

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри

_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)

“ _____ ” 2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення
інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Програмне забезпечення для класифікації текстових повідомлень

Виконав студент IV курсу, групи _____ ПП-11
(шифр групи)

Ющенко Андрій Вікторович
(прізвище, ім'я, по батькові)

_____ (підпис)

Керівник доцент, к.т.н., Олійник Ю. О.
(посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____ (підпис)

Консультант доцент, к.т.н., Лішук К. І.
(посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____ (підпис)

Рецензент доцент кафедри ІСТ, к.ф.-м.н., доц., Рибачук Л. В.
(посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____ (підпис)

Засвідчую, що у цьому дипломному проєкті
немає запозичень з праць інших авторів без
відповідних посилань.

Студент _____
(підпис)

Київ – 2025

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення
інформаційних систем

ЗАТВЕРДЖУЮ

Завідувач кафедри

(підпис) Едуард ЖАРІКОВ
(ім'я прізвище)

“ ____ ” _____ 2025 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Ющенку Андрію Вікторовичу

(прізвище, ім'я, по батькові)

1. Тема проєкту Програмне забезпечення для класифікації текстових повідомлень

керівник проєкту Олійник Юрій Олександрович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «23» травня 2025 р. №1705-с

2. Термін подання студентом проєкту «16» червня 2025 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки

1) Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.

2) Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури даних.

3) Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв'язання.

4) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів.

5) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань _____

2) Схема структурна компонентів програмного забезпечення _____

3) Креслення вигляду екранних форм _____

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «15» березня 2025 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	15.03.2025	
2	Аналіз існуючих методів розв'язання задачі	25.03.2025	
3	Постановка та формалізація задачі	05.04.2025	
4	Розробка інформаційного забезпечення	15.04.2025	
5	Алгоритмізація задачі	20.04.2025	
6	Обґрунтування вибору використаних технічних засобів	25.04.2025	
7	Розробка програмного забезпечення	15.05.2025	
8	Налагодження програми	20.05.2025	
9	Виконання графічних документів	26.05.2025	
10	Оформлення пояснювальної записки	28.05.2025	
11	Подання ДП на попередній захист	04.06.2025	
12	Подання ДП рецензенту	10.06.2025	
13	Подання ДП на основний захист	16.06.2025	

Студент

(підпис)

Андрій ЮЩЕНКО

(ініціали, прізвище)

Керівник

(підпис)

Юрій ОЛІЙНИК

(ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з п'яти розділів, містить 21 таблицю, 15 рисунків та 20 джерел – загалом 62 сторінки.

Дипломний проєкт присвячений розробці програмного забезпечення для класифікації текстових повідомлень.

Мета : підвищити ефективність та точність класифікації текстових повідомлень шляхом створення спеціалізованого програмного забезпечення.

У першому розділі наведено аналіз предметної області класифікації текстових повідомлень, порівняльний аналіз між реалізованим програмним застосунком та аналогами, виділено їх переваги та недоліки.

У другому розділі наведено основні варіанти використання програмного забезпечення, функціональні та нефункціональні вимоги до програмного забезпечення та сформульовано системні вимоги до нього, наведено результати аналізу економічних показників програмного забезпечення та наведено приблизну його собівартість.

У третьому розділі наведено архітектуру програмного забезпечення, наведено обґрунтування прийнятих основних архітектурних рішень та застосування використаних засобів розробки.

У четвертому розділі наведено опис процесів тестування та контрольного прикладу.

У п'ятому розділі наведено опис обраного способу розгортання розробленого програмного забезпечення, розгортання програмного забезпечення, супровід програмного забезпечення.

Програмне забезпечення впроваджено на хостингу Render.

КЛЮЧОВІ СЛОВА: ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, КЛАСИФІКАЦІЯ ДАНИХ, ТЕКСТОВІ ПОВІДОМЛЕННЯ, APACHE SPARK, APACHE KAFKA, DJANGO.

ABSTRACT

The explanatory note of the diploma project consists of five sections, contains 21 tables, 15 figures and 20 sources – in total 62 pages.

The diploma project is dedicated to the development of a software for classifying text messages.

The purpose is to improve the efficiency and accuracy of text message classification by creating specialised software.

The first chapter provides an analysis of the subject area of text message classification, a comparative analysis between the implemented software application and its analogues, and highlights their advantages and disadvantages.

The second chapter presents the main options for using the software, functional and non-functional requirements for the software, and formulates system requirements for it, presents the results of the analysis of the economic indicators of the software, and gives its approximate cost.

The third chapter presents the software architecture, justifies the main architectural decisions made, and describes the development tools used.

The fourth chapter describes the testing processes and a control example.

The fifth chapter describes the chosen method of deploying the developed software, software deployment, and software support.

The software is implemented on Render hosting.

KEYWORDS: SOFTWARE, DATA CLASIFICATION, TEXT MESSAGES, APACHE SPARK, APACHE KAFKA, DJANGO.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ” _____ 2025 р.

**ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ КЛАСИФІКАЦІЇ ТЕКСТОВИХ
ПОВІДОМЛЕНЬ**

Технічне завдання

КПІ.ПІ-1130.045440.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Юрій ОЛІЙНИК

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Андрій ЮЩЕНКО

Київ – 2025

ЗМІСТ

1	НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ	3
2	ПІДСТАВА ДЛЯ РОЗРОБКИ.....	4
3	ПРИЗНАЧЕННЯ РОЗРОБКИ	5
4	ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	6
4.1	Вимоги до функціональних характеристик	6
4.1.1	Запуск та зупинка обробки.....	6
4.1.2	Перегляд даних після класифікації	6
4.1.3	Введення параметрів класифікації	7
4.1.4	Введення інформації про класи.....	7
4.2	Вимоги до надійності.....	8
4.3	Умови експлуатації	8
4.3.1	Вид обслуговування.....	9
4.3.2	Обслуговуючий персонал.....	9
4.4	Вимоги до складу і параметрів технічних засобів.....	9
4.5	Вимоги до інформаційної та програмної сумісності.....	9
4.5.1	Вимоги до вхідних даних	9
4.5.2	Вимоги до вихідних даних	9
4.5.3	Вимоги до мови розробки	9
4.5.4	Вимоги до середовища розробки	10
4.5.5	Вимоги до представленню вихідних кодів.....	10
4.6	Вимоги до маркування та пакування	10
4.7	Вимоги до транспортування та зберігання	10
4.8	Спеціальні вимоги.....	10
5	ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ	11
5.1	Попередній склад програмної документації.....	11
5.2	Спеціальні вимоги до програмної документації	11
6	СТАДІЇ І ЕТАПИ РОЗРОБКИ	12
7	ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ	13

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення для класифікації текстових повідомлень.

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмного забезпечення TextClassifier, котре використовується для автоматичної класифікації текстових повідомлень та призначена для автоматичної класифікації текстових повідомлень на підприємствах.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки TextClassifier є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для автоматичної класифікації текстових повідомлень та спостереженням за процесом класифікації.

Метою розробки є розробка програмного забезпечення, яке значно полегшить та автоматизує процес класифікації текстових повідомлень.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1 Запуск та зупинка обробки

- користувач повинен мати змогу запускати та зупиняти обробку текстових повідомлень за допомогою кнопки;
- користувач має мати інформацію про поточний статус обробки.

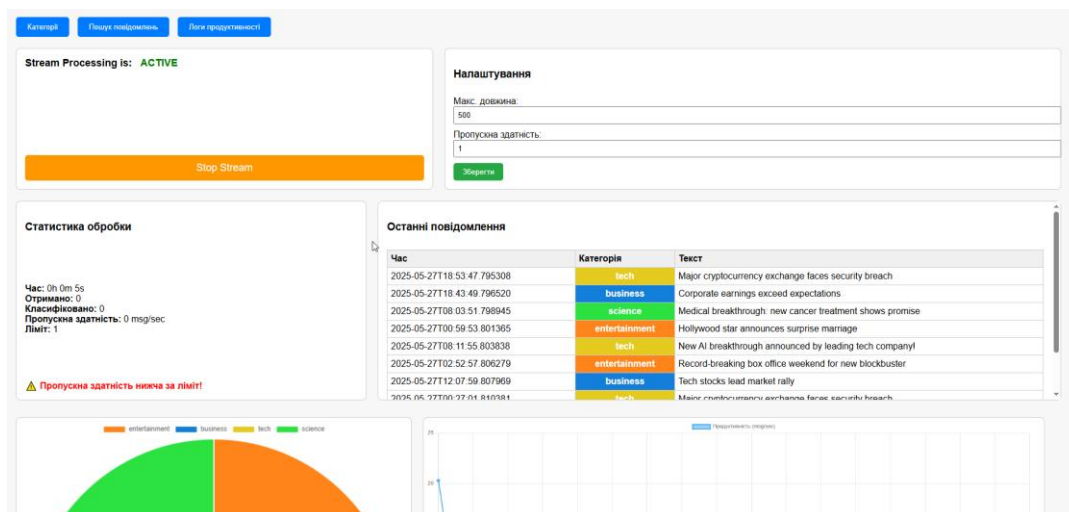


Рисунок 4.1 – Екранна форма запуску та зупинки обробки

4.1.2 Перегляд даних після класифікації

- користувач повинен мати змогу переглядати дані після класифікації;
- користувач повинен мати змогу фільтрувати та шукати повідомлення після процесу класифікації.

→ Back to Stream Control

Пошук класифікованих повідомлень

Пошук за текстом: business

Час	Категорія	Текст	Канал	Перегляди
2025-05-25T12:35:48.504549	business	Diplomatic relations between nations improve	Global Politics	815
2025-05-26T00:06:48.506999	business	Global supply chain disruptions continue	Business Daily	1257
2025-05-25T05:59:50.509404	business	International trade tensions escalate	Global Politics	397
2025-05-25T18:41:54.514718	business	Retail sector faces challenging quarter	Business Daily	819
2025-05-25T06:53:56.517159	business	UN Security Council holds emergency meeting	Global Politics	621
2025-05-25T14:30:58.520056	business	Global supply chain disruptions continue	Business Daily	1014
2025-05-25T12:53:00.522479	business	UN Security Council holds emergency meeting	Global Politics	341
2025-05-25T07:47:08.532909	business	Celebrity chef opens restaurant chain	Entertainment Buzz	2848
2025-05-25T10:37:10.535363	business	Tech stocks lead market rally	Business Daily	1483
2025-05-25T06:58:12.537737	business	Election results surprise political analysts	Global Politics	1029
2025-05-25T04:07:18.546128	business	Celebrity chef opens restaurant chain	Entertainment Buzz	183
2025-05-25T13:26:28.553884	business	Retail sector faces challenging quarter	Business Daily	651
2025-05-26T08:25:47.216749	business	International trade tensions escalate	Global Politics	1009
2025-05-26T01:25:49.219164	business	Corporate earnings exceed expectations	Business Daily	1721
2025-05-25T10:51:59.231201	business	Diplomatic relations between nations improve	Global Politics	725
2025-05-26T01:42:01.234108	business	Retail sector faces challenging quarter	Business Daily	1372

Рисунок 4.2 – Екранна форма пошуку та фільтрації повідомлень після процесу класифікації

4.1.3 Введення параметрів класифікації

- користувач повинен мати змогу вводити власні параметри для класифікації повідомлень;
- має відбуватись валідація введених даних користувачем.

Налаштування

Макс. довжина:

Пропускна здатність:

Рисунок 4.3 – Екранна форма введення параметрів для класифікації повідомлень

4.1.4 Введення інформації про класи

- користувач повинен мати змогу редагувати опис, назву, колір відображуваних класів
- має відбуватись валідація введених користувачем даних.

[← Back to Admin Control](#)

Налаштування категорій

ID	Label	Description	Color	Actions
0	business	Business-related news and updates, crypto	#147ed8	Edit Delete
1	entertainment	Movies, music, celebrities	#ff851b	Edit Delete
2	politics	Government, policies, and current events	#ff4136	Edit Delete
3	science	Scientific discoveries and advancements	#00a241	Edit Delete
4	tech	Technology trends and innovations	#e300ff	Edit Delete

Додати або оновити категорію

ID:

Description:

Color:

[Зберегти](#)

Рисунок 4.4 – Екранна форма введення інформації про класи

4.1.5 Моніторинг процесу класифікації

- користувач повинен мати інформаційне табло із основними метриками процесу класифікації текстових повідомлень;
- користувач повинен мати кругову діаграму, на якій зображене відсоткове співвідношення класів;
- користувач повинен мати графік продуктивності роботи системи.

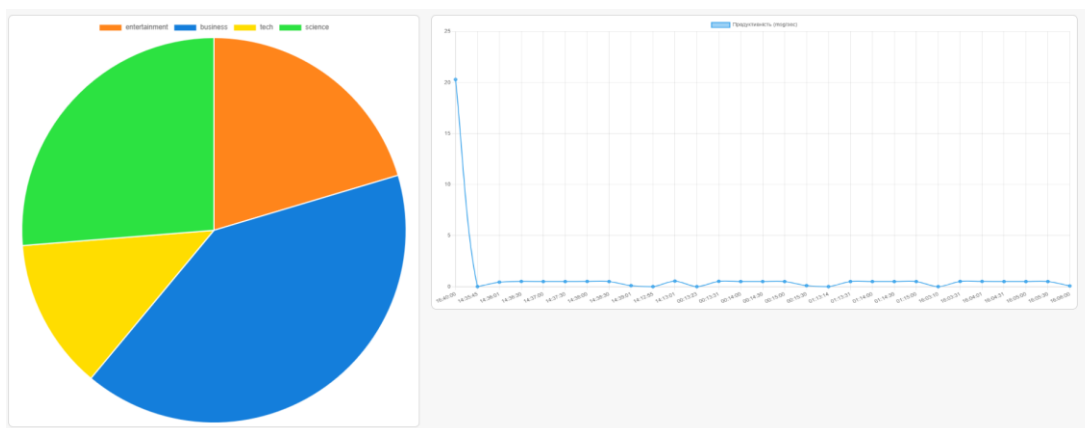


Рисунок 4.5 – Екранна форма звітності програми

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити належну безперебійність.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Мінімальна конфігурація технічних засобів:

- тип процесору: Intel Core i5;
- об'єм ОЗП: 8 Гб;
- підключення до мережі Інтернет зі швидкістю від 5 Мбіт/с.

Рекомендована конфігурація технічних засобів:

- тип процесору: Intel Core i7;
- об'єм ОЗП: 32 Гб;
- підключення до мережі Інтернет зі швидкістю від 1000 мегабіт.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням усіх операційних систем, які забезпечують з'єднання з Інтернетом і можуть надати доступ до веб-застосунку через браузер.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в форматі JSON файлу.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в форматі JSON файлу.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування Python

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформах PyCharm, Docker.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді гіт репозиторію за посиланням <https://github.com/degvor/text-msg-classifier>

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальні вимоги не висуваються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

- пояснювальна записка;
- технічне завдання;
- текст програми;
- програма та методика тестування;
- керівництво користувача.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

- схема структурна варіантів використання;
- схема структурна компонентів програмного забезпечення;
- креслення вигляду екранних форм.

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення рекомендованої літератури	15.03.2025	
2.	Аналіз існуючих методів розв'язання задачі	25.03.2025	Технічне завдання
3.	Постановка та формалізація задачі	05.04.2025	Специфікація програмного забезпечення
4.	Розробка інформаційного забезпечення	15.04.2025	
5.	Алгоритмізація задачі	20.04.2025	Формалізовані алгоритми програмного забезпечення
6.	Обґрунтування вибору використаних технічних засобів	25.04.2025	
7.	Розробка програмного забезпечення	15.05.2025	Тексти програмного забезпечення
8.	Налагодження програми	20.05.2025	Тести, результати тестування
9.	Виконання графічних документів	26.05.2025	Графічний матеріал проекту
10.	Оформлення пояснювальної записки	28.05.2025	Пояснювальна записка
11.	Подання ДП на попередній захист	04.06.2025	
12.	Подання ДП рецензенту	10.06.2025	
13.	Подання ДП на основний захист	16.06.2025	

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

**Пояснювальна записка
до дипломного проєкту**

на тему: Програмне забезпечення для класифікації текстових повідомлень

КПІ.ІІ-1130.045440.02.81

Київ 2025

ЗМІСТ

ВСТУП.....	5
1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ	6
1.1 Постановка завдання дипломного проєктування.....	6
1.2 Аналіз предметної області	7
1.3 Аналіз існуючих рішень.....	11
1.3.1 Аналіз відомих програмних продуктів.....	11
1.3.2 Аналіз відомих алгоритмічних та технічних рішень.....	15
1.4 Аналіз та моделювання бізнес-процесів	21
Висновки по розділу	22
2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	24
2.1 Варіанти використання програмного забезпечення	24
2.2 Розроблення функціональних вимог	28
2.3 Розроблення нефункціональних вимог	30
2.4 Аналіз системних вимог	31
2.5 Аналіз економічних показників програмного забезпечення	32
2.6 Постановка завдання на розробку програмного забезпечення	33
Висновки до розділу	35
3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	37
3.1 Архітектура програмного забезпечення.....	37
3.2 Архітектурні рішення та обґрунтування вибору засобів розробки	38
3.3 Конструювання програмного забезпечення	41
3.4 Аналіз безпеки даних	44
Висновки до розділу	44
4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	45
4.1 Аналіз якості ПЗ	45
4.2 Опис процесів тестування.....	45
4.3 Опис контрольного прикладу	48
Висновки до розділу	51
5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	53

5.1 Розгортання програмного забезпечення.....	53
5.2 Супровід програмного забезпечення.....	55
Висновки до розділу	56
ВИСНОВКИ	57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	59
ДОДАТОК А ЗВІТ ПОДІБНОСТІ	62

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	– Integrated Development Environment – інтегроване середовище розробки.
API	– Application programming interface, прикладний програмний Інтерфейс.
ІТ	– Інформаційні технології.
ОС	– Операційна система.
ПЗ	– Програмне забезпечення.

