МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп’ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

КАРАВЕЦЬ Роман Олексійович

Методи аналізу настроїв в соціальних мережах на основі технологій великих даних / Sentiment Analysis Methods in Social Networks Using Big Data Technologies

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп’ютерні науки

Кваліфікаційна робота

Виконав студент групи КНм-21
Р.О. Каравець
_____

Науковий керівник:
к.т.н., доцент І.В. Турченко
_____

Кваліфікаційну роботу
допущено до захисту:
«___» _____ 20___ р.
В.о. завідувача кафедри
_____ Н.В. Дзюбановська

ТЕРНОПІЛЬ – 2025

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «магістр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.М. Васильків
«____» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
КАРАВЦЮ Роману Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи

Методи аналізу настроїв в соціальних мережах на основі технологій великих даних / Sentiment Analysis Methods in Social Networks Using Big Data Technologies

керівник роботи к.т.н., доцент І.В. Турченко
затверджені наказами по університету від 20 грудня 2024 року № 938 та від 14 жовтня 2025 року № 724.

2. Строк подання студентом закінченої кваліфікаційної роботи 1 грудня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити

– проаналізувати предметну область аналізу настроїв у соціальних мережах;

– виконати порівняльний аналіз методів та засобів аналізу настроїв у текстових повідомленнях;

– провести аналіз відомих рішень для аналізу настроїв в соціальних мережах;

– сформулювати постановку задачі дослідження;

– розробити метод аналізу настроїв у соціальних мережах на основі технологій великих даних;

– розробити алгоритми витягування, трансформації та завантаження даних у потоковому режимі;

– розробити алгоритм класифікації повідомлень;

– розробити метод агрегації результатів класифікації;

– спроєктувати архітектуру та реалізувати програмні модулі для аналізу настроїв у соціальних мережах;

– провести експериментальні дослідження запропонованих рішень і виконати оцінювання їх продуктивності.

5. Перелік графічного матеріалу у роботі

– узагальнена схема методу аналізу настроїв у потоці соціальних повідомлень.

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 20 грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи.	до 01.01. 2025 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.03. 2025 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 20.05.2025 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 28.10. 2025 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 11.11.2025 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів.	до 25.11.2025 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту.	до 1.12.2025 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 04.12.2025 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії.	до 14.12. 2025 р.	

Студент _____ Р.О. Каравець
підпис

Керівник роботи _____ к.т.н., доцент І.В. Турченко
підпис

РЕЗЮМЕ

Кваліфікаційна робота на тему «Методи аналізу настроїв в соціальних мережах на основі технологій великих даних» на здобуття освітнього ступеня «Магістр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 78 сторінок і містить 1 ілюстрацію, 8 таблиць, 2 додатки та 82 використаних джерела.

Метою кваліфікаційної роботи є підвищення ефективності аналізу настроїв у соціальних мережах за рахунок розроблення потокового конвеєра обробки даних на основі технологій великих даних, який забезпечує безперервне витягування, трансформацію та завантаження повідомлень у режимі, наближеному до режиму реального часу, з подальшою класифікацією тональності та картографічною візуалізацією результатів.

Методи дослідження включають аналіз і синтез, системний підхід, порівняльний аналіз, методи математичної статистики та експериментального дослідження, методи машинного навчання для класифікації тональності, а також методи програмної інженерії для проєктування та реалізації програмного фреймворку.

Результати дослідження: удосконалено метод агрегації результатів класифікації тональності шляхом запровадження формалізованого індексу настрою та системи багатовимірних агрегувань, що забезпечує перехід від міток тональності окремих повідомлень до узагальнених показників настрою інформаційного потоку, придатних для стабільного моніторингу та порівняльного аналізу різних сегментів даних у середовищі великих даних.

Результати роботи можуть успішно застосовуватися для оперативного моніторингу настроїв у соціальних мережах, виявлення змін емоційного фону за заданими темами, мовами та часовими інтервалами.

Ключові слова: АНАЛІЗ НАСТРОЇВ, СОЦІАЛЬНІ МЕРЕЖІ, ПОТОКОВА ОБРОБКА ДАНИХ, ВЕЛИКІ ДАНІ, МАШИННЕ НАВЧАННЯ, АГРЕГАЦІЯ РЕЗУЛЬТАТІВ.

ABSTRACT

Qualification work on the topic «Sentiment Analysis Methods in Social Networks Using Big Data Technologies» for Master's degree on speciality 122 «Computer Science» educational and professional program «Computer Science» is written on 78 pages and it contains 1 figure, 8 tables, 2 annexes and 82 sources.

The purpose of the qualification thesis is to improve the efficiency of sentiment analysis in social networks by developing a streaming data processing pipeline based on big data technologies, which provides continuous extraction, transformation, and loading of messages in a near–real-time mode, followed by sentiment polarity classification and cartographic visualization of the results.

The research methods include analysis and synthesis, a systems approach, comparative analysis, methods of mathematical statistics and experimental research, machine learning methods for sentiment classification, as well as software engineering methods for designing and implementing the software framework.

Research results: the method for aggregating sentiment classification outputs has been improved by introducing a formalized sentiment index and a system of multidimensional aggregations, enabling the transition from sentiment labels of individual messages to generalized indicators of the sentiment of the information stream, suitable for stable monitoring and comparative analysis of different data segments in a big data environment.

The results of the work can be successfully applied for operational monitoring of sentiments in social networks and for detecting changes in the emotional background for specified topics, languages, and time intervals.

Keywords: SENTIMENT ANALYSIS, SOCIAL NETWORKS, STREAMING DATA PROCESSING, BIG DATA, MACHINE LEARNING, RESULTS AGGREGATION.

ЗМІСТ

ВСТУП

Актуальність роботи. У сучасних умовах соціальні мережі стали одним із ключових індикаторів суспільних реакцій, оскільки користувачі щоденно формують великий обсяг коротких повідомлень, у яких відображаються оцінки, емоції та ставлення до подій, продуктів і рішень. У межах даної роботи поняття аналіз настроїв у соціальних мережах трактується як прикладне завдання моніторингу емоційного фону інформаційного потоку, а аналіз тональності розглядається як базовий інструмент його формалізації. Це означає, що "настрій" потоку повідомлень у роботі описується через класифікацію полярності тексту на позитивну, нейтральну та негативну категорії, що дозволяє отримувати узагальнені висновки про домінування певних реакцій, відстежувати їх динаміку та порівнювати різні сегменти даних. Такий підхід є практично доцільним, оскільки тональність виступає універсальним, відтворюваним показником, який можна ефективно обчислювати для великих потоків повідомлень у режимі реального часу.

У сучасну добу соціальні мережі перетворилися на потужні джерела даних у режимі реального часу, які надають цінну інформацію для різних сфер, зокрема охорони здоров'я та бізнесу. Аналіз тональності (sentiment analysis, SA), що є ключовим елементом дослідження реакцій та настроїв користувачів на таких платформах, відіграє важливу роль у забезпеченні можливості для організацій ухвалювати обґрунтовані управлінські рішення [32]. Завдання автоматизованого аналізу тональності даних із соціальних мереж є складним через значний обсяг інформації та потребу в її обробці в режимі реального часу. Витягування та опрацювання даних у такому динамічному й швидкозмінному середовищі потребує застосування ефективних систем потокової обробки даних [45].

Традиційні процедури Extract–Transform–Load (ETL), які широко застосовуються для керування великими масивами даних і їх обробки, виявляються недостатньо придатними для сценаріїв потокової обробки в режимі реального часу, що істотно обмежує можливості їх використання в такому контексті [47]. Для подолання труднощів, пов'язаних з опрацюванням великих

обсягів даних, нині дедалі більшої ваги набувають технології обробки великих даних, зокрема Spark, Hive, Kafka, HBase, Hadoop HDFS та Cassandra [4, 6, 7, 39, 62, 63]. На відміну від класичного ETL, що працює зі статичними наборами даних і спирається на пакетну обробку, потоковий ETL спеціально розроблений для обробки безперервних потоків, таких як дані в режимі реального часу, зокрема соціальних мереж. Забезпечуючи безперервне й майже миттєве виконання етапів витягування, трансформації та завантаження даних, потоковий ETL долає обмеження, які виникають у випадку динамічних і швидкозмінних потоків.

У контексті середовищ обробки великих даних кількість робіт, що безпосередньо стосуються проєктування та реалізації потокового ETL, залишається обмеженою. Водночас більшість наявних досліджень зосереджуються на аналізі тональності повідомлень без явного зв’язку з компонентами потокового ETL [12, 58, 60, 71, 73, 75]. Окрему групу становлять роботи, у яких реалізовано потокову обробку даних для задач аналізу тональності повідомлень соціальних мереж [25, 26], де однією з ключових технологій виступає Spark. Натомість інша група досліджень використовує Hadoop в якості базової платформи [22, 54, 57]. Таким чином, актуальним є поєднання методів інтелектуального аналізу тексту з потоковою інфраструктурою великих даних, що забезпечує систематичне витягування, перетворення, збереження та інтерпретацію повідомлень із соціальних мереж для задач аналізу настроїв у режимі, наближеному до режиму реального часу.

Мета і завдання дослідження. Метою роботи є підвищення ефективності аналізу настроїв у соціальних мережах за рахунок розроблення потокового конвеєра обробки даних на основі технологій великих даних, який забезпечує безперервне витягування, трансформацію та завантаження повідомлень у режимі, наближеному до режиму реального часу, з подальшою класифікацією тональності та агрегацію результатів.

Для досягнення визначеної мети необхідно виконати ряд завдань:

– проаналізувати предметну область аналізу настроїв у соціальних мережах;

– виконати порівняльний аналіз методів та засобів аналізу настроїв у текстових повідомленнях;

– провести аналіз відомих рішень для аналізу настроїв в соціальних мережах;

– сформулювати постановку задачі дослідження;

– розробити метод аналізу настроїв у соціальних мережах на основі технологій великих даних;

– розробити алгоритми витягування, трансформації та завантаження даних у потоковому режимі;

– розробити алгоритм класифікації повідомлень;

– розробити метод агрегації результатів класифікації;

– спроєктувати архітектуру та реалізувати програмні модулі для аналізу настроїв у соціальних мережах;

– провести експериментальні дослідження запропонованих рішень і виконати оцінювання їх продуктивності.

Об’єкт дослідження – процес аналізу настроїв користувачів у соціальних мережах на основі потокових текстових повідомлень у середовищі великих даних.

Предмет дослідження – методи, алгоритми та програмні засоби потокового витягування, трансформації, класифікації текстових повідомлень соціальних мереж та агрегації результатів з використанням технологій великих даних.

Методи дослідження включають аналіз і синтез для узагальнення підходів до аналізу настроїв у соціальних мережах, системний підхід для проєктування потокового конвеєра обробки даних у середовищі великих даних, методи порівняльного аналізу для зіставлення відомих рішень і технологій, методи математичної статистики та експериментального дослідження для оцінювання продуктивності запропонованих рішень, методи машинного навчання для класифікації тональності текстових повідомлень, методи програмної інженерії для проєктування архітектури та реалізації програмного фреймворку.

Наукова новизна одержаних результатів полягає у вдосконаленні

методу агрегації результатів класифікації тональності шляхом запровадження формалізованого індексу настрою та системи багатовимірних агрегувань, що забезпечує перехід від міток тональності окремих повідомлень до узагальнених показників настрою інформаційного потоку, придатних для стабільного моніторингу та порівняльного аналізу різних сегментів даних у середовищі великих даних.

Практичне значення отриманих результатів полягає у реалізації програмного фреймворку для потокового аналізу настроїв у соціальних мережах на основі технологій великих даних, який забезпечує витягування, трансформацію та завантаження повідомлень, класифікацію їх тональності та агрегацію результатів для оперативного моніторингу й підтримки прийняття рішень.

Публікації та апробація КР. Результати кваліфікаційної роботи апробовані та опубліковані у матеріалах (додаток А):

– II міжнародної науково-практичної конференції «Progressive Approaches in Science and Engineering», November 26-28, 2025. Copenhagen, Denmark;

– II Всеукраїнської науково-практичної конференції «Інтелектуальні комп’ютерні системи та мережі», 25 листопада 2025 р., Тернопіль, Україна.

Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел та додатків.

1 АНАЛІЗ ВІДОМИХ МЕТОДІВ ТА ЗАСОБІВ АНАЛІЗУ НАСТРОЇВ В СОЦІАЛЬНИХ МЕРЕЖАХ

1.1 Опис предметної області

Процеси ETL є невід'ємною складовою технологій баз даних від моменту їх появи, хоча спочатку вони розглядалися як рутинні програмні процедури без окремої назви та виокремленого змістового значення [67]. Концепція ETL сформувалася тоді, коли розробники почали створювати програми для витягування записів з одного постійного файлу та заповнення або збагачення іншого файлу на основі отриманої інформації.

У контексті архітектури сховищ даних процеси ETL відіграють ключову, хоча й «невидиму» для кінцевого користувача роль. Першим етапом є витягування (extract), що передбачає визначення відповідної підмножини вихідних даних для подальшої обробки. На цьому кроці дані відшуковуються, зчитуються та витягуються з цільових джерел. Далі виконується етап трансформації (transform), під час якого дані обробляються й очищуються як на рівні схеми, так і на рівні окремих записів. Трансформація може включати нормалізацію та денормалізацію, переформатування, перерахунок показників, агрегування, об'єднання даних з кількох джерел, зміну структури ключів, урахування часових характеристик, обробку значень за замовчуванням тощо. Завершальним є етап завантаження (load), на якому оброблені дані записуються до цільового сховища або бази даних.

