, або управління чергами повідомлень.

3.6 Безпека системи

Забезпечення безпеки системи аналізу контенту веб-сайтів є надзвичайно важливим для захисту даних та підтримки надійності системи. Безпека є основним пріоритетом, оскільки система обробляє конфіденційну інформацію і піддається різноманітним кіберзагрозам. Захист даних користувачів та результатів аналізу від несанкціонованого доступу є критично важливим для збереження довіри клієнтів та запобігання фінансовим і репутаційним втратам. Таким чином, розробка ефективної стратегії безпеки є першочерговим завданням на всіх етапах створення та впровадження системи.

Безпека повинна бути інтегрованою у всі аспекти розробки та експлуатації системи, починаючи від початкового проектування до постійного моніторингу та оновлення. Це означає, що розробники повинні використовувати безпечні методології кодування, регулярно проводити аудит безпеки та тестування на вразливості. Підхід, заснований на "безпеці за проектом" security by design, і передбачає включення заходів безпеки на кожному етапі життєвого циклу розробки програмного забезпечення SDLC. Такий підхід допомагає виявляти та усувати потенційні загрози ще до того, як система буде розгорнута у виробничому середовищі.

Методи забезпечення безпеки даних, які включають шифрування як під час зберігання, так і під час передачі, а також впровадження надійних механізмів аутентифікації та авторизації. Шифрування даних допомагає захистити їх від перехоплення та несанкціонованого доступу, а аутентифікація та авторизація забезпечують, що тільки довірені користувачі мають доступ до системи та її ресурсів. Це свідчить, що комплексний підхід до безпеки даних є важливим для мінімізації ризиків та забезпечення цілісності та конфіденційності інформації.

Крім захисту даних, важливо також враховувати захист системи від зловмисних атак, таких як DDoS атаки [66], SQL ін'єкції та XSS [67]. Використання брандмауерів та систем виявлення і запобігання вторгненням IDS/IPS може значно підвищити рівень безпеки. Впровадження HTTPS забезпечує шифрування всього трафіку між клієнтом та сервером, що запобігає перехопленню та зміні даних під час передачі, Тому це свідчить про важливість використання багаторівневого підходу до безпеки, який включає як технічні засоби захисту, так і організаційні заходи для створення безпечного та надійного середовища для аналізу контенту веб-сайтів.

Шифрування даних у сховищі Data at Rest. Шифрування даних, коли вони зберігаються в базах даних або на дисках, забезпечує захист від несанкціонованого доступу до даних.

Приклад використання бібліотек шифрування, таких як «cryptography» у Python для шифрування конфіденційних даних перед збереженням у базі даних. Шифрування даних під час передачі Data in Transit виконується за допомогою Transport Layer Security [68] для шифрування даних, що передаються через мережу, тобто використовується шифрування для захисту даних, що передаються між клієнтом та сервером, або між різними модулями системи. Фрагмент реалізації наведено нижче:

```
from cryptography.fernet import Fernet
key = Fernet.generate_key()
cipher_suite = Fernet(key)
encrypted_data = cipher_suite.encrypt(b"My secret data")
decrypted_data = cipher_suite.decrypt(encrypted_data)
```

Шифрування даних під час передачі Data in Transit виконується за допомогою Transport Layer Security [68] для шифрування даних, що передаються через мережу, тобто використовується шифрування для захисту даних, що передаються між клієнтом та сервером, або між різними модулями системи.

Цей процес шифрування застосовується до даних, що передаються між клієнтом та сервером, або між різними модулями системи. Наприклад, при передачі конфіденційної інформації з клієнта на сервер або при взаємодії різних компонентів системи через мережу, дані будуть шифровані за допомогою TLS. Це дозволяє запобігти можливому перехопленню або зміні даних зловмисниками (рисунок 3.8).