ВСТУП

Розробка програмного забезпечення для класифікації текстових повідомлень є актуальною задачею, оскільки обсяги інформації, які передаються і вигляді тексту безперервно зростають. Щоденно мільйони користувачів обмінюються повідомленнями через електронну пошту, месенджери або інші цифрові платформи, що породжує необхідність ефективного аналізу та впорядкування цього потоку даних.

Сучасні алгоритми машинного навчання та методи обробки природньої мови відкривають нові можливості для точного та швидкого аналізу, розпізнавання та класифікації повідомлень. Враховуючи потребу у локалізованих, зручних інструментах, розробка програмного забезпечення для класифікації текстових повідомлень є логічним кроком у напрямку цифровізації та автоматизації інформаційних процесів.

Сфери застосування такого ПЗ можуть бути надзвичайно широкими, а саме: від автоматичної модерації коментарів у соціальних мережах до корпоративних систем фільтрації пошти.

Таким чином, розробка програмного забезпечення для класифікації текстових повідомлень є важливою задачею, що відповідає викликам часту та зростаючим потребам сучасного інформаційного суспільства.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання дипломного проєктування

У сучасних умовах стрімкого зростання обсягів даних виникає потреба в автоматизованій обробці текстової інформації, що генерується в режимі реального часу. Завданням дипломного проєктування є розробка програмного забезпечення для класифікації даних у текстових потоках. Іншими словами, необхідно створити програмне забезпечення, яка отримує безперервний потік текстових повідомлень і автоматично відносить кожне повідомлення до певної категорії. Розроблюване програмне забезпечення повинне функціонувати в режимі, наближеному до реального часу, обробляючи великі обсяги неструктурованих текстових даних з високою швидкістю. Це важливо, адже бізнесу потрібні оперативні аналітичні дані для прийняття рішень, витягнуті з текстових потоків. При постановці завдання враховуються основні виклики цієї предметної області:

- обробка потоків різної інтенсивності[1,10];
- деградація моделей[8,9];
- проблема донавчання нових класів до моделей[9];
- відсутність достатньої кількості україномовних засобів обробки[10].

Детальніше розглянемо проблеми та виклики.

Проблема обробки потоків різної інтенсивності полягає в тому, що текстові повідомлення генеруються безперервно та у великій кількості, що потребує високопродуктивних рішень для їх обробки[1]. Для забезпечення ефективної обробки в реальному часі застосовуються підходи на основі ковзних вікон, що дозволяють розбивати потік на блоки для подальшого аналізу[10].

Надалі розглянемо проблематику деградації моделі. Оскільки, з часом статистичні властивості даних можуть змінюватися, що призводить до

поступового зниження точності моделі, якщо її не оновлювати. Необхідно врахувати дрейф даних та концепції, коли категорії чи характер текстів змінюються, і забезпечити можливість адаптації або перенавчання моделі[6]. Програмне забезпечення повинне вміти перенавчатись на нових даних, або оновлювати частину параметрів без повного перенавчання[9].

Оскільки, з плином часу актуальність текстів змінюється і з'являються нові теми або категорії, яких не було в попередніх наборах даних, то розроблюване програмне забезпечення має бути побудована так, аби вона мала змогу перенавчатись у режимі наближеному до реального часу на нових даних, або ж оновлювати частину параметрів без повного перенавчання[9].

Отже, постановка завдання полягає у створенні масштабованого, продуктивного програмного комплексу, здатного збирати текстові дані у потоці, оперативно їх очищувати, аналізувати та класифікувати за допомогою методів машинного навчання, з можливістю адаптації до нових умов. Це програмне забезпечення має інтегруватися у бізнес-процеси організації, надаючи результати класифікації в реальному часі для підтримки прийняття рішень. Успішне виконання цього завдання автоматизує обробку великої кількості текстової інформації, що раніше могла потребувати значних людських ресурсів, та забезпечить більш швидку і послідовну реакцію на події, що відображаються у потоці текстових даних.

1.2 Аналіз предметної області

Предметна область проєкту охоплює перетин кількох галузей знань: потокова обробка даних (stream processing), обробка природної мови (NLP) та машинне навчання, орієнтоване на класифікацію тексту. Розглянемо ключові поняття та особливості цієї області. По-перше, текстовий потік — це послідовність текстових повідомлень, що надходять у режимі наближеному до реального часу. Джерелами текстових потоків можуть бути: соціальні мережі (стрічка Twitter, Facebook тощо), месенджери або чати, стрічки новин

інформагентств, тощо. Властивістю потоку є потенційно нескінченний обсяг даних, які надходять з високою швидкістю, тобто із великою частотою нових повідомлень. У контексті "великих даних" часто оперують концепцією так званих 5V-характеристик: обсяг (Volume), швидкість генерування (Velocity), різноманітність (Variety) і достовірність (Veracity) даних.

Розглянемо найбільш важливі метрики, які відповідають тематиці нашого завдання.

Перша метрика яку маємо максимізувати – швидкість. У контексті нашого завдання швидкість відповідає за надходження даних та їх оперативну обробку. Наступною метрикою є обсяг, який представляє собою сумарну кількість отримуваних повідомлень. Контроль цієї метрики дозволяє зрозуміти чи потрібно покращувати роботу нашого застосунку шляхом горизонтального масштабування, розподіленої обробки, тощо. Немало важливою метрикою є різноманітність даних, оскільки оброблювані тексти можуть містити різні мови, формати, сленги, спеціальні символи, емодзі тощо. Розроблюване програмне забезпечення має бути гнучким щодо формату вхідних даних і універсальним для різних тематик тексту. Останньою метрикою буде минушість даних. Для стрімких даних важливо те наскільки довго вони залишаються актуальними, оскільки інформація зі стрічки соціальних мереж має властивість швидко втрачати цінність. Тому оперативна обробка критично важлива для виконання поставленого завдання. Ця концепція підкреслює, що дані із потоку можуть втрачати свою цінність, тому давні записи можуть бути вже не актуальні для аналізу[4].

Також варто розглянути загальну модель потоку текстових даних на основі ковзного вікна.

Під формулюванням визначення потоку текстових даних маємо нескінченне надходження даних з одного або декількох джерел. В результаті надходження даних отримуємо пару (d_i, t_i) , в якій d_i відповідає за отримане

повідомлення, а t_i за час коли його було отримано[9]. При такому формулюванні ковзним вікном є часове вікно W_i яке має фіксований інтервал δ та бере свій початок у точці t_i .

На рис.1.1 представлено схему моделі потоку даних із часовим вікном.

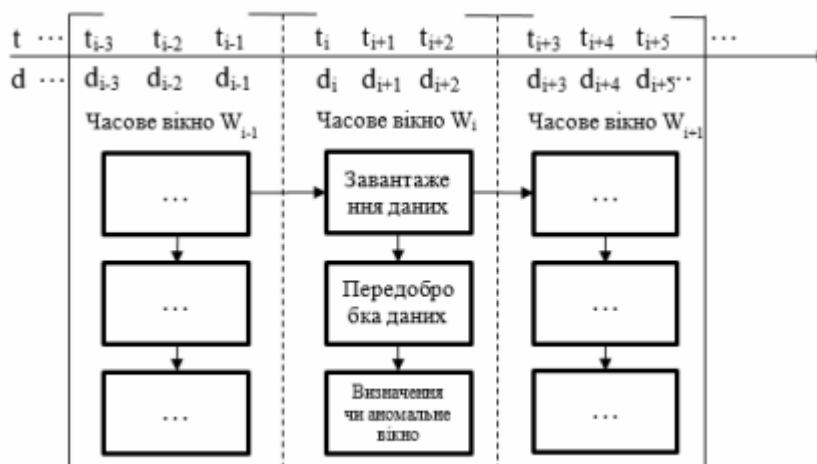


Рисунок 1.1 – Модель потоку даних на основі ковзного вікна [9]

Основним завданням нашого програмного забезпечення є класифікація тексту. Це задача віднесення певного фрагмента тексту до певної категорії. У статистичному випадку ця задача є добре дослідженою та реалізована багатьма методами, наприклад: наївний Баєс, метод опорних векторів, логістична регресія, дерева рішень, нейронні мережі, тощо. Проте у випадку потокової класифікації виникають нові додаткові складнощі. Розглянемо більш детально кожен з них.

По-перше, необхідно аби розроблене програмне забезпечення мало безперервний потік даних і водночас не виходило за обмеження пам'яті, оскільки алгоритм не може зберігати та обробляти весь потік повністю, як статистичний набір даних. Найбільш поширеним рішенням є – застосування ковзних вікон для збору даних. Це дозволяє обмежити одночасний набір даних, що необхідно знайти. Наступною проблемою, яку маємо вирішити є прийняття рішення щодо класу кожного нового повідомлення в режимі наближеному до реального часу. Ця умова накладає обмеження на складність моделі, оскільки

надто складні моделі можуть не встигати обробити усі повідомлення. Також необхідно пам'ятати про дрейф даних, одне із центральних понять потокового машинного навчання. Концепції дрейфу даних полягають в тому, що розподіл даних або внутрішні закономірності змінюються з часом[3]. Для текстових даних це означає що може додаватись нова лексика, мова, тема, тональність повідомлень. Існує декілька основних типів дрейфу – різкий, коли зміна відбувається раптово, поступовий, або в деяких джерелах повзучий, коли відбувається гладка еволюція тематики, сезонний, коли зміни відбуваються відносно часу. Варто пам'ятати, що дрейф даних призводить до деградації моделі, оскільки модель навчена на старих даних, з часом буде втрачати точність на нових даних[6]. Тому розроблюване програмне забезпечення мусить виявляти такі зміни та підлаштовуватись під них. Останнім пунктом буде необхідність попередньої обробки даних в потоці, оскільки текстові дані потребують NLP-перетворень, наприклад: очищення від HTML коду, перетворення регістру, нормалізація, токенизація, тощо. Ці кроки повинні використовуватись ефективно у реальному часі. Для прискорення можуть використовуватись розподілені обчислення або ж оптимізовані під таку задачу бібліотеки. Найбільш яким прикладом цього є Apache Spark, який надає API для паралельної обробки, очистки та токенизації даних[1].

Отже, предметна область включає Big Data аспекти: обробка великих масивів даних, масштабування, розподілені системи та машинне навчання по потоку даних. В останні роки сформувався окремий напрям — Data Stream Mining, що вивчає алгоритми класифікації, регресії, кластеризації та іншого аналізу для нескінченних потоків. Були розроблені спеціальні алгоритми, такі як Hoeffding для потокової побудови дерев, адаптивні ансамблі з детекцією дрейфу тощо. Оскільки завдання — практична розробка програмного засобу, важливо також розуміти сучасний ландшафт технологій у цій сфері. Виявлено, що комбінація Apache Kafka[11] та Apache Spark[12] є популярною архітектурою для побудови високопродуктивних конвеєрів обробки текстових

даних[1]. Kafka виступає як розподілена черга повідомлень для збору та проміжного зберігання текстових подій, тоді як Spark забезпечує механізми паралельної обробки та аналізу цих даних у пам'яті на кластері. Таке поєднання дозволяє досягти як надійності і масштабованості, так і швидкодії. Інші підходи включають використання конвеєрів на основі мікросервісів та бібліотеки Kafka Streams, або альтернативні фреймворки (наприклад, Apache Flink, Apache Storm), про які детальніше йтиметься у розділі 1.3. Насамкінець, важливо відзначити бізнес-цінність предметної області. Автоматична класифікація текстових потоків дозволяє компаніям швидко реагувати на інформаційні події: відфільтровувати небажаний контент (спам, токсичні повідомлення), миттєво перенаправляти звернення клієнтів до потрібних служб (класифікуючи тему звернення), моніторити публічну думку про бренд (аналіз тональності постів) тощо. Таким чином, дана технологія є ключовою для застосувань у сфері Big Data та реального часу.

1.3 Аналіз існуючих рішень

1.3.1 Аналіз відомих програмних продуктів

Для успішної реалізації проєкту необхідно вивчити наявні рішення – як готові програмні продукти, так і платформи та фреймворки, що можуть бути використані або надані інтегровані можливості. У сфері класифікації поточкових даних існують декілька категорій продуктів. Загалом їх можна представити таким списком: платформи потокової обробки, фреймворки та бібліотеки для навчання на потоках та комплексні рішення для аналізу текстових потоків.

Розглянемо платформи потокової обробки. Насамперед платформи потокової обробки – це інфраструктурні рішення для забезпечення прийому, буферизації та первинної обробки безперервного потоку даних. Першою платформою розглянемо Apache Kafka[11]. Ця платформа представляє собою розподілену систему обміну повідомленнями, яка спроектована саме для

обробки потоків даних у реальному часі. Kafka забезпечує високу пропускну здатність і низькі затримки, також її легко масштабувати горизонтально. Важливо зазначити, що Kafka сама по собі не класифікує, а лише транспортує дані, що робить її надійним транспортером та буфером для повідомлень. Kafka за часту використовується у поєднанні із іншими інструментами для виконання машинного навчання або аналітики[7]. Надалі розглянемо Apache Spark, який представляє собою розподілену платформу для обробки даних у пам'яті. Напочатку свого існування Apache Spark[12] був орієнтований на пакетну обробку, але після додавання модуля Spark Streaming є цілком підходящим для обробки даних у потоці. Основною перевагою нового модуля є мікропакетна обробка, коли потік розбивається на малі інтервали, задані користувачем, і кожен інтервал обробляється як пакет даних, звідси це дещо підвищує затримку, оскільки є необхідним час на збір мікропакету. Apache Spark в контексті класифікації тексту може використовуватись як для підготовки моделі, для навчання на великому історичному датаєті, так і для застосування моделі на потоці[12]. Apache Flink[13] – сучасний стрімінговий фреймворк, який орієнтований на подійну обробку з низькими затримками. Він підтримує як режим потокової обробки з точною семантикою, так і режим пакетної. Flink, на відміну від класичного Spark Streaming, не обмежений мікропакетами, він може обробляти кожну подію незалежно. Також Flink надає можливість збереження стану операції, що є досить корисним для складних алгоритмів класифікації. Тепер більш детально розглянемо Apache Storm, який є одним із перших розповсюджених систем реального часу. Він дозволяє визначити топологію потокової обробки у вигляді графу, а також забезпечує дуже низькі затримки, але для машинного навчання Storm менш зручний, оскільки не має власних бібліотек машинного навчання. Останнім із цієї групи розглянемо хмарні сервіси обробки потоків. В цій підгрупі можна виділити таких провайдерів як AWS, Google Cloud, Azure. Вони пропонують керовані служби для обробки потоків даних. Звідси у нашому контексті акцент робиться на

відкритих технологіях таких як Kafka або Spark, але слід знати і про існування хмарних альтернатив.

Окрім платформ загального призначення, існують спеціалізовані інструменти саме для машинного навчання на стрімінгових даних, отже наступною групою що ми розглянемо буде фреймворки та бібліотеки для навчання на потоках. Першим із цієї підгрупи буде фреймворк на Java під назвою MOA (Massive Online Analysis). Цей фреймворк є по суті бібліотекою алгоритмів для класифікації та кластеризації потоків даних і одночасно середовищем для експериментів[5]. Він містить реалізацію найбільш відомих алгоритмів, таких як: беггінг, бустинг, Hoeffding Tree, а також алгоритми кластеризації і засоби оцінки якості на потоці. Хоча MOA більше спрямований на дослідницькі задачі та експерименти, він закладає фундамент для реальних систем. Наприклад, на основі MOA був створений проект Apache SAMOA, який дозволяє запускати алгоритми MOA у розподіленому середовищі (на платформах як Flink, Storm, Samza). Тепер більш детально заглибимось у реалізації на мові Python. Першою такою реалізацією буде Scikit-multiflow / River. Ця реалізація є також інструментом для потокового машинного навчання. Scikit-multiflow був бібліотекою, що перенесла ідеї MOA на Python, реалізувавши потокоорієнтовані алгоритми. Наразі ця бібліотека об'єдналась із іншою під назвою Creme і перетворилась на River. Це python-бібліотека для онлайн-навчання, яка надає зручний інтерфейс, подібний до scikit-learn, але для потоків. Ця бібліотека підтримує роботу із текстовими даними в режимі онлайн та містить кілька детекторів дрейфу. River дозволяє будувати рішення безпосередньо на Python, що обробляють стрімінгові дані, але зазвичай для високонавантажених систем Python-бібліотек може бути недостатньо продуктивності. Також варто зазначити інші вузькоспеціалізовані рішення. Наприклад бібліотека sklearn має методи `partial_fit` у деяких популярних алгоритмах, наприклад наївний Баєс, що дозволяє навчати модель поступово, передаючи дані блоками. Окремо варто згадати бібліотеки для глибокого

навчання – самі по собі TensorFlow чи PyTorch не є стрімінговими, але їх можна використати для побудови та розгортання моделі, яка потім викликається в режимі реального часу[14].

Останньою групою розглянемо комплексні рішення для аналізу текстових потоків. Ця група в собі включає програмні продукти або платформи, які пропонують повний цикл збору та аналізу текстових даних у режимі реального часу. Першим прикладом буде Splunk[15]. Він збирає потоки логів і застосовує до них правила пошуку певних подій. Хоча варто зазначити, що Splunk більше орієнтований на пошук і агрегацію, а не на класифікацію за допомогою машинного навчання, його ми розглядаємо як приклад промислового підходу до роботи з потоками. Коротко розглянемо IBM Streams, що являє собою платформу для потокової аналітики, яка підтримує аналіз тексту і машинне навчання у потоці. Останнім прикладом буде Lambda/Kappa Architectures, що являє собою швидше архітектурні шаблони. Lambda-архітектура поєднує потокову і пакетну обробку, коли важливі події обробляються негайно у спід-шарі, тоді як для високої точності дані паралельно зберігаються та оброблюються у батч-шарі з більшим запізненням, після чого результат об'єднується. Карра-архітектура в свою чергу спрощує підхід, тим що використовує єдиний потік як джерело істини, і лише один шар обробки, що може повторно програвати весь журнал для повторної обробки в разі зміни логіки. Загалом із цих двох методів, для класифікації тексту найбільш підходящою може бути Карра-архітектура, де Kafka зберігає всі повідомлення, а класифікатор обробляє їх, а якщо модель оновлено то класифікатор можна запустити з початку потоку або з потрібного моменту, щоб перекласифікувати дані.