Традиційні пакетні ETL-процеси, що працюють зі статичними наборами даних, мають суттєві обмеження щодо опрацювання безперервних потоків даних. Потоковий ETL [43] покликаний подолати ці обмеження та забезпечує низку важливих переваг, зокрема можливість обробки даних у реальному часі, підтримку своєчасного ухвалення рішень, безперервну інтеграцію даних, масштабованість і підвищену ефективність.

Зростання популярності потокового ETL зумовлене потребою обробляти дані в режимі реального часу, що дозволяє організаціям зберігати конкурентоспроможність в умовах динамічного та швидкозмінного середовища.

Завдяки здатності працювати з великими обсягами інформації та високою швидкістю надходження даних stream ETL пропонує масштабований та ефективний підхід до обробки потокових даних із соціальних медіаплатформ. Це є особливо важливим для застосувань, пов’язаних з аналізом тональності та виявленням подій, де вирішальне значення мають оперативність отримання результатів і здатність системи швидко реагувати на зміни.

Система потокової обробки даних (Data Stream Processing System, DSPS) – це тип системи, що дає змогу подолати обмеження за затримкою, притаманні пакетній обробці, завдяки можливості опрацьовувати великі обсяги даних і отримувати корисну інформацію ще до їх збереження в довгострокових сховищах [33]. Потокова обробка забезпечує вагому конкурентну перевагу у вигляді своєчасності результатів і є доцільною там, де критично важливими є саме дані в реальному часі.

У DSPS зазвичай виділяють узагальнений конвеєр обробки даних [33]. Оскільки в подальшому розглядається побудова конвеєра для аналізу тональності повідомлень, далі увагу зосереджено на фонових аспектах шарів такого конвеєра з акцентом на ключових технологіях, що мають безпосередній стосунок до цього дослідження. Кожен із шарів розглядається нижче.

Надходження даних передбачає передання даних від джерела до цільової системи максимально ефективно та точно, із забезпеченням їхньої цілісності та мінімізацією ризику втрати або пошкодження [43]. Дані можуть надходити в різних форматах: це можуть бути CSV-файли, програмні інтерфейси, а також сучасні джерела, наприклад сенсори пристроїв Інтернету речей.

Доступ до даних може здійснюватися кількома підходами, зокрема шляхом вебскрапінгу (наприклад, за допомогою бібліотеки BeautifulSoup [52]), використанням сторонніх сервісів для скрапінгу (таких як Octoparse [48]) або шляхом звернення до публічно доступних наборів даних.

Для підтримки процесу витягування даних на цьому рівні доцільно використовувати систему повідомлень, яка забезпечує безперервну та надійну передачу даних між прикладними компонентами. Сучасна система обміну повідомленнями, орієнтована на роботу з великими джерелами даних, має бути

масштабованою, здатною обробляти великі обсяги повідомлень і забезпечувати низьку затримку. Серед таких систем можна виокремити RabbitMQ [68], Apache Kafka [63], Apache ActiveMQ [64], Apache Pulsar [65] та Amazon Simple Queue Service (SQS) [3].

У рамках даного дослідження як основну систему обрано Apache Kafka, оскільки вона поєднує високу продуктивність, стійкість до відмов і хорошу інтегровність з екосистемою обробки великих даних. Детальний емпіричний аналіз продуктивності Kafka наведено в роботі Hesse, Matthies та Uflacker [31].

Ключовим компонентом рівня обробки даних є рушії потокової обробки даних (Data Stream Processing Engines, DSPEs). DSPE характеризуються як розподілені паралельні архітектури з низькою затримкою, у яких конвеєри потокової обробки даних подаються у вигляді орієнтованого ациклічного графа (Directed Acyclic Graph, DAG) логічно пов’язаних завдань обробки потоків [33]. Базові компоненти DSPE включають джерела даних, приймачі (sinks), прикладні драйвери, менеджери потоків та обробники потоків; детальний опис цих компонентів подано в [33].

У цьому дослідженні для реалізації DSPE обрано Apache Spark [7]. Оскільки нас цікавить саме потокова обробка даних, одним з основних інтерфейсів програмування стає Spark Structured Streaming (Spark-SS) [9], за допомогою якого реалізується цільовий фреймворк. Spark-SS має низку переваг порівняно зі спадковим модулем Spark Streaming, що продемонстровано в роботі Ivanov та Taafe [35].

Рівні зберігання даних відіграють важливу роль у накопиченні та підготовці даних до подальшого аналізу, забезпечуючи ефективність аналітичних процедур. На цьому рівні можуть використовуватися різні підходи до зберігання, зокрема традиційні файлові системи, реляційні бази даних і NoSQL-рішення.

У межах даної роботи було обрано кілька технологій зберігання для виконання порівняльного аналізу продуктивності й отримання додаткової інформації щодо їх придатності до задач TSA. До складу розглядуваних рішень входять Apache Cassandra, HBase, Hive та HDFS [4, 6, 18, 39, 62]. Використання

декількох технологій зумовлене прагненням оцінити їхню продуктивність у контексті аналізу тональності на основі Twitter-даних. Для окремих компонентів уже існують відомі дослідження продуктивності, зокрема для HBase – робота Vora [69], для Hive – Camacho-Rodríguez та ін. [18], для Cassandra – Chaudhari та Mulay [20], а для Kafka – Hesse та ін. [31].

Рівень керування ресурсами стосується керування та координації обчислювальних вузлів у розподіленому середовищі. Зазвичай формується кластер, до складу якого входять диспетчер ресурсів і декілька обчислювальних вузлів. Існує низка технологій керування ресурсами; у межах даного дослідження використано кластер у режимі Spark standalone [8] та платформу Yet Another Resource Negotiator (YARN) [5].

Вихідний рівень відповідає за результати роботи конвеєра потокової обробки даних. На цьому рівні результати можуть передаватися до іншого застосунку, інтегруватися в подальший робочий процес або візуалізуватися за допомогою спеціальних інструментів.

У цій роботі основна увага приділяється саме візуалізації, яку можна розглядати з чотирьох позицій інструментарію [40]:

– візуалізація графів;
– візуалізація текстових даних;
– візуалізація карт;
– візуалізація багатовимірних даних.

Особливий інтерес становить картографічна візуалізація, для реалізації якої використано бібліотеки Geopy [28] та Folium [53]. Зазначені інструменти є відкритими й достатньо гнучкими, що спрощує їх інтеграцію до запропонованого фреймворку та дає змогу наочно відображати геопросторовий розподіл повідомлень.

1.2 Методи та засоби аналізу тональності

Аналіз тональності (Sentiment Analysis, SA), також відомий як opinion mining, пов’язаний із дослідженням ставлень, поглядів, емоцій та оцінок, які адресуються певним об’єктам чи подіям [72]. Відповідно, аналіз тональності – це процес визначення тональності висловлювань у повідомленнях з метою з’ясування, чи є висловлена думка переважно позитивною, негативною або нейтральною [76]. Ключовими етапами є надходження даних, попередня обробка, виокремлення ознак, відбір ознак (визначення суб’єктивності й об’єктивності тексту) та безпосередня класифікація тональності за полярністю [1, 38].

Дані з соціальних мереж доцільно оцінювати за кількома вимірами якості, зокрема читабельність (Readability), повнота (Completeness), корисність (Usefulness) та достовірність (Trustworthiness) [11]. Вимір читабельності забезпечує, щоб повідомлення після попередньої обробки залишалися коректними за змістом і зрозумілими для подальшого аналізу. Вимір повноти перевіряє, чи витягнуто всі необхідні компоненти повідомлення – текст, хештеги, згадки користувачів тощо. Вимір корисності зосереджений на тому, чи несе визначена тональність (позитивна, негативна або нейтральна) практичну цінність для аналізу. Нарешті, вимір достовірності пов’язаний із надійністю джерела та може базуватися на статусі верифікації облікового запису, кількості підписників, віці акаунта тощо. Опрацювання цих вимірів доцільно здійснювати на етапах надходження даних або попередньої обробки, щоб підвищити загальну якість вхідних даних TSA.

Поточні стрімінгові класифікатори тональності можуть оцінюватися за допомогою спеціалізованих метрик якості. У практиці аналізу тональності можливі як збалансовані, так і незбалансовані класи (наприклад, коли позитивних повідомлень суттєво менше, ніж нейтральних). У випадку збалансованих класів може застосовуватися, зокрема, prequential accuracy metric, рекомендована для оцінювання класифікаторів, що навчаються на часово змінних потоках даних [27]. Ця метрика є придатною тоді, коли кожен із класів

тональності (позитивний, негативний, нейтральний) представлено приблизно однаковою кількістю прикладів.

У ситуації незбалансованих класів використовують інші підходи, наприклад каппа-статистику [15]. Зазначена метрика дозволяє справедливіше оцінювати роботу класифікатора, оскільки нормує його точність відносно «випадкового» прогнозувача, враховуючи дисбаланс класів. Це забезпечує більш зважене й репрезентативне оцінювання ефективності моделей у потоках даних з нерівномірним розподілом тональностей.

Існуючі рішення узагальнюють і класифікують велику кількість підходів, однак у межах цього дослідження не ставиться за мету наводити вичерпний огляд. Натомість доцільно окреслити основні класифікаційні ракурси, запропоновані в уже наявних оглядах.

У роботі Zimbra та співавт. запропоновано класифікацію підходів із позиції викликів, характерних для даних соціальних мереж: стислості повідомлень, мовної різноманітності, наявності специфічних для платформи елементів комунікації, дисбалансу класів тональності та потокового характеру генерування повідомлень [76]. Серед розглянутих підходів виокремлюються:

– поширення інформації про тональність для виявлення нових, нетривіальних способів вираження емоцій;

– використання ансамблів і множин класифікаторів для подолання проблеми дисбалансу класів;

– застосування потокових класифікаторів для задоволення вимог до обробки великого обсягу та високої швидкості надходження потоку повідомлень.

Adwan та співавт. пропонують іншу перспективу й розподіляють підходи до TSA на чотири основні групи: методи машинного навчання (ML), лексиконно-орієнтовані, гібридні та графові підходи [1]. У свою чергу, Kumar і Jaiswal класифікують TSA з позиції «м’яких обчислень», виокремлюючи ML, нейронні мережі, ймовірнісні методи, еволюційні обчислення та нечітку логіку [38].

У ширшому контексті аналізу тональності (SA) Yadav і Vishwakarma запропонували таксономію, що узагальнює поширені глибинні архітектури (deep

learning) та обговорює наслідки їх застосування для задач SA [72]. Окрема група робіт стосується багатомовного аналізу тональності: Mercha та Benbrahim здійснили огляд підходів до аналізу тональності між різними мовами, зокрема в межах мультимовного (Multilingual Sentiment Analysis, MSA) і крослінгвального (Cross-Lingual Sentiment Analysis, CLSA) аналізу [46].

Таким чином, наявні огляди демонструють як різноманіття технічних підходів (ML, лексиконні, гібридні, графові, нейромережеві), так і різні виміри класифікації тональності – від типів обчислювальних методів до специфічних викликів, пов’язаних із природою потоків повідомлень.

1.3 Аналіз відомих рішень

Низка робіт присвячена потоковій обробці даних, де однією з ключових технологій виступає Spark, однак такі дослідження не фокусуються безпосередньо на аналізі тональності. Так, у праці Yadranjiaghdam та ін. запропоновано аналітичний фреймворк для витягування та аналізу у реальному режимі часу як структурованих, так і неструктурованих даних із використанням Apache Kafka для надходження даних, Spark – для їх обробки, NoSQL-сховищ – для зберігання та застосування методів ML; ефективність підходу продемонстровано на прикладі повідомлень про землетрус у Японії [73].

У роботі Aziz та ін. проведено порівняння продуктивності й архітектурних особливостей фреймворків Hadoop MapReduce та Apache Spark для задач обробки великих даних у реальному часі, окреслено обмеження Hadoop і наведено результати експериментального моделювання потокової обробки даних із використанням обох фреймворків [12]. Своєю чергою, Shah та співавт. запропонували систему TAGS, що поєднує Hadoop, Spark та HBase для збору й аналізу потокових даних із соціальних медіа в реальному часі з метою формування попереджень щодо перебігу надзвичайних подій і стихійних лих [58].

Інша група досліджень використовує Twitter-датасет і зосереджується вже на задачах аналізу тональності, але як базову технологічну платформу розглядає

Hadoop замість Spark. Зокрема, Cunha та ін. досліджують можливості фреймворку Apache Hadoop та бібліотеки Mahout для керування й аналізу великомасштабних медичних Twitter-даних, демонструючи потенціал цих технологій для вилучення корисної інформації як для користувачів, так і для фахівців галузі охорони здоров'я [22]. Rodrigues та ін. пропонують підхід до аналізу тональності в потоці Twitter-даних у режимі реального часу на основі екосистеми Hadoop, залучаючи Apache Flume для потокового надходження даних, Pig-скрипти – для їх витягування й попередньої обробки, а також словниковий метод класифікації повідомлень на позитивні, негативні та нейтральні [54]. Натомість Sehgal і Agarwal зосереджуються на задачі аналізу тональності в реальному часі для застосунків великих даних, використовуючи Twitter-дані в поєднанні з Hadoop-фреймворком [57].