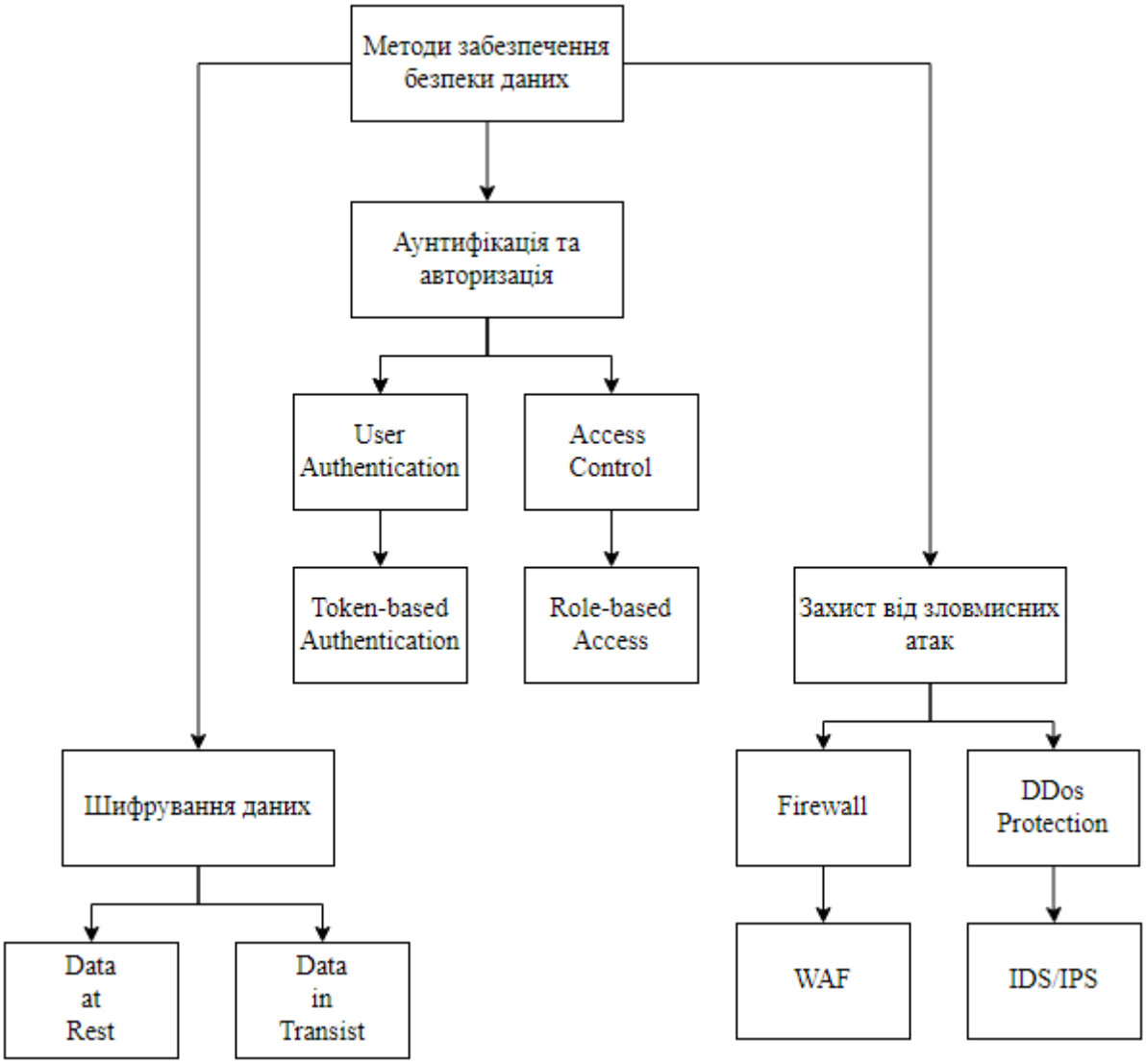


Рисунок 3.8 – Схема безпека даних

У контексті блок-схеми безпеки системи, цей елемент ілюструється як один з рівнів захисту, який взаємодіє з іншими методами безпеки, такими як шифрування

даних у спокої «Data at Rest», аутентифікація, контроль доступу і захист від зловмисних атак. Разом вони утворюють комплексну систему захисту, яка забезпечує безпеку даних та надійність системи на різних рівнях.

3.7 Розгортання та масштабування

Ключові аспекти розгортання та масштабування системи аналізу контенту на веб-сайтах. Забезпечення ефективного та надійного процесу розгортання, а також можливість масштабування системи відповідно до потреб, є важливими елементами для підтримки високої продуктивності та доступності (рисунок 3.9).

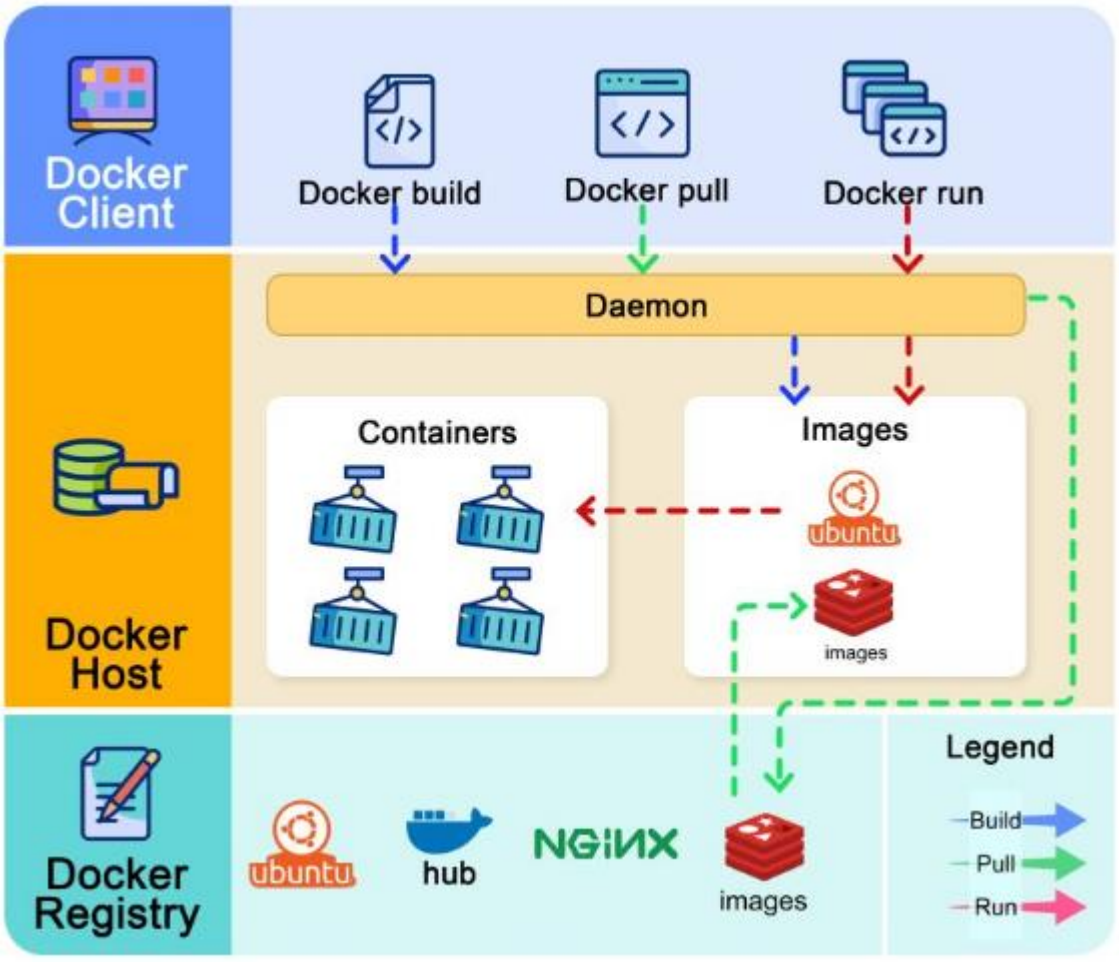


Рисунок 3.9 – Архітектура роботи Docker

Методи розгортання використання контейнерів Docker забезпечують ізольоване середовище для запуску додатків, що дозволяє уникнути проблем сумісності та спростити процес розгортання. Docker є однією з найпопулярніших платформ для роботи з контейнерами. Контейнери дозволяють розміщувати додатки та їх залежності в ізольованих середовищах, що забезпечує однакові умови для роботи додатків на різних платформах. Використання Docker дозволяє значно спростити процес розгортання, оскільки усі необхідні компоненти та залежності упаковані разом із додатком у контейнері. Це забезпечує портативність додатків, оскільки контейнери можуть бути запущені на будь-якій машині, що підтримує Docker [69]. Крім того, контейнери полегшують масштабування додатків та автоматизацію процесів розгортання, дозволяючи швидко та легко додавати нові екземпляри додатків або оновлювати існуючі.

Переваги використання Docker включають ізоляцію середовища, що запобігає конфліктам між залежностями різних компонентів системи, портативність, яка дозволяє легко переносити додатки між різними середовищами, та легкість управління, що дозволяє легко масштабувати додатки та автоматизувати процеси розгортання. Створення Dockerfile для кожного компонента системи та використання Docker Compose для організації та запуску контейнерів дозволяють ефективно управляти додатками та забезпечують їх надійність та стабільність.