Підсумовуючи, на ринку є широкий спектр рішень для обробки стрімінгових текстових даних. Відомі програмні продукти варіюються від спеціалізованих до масштабованих платформ загального призначення. Наш підхід – використати найкраще з обох світів, це – відкрите масштабоване

середовище з використанням Kafka та Spark, а також алгоритмічні напрацювання, можливо задіявши існуючі бібліотеки там, де це можливо.

Таблиця 1.1 – Порівняння наявних програмних аналогів

Програмне забезпечення/ Критерій	Amazon Comprehend	Google NL API	Prodigy	Розроблюване програмне забезпечення
Класифікація тексту	+	+	+	+
Потокова обробка	+	+	-	+
Можливість тренувати свою модель	-	-	+	+
Вартість використання	-	-	-	+

1.3.2 Аналіз відомих алгоритмічних та технічних рішень

На додаток до програмних платформ, слід розглянути алгоритмічні та технічні рішення, що застосовуються для класифікації даних у текстових потоках. Йдеться про конкретні методи машинного навчання та архітектурні підходи, які дозволяють будувати ефективні моделі в умовах, описаних у розділі 1.2. Ці рішення можна поділити на такі групи як алгоритми машинного навчання для потокової класифікації та технічні рішення і архітектурні патерни.

Першою групою розглянемо алгоритми машинного навчання для потокової класифікації. Традиційні алгоритми класифікації потребують адаптації для роботи з потоками. Деякі методи природно підходять для інкрементного навчання, інші – ні.

Першим методом із цієї групи буде наївний Баєсів класифікатор[16]. Це простий але ефективний метод для класифікації текстів. Його перевага полягає у легкості оновлення, оскільки модель зберігає підрахунки частот слів у кожному класі та загальні частоти класів. Якщо надходить новий приклад з відомим класом, достатньо оновити лічильники, але якщо класифікуємо потік без можливості онлайн навчання, наївний Баєс використовується лише для передбачень. За потреби присутня можливість швидко перерозрахувати частоти. Головний недолік методу наївного Баєса – це обмежена виразність моделі, можливо недостатня точність для складних задач, а з переваг цей метод є стійким до шуму і мало схильним до перенавчання, що є дуже корисним для роботи у потоці.

Далі розглянемо логістичну регресію[16]. Вона може навчатися інкрементно за допомогою стохастичного градієнтного спуску. У бібліотеці, про яку ми вже згадували вище `scikit-learn`, реалізовано метод, за допомогою якого ми можемо оновлювати модель по мірі надходження нових даних. Подібно працюють і нейронні мережі, яка навчалась шляхом логістичної регресії. Вона може продовжити навчання на нових даних, що дає змогу поступово донавчати мережу на потоці, але і присутні ризики, а саме – без належного контролю навчання, модель може забувати старі дані надто швидко, або і навпаки. Виникає поняття стабільність-пластичність, коли модель має бути достатньо пластичною, аби мати змогу адаптуватися, але в той самий час зберігати певну стабільність щодо старих даних. Тому в цьому методі часто застосовують стратегії ковзних вікон.

Надалі поговоримо про дерева рішень. Це алгоритм VFDT (Hoeffding Tree) спеціально розроблений для потоків, ідея якого полягає в поступовому рості дерев, коли стає статистично достовірно, що певне розбиття дерев є доцільним[17]. Критерій зведений на нерівності Хоффдінга, що дозволяє не зберігати всі дані, а поступово накопичувати інформацію для прийняття рішення про розбиття вузла. Дерева Хоффдінга можуть працювати в умовах

концепції дрейфу, якщо реалізувати механізм заміни піддерев. Адаптаційні дерева Хоффдінга включають детектор дрейфу, який відслідковує зниження точності в листках дерев і за умови фіксації зміни, починають тренувати альтернативні піддерева паралельно, замінюючи потім старі. Такі дерева можуть перевчитись на новий розподіл, не втрачаючи ефективності. На базі таких дерев будують ансамблі, такі як: адаптаційний випадковий ліс та експоненційно зважений ансамбль.

Говорячи про адаптаційний випадковий ліс, який представляє собою ансамбль з багатьох дерев Хоффдінга, кожне із яких навчається на потокових даних з буфером недавніх зразків. Адаптаційний випадковий ліс використовує детектори дрейфу для кожного дерева, з чого виходить що коли дрейф виявлено, відповідне дерево скидається і починає навчатись з нуля. В результаті, частина лісу завжди менш навчена на старих даних і більш пристосована до останніх, а інша частина навпаки. Цей алгоритм показує відмінні результати в багатьох дослідженнях концепції дрейфу.

Тепер більш детально про експоненційно-зважений ансамбль. Це алгоритми онлайн-беггінгу та онлайн підсилення, що дозволяють оновлювати ансамбль класифікаторів у потоці. Вони зазвичай працюють із фіксованим розміром ансамблю і при надходженні нового прикладу кожному класифікатору дають можливість оновитись.

Також розглянемо методи на основі відстаней та кластери. Відомо, що простий k -найближчих сусідів погано працюватиме в потоці, бо доводилося б зберігати усі приклади одночасно для коректної роботи програмного забезпечення. Для вирішення цієї проблеми варто розглянути підходи із стисненням даних. Наприклад MC-NN, який зберігає статистичні дані у вигляді мікрокластерів і виконує класифікацію, порівнюючи нові об'єкти із цими кластерами[4]. Це дозволить значно зменшити обсяг пам'яті і зробити алгоритм паралельним, адже різні кластери можна оновлювати окремо. Цей

підхід показує здатність адаптуватись до дрейфу даних, а також високий рівень масштабованості[4].

Тепер поговоримо про детекторів дрейфу, що являють собою важливий клас алгоритмів, які не займаються безпосередньо класифікацією, але допомагають виявити момент, коли модель стала погіршуватись. Грубо кажучи, детектор дрейфу моніторить потік даних або помилки моделі і сигналізує нам про зміну.

Розглянемо більш детально конкретні приклади детекторів дрейфу.

Розглянемо DDM або метод виявлення дрейфу, суть методу полягає у відстежуванні відсотку помилок класифікатора і порівнянні його статистики з очікуваною для стаціонарного процесу. Якщо похибка значно зросла – припускаємо дрейф[18,19].

Наступний, досить схожий метод, метод раннього виявлення дрейфу. Фактично, це є модифікацією попереднього, який є більш чутливим до поступових змін, фокусуючись на відстані між помилками.

Також цікавим для нас методом є ADWIN адаптаційне вікно[18,20]. Цей метод підтримує ковзне вікно і автоматично змінює розмір, для виявлення зміни в середньому значенні. Якщо статистики на початку і в кінці вікна значно відрізняються – виявлено зміну, і розмір вікна змінюється. Детектори дрейфу можуть працювати незалежно від моделей класифікації. Типовий підхід – додавання модуля моніторингу, якщо він сигналізує про наявність дрейфу, то розроблюване програмне забезпечення перевчає модель з нуля на останніх даних, або запускає процедуру адаптації. Важливо розуміти, що помилкові спрацювання детектора цілком можливі, тому важливо розробити стійку до цього стратегію.

Переходимо до другої групи яку ми маємо розглянути, а саме технічні рішення і архітектурні патерни. Розглянемо приклади представників цієї групи.

Першим таким рішенням для розгляду буде розділення навчання та передбачення. Це рішення практично можна описати як – офлайн-навчання, онлайн-інференс. Тобто модель періодично навчається на накопичених даних в умовно пакетному режимі, а потім у реальному часі лише застосовується. Це набагато спрощує розроблюване програмне забезпечення, проте створює затримки в адаптації до змін. Таке програмне забезпечення буде вимагати механізму моніторингу якості, коли наприклад у системі присутні еталонні дані, або відгуки користувачів про правильність класифікації, на основі чого можна судити про деградацію.

Далі варто зазначити про такий підхід як онлайн навчання у потоковому конвеєрі. Це є дещо складнішим підходом, але цікавішим рішенням, а саме вбудувати навчання моделі безпосередньо в потік. Для цього підходу варто виділити два потоки, перший потік навчальних даних, другий робочих. У реальному часі отримувати навчальні дані є проблематичним завданням, тому онлайн навчання зачасту здійснюється на псевдо-навчальних даних, наприклад в задачі фільтрації спаму алгоритм може приймати початкове рішення, а згодом коли користувач позначив що лист був спамом, додаємо цей лист у категорію псевдо-навчальних даних і оновлюємо модель. Технічно реалізацію такого завдання можна зробити у Spark Streaming або окремим процесом у Python, який слухає результати класифікації та порівнює їх із правдивими даними.

Далі розглянемо буферизацію та семплінг. Іноді, аби вирішити проблему дисбалансу старих та нових даних, варто семплювати з потоку вибірку зі стріму фіксованого розміру. Це буде корисно, якщо хочемо обмежити пам'ять, але в той самий час мати представницький набір даних для періодичного перенавчання моделі.

Розглянемо метод збереження стану моделі. При розподіленій обробці важливо вирішити, де зберігається модель та як вона оновлюється. Якщо розглядаємо реалізацію на Spark Streaming, то модель може зберігатись, в об'єкті “Трансляції” для передбачень, а оновлення моделі в окремий етап, що

запускається на драйвері. У Flink є поняття керованого стану, куди можна помістити модель, і вона буде реплікована для відмовостійкості. Kafka Streams дозволяє зберігати стан локально при необхідності[13]. Одним словом, архітектура повинна гарантувати, що при збої або рестарті програмного забезпечення модель не зникне або її можна відновити.

Останнім розглянемо системи моніторингу та оповіщення, які є невід'ємною частиною промислового рішення. Необхідно спостерігати за показниками швидкості обробки, затримки, використання пам'яті та якості класифікації. Як було зазначено у публікаціях, модель потребує постійного спостереження, щоб відстежувати дрейф даних та деградацію та вчасно реагувати. У разі виявлення погіршення, повинні генеруватися сигнали для перевчення або повідомлення користувачу. Також моніторинг охоплює технічні логи – щоб впевнитись, що конвеєр Kafka-Spark працює без втрат даних і збоїв.

Отже, в сучасній практиці алгоритмічні рішення для класифікації стрімінгових текстів полягають у комбінуванні стійких до дрейфу алгоритмів (адаптивні дерева, ансамблі, Байєс, SGD) з технічними заходами підтримки (моніторинг, механізми оновлення моделей). Відомі наукові розробки, дали багато готових компонентів, якими ми можемо керуватися при розробці власного рішення. У підсумку аналізу існуючих рішень можна зробити декілька висновків. По-перше, існують надійні інфраструктурні платформи для реалізації потокової обробки (Kafka, Spark), які ми використаємо як основу. По-друге, накопичено значний набір алгоритмів для класифікації та виявлення дрейфу, які потрібно адаптувати під наші вимоги. По-третє, жоден готовий продукт повністю не відповідає завданню "з коробки", але комбінація технологій з правильними алгоритмами дозволить побудувати ефективне програмне забезпечення.

1.4 Аналіз та моделювання бізнес-процесів

У межах розробки програмного забезпечення для класифікації текстових потоків важливо визначити логіку роботи, основні функціональні етапи та взаємодію між компонентами. Розроблюване програмне забезпечення повинне забезпечити безперервний процес отримання, обробки та класифікації даних у режимі, наближеному до реального часу.

Основні бізнес-процеси, які має реалізувати програмне забезпечення:

- збір текстових потоків даних;
- попередня обробка тексту;
- навчання та перенавчання моделей класифікації текстових даних;
- класифікація текстових даних;
- зберігання та логування результатів;
- взаємодія з користувачем (інтерфейс адміністратора).

Тепер більш детально по кожному пункту. Розроблюване програмне забезпечення має отримувати текстову інформацію із різних джерел у потоковому режимі. До таких джерел належать API новинних ресурсів, соціальні мережі, месенджери та інші сервіси. Збір має здійснюватись за допомогою брокера повідомлень, найкраще із попереднього аналізу для виконання цього пункту підійде Apache Kafka, оскільки цей брокер повідомлень забезпечує масштабовану та надійну передачу повідомлень між компонентами програмного забезпечення[2]. Зібрані повідомлення мають пройти процес очистки від зайвих символів HTML-розмітки, стоп-слів, а також процес токенізації. Ці етапи є обов'язковими для забезпечення конкретної роботи моделей класифікації. Після попередньої обробки текстові дані передаватимуться до модуля класифікації. Моделі машинного навчання, які натреновані заздалегідь на відповідному наборі даних, здійснюватимуть віднесення текстів до завчасно визначених категорій. Для виконання цього пункту найкраще за попереднім аналізом підійде Apache Spark Streaming, як

найбільш актуальний та одночасно перевірений часом фреймворк для обробки в реальному часі. Класифіковані дані зберігатимуться в базу даних, або передаватимуться далі іншим мікросервісам. Також варто створити логування всіх етапів обробки для забезпечення відстеження помилок та вчасного реагування на них. Взаємодія із користувачем має відбуватись через спеціальний інтерфейс адміністратора, що дає змогу переглядати статистику оброблень повідомлень, додавати нові категорії, а також запускати та зупиняти обробку потоків. Найкраще для цього пункту підійде реалізація простого інтерфейсу за допомогою фреймворків Django або Flask.

Нижче представлено наочне представлення бізнес-процесів за допомогою нотації BPMN.

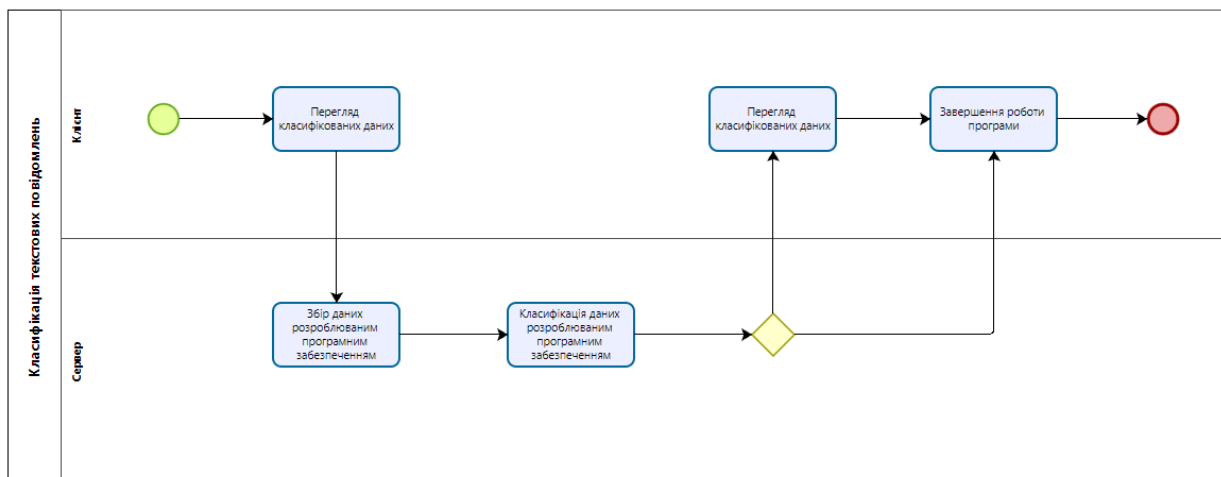


Рисунок 1.2 – BPMN діаграма розроблюваного програмного забезпечення

Висновки по розділу

У першому розділі було проведено передпроектне обстеження предметної області, що охоплює проблематику обробки великих обсягів неструктурованих текстових потоків. Було сформульовано основну мету дипломного проєктування — покращення засобів класифікації текстових потоків за рахунок створення відповідного програмного забезпечення для

автоматизованої класифікації текстових потоків у режимі, наближеному до реального часу.

Здійснено аналіз існуючих рішень у галузі класифікації текстів, зокрема програмних продуктів та алгоритмів, таких як моделі машинного навчання, засоби обробки природної мови (NLP), фреймворки потокової обробки даних: Apache Kafka, Apache Spark. На основі цього обґрунтовано необхідність розробки спеціалізованого ПЗ з урахуванням масштабованості, гнучкості та актуалізації моделей.

Також було проведено моделювання бізнес-процесів, які відображають логіку функціонування програмного забезпечення. Це дозволяє структурувати функціональні потреби до розроблюваного програмного забезпечення та окреслити його архітектурну побудову.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Основою наведеному аналізу існуючих підходів, сформулюємо вимоги до розроблюваного програмного забезпечення.

2.1 Варіанти використання програмного забезпечення

Програмне забезпечення, яке розробляється, орієнтоване на створення ефективного інструменту для аналізу даних в текстових потоках, яке надає інтуїтивно зрозумілий та гнучкий інтерфейс для роботи із ним.

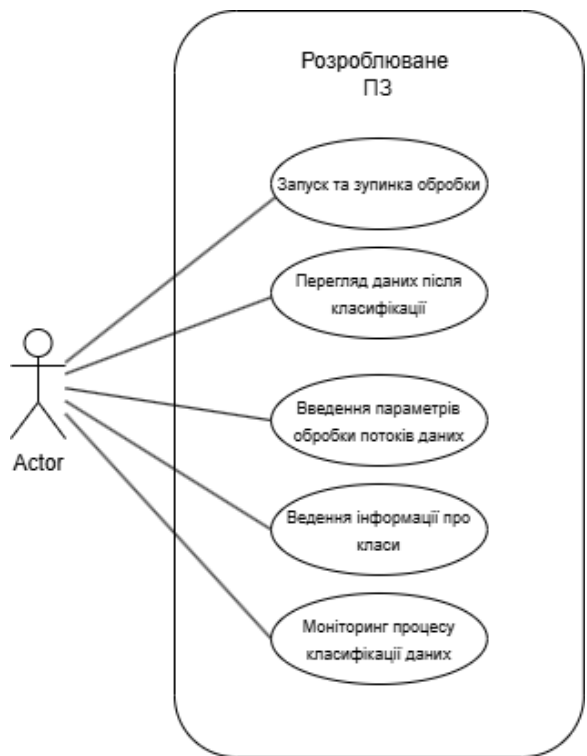


Рисунок 2.1 – Діаграма використання програмного забезпечення

У таблицях 2.1-2.5 наведено опис варіантів використання програмного забезпечення.