Окрім зазначених вище підходів, існує також низка праць, у яких розглядається потокова обробка даних у середовищі Big Data, але без фокусування саме на Twitter-датасеті. Так, Zhou та ін. запропонували систему онлайн-моніторингу інтернет-трафіку на основі Kafka та Spark Streaming для контролю й керування мережними ресурсами [75]. Sunny та ін. здійснили реалізацію системи рекомендацій телеканалів у реальному часі з використанням Apache Spark та Cassandra, оптимізованої для обробки потоків даних від set-top box-пристроїв [60]. Wu та ін. розробили потоковий ML-алгоритм на основі методу головних компонент і потокової лінійної регресії для точного й динамічного прогнозування концентрації газу в шахтах із застосуванням Spark Streaming [71]. Ed-Daoudy та Maalmi запропонували систему прогнозування серцевих захворювань у реальному часі, що поєднує потокову аналітику великих даних і методи ML на базі Spark Streaming та Spark MLlib, при цьому великі обсяги згенерованих даних зберігаються в Apache Cassandra [24]. Tun, Nyaung та Phyu розробили підхід до скорочення часу виявлення кіберзагроз на основі Apache Kafka та Spark Streaming з використанням датасету UNSWNB-15 [66].

До пов'язаних робіт, у яких безпосередньо розглядається потокова обробка даних для TSA із використанням Spark як однієї з ключових технологій, належать, зокрема, дослідження Elzayady та співавт. і Fahd та ін. [25, 26]. У

роботі Elzayady та ін. запропоновано рішення для аналізу тональності з використанням бібліотеки Apache Spark MLlib для обробки великих обсягів повідомлень із застосуванням алгоритмів наївного байєсівського класифікатора, логістичної регресії та дерев рішень [25]. Fahd та ін. розробили архітектуру системи аналізу тональності в реальному часі на основі багатьох вхідних джерел із соціальних медіа, реалізовану за допомогою Apache Kafka, Spark, кластерів YARN і MongoDB, із проведенням оцінювання продуктивності за різними показниками якості [26].

Застосування потокового ETL доцільно розглядати з позицій двох основних доменів: сховищ даних у реальному часі (real-time DWH) та сценаріїв, що не пов’язані безпосередньо з DWH [45].

Наявні роботи, у яких потоковий ETL вивчається саме в контексті DWH, представлені, зокрема, у дослідженнях Biswas, Sarkar та Mondal [16], Gorawski та Gorawska [30], Machado, Cunha, Pereira та Oliveira [41], Mehmood та Anees [44] і Pareek та ін. [49]. Так, Gorawski та Gorawska описують реалізацію процесу stream ETL для потокового сховища даних, що дає змогу завантажувати дані в реальному часі для підтримки процесів ухвалення рішень, а також подають результати аналізу точності й ефективності розробленого ETL-рушія [30]. Pareek та ін. наголошують на потребі доступу до корпоративних даних у реальному часі та на обмеженнях традиційних ETL-процесів; у відповідь на це вони пропонують платформу Striim – розподілене середовище потокового ETL та аналітики, орієнтоване на швидку розробку й розгортання стрімінгових застосунків і зосереджене на ключових функціональних можливостях [49].

Machado та ін. пропонують інструмент DOD-ETL, покликаний подолати «вузьке місце» класичного процесу Extract–Transform–Load у рішеннях бізнес-аналітики шляхом поєднання on-demand конвеєра потокових даних із розподіленою, паралельною, технологічно незалежною архітектурою, доповненою in-memory кешуванням та ефективним розподілом даних [41]. Biswas та колеги здійснюють порівняння відкритих кодових ETL-інструментів, розроблених у науковому середовищі, і пропонують ефективну модель ETL, орієнтовану на виконання вимог до обробки даних у режимі, наближеному до

реального часу, як альтернативу суто GUI-орієнтованим рішенням [16].

У контексті оцінювання продуктивності Mehmood та Anees досліджують проблеми й ефективність потокового з’єднання NoSQL-потоків, демонструючи, що використання пам’яті та час виконання залишаються стабільними незалежно від типу потоків даних; отримані результати можуть слугувати орієнтиром для розробників, які впроваджують схеми збереження даних у реальному часі на основі MongoDB [44].

Також проаналізовано україномовні джерела на цю тему [77, 78]. У сучасних дослідженнях моніторингу соціальних мереж акцент дедалі частіше зміщується від «простого збору повідомлень» до аналізу їхнього впливу на суспільні процеси. Так, у роботі Ткаченка та ін. [78] соціальні мережі розглядаються як середовище, де інформаційні потоки здатні формувати громадську думку, а розвиток штучного інтелекту одночасно створює і нові можливості аналізу, і нові загрози через дезінформацію та маніпулятивний контент. Автори підкреслюють потребу в інструментах, що автоматизують виявлення таких впливів і забезпечують аналіз у контексті гібридної війни, враховуючи роль алгоритмів платформ, «інформаційні бульбашки» та поведінкові ефекти користувачів. Водночас ця праця має переважно оглядово-аналітичний характер: вона добре обґрунтовує актуальність проблеми та напрямки застосування ШІ, але не формалізує практичний, відтворюваний конвеєр потокової обробки даних.

Інший напрям відомих рішень концентрується на якості й інтерпретованості алгоритмів аналізу тональності для конкретних мов. Зокрема, Ломовацький і Басюк [77] пропонують правило-орієнтований підхід до аналізу сентименту українською мовою та прямо вказують, що популярні англоцентричні інструменти (наприклад, VADER) демонструють низьку точність на українських текстах через складну морфологію, гнучкий синтаксис, часті заперечення та ідіоматичні конструкції. Для підвищення якості автори використовують розширений лексикон (зокрема EmoLex), модифікатори інтенсивності, а також складніші механізми врахування контексту, такі як синтаксичний аналіз залежностей і позиційно-орієнтоване оцінювання.

Водночас запропоноване рішення описує саме мовозалежний алгоритм оцінювання тональності та не вирішує системну частину задачі, пов’язану з високонавантаженим потоковим ETL.

Порівняно з наведеними роботами слід відзначити, що існує обмежена кількість досліджень, які розглядають застосування технологій великих даних як інтегроване рішення для потокового ETL, орієнтоване не стільки на класичні сценарії, скільки на потоки даних із соціальних мереж. Саме цю прогалину й покликане заповнити дане дослідження, спрямоване на розроблення ефективного методу для аналізу настроїв в соціальних мережах.

1.4 Постановка задачі дослідження

Соціальні медіа продукують безперервні потоки даних, що швидко змінюються й мають високу варіативність за мовою, тематикою та якістю. Для організацій, які приймають рішення на основі суспільних настроїв (державні інституції, медіа, бізнес), критичною є здатність оперативно перетворювати ці потоки на інсайти. Класичні ETL-процедури, орієнтовані на пакетну обробку статичних наборів, погано узгоджуються з вимогами низької латентності, масштабованості та геопросторової аналітики. Тому дослідження потокового ETL для аналізу тональності повідомлень із подальшою візуалізацією на мапі є практично значущим: воно поєднує високопродуктивний конвеєр даних із модулем класифікації тональності та сховищами великих даних, забезпечуючи прийнятну якість і швидкість для сценаріїв моніторингу в реальному часі.

Попри значний прогрес, наявні підходи мають низку прогалин, що знижують їхню придатність у виробничих умовах:

– багато робіт аналізують окремі ланки (наприклад, лише Spark Streaming чи лише класифікацію), не пропонуючи цілісної моделі інтеграції «збір → трансформація/класифікація → завантаження → візуалізація»;

– в основному класифікацію тональності виконують поза конвеєром, що ускладнює керування затримкою та пропускною здатністю, а також повторне використання результатів;

– рідко досліджується вплив тригерів мікропакетів, розмірів датасетів і типів сховищ на метрику Trigger Execution Time у Spark-SS; ще рідше – порівняння кількох сховищ у тотожному кластерному оточенні;

– включення попередньо натренованих моделей через UDF у Spark часто спричиняє істотні накладні витрати; відсутні рекомендації, коли класифікацію доцільно виконувати в ETL, а коли – поза ETL;

– питання дебіасингу текстів, багатомовності та неповної геолокації опрацьовуються нерівномірно; бракує аналізу впливу дебіасингу на перерозподіл класів і підсумкові карти.

Отже, метою роботи є підвищення ефективності аналізу настроїв у соціальних мережах за рахунок розроблення потокового конвеєра обробки даних на основі технологій великих даних, який забезпечує безперервне витягування, трансформацію та завантаження повідомлень у режимі, наближеному до режиму реального часу, з подальшою класифікацією тональності та агрегацію результатів.

Для досягнення визначеної мети необхідно виконати ряд завдань:

– проаналізувати предметну область аналізу настроїв у соціальних мережах;

– виконати порівняльний аналіз методів та засобів аналізу настроїв у текстових повідомленнях;

– провести аналіз відомих рішень для аналізу настроїв в соціальних мережах;

– сформулювати постановку задачі дослідження;

– розробити метод аналізу настроїв у соціальних мережах на основі технологій великих даних;

– розробити алгоритми витягування, трансформації та завантаження даних у потоковому режимі;

– розробити алгоритм класифікації повідомлень;

– розробити метод агрегації результатів класифікації;

– спроєктувати архітектуру та реалізувати програмні модулі для аналізу настроїв у соціальних мережах.

– провести експериментальні дослідження запропонованих рішень.

Висновки до розділу 1

1. Проаналізовано предметну область аналізу настроїв у соціальних мережах і встановлено, що потоки коротких повідомлень є цінним джерелом даних для моніторингу суспільних реакцій у різних сферах, однак характеризуються високою динамічністю, великими обсягами та неоднорідністю. Показано, що в межах прикладних задач моніторингу доцільно інтерпретувати “настрій” інформаційного потоку через аналіз тональності як базовий, відтворюваний індикатор емоційного фону повідомлень.

2. Виконано порівняльний огляд методів і засобів аналізу тональності текстових повідомлень, зокрема класичних підходів на основі правил і машинного навчання та сучасних нейромережевих методів. Визначено, що для багатомовних і реальних потоків даних найбільш перспективними є підходи на основі попередньо натренованих мовних моделей, які забезпечують кращу узагальнювальну здатність та стійкість до різноманіття мовних конструкцій.

3. Проаналізовано відомі рішення до задачі аналізу настроїв у соціальних мережах і встановлено, що значна частина робіт зосереджена або на підвищенні точності класифікації тональності, або на потоковій обробці даних як такій, без формалізації цілісного зв’язку між компонентами потокового ETL та інтелектуальною обробкою тексту. Також виявлено, що класичні ETL-процедури, орієнтовані на пакетну обробку статичних наборів даних, є недостатньо ефективними для сценаріїв режиму реального часу.

4. На основі виконаного аналізу сформульовано постановку задачі дослідження та визначено ключові вимоги до рішення: забезпечення потокового витягування, трансформації та завантаження повідомлень у середовищі великих даних; інтеграція алгоритму класифікації тональності як індикатора настрою; накопичення результатів у сховищах для подальшого аналізу; а також наочне подання результатів із можливістю фільтрації та інтерпретації.

2 МЕТОДИ ТА АЛГОРИТМИ АНАЛІЗУ НАСТРОЇВ В СОЦІАЛЬНИХ МЕРЕЖАХ НА ОСНОВІ ТЕХНОЛОГІЙ ВЕЛИКИХ ДАНИХ

2.1 Метод аналізу настроїв в соціальних мережах

У даній роботі запропоновано метод аналізу настроїв у соціальних мережах на основі технологій великих даних, орієнтований на обробку потоку повідомлень у реальному або наближеному до режиму реального часу. Під аналізом настроїв у межах розуміється оцінювання загального емоційного фону потоку повідомлень, який формалізується через аналіз тональності (позитивна / нейтральна / негативна) як найбільш універсальний і відтворюваний індикатор. Метод реалізовано як потоковий ETL-процес із можливістю масштабування та інтеграції інтелектуальної обробки у конвеєр великих даних.

Даний метод описується як послідовність кроків (рисунок 2.1), де кожен крок формує необхідні дані для наступного етапу та забезпечує перехід від сирого потоку соціальних повідомлень до інтерпретованих результатів аналізу настроїв.

Крок 1. Потокове витягування повідомлень і формування черги даних.

На першому кроці забезпечується безперервне одержання повідомлень із джерела даних та їх передавання в середовище великих даних. Для цього повідомлення записуються до Kafka-топіка за допомогою Kafka-Producer, після чого витягуються на обробку через Kafka-Consumer [63]. Використання Kafka дозволяє буферизувати потік, стабілізувати швидкість надходження даних та організувати надійну доставку повідомлень для наступних кроків обробки.

Крок 2. Потокова попередня обробка та перетворення даних.

На другому кроці здійснюється трансформація потоку: очищення, нормалізація, приведення полів до потрібного формату, вилучення службових або надлишкових елементів, а також підготовка даних до інтелектуального аналізу. Даний крок реалізується засобами Spark Structured Streaming (Spark-SS) та Spark SQL API [9, 61]. За потреби тут можуть застосовуватися фільтри за мовою, тематикою, часовими межами, а також формування набору ознак або полів, необхідних для класифікації.

Рисунок 2.1 – Узагальнена схема методу аналізу настроїв у потоці соціальних повідомлень

Крок 3. Обчислення настрою потоку на основі класифікації тональності.