Автоматизація розгортання Continuous Integration та Continuous Deployment забезпечують процес розгортання, що дозволяє швидко впроваджувати зміни та зменшує ризик помилок. CI/CD процеси включають автоматичне тестування кожного коміту в репозиторій, а також автоматичне розгортання успішно протестованих версій на сервери розробки та продуктивного середовища. Використання таких інструментів, як Jenkins, CircleCI та Travis CI, дозволяє автоматизувати процеси інтеграції та розгортання, що підвищує ефективність розробки та забезпечує високу якість коду.

Переваги CI/CD містять автоматизацію тестування та розгортання, що забезпечує автоматичне виконання тестів та розгортання нових версій додатків після внесення змін до коду, зменшення часу на розгортання завдяки автоматизованим процесам, та підвищення якості коду завдяки регулярному тестуванню та інтеграції змін. Впровадження CI/CD процесів дозволяє забезпечити постійний моніторинг якості коду та швидке реагування на виявлені помилки, що зменшує час на виправлення помилок та підвищує загальну стабільність системи.

Масштабування системи дозволяє підвищувати її продуктивність та доступність у разі збільшення навантаження. Горизонтальне масштабування передбачає додавання нових екземплярів додатків для розподілу навантаження, що забезпечує обробку більшої кількості запитів одночасно. Використання балансувальників навантаження дозволяє рівномірно розподіляти запити між екземплярами, що забезпечує високу доступність та надійність системи. Вертикальне масштабування передбачає збільшення ресурсів (CPU, RAM) на одному сервері для підвищення його продуктивності. Цей підхід підходить для додатків, що не можуть бути легко розділені на декілька екземплярів, та забезпечує підвищення продуктивності без необхідності додавання нових серверів.

Горизонтальне масштабування дозволяє збільшувати продуктивність системи шляхом додавання нових серверів з копіями додатку, що дозволяє обробляти більше запитів одночасно. Вертикальне масштабування забезпечує підвищення продуктивності існуючих серверів шляхом збільшення обсягів оперативної пам'яті та процесорних ресурсів, що дозволяє обробляти більш складні задачі швидше.

Використання оркестрації Kubernetes [70] забезпечує автоматизоване управління контейнерами, що дозволяє ефективно масштабувати та управляти додатками в контейнерах. Переваги Kubernetes включають автоматичне масштабування, самовідновлення та балансування навантаження. Kubernetes може автоматично додавати або видаляти екземпляри додатків на основі навантаження, що забезпечує гнучке масштабування системи відповідно до потреб.

Самовідновлення дозволяє Kubernetes автоматично перезапускати збоєві контейнери, замінювати або масштабує екземпляри додатків при збоях. Балансування навантаження забезпечує рівномірний розподіл запитів між контейнерами, забезпечує стабільну роботу системи.

Використання Kubernetes для оркестрації контейнерів дозволяє автоматизувати процеси управління додатками, що включає автоматичне масштабування, самовідновлення та балансування навантаження.

Це забезпечує високу гнучкість та надійність системи, дозволяючи швидко реагувати на зміни в навантаженні та забезпечуючи безперебійну роботу додатків. (рисунок 3.10).

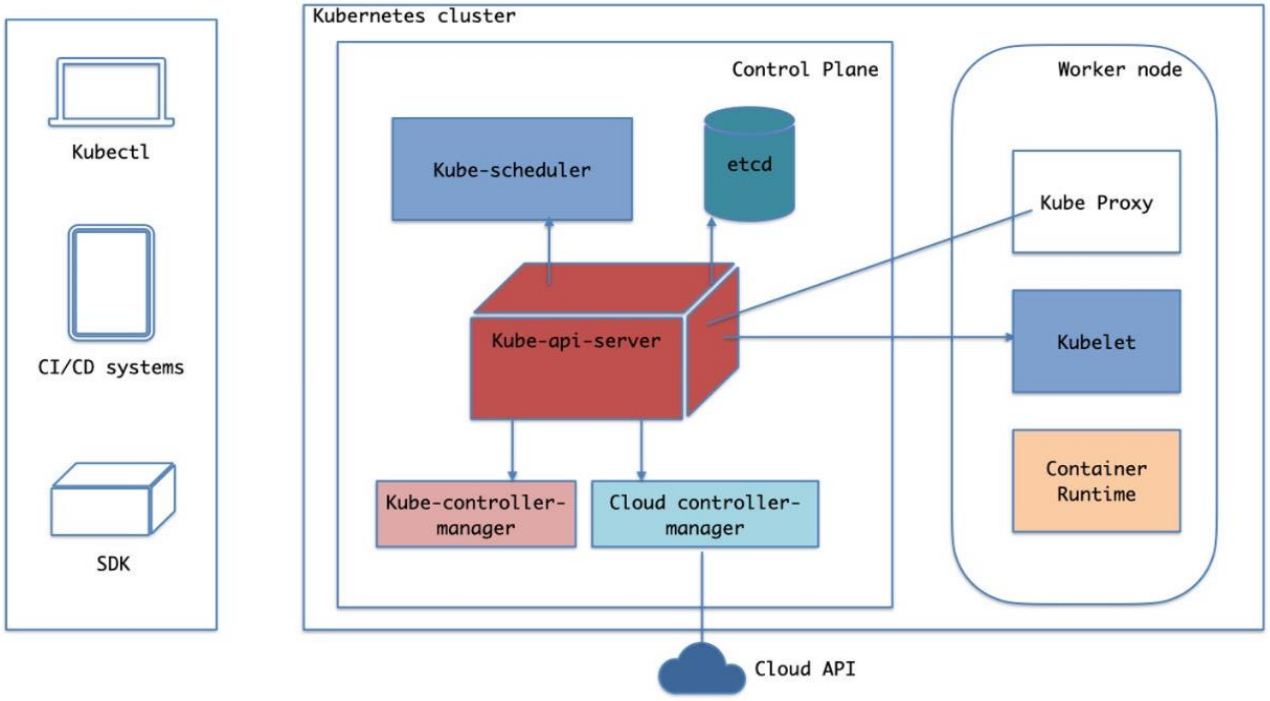


Рисунок 3.10 – Архітектура Kubernetes

Створення та налаштування Kubernetes кластеру, а також використання файлів конфігурації для визначення деплойментів, сервісів та інгресів, дозволяє ефективно управляти додатками та забезпечувати їх надійність та стабільність.