Таблиця 2.1 – Варіант використання UC-1

Use case name	Запуск та зупинка обробки
Use case ID	UC-01

Продовження таблиці 2.1

Goals	Надати адміністратору можливість запускати або зупиняти обробку потоку текстових повідомлень
Actors	Адміністратор
Trigger	Користувач натискає кнопку "Старт" або "Стоп" в інтерфейсі
Pre-conditions	Програмне забезпечення запущене, Kafka готовий до роботи
Flow of Events	Адміністратор входить у програмне забезпечення. Потім обирає потрібну кнопку «Старт» або «Стоп». Програмне забезпечення надсилає сигнал до обробника. Обробка зупиняється або активується
Extension	У разі помилки з Kafka виводиться повідомлення про збій
Post-condition	Обробка потоку або зупинена, або успішно триває

Таблиця 2.2 – Варіант використання UC-2

Use case name	Перегляд даних після класифікації
Use case ID	UC-02
Goals	Отримати список повідомлень з результатами класифікації
Actors	Адміністратор
Trigger	Відкриття вкладки з результатами в веб-інтерфейсі
Pre-conditions	Програмне забезпечення повинне бути з'єднана з базою даних
Flow of Events	Адміністратор відкриває інтерфейс перегляду. Розроблюване програмне забезпечення повертає список даних після класифікації

Продовження таблиці 2.2

Extension	У випадку відсутніх класифікованих даних відобразиться пуста сторінка
Post-condition	Дані виведені на екран

Таблиця 2.3 – Варіант використання UC-3

Use case name	Введення параметрів обробки потоків даних
Use case ID	UC-04
Goals	Ввести параметри для обробки потоків даних
Actors	Адміністратор
Trigger	-
Pre-conditions	Адміністратор увійшов до програмного забезпечення
Flow of Events	Користувач знаходиться сторінці запуску обробки текстових даних. Перед натисканням кнопки «Старт» користувач вводить у поля які знаходяться поряд необхідні параметри.
Extension	Виведення помилки при введені помилкових параметрів.
Post-condition	Програмне забезпечення почало обробку текстових даних за заданими параметрами.

Таблиця 2.4 – Варіант використання UC-4

Use case name	Моніторинг процесу класифікації даних
Use case ID	UC-05

Продовження таблиці 2.4

Goals	Забезпечити користувача можливістю переглядати основні показники ефективності обробки потоку даних у режимі наближеному до реального часу
Actors	Адміністратор
Trigger	Натискання кнопки «Моніторинг ресурсів»
Pre-conditions	Адміністратор увійшов до програмного забезпечення, та ввів необхідні параметри для запуску класифікації.
Flow of Events	Після старту класифікації адміністратор натискає кнопку «Моніторинг ресурсів», після чого на екрані виводиться інформація про швидкість обробки, затримки, кількість оброблених повідомлень, час роботи програми.
Extension	Виведення помилки при перевищенні встановлених порогів швидкості або помилок
Post-condition	Адміністратор отримує актуальні метрики роботи системи

Таблиця 2.5 – Варіант використання UC-5

Use case name	Введення інформації про класи
Use case ID	UC-06
Goals	Надати користувачу можливість вручну додавати, редагувати або видаляти інформацію про доступні категорії класифікації
Actors	Адміністратор
Trigger	Вибір розділу "Класи" в інтерфейсі
Pre-conditions	Адміністратор увійшов до програмного забезпечення

Продовження таблиці 2.5

Flow of Events	Користувач переходить у розділ класів. Користувач обирає необхідну дію із запропонованих, а саме: додати, редагувати, видалити. Збереження змін. Розроблюване програмне забезпечення оновлює інформацію про доступні класи.
Extension	Виведення помилок при дублюванні назв або видаленні класів що використовуються.
Post-condition	Класи оновлені в системі.

2.2 Розроблення функціональних вимог

Функціональні вимоги до розроблюваного програмного забезпечення представлені на таблиці 2.6.

Таблиця 2.6 – Функціональні вимоги до програмного забезпечення

Номер	Назва	Опис
FR-1	Старт/стоп обробки потоку	Користувач програмного забезпечення має мати можливість зупиняти або запускати обробку текстових повідомлень
FR-2	Підтвердження статусу обробки	Розроблюване програмне забезпечення має підтверджувати статус обробки даних у інтерфейсі
FR-3	Перегляд класифікованих повідомлень	Користувач має мати змогу переглядати результати класифікації
FR-4	Пошук та фільтрація результатів	Розроблюване програмне забезпечення має надавати можливість шукати і фільтрувати повідомлення за категоріями, датами, ключовими словами

Продовження таблиці 2.6

Номер	Назва	Опис
FR-5	Підтримка різних параметрів обробки даних	Користувач має мати можливість для введення власних параметрів для обробки потокових даних
FR-6	Виведення метрик класифікації	Розроблюване програмне забезпечення повинне виводити основні характеристики обробки даних
FR-7	Сповіщення при погіршенні показників	Розроблюване програмне забезпечення має повідомляти користувача про погіршення основних метрик обробки даних
FR-8	Логуювання продуктивності системи	Усі технічні показники мають автоматично зберігатись та логуватись для подальшого аналізу
FR-9	Керування класами	Користувач має мати можливість для додавання, редагування видалення класів
FR-10	Валідація введених даних	Розроблюване програмне забезпечення має валідувати введені дані користувачем

Нижче в таблиці 2.7 представлена матриця трасування вимог для розроблюваного програмного забезпечення.

Таблиця 2.7 – Матриця трасування вимог

	UC-1	UC-2	UC-3	UC-4	UC-5
FR-1	+				
FR-2	+				
FR-3		+			
FR-4		+			
FR-5			+		
FR-6				+	

Продовження таблиці 2.7

	UC-1	UC-2	UC-3	UC-4	UC-5
FR-7				+	
FR-8				+	
FR-9					+
FR-10			+		+

2.3 Розроблення нефункціональних вимог

Тепер розглянемо таблицю 2.8, в якій описані нефункціональні вимоги до розроблюваного програмного забезпечення.

Таблиця 2.8 – Нефункціональні вимоги до програмного забезпечення

Номер	Назва	Опис
NFR-1	Продуктивність	Розроблюване програмне забезпечення повинно обробляти потік повідомлень у режимі наближеному до реального часу.
NFR-2	Масштабованість	Розроблюване програмне забезпечення повинно мати можливість для горизонтального масштабування за кількістю одночасно оброблюваних потоків
NFR-3	Надійність	У разі збоїв, розроблюване програмне забезпечення має підтримувати повторну обробку після відновлення
NFR-4	Мовна універсальність	Розроблюване програмне забезпечення повинно мати можливість обробляти повідомлення які були написані українською та англійськими мовами.
NFR-5	Легкість інтеграції	Розроблюване програмне забезпечення має легко інтегруватись з базами даних, API.

Тепер розглянемо кожен пункт детально.

Під продуктивністю маємо на увазі те, що розроблюване програмне забезпечення має обробляти потік даних без затримок, а середній час обробки одного повідомлення має бути не більшим ніж 500 мс. Час обробки повідомлення є дуже важливим параметром для розроблюваного програмного забезпечення, оскільки низька затримка забезпечує високу актуальність рішення на основі класифікації.

Умова масштабованості розроблюваного програмного забезпечення дозволить нам забезпечити його коректну роботу при зростанні навантаження. Умовою для виконання цієї нефункціональної вимоги є підтримка обробки 1000+ повідомлень за секунду, з горизонтальним масштабуванням

Розроблюване програмне забезпечення має бути надійним, а саме уникнути втрати даних або збоїв при обробці. Основною метрикою для досягнення цієї вимоги буде оброблення 99.99% повідомлень без втрат, а також автоматичне відновлення розроблюваного програмного забезпечення за 1 с.

Вимога універсальності до мов класифікованих повідомлень є досить важливим аспектом для набуття актуальності нашим розроблюваним програмним забезпеченням. Метрикою для виконання цієї умови буде точність класифікації не менше ніж 85% для обох мов.

Останньою нефункційною вимогою до розроблюваного програмного забезпечення є легкість його інтеграції з зовнішніми сервісами. Умовою виконання цього пункту є підтримка REST API.

2.4 Аналіз системних вимог

Розглянемо системні вимоги для розроблюваного програмного забезпечення.

Мінімальними системними вимогами для роботи розроблюваного програмного забезпечення є:

- програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп'ютерах або серверних машинах з архітектурою x86 або x64;
- тип процесора: Intel Core i5 (8-го покоління) або AMD Ryzen 5 / аналог з багатопотоковістю;
- об'єм оперативної пам'яті — не менше 8 ГБ;
- обсяг вільного місця на жорсткому диску — не менше 100 ГБ;
- операційна система: Ubuntu 20.04 LTS / Windows 10 / macOS 11 або новіші версії;
- встановлене середовище виконання Python 3.9 або новіше, а також можливість розгортання Kafka, Spark та PostgreSQL;
- стабільне підключення до мережі Інтернет зі швидкістю від 5 Мбіт/с для забезпечення безперервного потоку повідомлень.

Вище перелічені системні вимоги основані на ресурсних потребах платформи потокової обробки Apache Spark, брокера повідомлень Apache Kafka, а також мінімальних вимогах до сучасних веб-браузерів і серверів.

2.5 Аналіз економічних показників програмного забезпечення

Для того аби оцінити економічну ефективність розроблюваного програмного забезпечення використаємо модель COCOMO, яка дозволить кількісно визначити такі характеристики проєкту, як: трудомісткість, тривалість, кількість працівників та орієнтована вартість.

У межах моделі COCOMO I використовується наступна формула для обрахунку трудомісткості розроблюваного програмного забезпечення:

$$Effort = a \cdot KLOC^b \quad (1.1)$$

Де, Effort – трудомісткість у людино-місяцях, KLOC – кількість тисяч рядків коду, а і b – емпіричні коефіцієнти, що залежать від типу проєкту.

Оскільки наше розроблюване програмне забезпечення є середньої складності, то коефіцієнт $a = 3.0$, а коефіцієнт $b = 1.12$. З цього також випливає, що кількість рядків коду у розроблюваному програмному забезпеченні буде на рівні 10 тисяч рядків.

Підставляємо значення у формулу, та отримуємо результат 39.5 людино-місяців, що показує приблизну кількість часу яка необхідна на розробку такого програмного забезпечення.

Тепер проводимо розрахунок тривалості розробки. Ця метрика обчислюється за наступною формулою:

$$Time = c \cdot Effort^d \quad (1.2)$$

Де, $Time$ – час який потрібен на розробку цього програмного забезпечення, c і d – емпіричні коефіцієнти.

Для нашого типу розроблюваного програмного забезпечення коефіцієнт $c = 2.5$, а коефіцієнт $d = 0.35$. Підставляємо значення у формулу, та проводимо розрахунки.

В результаті отримуємо, що час необхідний для розробки складає 9.05 місяців.

Тепер проведемо розрахунок необхідної кількості робітників. Розрахунки відбуваються за наступною формулою:

$$People = Effort/Time \quad (1.3)$$

Підставляємо значення отримані із попередніх формул і отримуємо, що для розробки програмного забезпечення є необхідним 5 робітників.

Ця оцінка основних економічних параметрів є теоретичною та відображає повну картину за класичною СОСОМО-моделлю без урахування повторного використання коду, сучасних фреймворків та бібліотек.

2.6 Постановка завдання на розробку програмного забезпечення

Метою розробки програмного забезпечення є створення застосунку для класифікації даних в текстових потоках в режимі наближеному до реального часу. Розроблюване програмне забезпечення має забезпечити ефективну обробку великих обсягів неструктурованої текстової інформації з можливістю адаптації до змін у даних, високу продуктивність, масштабованість та мінімальну затримку в обробці повідомлень.

Тепер опишемо основні функції яке має виконувати розроблюване програмне забезпечення. Визначено такі основні функції як:

- прийняття потоків текстових повідомлень із зовнішніх джерел у режимі наближеному до реального часу;
- попередня обробка текстових даних (очищення нормалізація, токенізація);
- здійснення класифікації кожного повідомлення відповідно до визначених категорій;
- забезпечення можливістю гнучкого управління категоріями та класами класифікації;
- зберігання результатів обробки для подальшого аналізу;
- забезпечення моніторингу стану обробки та класифікації.

Тепер опишемо основні задачі які мають бути вирішені у процесі розробки програмного забезпечення. Основні задачі:

- проектування архітектури програмного забезпечення, яка забезпечить масштабованість, стійкість до збоїв та розподілену обробку даних;
- розробка модулів збору, буферизації та попередньої обробки текстових даних;
- інтеграція алгоритмів машинного навчання для класифікації текстових даних, які орієнтовані на роботу в потоковому режимі;
- реалізація механізмів моніторингу якості обробки, які включають в себе: контроль швидкості, затримки, рівень деградації моделі;

- створення інтерфейсу адміністратора розроблюваного програмного забезпечення;
- розробка механізмів збереження та оновлення моделей машинного навчання.

Розроблюване програмне забезпечення має бути орієнтованим на реальні умови використання у середовищах з великою динамікою та обсягами даних, таких як: соціальні мережі, служби підтримки користувачів.

Успішна реалізація програмного забезпечення дозволить автоматизувати процес класифікації потоків текстових даних, підвищити оперативність обробки інформації, зменшити вплив людського фактору.

Висновки до розділу

У другому розділі було сформульовано вимоги до розроблюваного програмного забезпечення класифікації текстових даних у режимі наближеному до реального часу, визначено типові сценарії використання програмного забезпечення.

Також було сформульовані функціональні вимоги, які передбачають збирання даних, їх попередню обробку, класифікацію та інтеграцію з веб-інтерфейсом, а також і нефункціональні вимоги, які охоплюють надійність, масштабованість, адаптивність розроблюваного програмного забезпечення.

Було проведено аналіз системних вимог, які включають в себе потреби у обчислювальних ресурсах, зберігання даних, підтримку сучасних бібліотек та фреймворків. Також було здійснено попередню оцінку економічних показників розроблюваного програмного забезпечення, що свідчать про доцільність його розробки.

Таким чином у другому розділі було чітко поставлено завдання на розробку програмного забезпечення.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Для розробки програмного забезпечення для класифікації даних у текстових потоках було обрано мікросервісну архітектуру. Даний архітектурний підхід дозволить ефективно розділити компоненти системи відповідно з їх функціональним призначенням.

Розроблюване програмне забезпечення складається із наступних основних компонентів:

- Django серверна та клієнтська частини;
- Apache Kafka компонент, яка розподіляє текстові потоки;
- Apache Spark компонент, який безпосередньо займається класифікацією даних в текстових потоках;
- база даних для збереження класифікованих текстових даних.

На рис. 3.1 представлено діаграму розгортання розроблюваного програмного забезпечення.

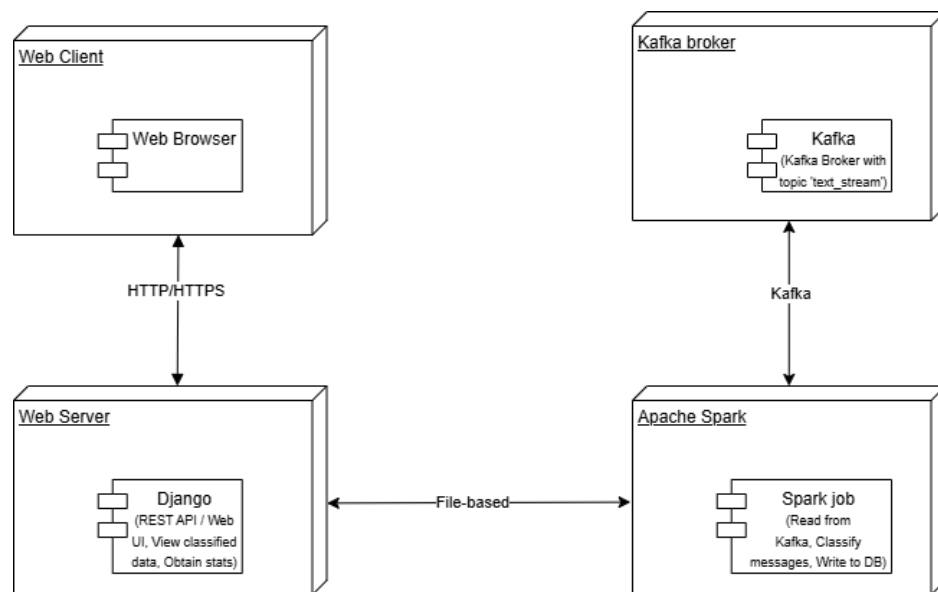


Рисунок 3.1 – Діаграма розгортання розроблюваного програмного забезпечення

3.2 Архітектурні рішення та обґрунтування вибору засобів розробки

Вибір розподіленої потокової архітектури базується на необхідності великих потоків текстових даних у режимі наближеному до реального часу, оскільки розподілена архітектура забезпечує:

- високу пропускну здатність;
- можливість горизонтального масштабування;
- відмовостійкість та безперервність роботи розроблюваного програмного забезпечення.

Також виконаємо порівняльний аналіз платформ для потокової обробки. Результати аналізу представлено на таблиці 3.1.

Таблиця 3.1 – Порівняльний аналіз платформ для потокової обробки даних

Платформа / Критерій	Apache Kafka	RabbitMQ	Apache Pulsar
Пропускна здатність	Дуже висока	Середня	Висока
Масштабованість	Відмінна	Середня	Відмінна
Затримка	Низька	Дуже низька	Низька
Складність налаштування	Середня	Низька	Висока
Можливості зберігання	Вбудовані	Обмежені	Розширені
Інтеграції з іншими системами	Відмінна	Добра	Добра

У порівнянні із альтернативами, Apache Kafka, пропонує найшвидшу пропускну здатність та зберігання історії повідомлень, що є критичним для аналізу та класифікації текстових даних.