На третьому кроці виконується інтелектуальна інтерпретація повідомлень як компонент аналізу настроїв. Кожне повідомлення відноситься до одного з трьох класів тональності: позитивного, нейтрального або негативного. Для цього використано трансформерну модель Twitter-XLM-RoBERTa-base, запропоновану Barbieri, Espinosa Anke та Camacho-Collados, що забезпечує класифікацію повідомлень за відповідними класами [13]. Отриманий клас тональності трактується як базовий показник “настрою” повідомлення, а сукупність класифікованих повідомлень формує оцінку загального емоційного фону потоку.

Залежно від обраного режиму роботи, цей крок може виконуватися двома способами:

– у потоковому режимі як частина кроку трансформації (для отримання

результатів у наближеному до реального часу), або

– після накопичення даних у сховищі (для пакетного аналізу та порівняння результатів).

Крок 4. Завантаження результатів у сховища даних.

На четвертому кроці підготовлені та класифіковані дані записуються у відповідні сховища для подальшого аналізу, агрегування та візуалізації. Даний крок реалізується засобами Spark-SS і відповідними механізмами запису, а також може використовувати Spark SQL API для формування необхідних структур даних [9, 61]. Результатом є накопичена та впорядкована база повідомлень із доданими атрибутами тональності та метаданими (час, мова, тема тощо).

Отже, запропонований метод поєднує обробку потоків даних і потоковий ETL у єдину послідовність кроків, що забезпечує перехід від безперервного потоку соціальних повідомлень до структурованих результатів аналізу настроїв. Ключовою умовою ефективності такого підходу є оптимізація продуктивності потокових обчислень.

2.2 Алгоритм витягування, трансформації та завантаження даних

Потоковий ETL-конвеєр у даній роботі реалізовано як послідовність взаємопов’язаних компонентів Stream Data → Kafka-Producer → Kafka-Consumer (Spark Structured Streaming) → трансформація → завантаження у сховище. Така організація забезпечує безперервне отримання повідомлень, їх буферизацію та транспортування через брокер повідомлень, подальшу потокову попередню обробку й накопичення результатів у вибраному сховищі для наступного етапу аналітики (класифікації тональності та агрегації).

Компонент Stream Data відповідає за доступ до джерела даних і витягування сирих повідомлень у напівструктурованому форматі (JSON). Для потокового отримання даних використовується Tweepy, зокрема клас StreamingClient, який ініціалізує з’єднання із сервісом. Фільтрація потоку задається правилами StreamRule, що дозволяють формувати вибірку за ключовими словами/гештегами, мовою та іншими умовами (наприклад, відбір

лише оригінальних публікацій). Запуск потоку здійснюється методом filter(), а приймання сирих повідомлень реалізується шляхом перевизначення on_data(). На цьому кроці з JSON вибираються необхідні атрибути (ідентифікатор повідомлення, атрибути користувача, час публікації, текст, вказане місцезнаходження тощо) та формується запис, придатний для подальшого передавання в потокову інфраструктуру.

Компонент Kafka-Producer виконує роль “вхідного шлюзу” в ETL-конвеєрі: він записує сформовані напівструктуровані повідомлення до Kafka-топіка, який використовується як буфер та транспортний рівень потокових даних. Producer реалізовано на Python із використанням бібліотеки Kafka-Python: ініціалізується об’єкт KafkaProducer, налаштовуються адреси брокерів та назва топіка, після чого кожен запис надсилається у Kafka через виклик send().

Компонент Kafka-Consumer реалізовано на основі Spark Structured Streaming, який виконує потокове зчитування з Kafka та формує Spark DataFrame для подальшої обробки. Зчитування здійснюється через readStream() із параметрами підключення: адреси брокерів, назва топіка та формат джерела. Після виконання load() дані потрапляють у DataFrame і переходять на етап трансформації.

Етап трансформації включає обов’язкову попередню обробку повідомлень, що забезпечує придатність даних для аналітики. У межах даної роботи попередня обробка охоплює:

– розбиття (data splitting) – перетворення вхідного payload (який у Kafka може зберігатися як один рядок) на окремі атрибути повідомлення з використанням withColumn() і split();

– очищення (data cleaning) – вилучення небажаних шаблонів і підрядків (згадки користувачів, гіперпосилання, службові символи тощо) із застосуванням regexp_replace() у поєднанні з withColumn();

– формування фінального набору полів – фіксація результатів трансформації через select() у новому DataFrame.

Трансформаційні операції виконуються засобами Spark SQL та DataFrame API, що забезпечує їх масштабованість у середовищі великих даних і

узгодженість із потоковою моделлю виконання Spark Structured Streaming.

Етап завантаження полягає у записі потоку оброблених даних до цільового сховища (наприклад, Cassandra, HDFS, Hive, HBase тощо). Запис виконується через writeStream() із налаштуванням:

– інтервалу періодичного запуску запису через trigger() (наприклад, 500 мс або фіксовані інтервали);

– режиму виведення результатів через outputMode() (наприклад, update);

– обробки кожної мікропартії через foreachBatch(), яка передає мікропакет у користувацьку функцію запису;

– ініціалізації потокового запиту через start().

Користувацька функція запису виконує фактичне збереження даних у вибране сховище через write.format(...).mode(...).save(...) з урахуванням специфіки технології зберігання.

Таким чином, описано завершений потоковий ETL-конвеєр, результатом якого є накопичення очищених і нормалізованих повідомлень у сховищі. Подальші етапи – класифікація тональності та агрегація результатів – виконуються вже на аналітичному.

Розглянемо пропоновані алгоритми.

Алгоритм 2.1 – Потоковий ETL: витягування → буферизація (Kafka) → трансформація (Spark) → завантаження у сховище

Вхідні дані: токен доступу до джерела (X API), правила фільтрації, Kafka-брокери, назва Kafka-топіка, параметри Spark, параметри сховища

1. Ініціалізація витягування даних: встановити з’єднання з джерелом із використанням токена.

2. Налаштування правил потоку: задати StreamRule (ключові слова/гештеги, мова, інші умови).

3. Ініціалізація Kafka-Producer: налаштувати брокери та топік, створити KafkaProducer.

4. Запуск потокового витягування: запустити filter() та приймати повідомлення через on_data().

5. Формування запису: виділити потрібні атрибути із JSON та сформувати напівструктурований запис.

6. Запис у Kafka: надіслати запис у Kafka-топік через KafkaProducer.send().

7. Ініціалізація Kafka-Consumer у Spark: створити потокове читання через readStream().format("kafka"), задати option() (брокери, топік).

8. Завантаження у DataFrame: виконати load() та отримати DataFrame потоку.

9. Трансформація:

9.1. Виконати розбиття payload на атрибути через withColumn(split(...));

9.2. Виконати очищення тексту через regexp_replace(...);

9.3. Сформувати фінальний DataFrame через select().

10. Завантаження у сховище:

10.1. Налаштувати writeStream();

10.2. Задати trigger(), outputMode();

10.3. Записувати мікропакети через foreachBatch() у користувацькій функції запису;

10.4. Запустити обробку через start()

2.3 Алгоритм класифікації даних

Класифікація тональності застосовується для віднесення повідомлень до одного з трьох класів: негативний, нейтральний, позитивний. У межах даної роботи не виконується навчання або оцінювання моделі; натомість використовується попередньо натренована трансформерна модель Twitter-XLM-RoBERTa-base та відповідний Tokenizer з Hugging Face . Модель навчено на великому масиві повідомлень і донавчено для багатомовного сценарію (зокрема англійської, іспанської та інших мов) .

Для інтеграції класифікації тональності у програмний комплекс визначається користувацька функція tokenize(), яка приймає очищений текст повідомлення, виконує токенізацію, подає тензори на вхід моделі, отримує

оцінки класів і повертає мітку "Negative" / "Neutral" / "Positive". Для вбудовування цієї логіки в аналітичний процес PySpark використовується udf(), що дозволяє формувати новий стовпець із мітками тональності для DataFrame .

Класифікація тональності повідомлення представлена алгоритмом 2.2, вхідними даними якого є: очищені повідомлення, завантажені зі сховища (наприклад, Cassandra/HDFS/Hive/HBase), модель Twitter-XLM-RoBERTa-base, токенайзер

1. Завантажити дані зі сховища: отримати DataFrame із очищеними повідомленнями (поля як мінімум text та службові атрибути ідентифікації).
2. Ініціалізувати модель та токенайзер: завантажити Twitter-XLM-RoBERTa-base і відповідний токенайзер.
3. Оголосити функцію tokenize(text):

3.1. Токенізувати вхідний текст та закодувати його у тензорне подання;

3.2. Передати тензори на вхід моделі;

3.3. Отримати “сирі” оцінки класів scores[];

3.4. Виконати нормалізацію prob = softmax(scores[]);

3.5. Визначити клас із максимальною ймовірністю label = argmax(prob);

3.6. Повернути текстову мітку "Negative", "Neutral" або "Positive".

4. Інтегрувати класифікацію у DataFrame: застосувати udf(tokenize) для формування нового стовпця sentiment_label.
5. Зберегти або передати результат:

5.1. Записати DataFrame з sentiment_label назад у сховище як окрему таблицю/зріз, або

5.2. Використати отримані мітки як вхід для наступного етапу агрегації.

Функція визначення мітки тональності tokenize() представлена алгоритмом 2.3, вхідними даними якого є: очищене повідомлення text

1. Ініціалізувати (або отримати із кешу) модель Twitter-XLM-RoBERTa-base.
2. Ініціалізувати токенайзер.

3. Закодувати text у тензорне подання (token IDs, attention mask).
4. Передати закодоване подання в модель.
5. Витягнути масив оцінок scores[] для класів тональності.
6. Обчислити prob = softmax(scores[]).
7. Визначити idx = argmax(prob).
8. Якщо idx == 0, повернути "Negative".
9. Якщо idx == 1, повернути "Neutral".
10. Якщо idx == 2, повернути "Positive".

Класифікація виконується після завантаження даних у сховище, тобто ETL-конвеєр завершується на кроці збереження очищених повідомлень, а модуль класифікації виступає окремим аналітичним етапом над накопиченими даними.

2.4 Метод агрегації результатів класифікації

Після виконання класифікації тональності кожне повідомлення перетворюється на структурований запис, який містить: ідентифікатор повідомлення, час створення або час надходження, мову повідомлення, тематичну категорію, а також визначений клас тональності (позитивна, нейтральна або негативна). За потреби також може зберігатися числова оцінка впевненості моделі. Такі дані є придатними для подальшого узагальнення, оскільки дають змогу обчислювати показники настрою не лише для окремих повідомлень, а й для інтервалів часу, мовних сегментів та тематичних зрізів.

Агрегація результатів класифікації виконується з метою отримання узагальнених показників, які відображають динаміку настрою та структуру емоційного фону інформаційного потоку. У даній роботі агрегація базується на трьох типах зрізів: часовому, мовному та тематичному. Загальна логіка агрегації реалізується як послідовність кроків.

Крок 1. Формування часових інтервалів агрегації.

На першому етапі потік класифікованих повідомлень дискретизується в часі, тобто кожному повідомленню призначається часовий інтервал, у межах

якого воно враховується при обчисленні агрегованих показників. Нехай Δt – тривалість інтервалу агрегації (наприклад, п’ять хвилин, одна година або один день). Для кожного повідомлення з часовою міткою t визначається межа інтервалу, до якого воно належить. У результаті кожне повідомлення однозначно відноситься до пари (t_{start}, t_{end}), де $t_{end} = t_{start} + \Delta t$. Такий підхід забезпечує узгоджене обчислення показників у режимі, наближеному до режиму реального часу, а також спрощує формування звітних зрізів за задані часові проміжки.

Крок 2. Обчислення базових частот тональності для часових інтервалів.

Для кожного інтервалу (t_{start}, t_{end}) підраховується кількість повідомлень кожного класу тональності: N_{pos} – кількість позитивних, N_{neu} – кількість нейтральних, N_{neg} – кількість негативних повідомлень. Загальна кількість повідомлень в інтервалі визначається як:

$$N = N_{pos} + N_{neu} + N_{neg}. \quad (2.1)$$

Окрім абсолютних значень доцільно обчислювати відносні частки, які підвищують порівнюваність результатів між інтервалами різної “насиченості” даними:

$$N_{pos} = \frac{N_{pos}}{N}, N_{neu} = \frac{N_{neu}}{N}, N_{neg} = \frac{N_{neg}}{N}. \quad (2.2)$$

Отриманий розподіл є основним узагальненням, що описує структуру настрою в інтервалі часу.

Крок 3. Обчислення індексу настрою.

Для компактного подання загального емоційного фону вводиться індекс настрою, який зводить розподіл тональності до одного числового показника. У даній роботі індекс настрою визначається як нормована різниця між позитивними та негативними повідомленнями:

$$SI = \frac{N_{pos} - N_{neu}}{N} \quad (2.3)$$

Зазначений індекс набуває значень у діапазоні $[-1; 1]$. Значення, близькі до 1, означають переважання позитивних висловлювань, значення, близькі до -1, свідчать про домінування негативних висловлювань, а значення поблизу нуля відповідають або збалансованому співвідношенню позитивних і негативних повідомлень, або переважанню нейтральних. Такий індекс є зручним для подальшого моніторингу, оскільки дозволяє швидко порівнювати інтервали між собою та виявляти тенденції.

Крок 4. Агрегація за мовою.