3.8 Висновки

У розділі програмно-технічної реалізації програми аналізу контенту на веб-сайтах розглянуто основні етапи створення та впровадження системи.

На першому етапі здійснюється реалізація модуля збору даних. Модуль відповідає за збирання інформації з веб-сайтів, використовуючи методи веб-скрапінгу та API. Важливим аспектом є забезпечення достовірності та повноти зібраних даних, а також обробка можливих помилок під час збору.

Далі йде обробка та аналіз тексту. Зібрані дані потребують попередньої обробки, яка включає очищення тексту від зайвих символів, нормалізацію, та токенізацію. Цей етап важливий для підвищення точності подальшого аналізу та забезпечення коректної роботи алгоритмів.

Наступний крок передбачає застосування машинного навчання. Використовуючи попередньо оброблені дані, система застосовує алгоритми машинного навчання для аналізу контенту, виявлення шаблонів та трендів. Важливим аспектом є вибір відповідних моделей та їх навчання на якісних даних для досягнення високої точності результатів.

В інтеграції компонентів системи розглядається взаємодія між модулями. Необхідно забезпечити безперебійну роботу всієї системи, щоб модулі збору даних, обробки тексту та машинного навчання ефективно взаємодіяли між собою. Інтеграція включає налагодження процесів передачі даних між модулями.

Питання безпеки системи є важливим етапом, який охоплює захист даних, та конфіденційності інформації. Включає впровадження заходів захисту від кібератак, забезпечення надійності збереження даних.

Останнім етапом є розгортання та масштабування системи. Після завершення розробки та тестування системи здійснюється її впровадження в робочу середу. Важливо забезпечити можливість масштабування системи для обробки збільшеної кількості даних та розширення функціональності відповідно до зростаючих вимог користувачів.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень було створено комплексну систему аналізу контенту веб-сайтів на базі формальних систем. Проведено детальне вивчення сучасних методів та технологій, які використовуються для обробки та аналізу текстових даних, що дозволило розробити ефективний програмно-технічний засіб для автоматизованого збору, аналізу та візуалізації інформації.

У першому розділі проведено аналіз існуючих систем аналізу контенту веб-сайтів. Вивчено основні підходи та методи, що використовуються в цій галузі, зокрема, обробку натуральної мови (NLP), машинне навчання (ML) та різні алгоритми аналізу тексту, такі як семантичний та синтаксичний аналіз. Було також розглянуто застосування формальних систем, що дозволяють підвищити точність та ефективність аналізу. Огляд показав, що сучасні системи здатні автоматизувати процеси класифікації тексту, визначення настроїв та виявлення ключових тем у великих обсягах даних.

У другому розділі проведено детальний аналіз проектування програмно-технічного засобу для системи аналізу контенту. Визначено ключові етапи розробки, включаючи збір даних, попередню обробку тексту, моделювання та візуалізацію результатів. Розглянуто методи web scraping та використання API для автоматичного отримання даних з веб-сайтів. Описані процеси попередньої обробки тексту, такі як токенізація, лематизація та видалення стоп-слів, що забезпечують якісну підготовку даних. Основний акцент зроблено на використанні машинного навчання та формальних систем для класифікації, кластеризації та аналізу тексту, що дозволило створити надійну та ефективну систему.

У третьому розділі проведено детальну програмно-технічну реалізацію системи аналізу контенту веб-сайтів. Реалізовано модулі збору даних за допомогою бібліотек BeautifulSoup та Scrapy, а також інтеграцію з API для отримання структурованих даних. Попередня обробка тексту виконувалася з використанням

бібліотек NLP, таких як NLTK та spaCy. Моделювання та аналіз здійснювалися за допомогою моделей машинного навчання та глибокого навчання, зокрема, методів SVM, CNN та RNN. Візуалізація результатів забезпечувалася інструментами Plotly та Dash, що надали користувачам зручний доступ до аналітичних даних. Завершальним етапом була інтеграція всіх компонентів системи та її тестування на різних наборах даних.

Отже, у цій роботі було розроблено та реалізовано ефективну систему аналізу контенту веб-сайтів на базі формальних систем, що здатна автоматично збирати, обробляти та аналізувати великі обсяги текстових даних. Система надає користувачам цінні інсайти та підтримку у прийнятті рішень, демонструючи високу точність та продуктивність у виконанні поставлених завдань.

Подальший розвиток системи може включати кілька напрямків. Додавання нових модулів для аналізу мультимедійного контенту (зображень, відео) та інтеграція з іншими джерелами даних, такими як соціальні мережі та новинні портали.

Розширення можливостей системи для аналізу контенту на різних мовах, включаючи автоматичний переклад та врахування культурних особливостей при аналізі настроїв та тематичному моделюванні. Створення більш інтерактивних та користувацько-орієнтованих інтерфейсів, які дозволять користувачам налаштовувати параметри аналізу відповідно до їхніх потреб та отримувати більш персоналізовані результати.

Реалізація цих напрямків дозволить підвищити цінність та ефективність системи, розширюючи її застосування у різних сферах та забезпечуючи користувачам ще більш глибокий та точний аналіз контенту веб-сайтів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Що таке аналіз контенту веб-сайтів? – URL: https://data-flair.training/blogs/djangoadvantages-and-disadvantages/ (дата звернення 06.01.2024).

2. Формальна система — ВУ. URL: https://vue.gov.ua/Формальна_система (дата звернення 06.01.2024).

3. HTML Підручник. Початок. Уроки для початківців. W3Schools українською URL: https://w3schoolsua.github.io/html/index.html#gsc.tab=0 (дата звернення 02.01.2024).

4. Що таке ППІ (API)? — служба підтримки Apple (UA). URL: https://support.apple.com/uk-ua/guide/shortcuts-mac/apd2e30c9d45/mac (дата звернення 03.01.2024).

5. General Architecture for Text Engineering (GATE) Developer. URL: https://www.oit.va.gov/Services/TRM/ToolPage.aspx?tid=6963 (дата звернення 05.01.2024).

6. Що таке токенізація? Токенізація — це процес заміни… | by VDS Group | Medium/ URL: https://medium.com/@vds-group/що-таке-токенізація-146a70e86153 (дата звернення 07.01.2024).

7. Що таке лематизація в тексті | Секрети від копірайтера | Fabrika Slov. URL: https://fabrika-slov.com/uk/chto-takoe-lemmatizacziya/ дата звернення 09.01.2024).

8. Іменовані сутності. URL: https://studfile.net/preview/3271234/page:59/ (дата звернення 10.01.2024).