Використання Apache Kafka як централізованого брокера повідомлень є ключовим архітектурним рішенням, яке забезпечує:

- зберігання потокових даних з можливістю повторного відтворення;
- незалежність та ізоляція виробників і споживачів текстових даних;
- гарантії доставки повідомлень навіть при збоях системи.

Виконаємо порівняльний аналіз фреймворків для розробки веб частини. Результати аналізу представлено на таблиці 3.2.

Таблиця 3.2 – Порівняльний аналіз фреймворків для веб розробки

Фреймворк / Критерій	Django	Flask	FastAPI
Функціональність “з коробки”	Висока	Низька	Середня
Масштабованість	Висока	Середня	Висока
Швидкість розробки	Висока	Середня	Висока
Документація	Відмінна	Добра	Відмінна
Інтеграція з фреймворками машинного навчання	Середня	Середня	Висока
Спільнота та підтримка	Велика	Велика	Зростаюча

У порівнянні із альтернативами, Django, надає найбільшу масштабованість, що є ледь не основним критерієм в розроблюваному програмному забезпеченні. Також важливою перевагою є висока безпека та надійність.

У розроблюваному програмному забезпеченні використовується реляційна база даних, оскільки вона забезпечує цілісність даних та ефективну роботу зі структурованими даними.

Для безпосередньої класифікації даних у режимі наближеному до реального часу обираємо Apache Spark, оскільки:

- легко інтегрується із Apache Kafka;
- має вбудовані бібліотеки машинного навчання;
- легке масштабування.

У розроблюваному програмному забезпеченні була використана бібліотека PySpark ML, яка є вбудованим компонентом Apache Spark. У порівнянні із альтернативними зовнішніми бібліотеками такими як scikit-learn або Tensorflow, PySpark ML має повну перевагу у повній інтеграції з потоковою обробкою Spark, що забезпечує ефективну роботу з великими обсягами даних у розподіленому середовищі. Також PySpark ML підтримує побудову моделі у вигляді PipelineModel, що дозволяє легко зберігати, переносити та використовувати її для класифікації повідомлень у режимі наближеному до реального часу. Основами моделей класифікації стали логістична регресія та випадковий ліс, які добре підходять для задачі багатокласового текстового аналізу.

Навчальна вибірка була попередньо підготовлена у вигляді CSV-файлу з текстовими повідомленнями та мітками класифікації. Навчальна вибірка була згенерована вручну. Було визначено основні категорії для класифікації текстових повідомлень, а саме: бізнес, розваги, політика, наука, технічні. Надалі навчальний набір даних розподілявся на тренувальну та тестові частини у співвідношенні 80% до 20% відповідно. Модель показала точність близько 99% на валідаційній вибірці, а показник F1 був в районі 0.98, що вказує на її достатню якість у контексті задачі.

3.3 Конструювання програмного забезпечення

Основою для розробки програмного забезпечення є алгоритм класифікації текстових даних, який є реалізованим за допомогою Apache Spark Streaming. Текстові повідомлення надходять у реальному часі через брокер повідомлень Apache Kafka, обробляються та проходять процес класифікації у Apache Spark. Дані після класифікації зберігаються в JSON файл. Управління обробкою, та отриманням статистики здійснюється за допомогою Django.

Для класифікації використовується модифікований підхід до роботи із об'єктами моделей Spark, коли модель тренується офлайн, а потім вже завантажується в робочий процес.

Алгоритм обробки повідомлень включає наступні етапи:

- отримання повідомлень Kafka-топіком з зовнішніх джерел;
- Spark отримує повідомлення у режимі потокової обробки;
- очищення, нормалізація, токенізація, векторизація текстових даних;
- класифікація даних на основі попередньо навченої моделі;
- збереження результату у файл формату JSON.

Для реалізації NLP-обробки та класифікації було використано бібліотеки Spark NLP, яка розширює стандартний функціонал Apache Spark можливістю роботи з природньою мовою, та PySpark ML, яка дозволяє створювати та зберігати моделі для подальшого запуску їх у потоці, а також виконувати попередню обробку текстових повідомлень. На етапі попередньої обробки тексту застосовуються стандартні компоненти Spark NLP, такі як: DocumentAssembler для формування вхідного документу, Tokenizer для розбиття повідомлення на токени, Normalizer для нормалізації, StopWordsCleaner для видалення стоп слів. Після цього токени передаються через компонент Finisher, який генерує готові текстові представлення. Після попередньої обробки текстових повідомлень застосовуємо класичні підходи до

векторизації із бібліотеки PySpark ML, такі як: CountVectorizer та IDF. Для побудови моделей використовуються логістична регресія та випадковий ліс, які дозволяють ефективно вирішувати задачу багатокласової класифікації. Отримані моделі зберігаються як PipelineModel та в подальшому використовуються в реальному часі для класифікації текстових повідомлень у режимі наближеному до реального часу.

Розроблюване програмне забезпечення отримує та зберігає дані у форматі JSON. Структуру вхідних та вихідних файлів представлено на таблицях 3.3 та 3.4.

Таблиця 3.3 – Структура вхідних даних

Назва поля	Опис
PostID	Ідентифікаційний номер повідомлення
PostText	Текст повідомлення
PostViewCount	Кількість переглядів на повідомленні
PostCreatedAt	Дата створення повідомлення
ChannellID	Ідентифікаційний номер каналу
ChannelTitle	Назва каналу, що є автором повідомлення

Таблиця 3.4 – Структура вихідних даних

Назва поля	Опис
PostID	Ідентифікаційний номер повідомлення
PostText	Текст повідомлення
tokens	Токенізоване повідомлення
ChannelTitle	Назва каналу, що є автором повідомлення
ChannelID	Ідентифікаційний номер каналу
PostViewCount	Кількість переглядів на повідомленні
PostCreatedAt	Дата створення повідомлення

Продовження таблиці 3.4

Назва поля	Опис
ProcessedAt	Дата класифікації повідомлення
Category	Категорія до якої було класифіковано повідомлення
Status	Статус обробки повідомлення
Processing_level	Статус попередньої обробки повідомлення

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено у таблиці 3.5.

Таблиця 3.5 – Опис утиліт

№	Назва утиліти	Опис застосування
1	PyCharm	Головне середовище розробки програмного забезпечення.
2	Postman	Програмне забезпечення необхідне для тестування REST запитів.
3	Docker	Платформа для контейнеризації програмних компонентів
4	Django	Веб-фреймворк для клієнтської та серверної частини
5	Apache Spark	Фреймворк для розподіленої обробки даних
6	Apache Kafka	Платформа для обробки поточкових даних

Тексти програмного коду наведені в окремому документі «Текст програми».

3.4 Аналіз безпеки даних

Під час розробки програмного забезпечення для класифікації даних у текстових потоках було проведено аналіз безпеки з урахуванням специфіки обробки потенційно конфіденційних текстових даних.

Оскільки розроблюване програмне забезпечення розгортається в контейнеризованому середовищі, впроваджено наступні заходи безпеки:

- використання мінімальних базових образів;
- реалізація принципу найменших привілеїв для контейнерів;
- сканування образів на наявність вразливостей.

Висновки до розділу

У розділі детально описано процес конструювання та розробки програмного забезпечення для класифікації даних у текстових потоках в режимі наближеному до реального часу за допомогою Apache Kafka, Apache Spark, Django. Також було представлено архітектуру програмного забезпечення з використанням мікросервісного підходу.

Також було обґрунтовано вибір технологічного стеку, з порівняльним аналізом альтернативних технологій, який підтверджує правильність обраних рішень.

Було проведено аналіз безпеки даних, розглянуто управління вразливостями, тестування безпеки та захисту контейнеризованих середовищ.

Під час розробки використовувалися такі інструменти як PyCharm, Postman, DataGrip, Docker, Django, Apache Spark, Apache Kafka. Вказано їх призначення та роль у процесі розробки програмного забезпечення.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Метою тестування є наступне:

- перевірка коректності роботи розробленого програмного забезпечення відповідно до функціональних вимог;
- перевірка правильності збереження даних;
- перевірка зручності користувацького інтерфейсу.

4.2 Опис процесів тестування

Тестування виконується згідно документу «Програма та методика тестування».

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 4.1-4.7.

Таблиця 4.1 – Тест 1.1 Запуск обробки

Початковий стан системи	Користувач знаходиться на основній сторінці
Вхідні дані	-
Опис проведення тесту	Натискається кнопка “Start Stream” для початку процесу класифікації.
Очікуваний результат	Процес класифікації запущено. На екрані змінився статус обробки.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.2 – Тест 1.2 Налаштування максимальної довжини повідомлення

Початковий стан системи	Користувач знаходиться на основній сторінці
Вхідні дані	-
Опис проведення тесту	У відповідне поле вводиться значення максимальної довжини повідомлення, яке буде підлягати класифікації.
Очікуваний результат	Введене число було збережено у відповідний конфіг файл. На екрані виведено нове встановлене обмеження.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.3 – Тест 1.3 Налаштування пропускної здатності програмного забезпечення

Початковий стан системи	Користувач знаходиться на основній сторінці
Вхідні дані	-
Опис проведення тесту	У відповідне поле вводиться значення очікуваної пропускної здатності програмного забезпечення.
Очікуваний результат	Введене число було збережено у відповідний конфігураційний файл. На екрані виведено нове встановлене обмеження.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.4 – Тест 1.4 Налаштування кольору категорії

Початковий стан системи	Користувач знаходиться на сторінці налаштування категорій.
Вхідні дані	Наявні категорії.
Опис проведення тесту	Натискається кнопка “Edit” поряд із бажаною категорією. Натискається поле “Color” та обирає бажаний колір для цієї категорії.
Очікуваний результат	Колір категорії змінено.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.5 – Тест 1.5 Налаштування опису категорії

Початковий стан системи	Користувач знаходиться на сторінці налаштування категорій.
Вхідні дані	Наявні категорії.
Опис проведення тесту	Натискається кнопка “Edit” поряд із бажаною категорією. Натискається поле “Description” та вводить бажаний опис категорії.
Очікуваний результат	Опис категорії змінено.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.6 – Тест 1.6 Пошук класифікованого повідомлення за категорією

Початковий стан системи	Користувач знаходиться на сторінці пошуку класифікованих повідомлень.
-------------------------	---

Продовження таблиці 4.6

Вхідні дані	Наявність класифікованих повідомлень.
Опис проведення тесту	Відкривається випадаючий список із наявними категоріями, та обирається необхідна. Після цього натискається кнопка “Пошук”.
Очікуваний результат	Усі повідомлення з шуканої категорії виведені на екран.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.7 – Тест 1.7 Пошук класифікованого повідомлення за текстом

Початковий стан системи	Користувач знаходиться на сторінці пошуку класифікованих повідомлень.
Вхідні дані	Наявність класифікованих повідомлень.
Опис проведення тесту	У відповідне поле пошуку вводиться текст повідомлення, або його частина. Після цього натискається кнопка “Пошук”.
Очікуваний результат	Виведено усі повідомлення які відповідають шуканим параметрам.
Фактичний результат	Збігається з очікуваним.

4.3 Опис контрольного прикладу

Для опису контрольного прикладу відкриваємо головну сторінку застосунку та натискаємо кнопку «Start Stream».

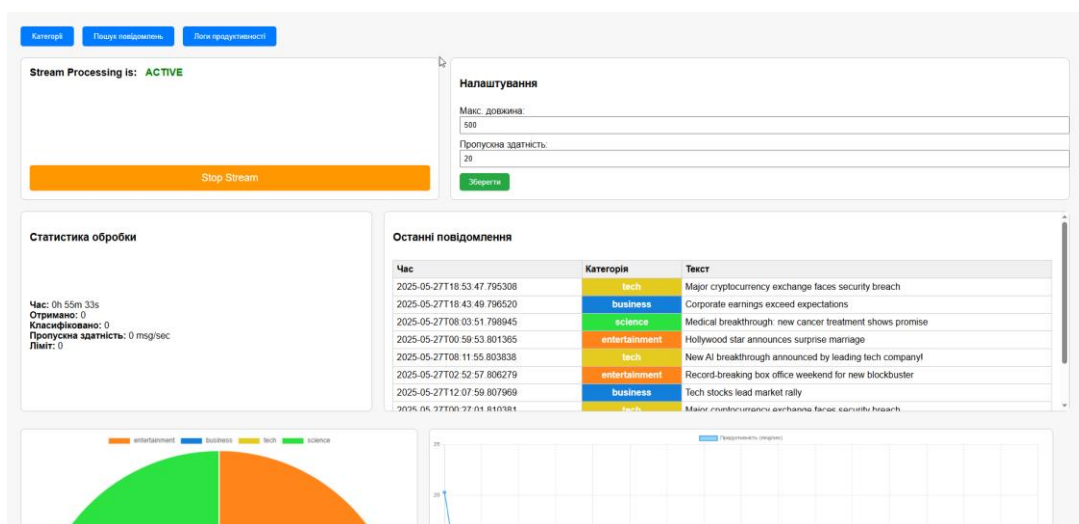


Рисунок 4.1 – Запуск процесу класифікації повідомлень

Надалі вводимо параметри налаштування обробника. Поля було заповнено такими даними:

- максимальна довжина: 500;
- пропускна здатність: 1.

Натискаємо кнопку «Зберегти». Результат введення параметрів налаштувань обробника представлено на рис. 4.2.

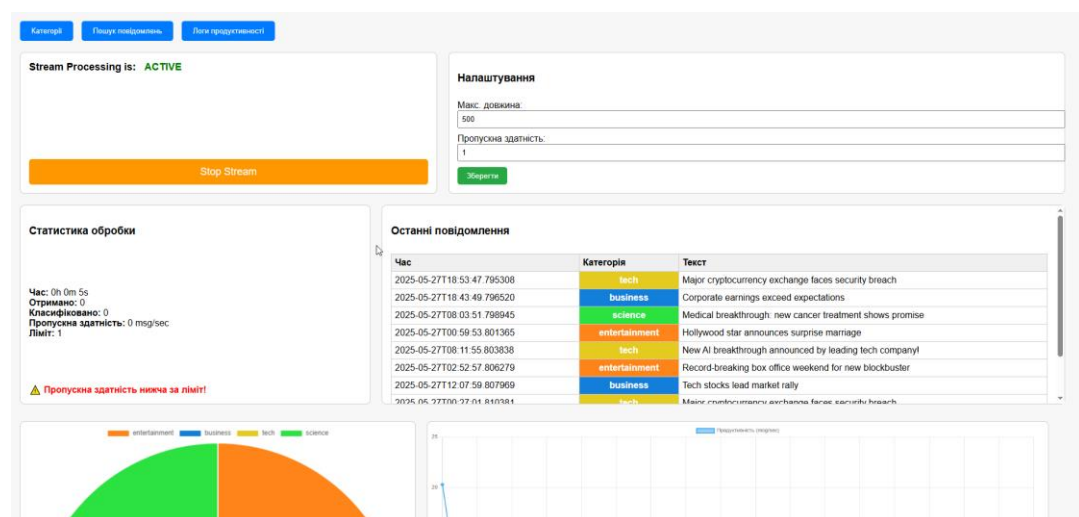


Рисунок 4.2 – Налаштування параметрів процесу класифікації

Переходимо до вкладки «Категорії». Та змінюємо колір і опис у категорії «tech» шляхом натискання на кнопку «Edit» поряд. Категорія до змін та у процесі редагування представлена на рис. 4.3.

[← Back to Stream Control](#)

Налаштування категорій

ID	Label	Description	Color	Actions
0	business	Business-related news and updates, crypto	#1478db	Edit Delete
1	entertainment	Movies, music, celebrities	#ff351b	Edit Delete
2	politics	Government, policies, and current events	#ff4136	Edit Delete
3	science	Scientific discoveries and advancements	#2ecc71	Edit Delete
4	tech	Technology trends and innovations	#e67e22	Edit Delete

Додати або оновити категорію

ID:

Description:

Color:

[Зберегти](#)

Рисунок 4.3 – Категорія в процесі редагування

Змінюємо колір на фіолетовий та зберігаємо зміни. Категорія після змін представлена на рис. 4.4.

[← Back to Stream Control](#)

Налаштування категорій

ID	Label	Description	Color	Actions
0	business	Business-related news and updates, crypto	#1478db	Edit Delete
1	entertainment	Movies, music, celebrities	#ff351b	Edit Delete
2	politics	Government, policies, and current events	#ff4136	Edit Delete
3	science	Scientific discoveries and advancements	#2ecc71	Edit Delete
4	tech	Technology trends and innovations	#8e44ad	Edit Delete

Додати або оновити категорію

ID:

Description:

Color:

[Зберегти](#)

Рисунок 4.4 – Відредагована категорія

Повертаємось до головної сторінки та переходимо до вкладки «Пошук повідомлень». Натискаємо на випадаючий список, та обираємо необхідну категорію. Натискаємо кнопку пошук. Результат пошуку класифікованих повідомлень за певною категорією представлено на рис. 4.5.

← Back to Stream Control

Пошук класифікованих повідомлень

Пошук за текстом: business

Час	Категорія	Текст	Канал	Перегляди
2025-05-25T12:35:46.504549	business	Diplomatic relations between nations improve	Global Politics	815
2025-05-26T00:06:48.506990	business	Global supply chain disruptions continue	Business Daily	1257
2025-05-25T05:59:50.509404	business	International trade tensions escalate	Global Politics	397
2025-05-25T16:41:54.514718	business	Retail sector faces challenging quarter	Business Daily	819
2025-05-25T06:53:56.517159	business	UN Security Council holds emergency meeting	Global Politics	621
2025-05-25T14:30:58.520056	business	Global supply chain disruptions continue	Business Daily	1014
2025-05-25T12:53:00.522479	business	UN Security Council holds emergency meeting	Global Politics	341
2025-05-25T07:47:08.532909	business	Celebrity chef opens restaurant chain	Entertainment Buzz	2848
2025-05-25T10:37:10.535363	business	Tech stocks lead market rally	Business Daily	1483
2025-05-25T06:58:12.537737	business	Election results surprise political analysts	Global Politics	1029
2025-05-25T04:07:18.546128	business	Celebrity chef opens restaurant chain	Entertainment Buzz	183
2025-05-25T13:26:26.553884	business	Retail sector faces challenging quarter	Business Daily	651
2025-05-26T08:25:47.216749	business	International trade tensions escalate	Global Politics	1009
2025-05-26T01:25:49.219164	business	Corporate earnings exceed expectations	Business Daily	1721
2025-05-25T10:51:59.231201	business	Diplomatic relations between nations improve	Global Politics	725
2025-05-26T01:42:01.234108	business	Retail sector faces challenging quarter	Business Daily	1372

Рисунок 4.5 – Результат пошуку класифікованих повідомлень

Повертаємось до головної сторінки та переходимо до вкладки «Лог продуктивності».