Для аналізу відмінностей настроїв між мовними сегментами потоку виконується агрегація показників окремо для кожної мови. Для цього для кожного інтервалу часу формується групування за ознакою мови $lang$, після чого повторюються кроки 2–3 у межах кожної групи. У результаті отримуються показники виду $(t_{start}, t_{end}, lang, N_{pos}, N_{neu}, N, SI)$. Такий підхід дозволяє, наприклад, зафіксувати ситуації, коли загальний настрій потоку є нейтральним, але в окремій мовній групі спостерігається виражене зростання негативного фону.

Крок 5. Агрегація за тематикою.

Аналогічно до мовного зрізу виконується агрегація за тематичними категоріями $topic$. Для кожного інтервалу часу повідомлення групуються за темою, після чого для кожної теми обчислюються частоти тональності, частки та індекс настрою. Результат має вигляд $(t_{start}, t_{end}, topic, N_{pos}, N_{neu}, N, SI)$. Тематична агрегація дає змогу виділяти теми, що формують негативний фон, та відокремлювати їх від тем із нейтральною або позитивною динамікою.

Крок 6. Комбінований зріз “мова–тематика–час”.

Для детальнішого аналізу, коли необхідно одночасно враховувати мовну та тематичну специфіку, застосовується комбінована агрегація за трійкою ознак $(lang, topic, time\_window)$. У такому разі обчислюються показники для груп $(t_{start}, lang, t_{end}, topic$. Комбінований зріз є інформативним, однак може призводити до появи малочисельних груп. Тому доцільно використовувати мінімальний поріг $N \geq N_{min}$ для врахування груп у звітах, щоб зменшити вплив

випадкових коливань при малих обсягах даних.

Крок 7. Оцінювання динаміки настрою в часі.

Для моніторингу змін емоційного фону важливо аналізувати не лише значення індексу в окремому інтервалі, а й його зміну між сусідніми інтервалами. Для цього може обчислюватися приріст:

$$\Delta SI = SI(t_k) - SI(t_{k-1}) \quad (2.4)$$

де t_k – поточний часовий інтервал;

t_{k-1}) – попередній.

Додатково може застосовуватися згладжування (наприклад, ковзне усереднення індексу) для зменшення впливу короткочасних сплесків.

У підсумку, запропонований метод агрегації забезпечують перехід від класифікації окремих повідомлень до системи узагальнених показників, які відображають структуру та динаміку настрою інформаційного потоку. Зокрема, формуються розподіли тональності та індекси настрою у часових інтервалах, а також у мовних і тематичних зрізах.

Висновки до розділу 2

1. Запропоновано метод аналізу настроїв, у якій “настрій” інформаційного потоку інтерпретується через тональність повідомлень як базовий показник емоційного фону. Визначено загальну логіку функціонування рішення та структуру обробки, що поєднує потокову інфраструктуру великих даних із інтелектуальним аналізом текстового контенту.

2. Розроблено алгоритми витягування, трансформації та завантаження даних у потоковому режимі. Обґрунтовано використання брокера повідомлень для приймання та буферизації потоку, а також засобів потокової обробки і SQL-операцій для очищення, нормалізації та підготовки даних до подальшої інтерпретації. Визначено, що ключовими параметрами ефективності такого підходу є продуктивність обробки потоку та коректне налаштування тригерів

виконання мікропакетів.

3 Розроблено алгоритм класифікації даних, який призначений для визначення тональності повідомлень із поділом на позитивні, нейтральні та негативні. Описано логіку інтеграції класифікації в загальний конвеєр обробки, що забезпечує формування атрибутів настрою для кожного повідомлення та створює основу для подальшої аналітики й візуалізації.

4. Запропоновано метод агрегації результатів класифікації, який забезпечує перехід від міток окремих повідомлень до системи узагальнених показників. Показано, що через дискретизацію в часі та групування за мовою і тематикою можна обчислювати розподіли тональності, індекс настрою та динаміку зміни настрою, а також формувати комбіновані зрізи “час–мова–тема” для подальшого моніторингу й підготовки звітних узагальнень.

3 РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ЗАПРОПОНОВАНИХ РІШЕНЬ

3.1 Архітектура програмного забезпечення

Програмна реалізація запропонованого методу виконана у вигляді модульного потокового фреймворку, який поєднує брокер повідомлень Kafka, потокову обробку Spark Structured Streaming та аналітичну обробку Spark SQL. Архітектура орієнтована на відокремлення задач потокового ETL від задач інтелектуальної інтерпретації й узагальнення, що відповідає варіанту B і дає змогу підвищити керованість продуктивністю, забезпечити повторюваність аналітики та зменшити вплив обчислювально “важкої” класифікації на стабільність конвеєра надходження даних.

Архітектура включає такі логічні підсистеми:

1. Підсистема витягування та транспортування потоку.

Містить компонент Stream Data (отримання повідомлень із джерела за заданими правилами фільтрації) та Kafka-Producer, який серіалізує повідомлення й записує їх у Kafka-топік. Kafka виконує роль буфера та транспортного шару, вирівнює нерівномірність потоку, підвищує надійність доставки і роз’єднує джерело даних та обчислювальну частину.

2. Підсистема потокового ETL.

Реалізована як Spark Structured Streaming додаток, який виконує Kafka-Consumer (readStream), формує потоковий DataFrame та застосовує трансформації попередньої обробки: розбиття payload на атрибути, очищення тексту, нормалізацію і формування фінального набору полів. Результат ETL записується у сховище через writeStream/foreachBatch з параметром trigger, що дозволяє гнучко керувати частотою формування мікропакетів.

3. Підсистема зберігання.

Виступає базою для подальшої аналітики. У рамках реалізації передбачено можливість використання різних технологій зберігання (наприклад, HDFS/Hive/HBase/Cassandra), що дозволяє вибирати сховище залежно від того, чи пріоритетом є швидке накопичення даних, чи підтримка SQL-аналітики та

звітних зрізів.

4. Підсистема аналітичної обробки (Analytics Layer).

Працює поверх накопичених даних і включає два ключові кроки:

– класифікація тональності трансформерною моделлю Twitter-XLM-RoBERTa-base з формуванням поля sentiment_label;

– агрегація результатів (розподіли за часом/мовою/темою та індекс настрою) з підготовкою узагальнених таблиць/зрізів для звітності.

Аналітика реалізована через Spark SQL (таблиці, представлення, групування, віконні перетворення), що забезпечує масштабоване виконання узагальнень.

5. Підсистема моніторингу та відтворюваності експериментів.

Для контролю продуктивності використано метрики Spark Structured Streaming (зокрема triggerExecution), що дає змогу порівнювати конфігурації інтервалів тригера, різні розміри датасетів і технології сховищ у єдиній системі координат.

3.2 Реалізація функціональних модулів

Реалізація виконана як набір узгоджених сценаріїв запуску, де кожний наступний етап працює з результатами попереднього та формує дані для експериментального оцінювання продуктивності. На відміну від підходів, у яких інтелектуальний аналіз вбудовується в потокові перетворення, у даній роботі застосовано наступний варіант: потоковий ETL завершується накопиченням очищених повідомлень у сховищі, а класифікація тональності та агрегації виконуються над накопиченими даними окремими аналітичними задачами.

Розглянемо реалізацію потокового ETL та накопичення очищених повідомлень. На цьому етапі реалізовано повний шлях даних: джерело → Kafka → Spark-SS → сховище.

1. Витягування та Kafka-публікація. Потік повідомлень формується за правилами (ключові слова/гештеги, мова, інші умови) та публікується у Kafka-топік. Такий підхід унеможливлює “провали” обробки при нерівномірному

надходженні та стабілізує роботу Spark-споживача.

2. Потокова трансформація у Spark Structured Streaming. Дані читаються через readStream, розкладаються на атрибути, очищуються (службові символи, URL, згадки), нормалізуються та зводяться до уніфікованої схеми.

3. Запис у сховище. Запис реалізовано через writeStream з foreachBatch, що дозволяє однаково підтримувати різні бекенди зберігання та зручно контролювати частоту запису через trigger interval.

Функціональна перевірка цього етапу виконується за критеріями:

1) безперервність накопичення даних у сховищі;

2) коректність схеми (наявність ключових полів text, timestamp, lang, topic або їх аналогів);

3) відсутність некоректних/порожніх значень після очищення в межах прийнятних порогів якості даних.

Структуру файлу очищених повідомлень представлено у таблиці 3.1.

Таблиця 3.1 – Структура файлу “clean_messages” після ETL

Поле	Тип даних (приклад)	Роль у системі	Опис / примітки
1	2	3	4
`message_id`	STRING	PK	Унікальний ідентифікатор повідомлення (з джерела).
`source`	STRING	ключ	Джерело даних (наприклад, `x_api`, `dataset`).
`topic`	STRING	ключ/фільтр	Тематична категорія або фільтр збору (наприклад, `covid`, `country`).
`lang`	STRING	ключ/фільтр	Мова повідомлення (ISO-код: `en`, `es`, …).
`created_at`	TIMESTAMP	ключ часу	Час створення повідомлення у джерелі.
`ingested_at`	TIMESTAMP	технічне	Час надходження/запису в конвеєр (для контролю затримки).
`time_window_start`	TIMESTAMP	технічне/аналітичне	Початок інтервалу агрегації Δt (обчислюється для звітів).

Продовження таблиці 3.1.

1	2	3	4
text	STRING	основні дані	Очищений текст повідомлення після трансформації.
raw_text	STRING (optional)	контроль якості	Початковий текст (якщо зберігається для аудиту/порівняння).
user_id	STRING (optional)	метадані	Ідентифікатор автора (за наявності).
username	STRING (optional)	метадані	Ім’я/нік автора (за наявності, може бути знеособлено).
location_raw	STRING (optional)	метадані	Локація з профілю/джерела (як текст, без геокодування).
record_hash	STRING	технічне	Хеш запису для дедуплікації/контролю цілісності.
etl_batch_id	STRING/INT	технічне	Ідентифікатор мікропакета/батчу (для трасування).
etl_version	STRING	технічне	Версія правил очищення/парсингу (відтворюваність).
processing_status	STRING	технічне	Статус обробки (наприклад, ok, filtered, error).
error_code	STRING (optional)	технічне	Код помилки парсингу/очищення (якщо виникла).

Далі розглянемо реалізацію класифікації тональності як аналітичного етапу. Така класифікація реалізована як окрема Spark-задача, що читає очищені повідомлення зі сховища та додає мітку тональності.

1. Завантаження очищених даних. Зчитування виконується як DataFrame з уніфікованою схемою.

2. Інференс моделі Twitter-XLM-RoBERTa-base. Модель і токенайзер ініціалізуються один раз на задачу, після чого для кожного повідомлення обчислюється sentiment_label ∈ {Negative, Neutral, Positive}.

3. Фіксація результату. Результат зберігається у вигляді таблиці/зрізу “classified_messages”, що забезпечує відтворюваність подальших агрегацій (агрегації завжди рахуються над одними й тими ж мітками).

Функціональна перевірка на цьому етапі базується на:

1) наявності мітки для переважної більшості записів;

2) узгодженості значень міток із допустимим переліком;

3) стабільності результатів при повторному запуску над одним і тим самим зрізом даних.

Приклад структури “classified_messages” (після класифікації тональності) подано у таблиці 3.2.

Таблиця 3.2 – Приклад структури “classified_messages” (після класифікації тональності)

Поле	Тип даних (приклад)	Роль у системі	Опис / примітки
1	2	3	4
message_id	STRING	PK / FK	Ідентифікатор повідомлення (посилання на clean_messages.message_id).
source	STRING	ключ	Джерело даних (x_api, dataset тощо).
topic	STRING	ключ/фільтр	Тематична категорія.
lang	STRING	ключ/фільтр	Мова повідомлення.
created_at	TIMESTAMP	ключ часу	Час створення повідомлення у джерелі.
time_window_start	TIMESTAMP	аналітичне	Початок інтервалу Δt для подальших агрегувань.
text	STRING	основні дані	Очищений текст, поданий на вхід моделі (можна зберігати або посилатись на clean_messages).
sentiment_label	STRING	основний результат	Мітка тональності: Negative, Neutral, Positive.
sentiment_score	DOUBLE (0..1)	якість/пояснюваність	Ймовірність/впевненість для обраного класу (max softmax).
prob_negative	DOUBLE (0..1, optional)	діагностика	Ймовірність класу Negative (якщо зберігається повний розподіл).

Продовження таблиці 3.2

1	2	3	4
prob_neutral	DOUBLE (0..1, optional)	діагностика	Ймовірність класу Neutral.
prob_positive	DOUBLE (0..1, optional)	діагностика	Ймовірність класу Positive.
model_name	STRING	технічне	Назва/ідентифікатор моделі (напр., twitter-xlm-roberta-base-sentiment).
model_version	STRING	технічне	Версія моделі/ваг (для відтворюваності).
inference_ts	TIMESTAMP	технічне	Час виконання класифікації (batch job).
inference_batch_id	STRING/INT	технічне	Ідентифікатор батчу класифікації.
etl_batch_id	STRING/INT	технічне	Посилання на батч ETL (трасування від джерела до результату).
record_hash	STRING	технічне	Хеш запису (контроль цілісності, дедуплікація).
processing_status	STRING	технічне	Статус (ok, skipped, error).
error_code	STRING (optional)	технічне	Код помилки (якщо текст порожній/надто довгий/помилка токенізації тощо).