9. Види машинного перекладу. URL: https://profpereklad.ua/vidi-mashinnogo-perekladu/ (дата звернення 08.01.2024).

10. What Is a Chatbot? | IBM. URL: https://www.ibm.com/topics/chatbots (дата звернення 10.01.2024).

11. Hemingway Editor. URL: https://hemingwayapp.com (дата звернення 09.01.2024).

12. Overview – CoreNLP. URL: https://stanfordnlp.github.io/CoreNLP/ (дата звернення 09.01.2024).

13. A Brief Introduction to BERT | MKAI. URL: https://mkai.org/a-brief-introduction-to-bert/ (дата звернення 10.01.2024).

14. What is Machine Learning and How Does It Work? In-Depth Guide. URL: https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML (дата звернення 10.01.2024).

15. Що таке нейронна мережа та як вона працює. URL: https://mc.today/uk/shho-take-nejronna-merezha/ (дата звернення 11.01.2024).

16. Step by step: Програмне забезпечення. URL: https://step.org.ua/konspekt/pz/tema1 (дата звернення 15.01.2024).

17. A Web Scraping Project using Scrapy | by Tanja Adžić | Medium. URL: https://adzic-tanja.medium.com/a-web-scraping-project-with-scrapy-bba1e2037c4d (дата звернення 14.01.2024).

18. Beautiful Soup Tutorial - How to Parse Web Data With Python. URL: https://oxylabs.io/blog/beautiful-soup-parsing-tutorial (дата звернення 17.01.2024).

19. Scrapy | A Fast and Powerful Scraping and Web Crawling Framework. URL: https://scrapy.org (дата звернення 17.01.2024).

20. Puppeteer | Puppeteer URL: https://pptr.dev (дата звернення 17.01.2024).

21. Selenium URL: https://www.selenium.dev (дата звернення 20.01.2024).

22. XPath Tutorial. URL: https://www.w3schools.com/xml/xpath_intro.asp (дата звернення 21.01.2024).

23. CSS-селекторы| MDN. URL: https://developer.mozilla.org/en/docs/Web/CSS/CSS_selectors (дата звернення 20.01.2024).

24. Про регулярні вирази - Analytics Довідка. URL: https://support.google.com/analytics/answer/1034324?hl=uk (дата звернення 21.01.2024).

25. Що таке Unix? Огляд UNIX-подібних систем | Блог HyperHost.UA. URL: https://hyperhost.ua/info/uk/shcho-take-unix-oglyad-unix-podibnikh-sistem (дата звернення 22.01.2024).

26. How to create an automated task using Task Scheduler on Windows 10 | Windows Central URL: https://www.windowscentral.com/how-create-automated-task-using-task-scheduler-windows-10 (дата звернення 23.01.2024).

27. Apache Airflow. URL: https://airflow.apache.org (дата звернення 20.01.2024).

28. JavaScript language overview - JavaScript | MDN. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language_overview дата звернення 12.02.2024).

29. Node.js — The V8 JavaScript Engine. URL: https://nodejs.org/en/learn/getting-started/the-v8-javascript-engine (дата звернення 10.02.2024).

30. AJAX Introduction. URL: https://www.w3schools.com/xml/ajax_intro.asp (дата звернення 11.02.2024).

31. lxml - Processing XML and HTML with Python URL: https://lxml.de (дата звернення 12.02.2024).

32. What is metadata and why is it as important as the data itself?. URL: https://www.opendatasoft.com/en/blog/what-is-metadata-and-why-is-it-important-data/ (дата звернення 13.02.2024).

33. NLTK :: Natural Language Toolkit. URL: https://www.nltk.org (дата звернення 14.02.2024).

34. spaCy·Industrial-strength Natural Language Processing in Python. URL: https://spacy.io (дата звернення 15.02.2024).

35. Porter Stemming Algorithm. URL: https://tartarus.org/martin/PorterStemmer/ (дата звернення 15.02.2024).

36. Understanding Snowball Stemmer in NLP URL: https://www.tutorialspoint.com/understanding-snowball-stemmer-in-nlp (дата звернення 17.02.2024).

37. Rabin. URL: https://www.geeksforgeeks.org/rabin-karp-algorithm-for-pattern-searching/ (дата звернення 18.02.2024).

38. KMP Algorithm for Pattern Searching – GeeksforGeeks. URL: https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/ (дата звернення 19.02.2024).

39. Minhash LSH Implementation Walkthrough: Deduplication – Dzone. URL: https://dzone.com/articles/minhash-lsh-implementation-walkthrough (дата звернення 24.02.2024).

40. Locality Sensitive Hashing (LSH): The Illustrated Guide | Pinecone. URL: https://www.pinecone.io/learn/series/faiss/locality-sensitive-hashing/ (дата звернення 26.02.2024).

41. Latent Dirichlet Allocation. Latent Dirichlet Allocation, or LDA for… | by Cory Maklin | Medium. URL: https://medium.com/@corymaklin/latent-dirichlet-allocation-dfcea0b1fddc (дата звернення 27.02.2024).

42. Gensim: Topic modelling for humans. URL: https://radimrehurek.com/gensim/ (дата звернення 27.02.2024).

43. Exploring Word2Vec. Here we have represented words with… | by Abhishek Jain | Medium. URL: https://medium.com/@abhishekjainindore24/exploring-word2vec-fb05de8c280d (дата звернення 01.03.2024).

44. TF-IDF in NLP (Term Frequency Inverse Document Frequency) | by Abhishek Jain | Medium. URL: https://medium.com/@abhishekjainindore24/tf-idf-in-nlp-term-frequency-inverse-document-frequency-e05b65932f1d (дата звернення 02.03.2024).

45. What is Inverse Document Frequency (IDF)? - Kavita Ganesan, PhD. URL: https://kavita-ganesan.com/what-is-inverse-document-frequency/ (дата звернення 01.03.2024).

46. MySQL. URL: https://www.mysql.com/ (дата звернення 01.03.2024).

47. PostgreSQL: The world's most advanced open source database. URL: https://www.postgresql.org/ (дата звернення 01.03.2024).

48. MongoDB: The Developer Data Platform | MongoDB. URL: https://www.mongodb.com/ (дата звернення 05.03.2024).

49. Getting Started with Apache Cassandra | Datastax. URL: https://www.datastax.com/blog/getting-started-apache-cassandra (дата звернення 05.03.2024).

50. What is a CDN? - Content Delivery Network Explained – AWS. URL: https://aws.amazon.com/what-is/cdn/ (дата звернення 10.03.2024).