← Back to Stream Control

Історія продуктивності системи

Час	Отримано	Класифіковано	Пропускна здатність	Ліміт	Аптайм (сек)	Статус
2025-05-28T01:14:30.904949	38	38	0.5	1	78	SLOW
2025-05-28T01:14:00.896941	23	23	0.5	1	48	SLOW
2025-05-28T01:13:31.025339	8	8	0.5	1	18	SLOW
2025-05-28T01:13:14.978807	0	0	0	1	2	SLOW
2025-05-28T00:15:30.828109	52	52	0.1	20	130	SLOW
2025-05-28T00:15:00.934555	49	49	0.5	20	100	SLOW
2025-05-28T00:14:30.943927	34	34	0.5	20	70	SLOW
2025-05-28T00:14:00.929287	19	19	0.5	20	40	SLOW
2025-05-28T00:13:31.217556	4	4	0.52	20	10	SLOW
2025-05-28T00:13:23.509323	0	0	0	1	2	SLOW

Рисунок 4.6 – Журнал роботи програми

Висновки до розділу

У розділі проведено аналіз якості програмного забезпечення та його тестування. Основними задачами тестування були: перевірка виконання функційних вимог, виявлення та усунення помилок, збереження даних.

Було проведено мануальне тестування розробленого програмного забезпечення, результати якого наведені в таблицях 4.1 – 4.7. В кожному тесті було описано початковий стан системи, вхідні дані, процес тестування, очікуваний та фактичний результати. Усі тести завершилися успішно, що підтверджує правильність розробленого програмного забезпечення.

Було описано контрольний приклад, в якому було висвітлено основний функціонал з усіма можливими розгалуженнями.

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Для розгортання розробленого програмного забезпечення було обрано платформу Render. Цей сервіс поєднує в собі легкість розгортання програмного забезпечення, а також дозволяє мати повний контроль над використаними ресурсами.

Після реєстрації на платформі, потрапляємо на головну сторінку, де пропонується створити один із сервісів, які наведені на рис. 5.1. У відповідності до розробленого програмного забезпечення, необхідно створити три сервіси. Перший для Apache Kafka для отримання повідомлень, другий для Apache Spark для попередньої обробки та класифікації повідомлень, третій для Django частини розробленого програмного забезпечення, яка відповідає за керування процесом класифікації, вивід статистики та результатів.

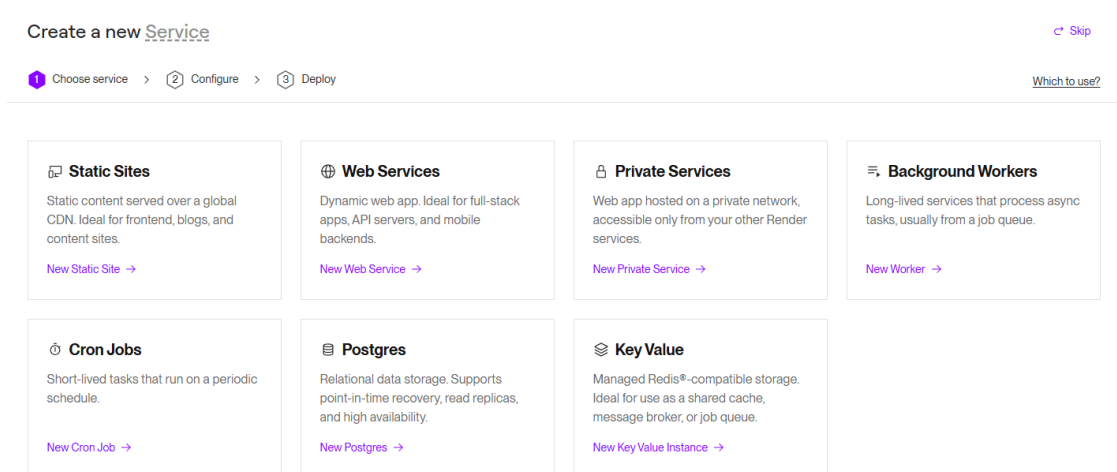


Рисунок 5.1 – Головна сторінка із запропонованими сервісами

Оскільки розроблене програмне забезпечення розроблювалось з самого початку за допомогою Docker, це сильно полегшує задачу розгортання.

Для початку створюємо сервіс для Kafka-брокеру повідомлень. Для успішного розгортання сервісу, необхідно додатково прописати шлях до

конкретного Docker файлу, та виконати розгортання. Розгортання сервісу для Kafka представлено на рис. 5.2.

You are deploying a Background Worker
 You seem to be using **Docker**, so we've autofilled some fields accordingly. Make sure the values look right to you!

Source Code degvor / text-msg-classifier · 38m ago Edit

Name
 A unique name for your background worker.
 text-msg-classifier-kafka

Project Optional
 Add this background worker to a [project](#) once it's created.

Create a new project to add this to?

You don't have any projects in this workspace. [Projects](#) allow you to group resources into environments so you can better manage related resources.

[+ Create a project](#)

Language
 Docker

Branch
 The Git branch to build and deploy.
 main

Рисунок 5.2 – Створення сервісу для Kafka брокеру повідомлень

Наступним кроком створимо сервіс для Spark обробника повідомлень. Так само як і з попереднім сервісом додатково прописуємо шлях до Docker файлу який призначений для цього сервісу. Розгортання сервісу представлено на рис. 5.3.

You are deploying a Background Worker
 You seem to be using **Docker**, so we've autofilled some fields accordingly. Make sure the values look right to you!

Source Code degvor / text-msg-classifier · 1h ago Edit

Name
 A unique name for your background worker.
 text-msg-classifier-spark

Project Optional
 Add this background worker to a [project](#) once it's created.

Create a new project to add this to?

You don't have any projects in this workspace. [Projects](#) allow you to group resources into environments so you can better manage related resources.

[+ Create a project](#)

Language
 Docker

Branch
 The Git branch to build and deploy.
 main

Рисунок 5.3 – Створення сервісу для Spark обробника повідомлень

Останнім кроком розгортаємо наш веб сервіс Django, який відповідає за керування процесу класифікації, вивід результатів та статистики.

You are deploying a Web Service

You seem to be using **Docker**, so we've autofilled some fields accordingly. Make sure the values look right to you!

Source Code degvor / text-msg-classifier · 1h ago Edit

Name
A unique name for your web service.

Project Optional
Add this web service to a [project](#) once it's created.

Create a new project to add this to?

You don't have any projects in this workspace. [Projects](#) allow you to group resources into environments so you can better manage related resources.

[+ Create a project](#)

Language

Branch
The Git branch to build and deploy.

Рисунок 5.4 – Створення веб частини застосунку

На рис. 5.5 представлений кінцевий вигляд розроблюваного програмного забезпечення після розгортання.

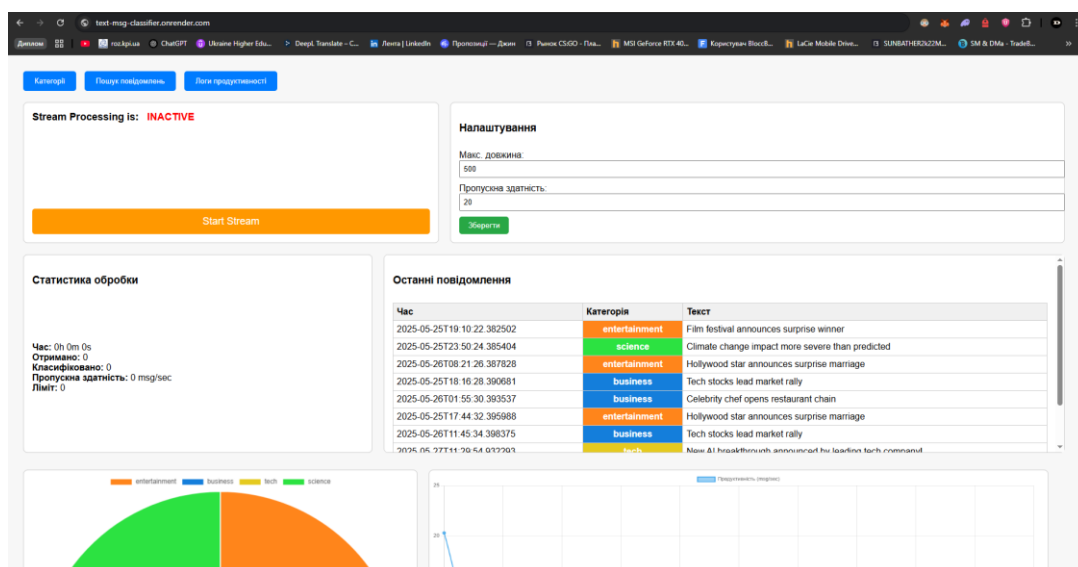


Рисунок 5.5 – Результат розгортання розробленого програмного забезпечення

5.2 Супровід програмного забезпечення

Інструкція користувача наведена в окремому документі «Керівництво Користувача».

Користувачам потрібна буде надана можливість отримання доступу до останньої версії застосунку з кожним його оновленням. Для виконання цієї задачі використовується функція Auto-Deploy сервісу Render. Тепер, з використанням цієї функції, після оновлення основної гілки репозиторію на Github, усі сервіси розгортаються заново. Таким чином після додавання нової версії застосунку, користувач одразу отримає доступ до актуальних сервісів.

Висновки до розділу

У розділі було розглянуто переваги розгортання розробленого програмного забезпечення за допомогою сервісу Render. Було висвітлено ключові аспекти, які роблять цей сервіс зручним і ефективним інструментом для розгортання. Зокрема, було зазначено, що Render забезпечує простий і зрозумілий інтерфейс, що дозволяє швидко налаштовувати необхідні параметри для запуску застосунку.

Окрему увагу приділено опису процесу створення сервісів, які необхідні для коректного розгортання застосунку.

Крім того, було розглянуто і вирішено одну із важливих проблем, що виникає при оновленні застосунку – забезпечення користувачів актуальною версією. У розділі наведено налаштування сервісу Render, яке дозволяє мінімізувати час між внесенням зміни до проєкту та їх появою у користувача.

ВИСНОВКИ

В результаті розробки програмного забезпечення було виконано проєктування застосунку для класифікації текстових повідомлень.

На етапі аналізу предметної області було визначено основні виклики, пов'язані з потоковою класифікацією, а саме: висока швидкість надходження даних, різноманітність форматів і мов, обмеження в пам'яті, потреба в попередній обробці тексту. Це зумовило необхідність створення архітектури здатної підтримувати гнучке масштабування, високу продуктивність.

Було розглянуто та проаналізовано наявні варіанти вирішення поставленого завдання. В результаті аналізу було визначено, що найкращою комбінацією буде Apache Kafka, у ролі брокера повідомлень, Apache Spark для попередньої обробки та подальшої класифікації текстових повідомлень, а також Django для створення інтерфейсу керування Spark-обробником, виводу основних метрик та результатів процесу класифікації.

У рамках проєкту було обрано мікросервісну архітектуру, що дозволила ефективно розділити логіку системи на окремі незалежні компоненти. Архітектура включає:

- компонент збору текстових потоків за допомогою Apache Kafka;
- класифікатор на базі Spark Streaming;
- клієнтська та серверна частини на Django.

Було реалізовано підхід до класифікації, в якому модель машинного навчання тренується офлайн на історичних даних із подальшим збереженням у вигляді PipelineModel. У навчальному пайплайні використовуються модулі попередньої обробки тексту з бібліотеки Spark NLP, а також компоненти векторизації та класифікатор з бібліотеки PySpark ML. Класифікацію текстових повідомлень було реалізовано за допомогою таких методів, як: логістична регресія та випадковий ліс. Точність класифікації текстових повідомлень складає 99%.

Цей підхід дозволив досягнути балансу між швидкістю, точністю та можливістю повторного використання моделі в режимі наближеному до реального часу.

Було розроблено програмне забезпечення з наступними можливостями:

- зупинка та запуск процесу класифікації текстових повідомлень;
- налаштування категорій класифікації;
- отримувати основні метрики процесу класифікації в режимі наближеному до реального часу;
- зручний пошук по повідомленнях після процесу класифікації по тексту повідомлення або по категорії;
- логування продуктивності системи.

Окрему увагу було приділено формалізації структури вхідних та вихідних даних. На вхід надходять повідомлення які містять такі атрибути, як: ідентифікатор, текст, кількість переглядів, дата створення та канал надходження. Результатом обробки є розширена структура, як включає в себе усі атрибути із вхідних даних, а також токенізований текст, категорію класифікації, статус обробки та час класифікації. Такий формат забезпечує універсальність обробки та зручний подальший аналіз.

Під час тестування розробленого програмного забезпечення було виявлено, що воно відповідає усім поставленим функціональним та нефункціональним вимогам.

Розроблене програмне забезпечення було успішно розгорнуто за допомогою сервісу Render, а також була вирішена проблема отримання оновлень застосунку користувачами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) Sabare V. O. Building a data pipeline for natural language processing with Apache Kafka and Apache Spark [Електронний ресурс] – Режим доступу: <https://medium.com/@sabarevictor/building-a-data-pipeline-for-natural-language-processing-with-apache-kafka-and-apache-spark-58a302699c5b>
- 2) Velotio Technologies. Real-Time Text Classification Using Kafka and Scikit-learn [Електронний ресурс] – Режим доступу: <https://medium.com/velotio-perspectives/real-time-text-classification-using-kafka-and-scikit-learn-c2875ad80b3c>
- 3) Garcia C. M. et al. Concept Drift Adaptation in Text Stream Mining Settings: A Comprehensive Review [Електронний ресурс] – Режим доступу: <https://arxiv.org/abs/2312.02901>
- 4) Tennant M. et al. Scalable real-time classification of data streams with concept drift [Електронний ресурс] – Режим доступу: <https://doi.org/10.1016/j.future.2017.03.014>
- 5) Bifet A. et al. MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering [Електронний ресурс] – Режим доступу: <http://proceedings.mlr.press/v11/bifet10a/bifet10a.pdf>
- 6) Komolafe A. Retraining Model During Deployment: Continuous Training and Continuous Testing [Електронний ресурс] – Режим доступу: <https://neptune.ai/blog/retraining-model-during-deployment-continuous-training-continuous-testing>
- 7) Amazon Web Services. What's the Difference Between Kafka and Spark? [Електронний ресурс] – Режим доступу: <https://aws.amazon.com/compare/the-difference-between-kafka-and-spark/>

8) Kwon H.-Y. et al. Distributed Classification Model of Streaming Tweets based on Dynamic Model Updates [Електронний ресурс] – Режим доступу: <https://ieeexplore.ieee.org/document/9689997>

9) Ю.О. Олійник, О.Є. Афанасьєва, Г.Д.Аршакян Підхід до виявлення аномалій в потоках текстових даних [Електронний ресурс] – Режим доступу: <https://doi.org/10.34185/1562-9945-2-127-2020-10>

10) Ю.О. Олійник Система аналізу текстових потоків даних [Електронний ресурс] – Режим доступу: <https://journals.kntu.net.ua/index.php/aqmm/article/view/524>

11) Franklin J. Apache Kafka Use Cases: When To Use It? When Not To? [Електронний ресурс] – Режим доступу: <https://www.upsolver.com/blog/apache-kafka-use-cases-when-to-use-not>

12) What is Apache Spark? [Електронний ресурс] – Режим доступу: <https://www.ibm.com/think/topics/apache-spark>

13) What is Apache Flink? [Електронний ресурс] – Режим доступу: <https://www.confluent.io/learn/apache-flink/>

14) Kurama V. PyTorch vs. TensorFlow: Key Differences to Know for Deep Learning [Електронний ресурс] – Режим доступу: <https://builtin.com/data-science/pytorch-vs-tensorflow#:~:text=PyTorch%20is%20a%20relatively%20young,for%20high%2DIevel%20model%20development.>

15) What Is Splunk & What Does It Do? A Splunk Intro [Електронний ресурс] – Режим доступу: https://www.splunk.com/en_us/blog/learn/what-splunk-does.html

16) Generative and discriminative classifiers: Native Bayes and logistic regression [Електронний ресурс] – Режим доступу: <https://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf>

17) A Beginner's Guide to Hoeffding Tree with Python Implementation [Электронный ресурс] – Режим доступа: <https://medium.com/we-talk-data/a-beginners-guide-to-hoeffding-tree-with-python-implementation-64bb90b5a91c>

18) Wang J. Model drift detection [Электронный ресурс] – Режим доступа: <https://medium.com/@j.wang.mlds/model-drift-detection-100a35a5edfa>

19) Parti A. Understanding Model Drift In Machine Learning [Электронный ресурс] – Режим доступа: <https://pareto.ai/blog/model-drift-in-machine-learning>

20) Bifet A. and Gavalda R. Learning from Time-Changing Data with Adaptive Windowing [Электронный ресурс] – Режим доступа: <https://epubs.siam.org/doi/10.1137/1.9781611972771.42>

ДОДАТОК А ЗВІТ ПОДІБНОСТІ



Дата звіту 6/2/2025
Дата редагування 6/2/2025

Документ прийнятий

Звіт подібності

метадані

Назва організації

National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute

Заголовок

ІП-11_Ющенко_ПЗ

Автор

Науковий керівник / Експерт

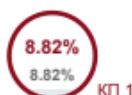
ІП-11_ЮщенкоОлійник Ю.О.