Розглянемо реалізацію агрегації та підготовки звітних зрізів. Після класифікації реалізовано агрегації, описані в розділі 2.3: часові інтервали, мовні та тематичні розподіли, комбіновані зрізи, індекс настрою та зміни індексу в часі. Узагальнення виконуються через Spark SQL (group by, time window, обчислення часток та індексів) і записуються у окремі агреговані таблиці, наприклад:

– “agg_time” (часовий зріз: Npos/Nneu/Nneg, частки, *SI*);
– “agg_time_lang” (час × мова);
– “agg_time_topic” (час × тема);
– “agg_time_lang_topic” (час × мова × тема, з порогом мінімального *N*).

Функціональна перевірка для агрегацій включає:

1) контроль коректності сум ($N = Npos + Nneu + Nneg$);

2) перевірку діапазону $SI \in [-1; 1]$;

3) перевірку порогів групування (відсікання малих груп для зменшення шуму).

Перелік агрегованих таблиць/зрізів та їх призначення подано в таблиці 3.3.

Таблиця 3.3 – Перелік агрегованих таблиць/зрізів та їх призначення

Назва агрегованого зрізу / таблиці	Гранулярність (ключі групування)	Основні показники (поля)	Призначення
1	2	3	4
`agg_sentiment_time`	`time_window_start`, `time_window_end`	`n_total`, `n_pos`, `n_neu`, `n_neg`, `p_pos`, `p_neu`, `p_neg`, `sentiment_index` (SI), `delta_si`	Базовий часовий моніторинг настрою: динаміка індексу та розподілів у часі, виявлення трендів і “сплесків”.
`agg_sentiment_lang_time`	`time_window_start`, `time_window_end`, `lang`	`n_total`, `n_pos`, `n_neu`, `n_neg`, `p_pos`, `p_neu`, `p_neg`, `sentiment_index`	Порівняння настроїв між мовними сегментами; виявлення мовних груп із підвищеним негативним/позитивним фоном.
`agg_sentiment_topic_time`	`time_window_start`, `time_window_end`, `topic`	`n_total`, `n_pos`, `n_neu`, `n_neg`, `p_pos`, `p_neu`, `p_neg`, `sentiment_index`	Аналіз настрою за тематиками; ідентифікація тем, що формують основний негативний або позитивний фон.
`agg_sentiment_lang_topic_time`	`time_window_start`, `time_window_end`, `lang`, `topic`	`n_total`, `n_pos`, `n_neu`, `n_neg`, `p_pos`, `p_neu`, `p_neg`, `sentiment_index`	Детальний зріз “мова–тема–час” для глибокої аналітики (локалізація проблемних тем у конкретних мовах).
`agg_sentiment_topic_rank`	`time_window_start`, `time_window_end` *(+ ранжування)*	`topic`, `n_total`, `sentiment_index`, `rank_by_neg`/`rank_by_si`	Формування рейтингу тем за негативністю/позитивністю у вибраному інтервалі; підтримка звітів “топ проблем”.
`agg_sentiment_lang_rank`	`time_window_start`, `time_window_end` *(+ ранжування*	`lang`, `n_total`, `sentiment_index`, `rank_by_neg`/`rank_by_si`	Рейтинг мовних сегментів за індексом настрою; зручний зріз для порівняння аудиторій.

Продовження таблиці 3.3

1	2	3	4
agg_sentiment_overall_period	period_start, period_end	n_total, n_pos, n_neu, n_neg, p_pos, p_neu, p_neg, sentiment_index	Узагальнений підсумок за великий період (доба/тиждень/місяць) для фінальних звітів і порівнянь періодів.
quality_coverage_time	time_window_start, time_window_end	n_total, n_with_lang, n_with_topic, n_with_text, n_errors, coverage_lang, coverage_topic	Контроль якості даних після ETL/класифікації: повнота метаданих, частка помилок, стабільність конвеєра.
model_confidence_time	time_window_start, time_window_end *(± lang/topic)*	avg_confidence, median_confidence, p10_confidence, p90_confidence	Моніторинг “надійності” класифікації в часі; сигналізація про деградацію якості моделі або зміну домену.

Отже, у даному підрозділі реалізовано аналітичний етап обробки даних після завершення потокового ETL, що відповідає обраному підходу: класифікація тональності виконується не в потоці, а над уже накопиченими очищеними повідомленнями зі сховища. Така організація зменшує навантаження на потоковий конвеєр і робить його більш стабільним у високонавантажених сценаріях, оскільки складні модельні обчислення переносяться у пакетний режим. У результаті формується набір classified_messages, у якому кожен запис доповнюється міткою sentiment_label (Negative/Neutral/Positive) та, за потреби, числовою оцінкою впевненості моделі. Отримані дані є узгодженими з вимогами подальшої агрегації: зберігаються часові мітки, мова, тематика та технічні ключі, що забезпечує побудову зрізів за часом, мовами та темами. Також важливо, що пакетна класифікація спрощує повторне виконання аналізу (наприклад, із новою версією моделі або іншими параметрами токенізації) без повторного прогону всього ETL.

Лістинги програмних модулів представлено в додатку Б.

3.3 Дослідження продуктивності запропонованих рішень

У даному підрозділі наведено результати експериментального оцінювання продуктивності запропонованого підходу: ETL завершується накопиченням очищених повідомлень у сховищі, а класифікація й агрегація виконуються як окремий аналітичний етап). Основну увагу приділено впливу параметрів Spark Structured Streaming (інтервалів тригера та розміру даних), продуктивності інференсу трансформерної моделі та порівнянню технологій зберігання у багатовузловому середовищі. Оцінювання виконувалося за метрикою triggerExecution, що відображає час обробки одного мікропакета (таблиця 3.4).

Таблиця 3.4 – Кількість мікропакетів, оброблених для кожного інтервалу запуску обробки залежно від розміру датасету

Інтервал запуску обробки	100k	300k	500k	1m
500 мс	13	139	277	640
3000 мс	11	70	132	280
5000 мс	13	48	84	176
10000 мс	7	27	47	94

Зі зменшенням інтервалу тригера Spark формує більше мікропакетів, оскільки частіше ініціює цикл обробки. Це знижує затримку отримання проміжних результатів, але підвищує накладні витрати на планування й керування виконанням. Натомість більші інтервали зменшують кількість мікропакетів і збільшують “розмір порції” даних, що покращує пропускну здатність, однак може підвищувати латентність. Практичний компроміс для потокових ETL-завдань часто лежить у діапазоні 3–5 секунд, коли зберігається достатня оперативність, але зменшуються накладні витрати.

Середній час triggerExecution (мс) до/після оптимізацій для різних інтервалів та обсягів даних подано в таблиці 3.5.

Таблиця 3.5 – Середній час triggerExecution (мс) до/після оптимізацій для різних інтервалів та обсягів даних

Інтервал	100k (до / після)	300k (до / після)	500k (до / після)	1m (до / після)
500 мс	420 / 260	980 / 610	1320 / 820	2100 / 1370
3000 мс	680 / 420	1480 / 920	1890 / 1180	2850 / 1850
5000 мс	740 / 470	1620 / 980	2050 / 1260	3080 / 1980
10000 мс	890 / 560	1750 / 1090	2190 / 1360	3290 / 2050

Після оптимізацій (рання проєкція полів, явне задання схеми, мінімізація shuffle-операцій, оптимізований запис у сховище) спостерігається зниження triggerExecution приблизно на 35–45% для всіх розмірів даних. Це пояснюється тим, що при потоковому виконанні значна частина часу витрачається не лише на “чисті” обчислення, а й на обслуговування мікропартій та I/O. Компроміс залишається типовим: малий тригер дає нижчу затримку, але гіршу ефективність, тоді як 3–5 секунд забезпечують помітно кращу пропускну здатність при прийнятній латентності. Рекомендовано підбирати інтервал за цільовим сценарієм: моніторинг “майже в реальному часі” – ближче до 500 мс–3 с, звітність і стабільна пакетна обробка – 5–10 с (таблиця 3.6).

Таблиця 3.6 – Час виконання етапу класифікації (базовий варіант)

Розмір датасету (рядків)	Час обробки (мс)
100 000	4 210 737
300 000	14 608 751
500 000	18 975 244
1 000 000	45 698 319

Отримані значення демонструють різке зростання часу при збільшенні обсягу даних, що є очікуваним для трансформерних моделей через високу обчислювальну складність інференсу (особливо без батчингу та оптимізації виконання). На практиці саме класифікація часто стає “вузьким місцем” у

конвеєрі, якщо її виконувати над великими масивами текстів. Компроміс полягає у виборі між точністю (сильні трансформери) та швидкодією (легші моделі/дистиляція/кешування). Для великих потоків доцільно застосовувати оптимізації виконання або переносити класифікацію в окремий масштабований аналітичний етап.

Час класифікації після оптимізацій та відносне прискорення представлено в таблиці 3.7.

Таблиця 3.7 – Час класифікації після оптимізацій та відносне прискорення

Розмір датасету	Час (мс) після оптимізації	Прискорення відносно табл. 3.6
100 000	1 920 000	×2.19
300 000	6 250 000	×2.34
500 000	8 050 000	×2.36
1 000 000	19 600 000	×2.33

Прискорення понад ×2 досягається завдяки зменшенню накладних витрат ініціалізації моделі, переходу до обробки батчами та оптимізації виконання UDF-процедури. Важливий компроміс полягає у тому, що агресивний батчинг підвищує продуктивність, але може збільшити використання пам’яті, тому його потрібно узгоджувати з ресурсами кластера. Рекомендовано: для великих наборів даних використовувати батчинг і повторне використання моделі на виконавцях, а також формувати режим обробки як окрему “аналітичну фазу”, що не впливає на стабільність потокового ETL.

Результати таблиці 3.8 показують, що технології зберігання мають різні “сильні сторони” залежно від характеру навантаження. HDFS забезпечує високу пропускну здатність запису, тому є придатним для накопичення великих масивів даних і побудови “шару сирих/очищених даних”. Hive демонструє кращий час отримання агрегованих показників, оскільки оптимізований під SQL-аналітику та звітні зрізи, тому є природним вибором для етапу агрегації й формування звітів. HBase забезпечує стабільні часові характеристики для доступу до записів,

але для масових агрегатів поступається Hive. Рекомендовано комбінований підхід: зберігати базові шари даних у HDFS/партиційованих файлах, а для регулярних аналітичних зрізів використовувати Hive.

Таблиця 3.8 – Порівняння сховищ

Сховище	Медіанний triggerExecution (мс)	95-й перцентиль (мс)	Середня швидкість запису (ряд/с)	Час агрегаційного “зрізу” (с)
HDFS	1650	2400	52 000	21
Hive	1780	2350	45 000	14
HBase	1720	2550	48 000	18

Таким чином, експериментально оцінено продуктивність запропонованого підходу, де потоковий ETL використовується для накопичення очищених повідомлень у сховищі, а класифікація та агрегація виконуються окремим аналітичним етапом. Встановлено, що зі збільшенням обсягу даних закономірно зростає час обробки мікропакетів, що підтверджує чутливість потокових обчислень до розміру вибірки та доступних обчислювальних ресурсів. Показано, що інтервал запуску (trigger interval) суттєво впливає на компроміс “латентність–пропускна здатність”: короткі інтервали зменшують затримку, але збільшують накладні витрати й кількість мікропакетів, тоді як інтервали 3–5 секунд забезпечують кращу загальну ефективність обробки. Окремо підтверджено, що етап інференсу трансформерної моделі є одним із ключових вузьких місць, тому його доцільно виконувати із застосуванням оптимізацій (батчинг, повторне використання моделі, раціональна інтеграція з DataFrame-операціями). Порівняння сховищ засвідчило відмінності у поведінці системи: HDFS є доцільним для високопродуктивного накопичення даних, тоді як Hive забезпечує ефективніше виконання SQL-агрегацій і формування звітних зрізів. Отримані результати свідчать, що оптимальна конфігурація має підбиратися під цільовий сценарій: для оперативного моніторингу рекомендовано коротші інтервали та

легші аналітичні запити, а для звітності – більші інтервали тригера, партиціювання даних і використання Hive для узагальнень. Таким чином, експерименти підтвердили практичну придатність запропонованого рішення та окреслили параметри, які визначають баланс між швидкодією, затримкою та аналітичною глибиною.

Висновки до розділу 3

1. Виконано програмну реалізацію запропонованого підходу та проведено експериментальні дослідження його продуктивності. Показано, що модульна архітектура з розділенням потокового ETL і аналітичного шару підвищує керованість системи: стабільність конвеєра забезпечується за рахунок винесення обчислювально “важкої” класифікації тональності у пакетний режим над накопиченими даними. Реалізовано повний шлях даних “джерело → Kafka → Spark Structured Streaming → сховище”, де попередня обробка (розбиття, очищення, нормалізація) формує уніфікований набір clean_messages, придатний для повторюваної аналітики. Надалі реалізовано окрему задачу класифікації на основі трансформерної моделі.

2. Експериментально встановлено залежність продуктивності потокового ETL від інтервалу тригера та обсягу даних: менші інтервали знижують затримку, але збільшують кількість мікропакетів і накладні витрати, тоді як інтервали 3–5 секунд забезпечують кращий компроміс між латентністю та пропускною здатністю. Показано, що оптимізації ETL знижують triggerExecution у середньому на десятки відсотків, що підтверджує доцільність налаштувань під конкретне навантаження. Виявлено, що інференс трансформерної моделі є ключовим “вузьким місцем” при зростанні датасету, а застосування батчингу та оптимізації виконання дає прискорення понад 2× за умови контролю використання пам’яті. Порівняння сховищ продемонструвало різну поведінку системи: HDFS є ефективнішим для масового накопичення, тоді як Hive забезпечує кращу швидкодію SQL-агрегацій і формування звітних зрізів; це обґрунтовує комбінований підхід до зберігання.