51. What is SSL/TLS Certificate? - SSL/TLS Certificates Explained – AWS. URL: https://aws.amazon.com/what-is/ssl-certificate/?nc1=h_ls (дата звернення 12.03.2024).

52. Introduction to NoSQL – GeeksforGeeks. URL: https://www.geeksforgeeks.org/introduction-to-nosql/ дата звернення 13.03.2024).

53. SQL запити: основи, обчислення та визначення складності. URL: https://foxminded.ua/sql-zapyty/ (дата звернення 14.03.2024).

54. The web framework for perfectionists with deadlines | Django. URL: https://www.djangoproject.com/ (дата звернення 15.03.2024).

55. Welcome to Flask — Flask Documentation (3.0.x). URL: https://flask.palletsprojects.com/en/3.0.x/ (дата звернення 15.03.2024).

56. Welcome to Python.org. URL: https://www.python.org/ (дата звернення 15.03.2024).

57. PyTorch. URL: https://pytorch.org/ (дата звернення 16.03.2024).

58. Acid | Definition, Examples, Types, Uses, & Facts | Britannica. URL: https://www.britannica.com/science/acid (дата звернення 17.03.2024).

59. What is Natural Language Processing (NLP)? | Oracle Egypt. URL: https://www.oracle.com/eg/artificial-intelligence/what-is-natural-language-processing/ (дата звернення 18.03.2024).

60. Computer Analysis of Text Tonality Based on the JSM Method | IEEE Conference Publication | IEEE Xplore. URL: https://ieeexplore.ieee.org/document/8656847 (дата звернення 18.03.2024).

61. Visual Studio Code - Code Editing. Redefined. URL: https://code.visualstudio.com/ (дата звернення 12.04.2024).

62. JSON - JavaScript | MDN. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON (дата звернення 21.04.2024).

63. CSV (comma separated values): everything you need to know about this file format. URL: https://datascientest.com/en/csv-comma-separated-values-everything-you-need-to-know-about-this-file-format (дата звернення 21.042024).

64. TextBlob: Simplified Text Processing — TextBlob 0.18.0.post0 documentation. URL: https://textblob.readthedocs.io/en/dev/ (дата звернення 22.03.2024).

65. scikit-learn: machine learning in Python — scikit-learn 1.5.0 documentation. URL: https://scikit-learn.org/stable/ (дата звернення 23.03.2024).

66. What Is a Denial-of-Service (DoS) Attack? | Zscaler. URL: https://www.zscaler.com/resources/security-terms-glossary/what-is-a-denial-of-service-attack (дата звернення 24.04.2024).

67. Найвідоміші вразливості веб застосунків. XSS та SQL ін’єкції, вразливості автентифікації | DOU. URL: https://dou.ua/forums/topic/40613/ (дата звернення 24.04.2024).

68. What is Transport Layer Security (TLS)? | Cloudflare. URL: https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/ (дата звернення 27.04.2024).

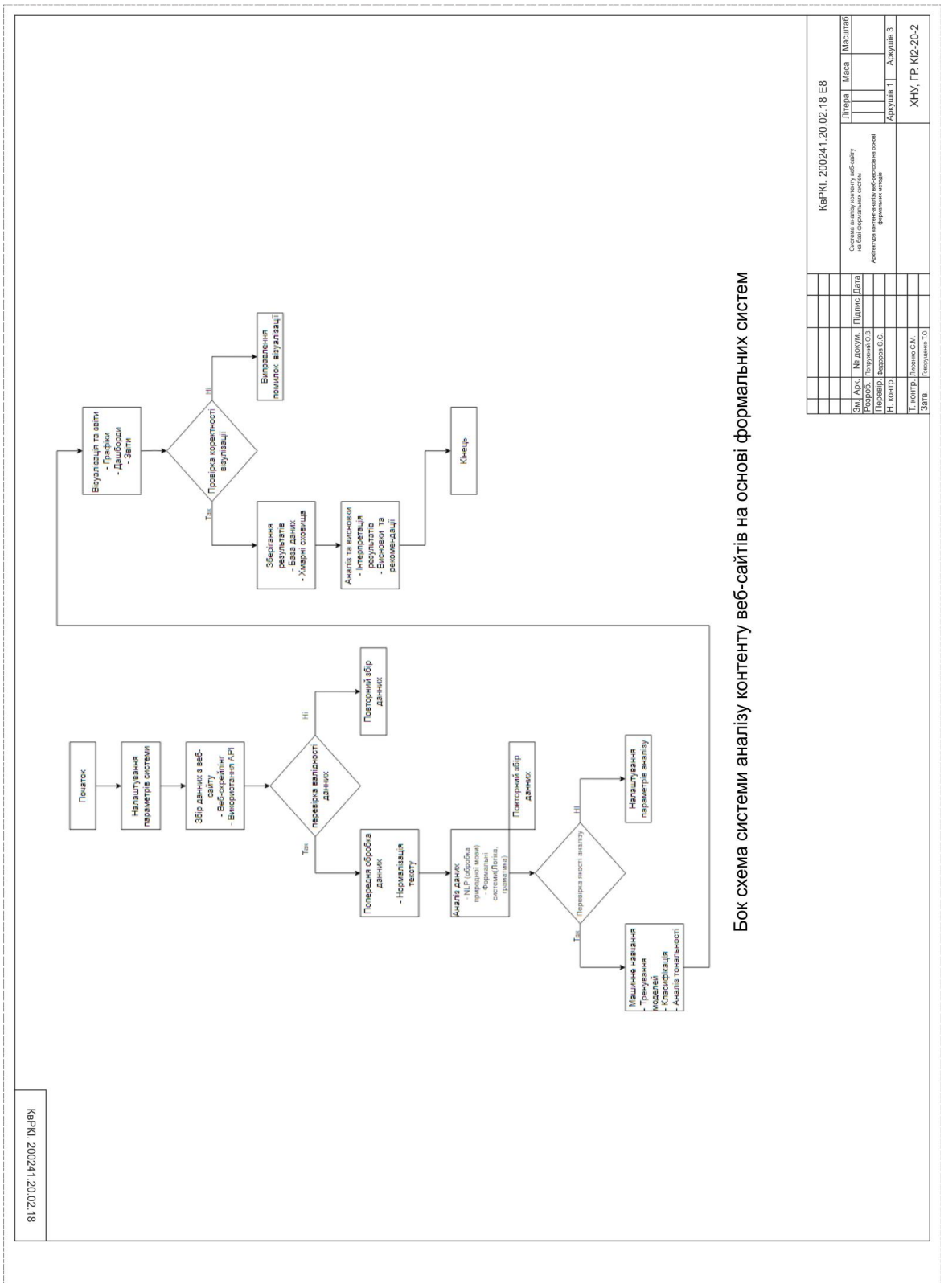
69. Docker: Accelerated Container Application Development. URL: https://www.docker.com/ (дата звернення 10.05.2024).