підрозділ

ФІОТ, К-а інформатики та програмної інженерії

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



10

Довжина фрази для коефіцієнта подібності 2

8830

Кількість слів

69624

Кількість символів

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2025 р.

**ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ КЛАСИФІКАЦІЇ ТЕКСТОВИХ
ПОВІДОМЛЕНЬ**

Текст програми

КПІ.ПІ-1130.045440.03.12

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Юрій ОЛІЙНИК

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Андрій ЮЩЕНКО

Київ – 2025

Посилання на репозиторій з повним текстом програмного коду

<https://github.com/degvor/text-msg-classifier>

Файл `spark_kafka_consumer.py`

Реалізація функціональної задачі логування продуктивності системи:

`@staticmethod`

```
def write_to_json_file(batch_df, batch_id):
    data = batch_df.select(to_json(col("classified_data"))).rdd.map(lambda r: r[0]).collect()
    with open("/data/all_messages.json", "a", encoding="utf-8") as f:
        for json_line in data:
            f.write(json_line + "\n")

    with open("/data/stream_metrics.json", "r", encoding="utf-8") as f:
        metrics = json.load(f)

    with open("/data/config.json", "r", encoding="utf-8") as f:
        config = json.load(f)

    throughput_limit = config.get("throughput_limit", 1)
    now = datetime.datetime.utcnow()
    prev_time = datetime.datetime.fromisoformat(metrics.get("last_update", datetime.datetime.utcnow()))
    elapsed = (now - prev_time).total_seconds() if prev_time else 0
    current_throughput = len(data) / elapsed if elapsed > 0 else 0

    metrics["current_throughput"] = round(current_throughput, 2)
    metrics["throughput_limit"] = throughput_limit
    metrics["status"] = "ok" if current_throughput >= throughput_limit else "slow"

    metrics['received_count'] += len(data)
    metrics['classified_count'] += len(data)
    metrics['last_update'] = datetime.datetime.utcnow().isoformat()

    metrics["uptime_seconds"] = int(
        (now - datetime.datetime.fromisoformat(metrics["start_time"])).total_seconds())

    with open("/data/stream_metrics.json", "w", encoding="utf-8") as f:
        json.dump(metrics, f, indent=2)

    log_entry = {
```

```

        "timestamp": now.isoformat(),
        "received": metrics["received_count"],
        "classified": metrics["classified_count"],
        "throughput": metrics["current_throughput"],
        "limit": metrics["throughput_limit"],
        "uptime": metrics["uptime_seconds"],
        "status": metrics["status"]
    }

    log_history = []

    with open("/data/performance_logs.json", "r", encoding="utf-8") as f:
        log_history = json.load(f)

    log_history.append(log_entry)

    with open("/data/performance_logs.json", "w", encoding="utf-8") as f:
        json.dump(log_history, f, indent=2)

```

Файл views.py

Реалізація функціональної задачі старту та зупинки обробки:

```

def toggle_stream(request):
    control = StreamStatus.objects.first()
    control.is_active = not control.is_active
    control.save()
    return redirect("stream_status")

```

Реалізація функціональної задачі підтвердження статусу обробки:

```

def stream_status(request):
    control = StreamStatus.objects.first()
    return render(request, "stream_status.html", {"control": control})

```

Реалізація функціональної задачі перегляду класифікованих повідомлень:

```

def latest_messages_json(request):
    messages_path = os.path.join(settings.BASE_DIR, 'data', 'all_messages.json')
    categories_path = os.path.join(settings.BASE_DIR, 'data/models', 'label_mapping.json')
    meta_path = os.path.join(settings.BASE_DIR, 'data', 'category_metadata.json')

    messages = []
    categories = {}

```

```

meta_map = {}
if os.path.exists(meta_path):
    with open(meta_path, 'r', encoding='utf-8') as f:
        meta_map = json.load(f)

try:
    with open(messages_path, 'r', encoding='utf-8') as file:
        for line in file:
            line = line.strip()
            if line:
                try:
                    message = json.loads(line)
                    messages.append(message)
                except json.JSONDecodeError:
                    continue

except FileNotFoundError:
    return JsonResponse({"error": "Файл не знайдено."}, status=404)
except Exception as e:
    return JsonResponse({"error": f"Помилка зчитування файлу: {str(e)}"}, status=500)

try:
    with open(categories_path, 'r', encoding='utf-8') as file:
        categories = json.load(file)
except FileNotFoundError:
    return JsonResponse({"error": "Файл з категоріями не знайдено."}, status=404)
except json.JSONDecodeError:
    return JsonResponse({"error": "Формат категорій некоректний."}, status=500)

latest_10 = messages[-10:]

for msg in latest_10:
    category_id = str(msg.get("category"))
    msg["category_label"] = categories.get(category_id, "unknown")
    msg["category_color"] = meta_map.get(category_id, {}).get("color", "#cccccc")

return JsonResponse({"messages": latest_10}, safe=False)

```

Реалізація функціональної задачі пошуку та фільтрації результатів класифікації:

```

def search_classified_messages(request):
    query = request.GET.get("q", "").lower()
    category = request.GET.get("category")

    file_path = os.path.join(settings.BASE_DIR, 'data', 'all_messages.json')
    labels_path = os.path.join(settings.BASE_DIR, 'data/models', 'label_mapping.json')

    try:
        with open(labels_path, "r", encoding="utf-8") as f:
            label_map = json.load(f)
    except:
        label_map = {}

    results = []

    try:
        with open(file_path, 'r', encoding='utf-8') as f:
            for line in f:
                line = line.strip()
                if not line:
                    continue
                try:
                    msg = json.loads(line)
                    text = msg.get("PostText", "").lower()
                    cat_id = str(msg.get("category"))
                    if query and query not in text:
                        continue
                    if category and category != cat_id:
                        continue
                    msg["category_label"] = label_map.get(cat_id, "unknown")
                    results.append(msg)
                except json.JSONDecodeError:
                    continue
    except FileNotFoundError:
        return JsonResponse({"messages": []})

    return JsonResponse({"messages": results[-100:]})

```

Реалізація функціональної задачі підтримки різних параметрів обробки даних:

```

@csrf_exempt
def update_config(request):
    if request.method != "POST":

```

```

return JsonResponse({"error": "POST only"}, status=405)

try:
    config_path = os.path.join(settings.BASE_DIR, "data", "config.json")
    data = json.loads(request.body)
    config = {
        "max_message_length": int(data.get("max_message_length", 1000)),
        "throughput_limit": int(data.get("throughput_limit", 1))
    }
    with open(config_path, "w", encoding="utf-8") as f:
        json.dump(config, f, indent=2)
    return JsonResponse({"status": "saved"})
except Exception as e:
    return JsonResponse({"error": str(e)}, status=500)

```

Реалізація функціональної задачі виведення метрик класифікації:

```

def stream_metrics_json(request):
    metrics_path = os.path.join(settings.BASE_DIR, 'data', 'stream_metrics.json')

    with open(metrics_path, 'r', encoding='utf-8') as file:
        metrics = json.load(file)

    control = StreamStatus.objects.first()
    is_active = control.is_active if control else False

    if is_active:
        start_time_str = metrics.get("last_start_time") or metrics.get("start_time")
        if start_time_str:
            try:
                start_time = datetime.fromisoformat(start_time_str)
                uptime = int((datetime.utcnow() - start_time).total_seconds())
            except:
                uptime = 0
        else:
            uptime = 0
    else:
        uptime = metrics.get("uptime_seconds", 0)

    return JsonResponse({
        "received_count": metrics.get("received_count", 0),
        "classified_count": metrics.get("classified_count", 0),
        "uptime_seconds": uptime,
    })

```

```

    "current_throughput": metrics.get("current_throughput", 0),
    "throughput_limit": metrics.get("throughput_limit", 0),
    "status": metrics.get("status", "unknown")
})

```

Реалізація функціональної задачі керування класами:

```

def get_category_data():
    labels_path = os.path.join(settings.BASE_DIR, 'data/models', 'label_mapping.json')
    meta_path = os.path.join(settings.BASE_DIR, 'data', 'category_metadata.json')

    with open(labels_path, 'r', encoding='utf-8') as f:
        labels = json.load(f)

    try:
        with open(meta_path, 'r', encoding='utf-8') as f:
            metadata = json.load(f)
    except FileNotFoundError:
        metadata = {}

    result = []
    for cat_id, label in labels.items():
        meta = metadata.get(cat_id, {})
        result.append({
            "id": cat_id,
            "label": label,
            "description": meta.get("description", ""),
            "color": meta.get("color", "#cccccc")
        })

    return result, metadata

def get_categories(request):
    try:
        categories, _ = get_category_data()
        return JsonResponse({"categories": categories})
    except Exception as e:
        return JsonResponse({"error": str(e)}, status=500)

@csrf_exempt
def update_category_metadata(request):

```

```

if request.method not in ['POST', 'PUT']:
    return JsonResponse({"error": "Only POST or PUT allowed"}, status=405)

meta_path = os.path.join(settings.BASE_DIR, 'data', 'category_metadata.json')

try:
    data = json.loads(request.body)
    cat_id = str(data["id"])
    description = data.get("description", "")
    color = data.get("color", "#cccccc")

    _, metadata = get_category_data()
    metadata[cat_id] = {
        "description": description,
        "color": color
    }

    with open(meta_path, 'w', encoding='utf-8') as f:
        json.dump(metadata, f, indent=2, ensure_ascii=False)

    return JsonResponse({"message": "Metadata updated", "category": metadata[cat_id]})
except Exception as e:
    return JsonResponse({"error": str(e)}, status=500)

@csrf_exempt
def delete_category_metadata(request, cat_id):
    if request.method != 'DELETE':
        return JsonResponse({"error": "Only DELETE allowed"}, status=405)

    meta_path = os.path.join(settings.BASE_DIR, 'data', 'category_metadata.json')

    try:
        _, metadata = get_category_data()
        if cat_id in metadata:
            deleted = metadata.pop(cat_id)
            with open(meta_path, 'w', encoding='utf-8') as f:
                json.dump(metadata, f, indent=2, ensure_ascii=False)
            return JsonResponse({"message": f"Metadata for {cat_id} deleted", "deleted": deleted})
        else:
            return JsonResponse({"error": "Metadata not found"}, status=404)

```

except Exception as e:

```
return JsonResponse({"error": str(e)}, status=500)
```