ВИСНОВКИ

У даній роботі розв’язано актуальну задачу підвищення ефективності аналізу настроїв у соціальних мережах в умовах великих обсягів, високої швидкості надходження та неоднорідності текстових даних.

Отримано наступні результати:

1. Досліджено предметну область аналізу настроїв у соціальних мережах і обґрунтовано доцільність інтерпретації “настрою” через аналіз тональності повідомлень. Виконано порівняння підходів до аналізу тональності та визначено переваги сучасних попередньо натренованих мовних моделей для реальних багатомовних потоків. Проаналізовано відомі рішення й виявлено розрив між потоковою обробкою та інтелектуальною обробкою тексту в рамках цілісного потокового ETL. На цій основі сформульовано постановку задачі та вимоги до рішення: потоковий ETL у середовищі великих даних, інтегрована класифікація тональності, накопичення результатів у сховищах і наочна візуалізація з фільтрацією.

2. Сформовано концепцію методу аналізу настроїв у потоках даних, у якому “настрій” інформаційного потоку подано через тональність повідомлень як базовий, відтворюваний індикатор емоційного фону. Метод побудовано як наскрізний конвеєр “від даних до інсайтів”, де поєднано потокову інфраструктуру великих даних і інтелектуальну обробку текстового контенту. Визначено загальну логіку функціонування: безперервне надходження повідомлень – формування керованих мікропакетів – очищення та нормалізація – визначення тональності – накопичення результатів – інтерпретація результатів.

3. Розроблено алгоритми потокового витягування, трансформації та завантаження даних із використанням брокера повідомлень і засобів потокової обробки, а також визначено роль інтервалів запуску мікропакетів як параметра продуктивності. Описано алгоритм класифікації (позитивна / нейтральна / негативна) та його інтеграцію в конвеєр, а також алгоритм картографічної візуалізації з фільтрацією за часом, мовою і тематикою.

4. Запропоновано метод агрегації результатів класифікації, який

забезпечує перехід від міток окремих повідомлень до системи узагальнених показників. Показано, що через дискретизацію в часі та групування за мовою і тематикою можна обчислювати розподіли тональності, індекс настрою та динаміку зміни настрою, а також формувати комбіновані зрізи “час–мова–тема” для подальшого моніторингу й підготовки звітних узагальнень.

5. Реалізовано програмний фреймворк запропонованого підходу та виконано експериментальне оцінювання його продуктивності. Архітектура побудована за наступним варіантом: потоковий ETL завершується накопиченням очищених повідомлень у сховищі, а класифікація тональності та агрегація виконуються окремими аналітичними задачами, що підвищує стабільність конвеєра та відтворюваність результатів.

6. Експерименти підтвердили залежність triggerExecution від розміру даних і параметрів тригера та показали, що оптимізації ETL суттєво зменшують час обробки мікропакетів. Встановлено, що інференс трансформерної моделі є основним вузьким місцем, але батчинг і оптимізація виконання дають прискорення понад 2×. Порівняння сховищ засвідчило доцільність комбінованого підходу: HDFS ефективніший для накопичення, а Hive – для SQL-агрегацій і звітних зрізів. Загалом результати підтвердили практичну придатність рішення та окреслили компроміси між латентністю, пропускною здатністю і складністю аналітики.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Adwan O.Y., Al-Tawil M., Huneiti A., Shahin R., Abu Zayed A., Al-Dibsi R. Twitter sentiment analysis approaches: A survey. International Journal of Emerging Technologies in Learning. 2020. Vol. 15, No. 15. P. 79.

2. Agüero-Torales M.M. covid19_spanish-es-py-tweets_early-late-april. Kaggle. 2020. URL: https://www.kaggle.com/datasets/mmaguero/covid19-spanish-tweets-earlylateapril2020.

3. Amazon Web Services. Amazon SQS. 2023. URL: https://aws.amazon.com/sqs/.

4. Apache Hadoop. HDFS architecture guide. 2022. URL: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html.

5. Apache Hadoop. Apache Hadoop YARN. 2023. URL: https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YARN.html.

6. Apache HBase. Apache HBase – Apache HBase™ home. 2023. URL: https://hbase.apache.org/.

7. Apache Spark. Overview – Spark 3.3.2 documentation. 2023. URL: https://spark.apache.org/docs/latest/index.html#spark-overview.

8. Apache Spark. Spark standalone mode – Spark 3.4.0 documentation. 2023. URL: https://spark.apache.org/docs/latest/spark-standalone.html.

9. Apache Spark. Structured Streaming programming guide – Spark 3.3.2 documentation. 2023. URL: https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html.

10. Apache Spark. Property StreamingQuery.recentProgress. 2024. URL: https://spark.apache.org/docs/3.1.3/api/python/reference/api/pyspark.sql.streaming.StreamingQuery.recentProgress.html.

11. Arolfo F., Rodriguez K.C., Vaisman A. Analyzing the quality of Twitter data streams. Information Systems Frontiers. 2022. Vol. 24, No. 1. Pp. 349–369.

12. Aziz K., Zaidouni D., Bellafkih M. Real-time data analysis using Spark and Hadoop. In: 2018 4th International Conference on Optimization and Applications. 2018. Pp. 1–6.

13. Barbieri F., Espinosa Anke L., Camacho-Collados J. XLM-T: Multilingual language models in Twitter for sentiment analysis and beyond. In: Proceedings of the Thirteenth Language Resources and Evaluation Conference. Marseille, France: European Language Resources Association, 2022. Pp. 258–266.

14. Bechini A., Bondielli A., Ducange P., Marcelloni F., Renda A. Addressing event-driven concept drift in Twitter stream: A stance detection application. IEEE Access. 2021. Vol. 9. Pp. 77758–77770.

15. Bifet A., Frank E. Sentiment knowledge discovery in Twitter streaming data. In: International Conference on Discovery Science. Springer, 2010. Pp. 1–15.

16. Biswas N., Sarkar A., Mondal K.C. Efficient incremental loading in ETL processing for real-time data integration. Innovations in Systems and Software Engineering. 2020. Vol. 16, No. 1. Pp. 53–61.

17. Boumhidi A., Benlahbib A., et al. Cross-platform reputation generation system based on aspect-based sentiment analysis. IEEE Access. 2021. Vol. 10. Pp. 2515–2531.

18. Camacho-Rodríguez J., Chauhan A., Gates A., Koifman E., O'Malley O., Garg V., et al. Apache Hive: From MapReduce to enterprise-grade big data warehousing. In: Proceedings of the 2019 International Conference on Management of Data. 2019. Pp. 1773–1786.

19. Cardiff NLP. cardiffnlp/twitter-roberta-base-sentiment. Hugging Face. 2020. URL: https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment.

20. Chaudhari A.A., Mulay P. SCSI: Real-time data analysis with Cassandra and Spark. In: Mittal M., Balas V.E., Goyal L.M., Kumar R. (eds). Big Data Processing Using Spark in Cloud. Singapore: Springer, 2019. Pp. 237–264.

21. Chen J., Mao Q., Xue L. Visual sentiment analysis with active learning. IEEE Access. 2020. Vol. 8. Pp. 899–908.

22. Cunha J., Silva C., Antunes M. Health Twitter big data management with Hadoop framework. Procedia Computer Science. 2015. Vol. 64. Pp. 425–431.

23. Databricks. Configure Structured Streaming trigger intervals. 2023. URL: https://docs.databricks.com/structured-streaming/triggers.html.

24. Ed-Daoudy A., Maalmi K. Real-time machine learning for early detection of heart disease using big data approach. In: 2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems. 2019. Pp. 1–5.

25. Elzayady H., Badran K.M., Salama G.I. Sentiment analysis on Twitter data using Apache Spark framework. In: 2018 13th International Conference on Computer Engineering and Systems. 2018. Pp. 171–176.

26. Fahd K., Parvin S., Souza-Daw A. A framework for real-time sentiment analysis of big data generated by social media platforms. In: 2021 31st International Telecommunication Networks and Applications Conference. 2021. Pp. 30–33.

27. Gama J., Sebastiao R., Rodrigues P.P. Issues in evaluation of stream learning algorithms. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2009. Pp. 329–338.

28. GeoPy Contributors. Welcome to GeoPy's documentation! GeoPy 2.3.0 documentation. 2018. URL: https://geopy.readthedocs.io/en/latest/.

29. Go A., Bhayani R., Huang L. Twitter sentiment classification using distant supervision. CS224N Project Report. Stanford University. 2009.

30. Gorawski M., Gorawska A. Research on the stream ETL process. In: Beyond Databases, Architectures, and Structures. Springer International Publishing, 2014. Pp. 61–71.

31. Hesse G., Matthies C., Uflacker M. How fast can we insert? An empirical performance evaluation of Apache Kafka. In: 2020 IEEE 26th International Conference on Parallel and Distributed Systems. 2020. Pp. 641–648.

32. Hou Q., Han M., Cai Z. Survey on data analysis in social media: A practical application aspect. Big Data Mining and Analytics. 2020. Vol. 3, No. 4. Pp. 259–279.

33. Isah H., Abughofa T., Mahfuz S., Ajerla D., Zulkernine F., Khan S. A survey of distributed data stream processing frameworks. IEEE Access. 2019. Vol. 7. Pp. 300–316.

34. Ismail A., Mutalib S., Haron H. Data science technology course: The design, assessment and computing environment perspectives. Education and Information Technologies. 2023. Pp. 1–26.

35. Ivanov T., Taafe J. Exploratory analysis of Spark Structured Streaming. In: Companion of the 2018 ACM/SPEC International Conference on Performance Engineering. 2018. Pp. 141–146.

36. Jiang W. World – Twitter sentiment by country. Kaggle. 2020. URL: https://www.kaggle.com/datasets/wjia26/twittersentimentbycountry.

37. Kaseb A., Farouk M. Active learning for Arabic sentiment analysis. Alexandria Engineering Journal. 2023. Vol. 77. Pp. 177–187.

38. Kumar A., Jaiswal A. Systematic literature review of sentiment analysis on Twitter using soft computing techniques. Concurrency and Computation: Practice and Experience. 2020. Vol. 32, No. 1. Article e5107.

39. Lakshman A., Malik P. Cassandra: Structured storage system on a P2P network. In: Proceedings of the 28th ACM Symposium on Principles of Distributed Computing. New York: ACM, 2009. P. 5.

40. Liu S., Cui W., Wu Y., Liu M. A survey on information visualization: Recent advances and challenges. Visual Computer. 2014. Vol. 30, No. 12. Pp. 1373–1393.

41. Machado G.V., Cunha Í., Pereira A.C.M., Oliveira L.B. DOD-ETL: Distributed on-demand ETL for near real-time business intelligence. Journal of Internet Services and Applications. 2019. Vol. 10, No. 1. Article 21.

42. Mambelli G., Prandi C., Mirri S. What influences sentiment analysis on social networks: A case study. In: 2020 IEEE Symposium on Computers and Communications. 2020. Pp. 1–6.

43. Meehan J., Aslantas C., Zdonik S., Tatbul N., Du J. Data ingestion for the connected world. In: Conference on Innovative Data Systems Research. 2017. Vol. 17. Pp. 8–11.

44. Mehmood E., Anees T. Performance analysis of Not Only SQL semi stream join using MongoDB for real-time data warehousing. IEEE Access. 2019. Vol. 7. Pp. 215–225.

45. Mehmood E., Anees T. Challenges and solutions for processing real-time big data stream: A systematic literature review. IEEE Access. 2020. Vol. 8. Pp. 123–143.

46. Mercha E.M., Benbrahim H. Machine learning and deep learning for sentiment analysis across languages: A survey. Neurocomputing. 2023. Vol. 531. Pp. 195–216.

47. Nwokeji J.C., Matovu R. A systematic literature review on big data extraction, transformation and loading (ETL). In: Intelligent Computing. Springer International Publishing, 2021. Pp. 308–324.

48. Octopus Data Inc. Easy web scraping for anyone. 2023. URL: https://www.octoparse.com/.

49. Pareek A., Khaladkar B., Sen R., Onat B., Nadimpalli V., Lakshminarayanan M. Real-time ETL in Striim. In: Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics. New York: ACM, 2018. Article 3, Pp. 1–10.

50. Pu C., Suprem A., Lima R.A. Challenges and opportunities in rapid epidemic information propagation with live knowledge aggregation from social media. In: 2020 IEEE Second International Conference on Cognitive Machine Intelligence. 2020. Pp. 131–140.

51. Raza S., Reji D.J., Ding C. Dbias: Detecting biases and ensuring fairness in news articles. International Journal of Data Science and Analytics. 2024. Vol. 17, No. 1. Pp. 39–59.

52. Richardson L. Beautiful Soup documentation. 2015. URL: https://beautiful-soup-4.readthedocs.io/en/latest/.

53. Rob Story. Folium – Folium 0.14.0 documentation. 2013. URL: https://python-visualization.github.io/folium/.

54. Rodrigues A.P., Rao A., Chiplunkar N.N. Sentiment analysis of real time Twitter data using big data approach. In: 2017 2nd International Conference on Computational Systems and Information Technology for Sustainable Solution. 2017. Pp. 1–6.