70. Kubernetes. URL: https://kubernetes.io/ (дата звернення 12.05.2024).

Додаток А

(обов’язковий)

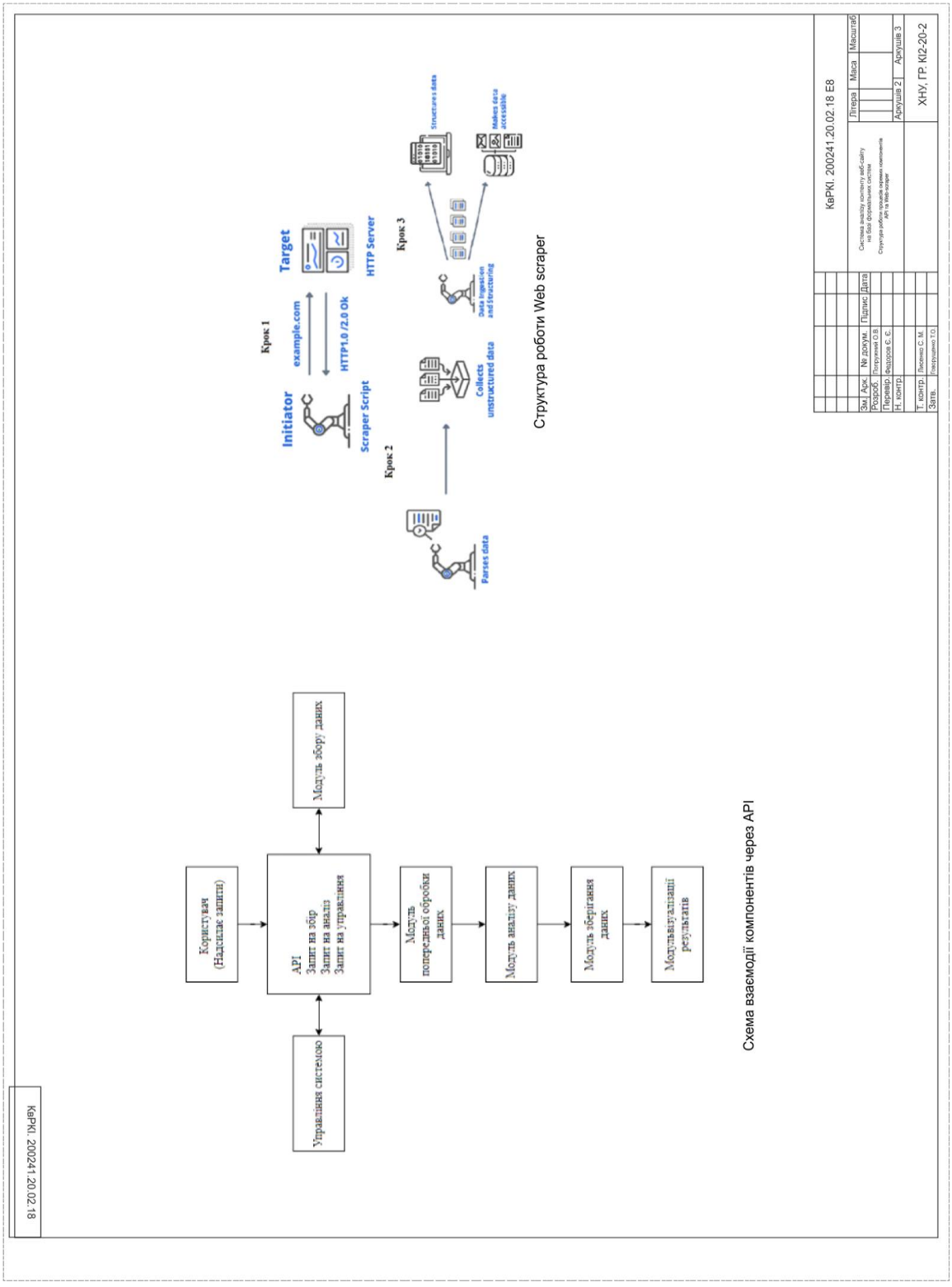
Копія креслення «Архітектура контент-аналізу веб-ресурсів на основі формальних методів»



Додаток Б

(обов'язковий)

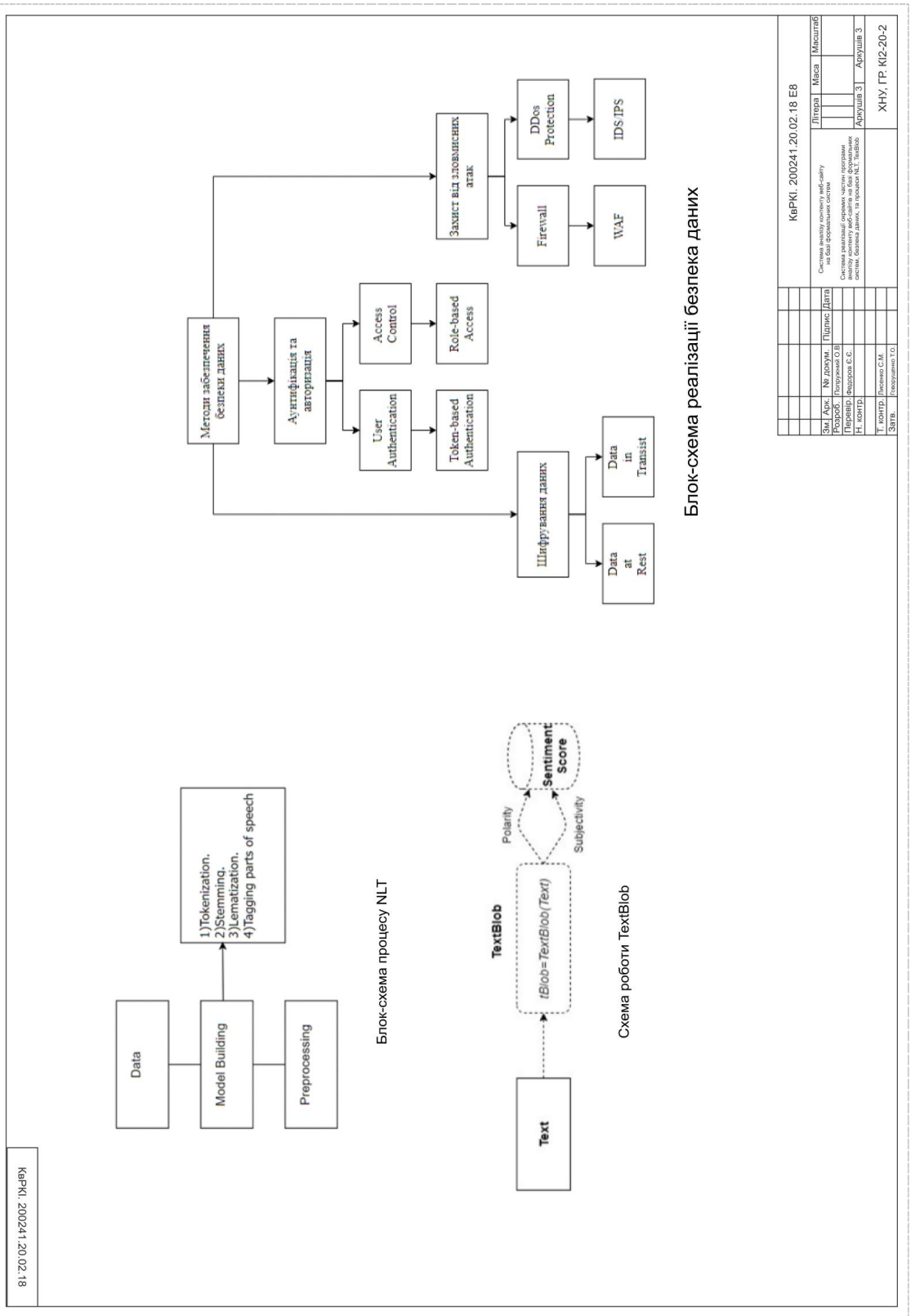
Копія креслення «Структура роботи процесів окремих компонентів API та Web-scraper»