Файл `stream_status.html`

Реалізація функціональної задачі сповіщення при погіршенні показників:

```
function fetchStreamMetrics() {
  fetch("% url 'stream_metrics_json' %}")
    .then(res => res.json())
    .then(data => {
      document.getElementById("uptime").textContent = formatUptime(data.uptime_seconds);
      document.getElementById("received").textContent = data.received_count;
      document.getElementById("classified").textContent = data.classified_count;
      document.getElementById("throughput").textContent = data.current_throughput;
      document.getElementById("limit").textContent = data.throughput_limit;

      const alertBox = document.getElementById("alert-box");
      alertBox.textContent = data.status === "slow" ? "Пропускна здатність нижча за ліміт!" : "";
    });
}
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ” _____ 2025 р.

**ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ КЛАСИФІКАЦІЇ ТЕКСТОВИХ
ПОВІДОМЛЕНЬ**

Програма та методика тестування

КПІ.ПІ-1130.045440.04.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Юрій ОЛІЙНИК

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Андрій ЮЩЕНКО

Київ – 2025

ЗМІСТ

1	ОБ’ЄКТ ВИПРОБУВАНЬ	3
2	МЕТА ТЕСТУВАННЯ.....	4
3	МЕТОДИ ТЕСТУВАННЯ	5
4	ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ	6

1 ОБ'ЄКТ ВИПРОБУВАНЬ

Об'єктом випробування є тестування коректності роботи програмного забезпечення, а саме: класифікація повідомлень, збереження даних, попередня обробка текстових повідомлень, обробка помилок.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

- перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
- перевірка збереження даних;
- знаходження проблем, помилок і недоліків з метою їх усунення.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

- статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;
- функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікувань;
- мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Під час проведення тестування будуть використовуватись наступні допоміжні засоби:

- Postman для тестування API запитів;
- Codacy для статичного аналізу коду.

Порядок проведення тестування буде наступним:

- статичний аналіз коду;
- тестування на виведення повідомлень про помилку, коли це необхідно;
- функціональне тестування на відповідність функціональним вимогам.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ” _____ 2025 р.

**ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ КЛАСИФІКАЦІЇ ТЕКСТОВИХ
ПОВІДОМЛЕНЬ**

Керівництво користувача

КПІ.ПІ-1130.045440.05.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Юрій ОЛІЙНИК

Нормоконтроль:

_____ Катерина ЛЩУК

Виконавець:

_____ Андрій ЮЩЕНКО

Київ – 2025

ЗМІСТ

1	ПРИЗНАЧЕННЯ ПРОГРАМИ	3
2	ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ	4
2.1	Системні вимоги для коректної роботи	4
2.2	Завантаження застосунку.....	4
2.3	Перевірка коректної роботи.....	4
3	ВИКОНАННЯ ПРОГРАМИ.....	5

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

«TextClassifier» – це програмне забезпечення для автоматичної класифікації текстових повідомлень та моніторингу процесу класифікації. Також користувачу доступна функція зміни опису категорій класифікації, і зручний інтерфейс.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Мінімальна конфігурація технічних засобів:

- тип процесору: Intel Core i5;
- об'єм ОЗП: 8 Гб;
- підключення до мережі Інтернет зі швидкістю від 5 Мбіт/с.

Рекомендована конфігурація технічних засобів:

- тип процесору: Intel Core i7;
- об'єм ОЗП: 32 Гб;
- підключення до мережі Інтернет зі швидкістю від 1000 мегабіт.

2.2 Завантаження застосунку

Наразі вебзастосунок розгорнутий за адресою <https://txt-msg-classifier.onrender.com>. Для цього необхідно відкрити дане посилання в будь-якому браузері на комп'ютері.

2.3 Перевірка коректної роботи

По завершенню завантаження програмного забезпечення у вікні браузера має відобразитись головна сторінка для роботи із програмним забезпеченням.

3 ВИКОНАННЯ ПРОГРАМИ

При відкритті програмного забезпечення користувач направляється на основну сторінку керуванням процесом класифікації текстових повідомлень (рис. 3.1)

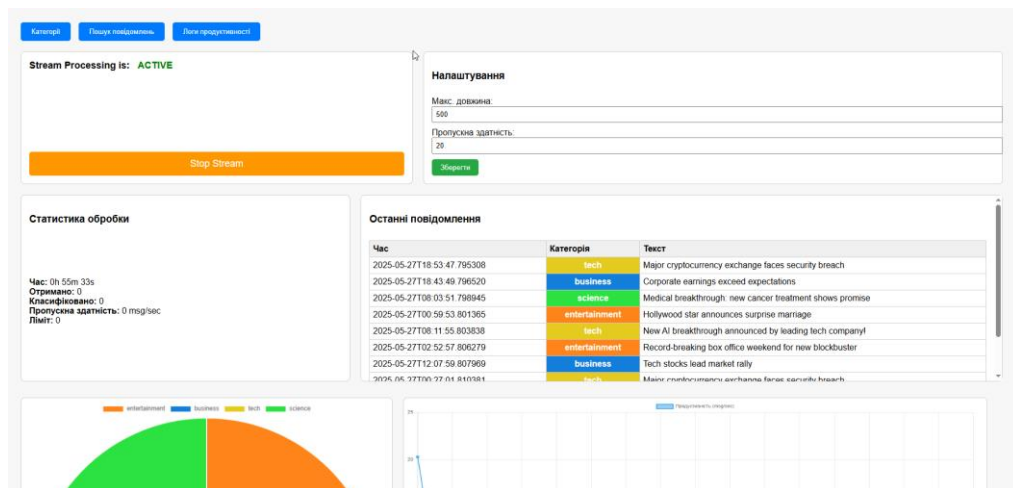


Рисунок 3.1 – Головна сторінка застосунку

Після потрапляння на головну сторінку застосунку користувач може ввести бажані параметри для обробки повідомлень та натисну кнопку «Start Stream» для початку обробки (рис. 3.2).

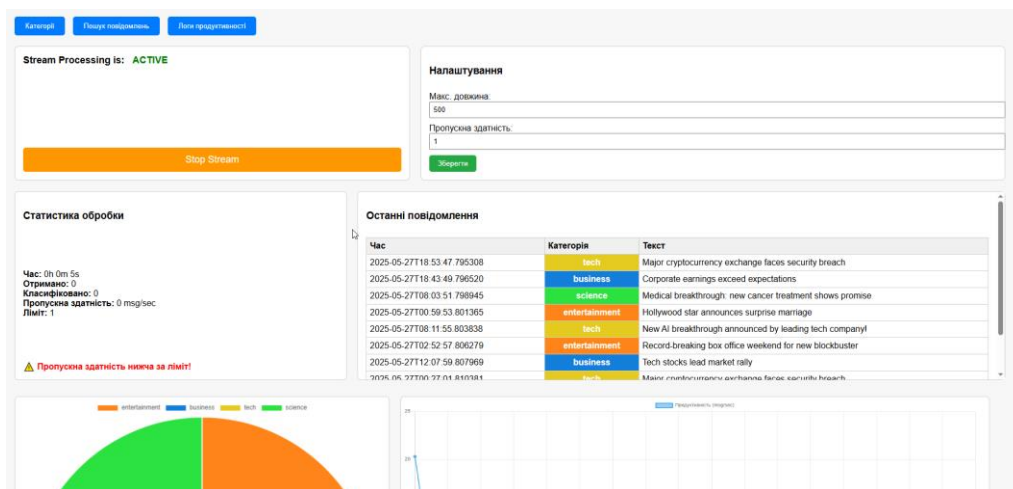


Рисунок 3.2 – Введення параметрів обробки повідомлень

Після початку обробки повідомлень користувач переходить у налаштування категорій (рис. 3.3). Тут він може змінити опис, назву та колір категорії, який буде відображатись у табличках та у круговій діаграмі.

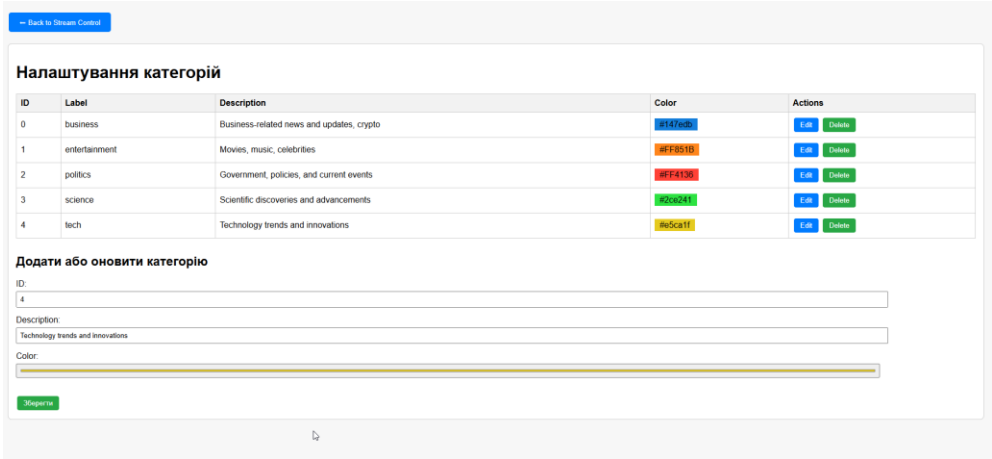


Рисунок 3.3 – Сторінка налаштування категорій

Після цього користувач повертається до головної сторінки і звідти переходить на сторінку класифікованих повідомлень (рис. 3.4). Тут користувач може почати пошук по тексту повідомлення, або за категорією.

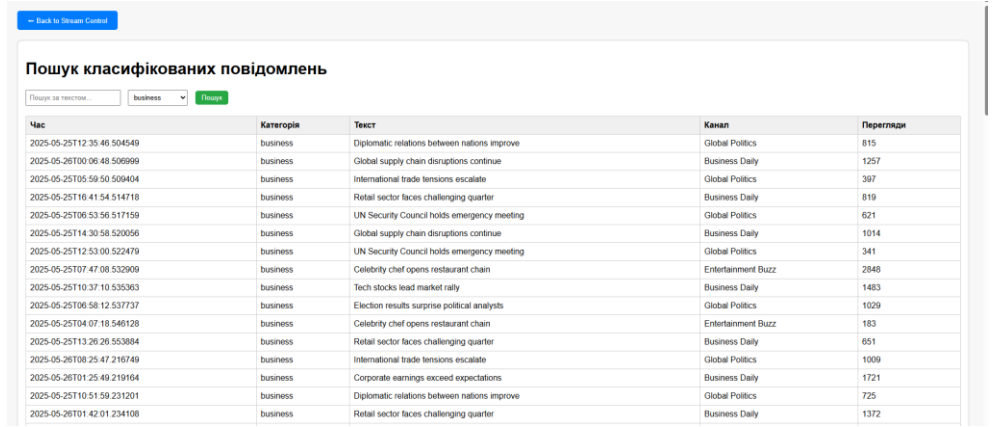


Рисунок 3.4 – Сторінка класифікованих повідомлень

Після цього користувач повертається до головної сторінки і переходить на сторінку трекінгу продуктивності системи (рис. 3.5). На цій сторінці користувачу показано у вигляді таблиці продуктивність роботи розроблюваного програмного забезпечення.

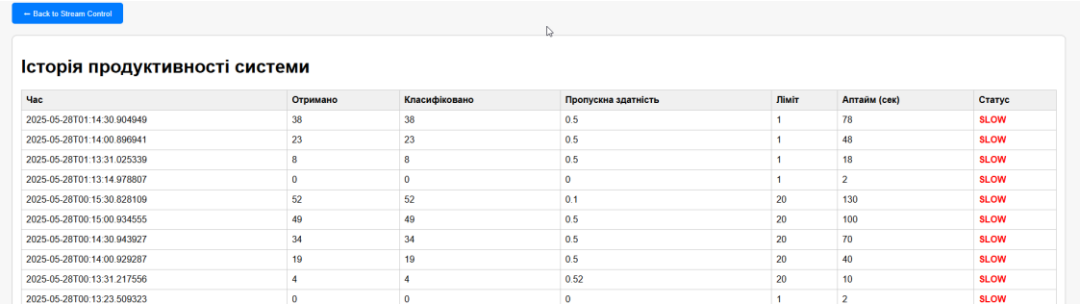


Рисунок 3.5 – Сторінка продуктивності роботи програмного забезпечення.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“ ____ ” _____ 2025 р.

**ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ КЛАСИФІКАЦІЇ ТЕКСТОВИХ
ПОВІДОМЛЕНЬ**

Графічний матеріал

КПІ.ПІ-1130.045440.06.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Юрій ОЛІЙНИК

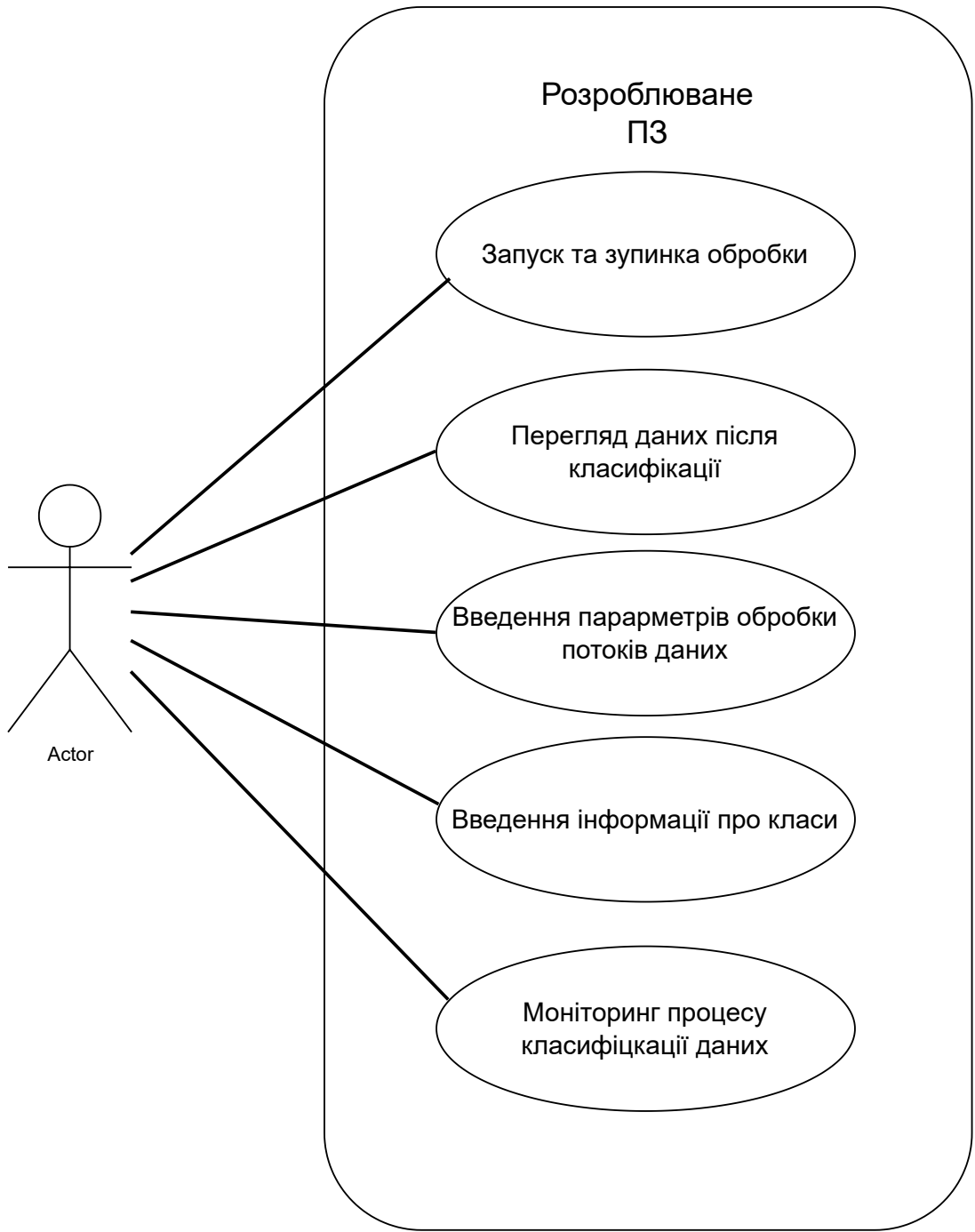
Нормоконтроль:

_____ Катерина ЛЩУК

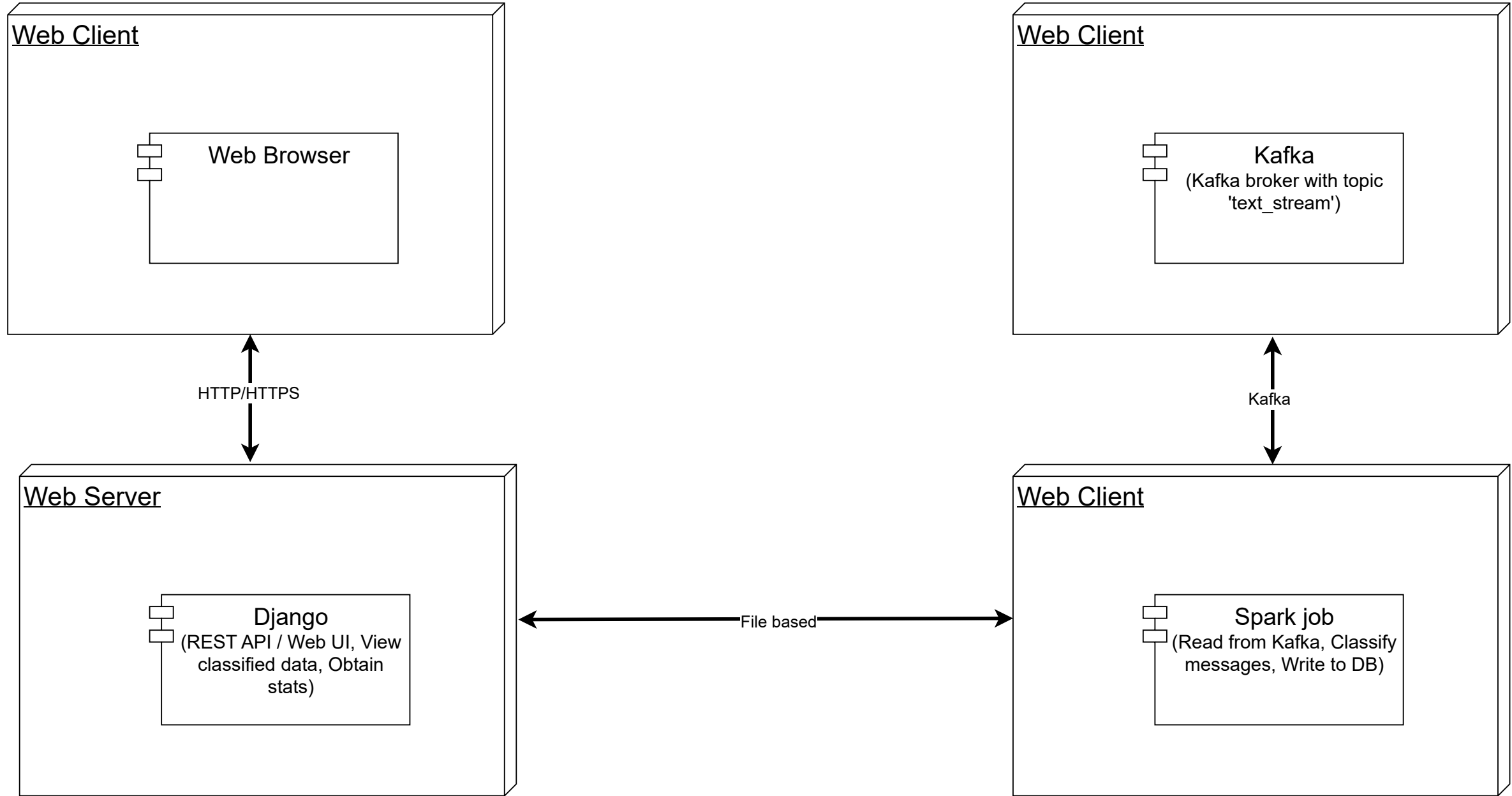
Виконавець:

_____ Андрій ЮЩЕНКО

Київ – 2025



					КПІ.ІП-1130.045440.06.99.CCB						
					Схема структурна варіантів використання	Літера			Маса	Масштаб	
Зм.	Арк.	№ документа	Підпис	Дата							
Розробив		Ющенко А.В.									
Перевішив		Олійник Ю.О.									
Т. контр.						Аркуш 1			Аркушів 3		
					Програмне забезпечення для класифікації текстових повідомлень	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-11					
Н. контр.		Ліщук К.І.									
Затвердив		Жаріков Е.В.									



					КПІ.ІП-1130.045440.06.99.CCM									
					Схема структурна компонентів програмного забезпечення	Літера			Маса		Масштаб			
Зм.	Арк.	№ документа	Підпис	Дата										
Розробив		Ющенко А.В.												
Перевірів		Олійник Ю.О.												
Т. контр.														
					Програмне забезпечення для класифікації текстових повідомлень	Аркуш 2			Аркушів 3					
Н. контр.		Ліщук К.І.				КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-11								
Затвердив		Жаріков Е.В.												

[← Back to Stream Control](#)

Пошук класифікованих повідомлень

business Пошук

Час	Категорія	Текст	Канал	Перегляди
2025-05-25T12:35:46.504549	business	Diplomatic relations between nations improve	Global Politics	815
2025-05-26T00:06:48.506999	business	Global supply chain disruptions continue	Business Daily	1257
2025-05-25T05:59:50.509404	business	International trade tensions escalate	Global Politics	397
2025-05-25T16:41:54.514718	business	Retail sector faces challenging quarter	Business Daily	819
2025-05-25T06:53:56.517159	business	UN Security Council holds emergency meeting	Global Politics	621
2025-05-25T14:30:58.520056	business	Global supply chain disruptions continue	Business Daily	1014
2025-05-25T12:53:00.522479	business	UN Security Council holds emergency meeting	Global Politics	341
2025-05-25T07:47:08.532909	business	Celebrity chef opens restaurant chain	Entertainment Buzz	2848
2025-05-25T10:37:10.535363	business	Tech stocks lead market rally	Business Daily	1483
2025-05-25T06:58:12.537737	business	Election results surprise political analysts	Global Politics	1029
2025-05-25T04:07:18.546128	business	Celebrity chef opens restaurant chain	Entertainment Buzz	183
2025-05-25T13:26:26.553884	business	Retail sector faces challenging quarter	Business Daily	651
2025-05-26T06:25:47.216749	business	International trade tensions escalate	Global Politics	1009
2025-05-26T01:25:49.219164	business	Corporate earnings exceed expectations	Business Daily	1721
2025-05-25T10:51:59.231201	business	Diplomatic relations between nations improve	Global Politics	725
2025-05-26T01:42:01.234108	business	Retail sector faces challenging quarter	Business Daily	1372

Категорії

Пошук повідомлень

Логі продуктивності

Stream Processing is: **ACTIVE**

Stop Stream

Налаштування

Макс. довжина: 500

Пропускна здатність: 20

Зберегти

Статистика обробки

Час: 0h 55m 33s

Отримано: 0

Класифіковано: 0

Пропускна здатність: 0 msg/sec

Ліміт: 0

Останні повідомлення

Час	Категорія	Текст
2025-05-27T18:53:47.795308	tech	Major cryptocurrency exchange faces security breach
2025-05-27T18:43:49.796520	business	Corporate earnings exceed expectations
2025-05-27T08:03:51.798945	science	Medical breakthrough: new cancer treatment shows promise
2025-05-27T00:59:53.801365	entertainment	Hollywood star announces surprise marriage
2025-05-27T08:11:55.803838	tech	New AI breakthrough announced by leading tech company
2025-05-27T02:52:57.806279	entertainment	Record-breaking box office weekend for new blockbuster
2025-05-27T12:07:59.807969	business	Tech stocks lead market rally
2025-05-27T00:22:01.810181	tech	Major cybersecurity vulnerabilities discovered

entertainment

business

tech

science

Topic trends (top100)

[← Back to Stream Control](#)

Історія продуктивності системи

Час	Отримано	Класифіковано	Пропускна здатність	Ліміт	Алтайм (сек)	Статус
2025-05-28T01:14:30.904949	38	38	0.5	1	78	SLOW
2025-05-28T01:14:00.896941	23	23	0.5	1	48	SLOW
2025-05-28T01:13:31.025339	8	8	0.5	1	18	SLOW
2025-05-28T01:13:14.978807	0	0	0	1	2	SLOW
2025-05-28T00:15:30.828109	52	52	0.1	20	130	SLOW
2025-05-28T00:15:00.934555	49	49	0.5	20	100	SLOW
2025-05-28T00:14:30.943627	34	34	0.5	20	70	SLOW
2025-05-28T00:14:00.929287	19	19	0.5	20	40	SLOW
2025-05-28T00:13:31.217596	4	4	0.52	20	10	SLOW
2025-05-28T00:13:23.509323	0	0	0	1	2	SLOW

[← Back to Stream Control](#)

Налаштування категорій

ID	Label	Description	Color	Actions
0	business	Business-related news and updates, crypto	#147ed6	Видалити

| 1 | entertainment | Movies, music, celebrities | #ff851b | Видалити |

| 2 | politics | Government, policies, and current events | #ff4136 | Видалити |

Додати або оновити категорію

ID: 4

Description: Technology trends and innovations

Color:

Зберегти

					КПІ.ІП-1130.045440.06.99.КЕ				
					Креслення виду екранних форм				
Зм.	Арк.	№ документа	Підпис	Дата					
Розробив		Ющенко А.В.			Програмне забезпечення для класифікації текстових повідомлень				
Перевірів		Олійник Ю.О.							
Т. контр.									
Н. контр.		Ліщук К.І.			КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-11				
Затвердив		Жаріков Е.В.							