Додаток В

(обов’язковий)

Копія креслення «Система реалізації окремих частин програми аналізу контенту веб-сайтів на базі формальних систем, безпека даних, та процеси NLT, TexBlob»



UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КІ

ID перевірки:
1016370691

Дата перевірки:
18.06.2024 08:22:52 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
18.06.2024 08:31:39 EEST

ID користувача:
100005591

Назва документа: **Попружний_Система аналізу контенту веб-сайту на базі формальних систем**

Кількість сторінок: **71** Кількість слів: **13897** Кількість символів: **112188** Розмір файлу: **1.54 MB** ID файлу: **1016177812**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

8.04%
Схожість

Найбільша схожість: **1.19%** з Інтернет-джерелом (https://dokumen.tips/documents/-32339-f-ff-f.html)

7.73% Джерела з Інтернету	762	Сторінка 73
3.43% Джерела з Бібліотеки	147	Сторінка 83

0% Цитат

Не знайдено жодних цитат

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	2
Підозріле форматування	12 сторінок

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 12%**

ID: 131209 Назва: БКР Система аналізу контенту веб-сайту на базі формальних систем Додано в БД: 2024-06-18 Автора: О.В. Попружний Керівники: Є. Є. Федоров Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	97779	722	2268 (2%)	28 (4%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Попружний Олександр Васильович

Тема: Система аналізу контенту веб-сайту на базі формальних систем

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 Кількість сторінок записки 62

1. Короткий зміст роботи та прийнятих рішень: Метою даної дипломної роботи є підвищення ефективності використання системи аналізу контенту на веб-сторінках на базі формальних систем.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи:

У першому розділі проведено аналіз існуючих систем аналізу контенту веб-сайтів. Вивчено основні підходи та методи, що використовуються в цій галузі, зокрема, обробку натуральної мови (NLP), машинне навчання (ML) та різні алгоритми аналізу тексту, такі як семантичний та синтаксичний аналіз. Було також розглянуто застосування формальних систем, що дозволяють підвищити точність та ефективність аналізу. У другому розділі проведено детальний аналіз проектування програмно-технічного засобу для системи аналізу контенту. Визначено ключові етапи розробки, включаючи збір даних, попередню обробку тексту, моделювання та візуалізацію результатів. Розглянуто методи web scraping та використання API для автоматичного отримання даних з веб-сайтів. Описані процеси попередньої обробки тексту, такі як токенізація, лематизація та видалення стоп-слів, що забезпечують якісну підготовку даних. У третьому розділі проведено детальну програмно-технічну реалізацію системи аналізу контенту веб-сайтів. Реалізовано модулі збору даних за допомогою бібліотек BeautifulSoup та Scrapy, а також інтеграцію з API для отримання структурованих даних. Попередня обробка тексту виконувалася з

використанням бібліотек NLP, таких як NLTK та spaCy. Моделювання та аналіз здійснювалися за допомогою моделей машинного навчання та глибокого навчання, зокрема, методів SVM, CNN та RNN. Моделювання та аналіз здійснювалися за допомогою моделей машинного навчання та глибокого навчання, зокрема, методів SVM, CNN та RNN. Візуалізація результатів забезпечувалася інструментами Plotly та Dash, що надали користувачам зручний доступ до аналітичних даних. Завершальним етапом була інтеграція всіх компонентів системи та її тестування на різних наборах даних.

4. Позитивні сторони роботи: система може аналізувати контент на веб-сторінках, а також вона дозволяє автоматизувати цей процес.

5. Негативні сторони роботи: потрібно провести більш детальне дослідження обробки природної мови.

Потрібно навести більше прикладів промислових систем розпізнавання обробки тексту з веб-сторінок з детектуванням обєктів.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження:

Бажано розглянути також аналіз англомовного контенту за допомогою розглянутих кваліфікацьною роботою методів мови.

Бажано б більш детально розглянути аналіз українських словоформ за допомогою пакетів пайтон.

9. Оцінка дипломної роботи: добре

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) ______________

Бадоліан Леонід Петрович

зав. каф. ІПЗ

"15" червня 2024 р. ______________ (підпис)

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т. О.

Попружний Олександр Васильович
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-20-2

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

11 червня 2024 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Система аналізу контенту веб-сайту на базі формальних систем

Автор: Попружний Олександр Васильович

Спеціальність: 123– Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Федоров Євген Євгенович, д.т.н, професор.

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з 10-40 джерелами на один фрагмент речення;
4) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 8.04% і адресується до 909 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи Федоров Є. Є. Федоров

Гарант ОП С. М. Лисенко

Завідувач кафедри КІС Т. О. Говорущенко