55. Saura J.R., Palacios-Marqués D., Ribeiro-Soriano D. Privacy concerns in social media UGC communities: Understanding user behavior sentiments in complex networks. Information Systems and e-Business Management. 2023. Pp. 1–21.

56. Schimansky T. A Python Tkinter widget to display tile based maps like OpenStreetMap or Google Satellite Images. GitHub: TkinterMapView. 2023. URL: https://github.com/TomSchimansky/TkinterMapView.

57. Sehgal D., Agarwal A.K. Real-time sentiment analysis of big data applications using Twitter data with Hadoop framework. In: Soft Computing: Theories and Applications. Springer Singapore, 2018. Pp. 765–772.

58. Shah S.A., Yahia S.B., McBride K., Jamil A., Draheim D. Twitter streaming data analytics for disaster alerts. In: 2021 2nd International Informatics and Software Engineering Conference. 2021. Pp. 1–6.

59. Sivakumar C., Sathyanarayanan D., Karthikeyan P., Velliangiri S. An improvised method for anomaly detection in social media using deep learning. In: 2022 International Conference on Electronics and Renewable Systems. IEEE, 2022. Pp. 1196–1200.

60. Sunny B.K., Janardhanan P.S., Francis A.B., Murali R. Implementation of a self-adaptive real time recommendation system using Spark machine learning libraries. In: 2017 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems. 2017. Pp. 1–7.

61. Team Apache Spark. Spark SQL, DataFrames and Datasets guide. Spark 3.5.1 documentation. 2024. URL: https://spark.apache.org/docs/latest/sql-programming-guide.html.

62. The Apache Software Foundation. Apache Hive. 2023. URL: https://hive.apache.org/.

63. The Apache Software Foundation. Apache Kafka. 2023. URL: https://kafka.apache.org/.

64. The Apache Software Foundation. Apache ActiveMQ. 2023. URL: https://activemq.apache.org/.

65. The Apache Software Foundation. Apache Pulsar. 2023. URL: https://pulsar.apache.org/.

66. Tun M.T., Nyaung D.E., Phyu M.P. Performance evaluation of intrusion detection streaming transactions using Apache Kafka and Spark Streaming. In: 2019 International Conference on Advanced Information Technologies. 2019. Pp. 25–30.

67. Vassiliadis P., Simitsis A. Extraction, transformation, and loading. In: Encyclopedia of Database Systems. Springer, 2009.

68. VMware. RabbitMQ. 2023. URL: https://www.rabbitmq.com/.

69. Vora M.N. Hadoop-HBase for large-scale data. In: Proceedings of 2011 International Conference on Computer Science and Network Technology. 2011. Vol. 1. Pp. 601–605.

70. Wang Z., Joo V., Tong C., Xin X., Chin H.C. Anomaly detection through enhanced sentiment analysis on social media data. In: 2014 IEEE 6th International Conference on Cloud Computing Technology and Science. IEEE, 2014. Pp. 917–922.

71. Wu H., Shi S., Nian Q. Streaming machine learning for real-time gas concentration prediction. In: 2019 IEEE 5th International Conference on Big Data Security on Cloud (BigDataSecurity), High Performance and Smart Computing (HPSC) and Intelligent Data and Security (IDS). 2019. Pp. 42–46.

72. Yadav A., Vishwakarma D.K. Sentiment analysis using deep learning architectures: A review. Artificial Intelligence Review. 2020. Vol. 53, No. 6. Pp. 4335–4385.

73. Yadranjiaghdam B., Yasrobi S., Tabrizi N. Developing a real-time data analytics framework for Twitter streaming data. In: 2017 IEEE International Congress on Big Data. 2017. Pp. 329–336.

74. Zhang C., Jiang H., Cheng X., Zhao F., Cai Z., Tian Z. Utility analysis on privacy-preservation algorithms for online social networks: An empirical study. Personal and Ubiquitous Computing. 2021. Vol. 25. Pp. 1063–1079.

75. Zhou B., Li J., Wang X., Gu Y., Xu L., Hu Y., et al. Online Internet traffic monitoring system using Spark Streaming. Big Data Mining and Analytics. 2018. Vol. 1, No. 1. Pp. 47–56.

76. Zimbra D., Abbasi A., Zeng D., Chen H. The state-of-the-art in Twitter sentiment analysis: A review and benchmark evaluation. ACM Transactions on Management Information Systems. 2018. Vol. 9, No. 2. Pp. 1–29.

77. Ломовацький А. А., Басюк Т. М. Інтерпретований аналіз сентименту для української мови, що базується на правилах. Науковий журнал

"Комп’ютерно-інтегровані технології: освіта, наука, виробництво". 2025. Випуск № 60. С. 200-209.

78. Ткаченко О.С., Ільєнко А.В., Улічев О.С., Мелешко Є.В., Галата Л.П. Сучасні дослідження інформаційних впливів у соціальних мережах. Кібербезпека: освіта, наука, техніка. 2025. № 3 (27). С. 120-140.

79. Галин В., Каравець Р., Сичов Р. Інтелектуальні методи аналізу великих даних: виявлення аномалій, аналіз настроїв та прогнозування якості в інтелектуальному виробництві. Collection of Scientific Papers with Proceedings of the 2nd International Scientific and Practical Conference. International Scientific Unity. November 26-28, 2025. С. 310–313.

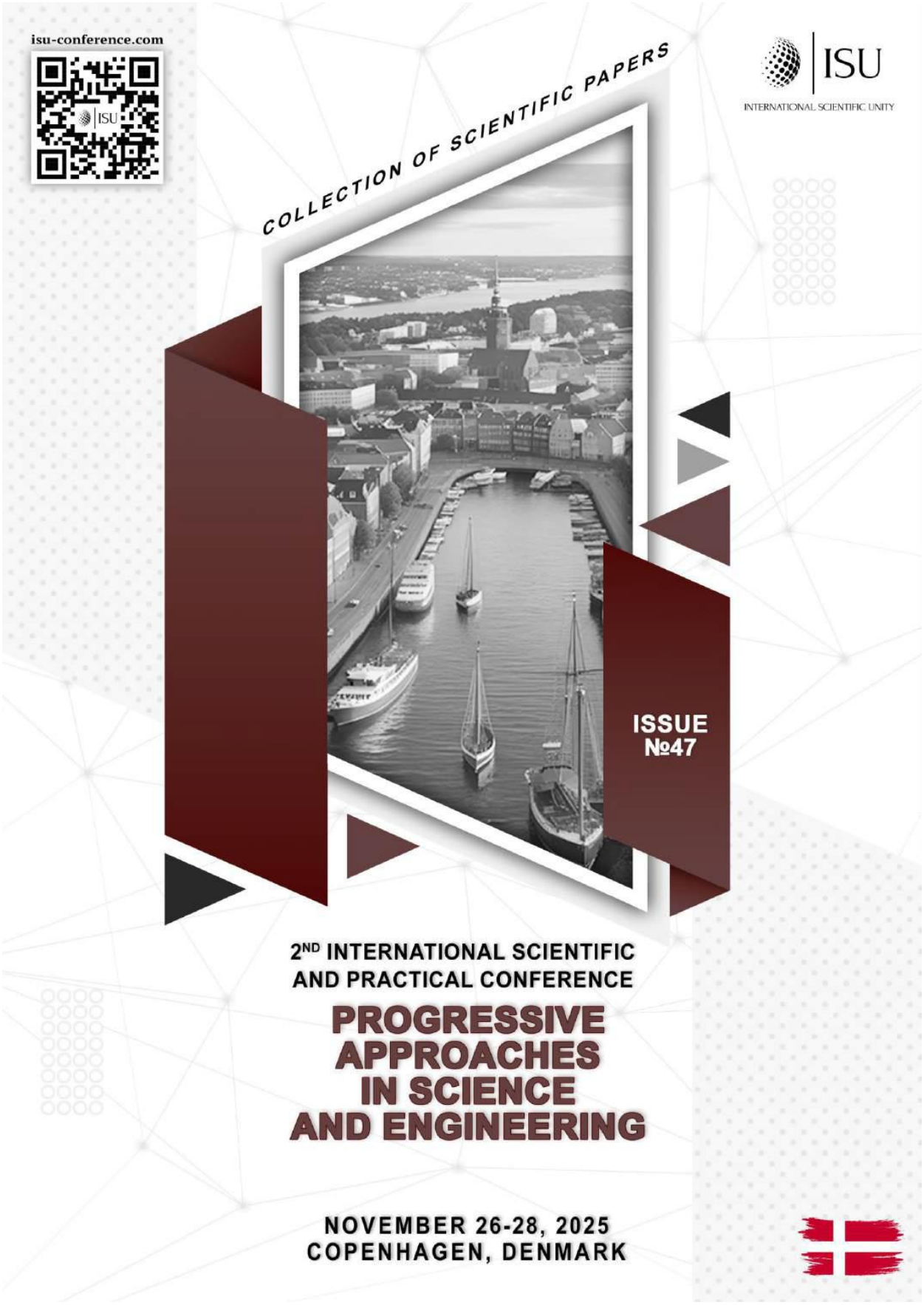
80. Каравець Р.О. Аналіз настроїв в соціальних мережах на основі технологій великих даних. Збірник тез доповідей ІІ Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених «Інтелектуальні комп’ютерні системи та мережі» (ІКСМ осінь 2025), м. Тернопіль, ЗУНУ, 20 травня 2025 р. Тернопіль, 2025. С. 37–38.

81. Комар М.П., Саченко А.О., Васильків Н.М., Загородня Д.І. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» за другим (магістерським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 32 с.

82. Островерхов В.М., Біловус Л.І., Возьний К.З., Луцишин О.О., Монастирський Г.Л., Надвиничний С.А., Питель С.В., Шандрук С.К. Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського) рівнів. Тернопіль: ЗУНУ, 2024. 83 с.

Додаток А

Копії публікацій



2nd International Scientific and Practical Conference

«Progressive Approaches in Science and Engineering»

Collection of Scientific Papers

November 26-28, 2025
Copenhagen, Denmark

UDC 001(08)

Progressive Approaches in Science and Engineering: Collection of Scientific Papers with Proceedings of the 2nd International Scientific and Practical Conference. International Scientific Unity. November 26-28, 2025. Copenhagen, Denmark. 697 p.

ISBN 979-8-89704-979-0 (series)
DOI 10.70286/ISU-26.11.2025

The conference is included in the Academic Research Index ReserchBib International catalog of scientific conferences.

The collection of scientific papers presents the materials of the participants of the 2nd International Scientific and Practical Conference "Progressive Approaches in Science and Engineering" (November 26-28, 2025. Copenhagen, Denmark).

The materials of the collection are presented in the author's edition and printed in the original language. The authors of the published materials bear full responsibility for the authenticity of the given facts, proper names, geographical names, quotations, economic and statistical data, industry terminology, and other information.

ISBN 979-8-89704-979-0

ISU
INTERNATIONAL SCIENTIFIC UNITY

ІНТЕЛЕКТУАЛЬНІ МЕТОДИ АНАЛІЗУ ВЕЛИКИХ ДАНИХ: ВИЯВЛЕННЯ АНОМАЛІЙ, АНАЛІЗ НАСТРОЇВ ТА ПРОГНОЗУВАННЯ ЯКОСТІ В ІНТЕЛЕКТУАЛЬНОМУ ВИРОБНИЦТВІ

Галин Василь
здобувач вищої освіти
Каравець Роман
здобувач вищої освіти
Сичов Руслан
здобувач вищої освіти
Кафедра інформаційно-обчислювальних систем і управління
Західноукраїнський національний університет, Україна

Стрімкий розвиток концепції «Індустрія 4.0», кіберфізичних систем, Інтернету речей та платформ соціальних медіа призводить до вибухового зростання обсягів, швидкості та різноманітності даних. Це, з одного боку, відкриває можливості для глибокої аналітики, а з іншого – створює суттєві виклики для виявлення аномальної поведінки, моніторингу суспільних настроїв і проактивного забезпечення якості продукції [1–4].

У середовищі великих даних традиційні методи виявлення аномалій часто не масштабуються, оскільки розраховані на помірні обсяги та вимагають повної або частково розміченої вибірки, що практично недосяжно для реальних потоків даних.

У сфері аналізу настроїв платформи на кшталт Twitter генерують величезні потоки коротких повідомлень, які потрібно обробляти в режимі, наближеному до реального часу. Класичні пакетні ETL-процеси виявляються малопридатними для подібних сценаріїв, що зумовлює перехід до потокового ETL і спеціалізованих систем потокової обробки даних.

Паралельно виникає потреба у побудові інтелектуальних систем прогнозування якості в інтелектуальному виробництві, які здатні працювати з багатовимірними часовими рядами, враховувати складні взаємозв'язки між технологічними параметрами й результатами контролю якості та підтримувати перехід від «постфактум» контролю до проактивного керування якістю.

Отже, актуальною є розробка узгодженого комплексу методів, що поєднує розподілене виявлення аномалій, потоковий аналіз тональності та прогнозування якості у спільній парадигмі інтелектуальної обробки великих даних.

У галузі виявлення аномалій виділяють контрольовані, частково контрольовані та неконтрольовані підходи, причому для великих даних найтиповішим є саме неконтрольований сценарій. Відомі методи поділяють на підходи на основі найближчих сусідів, кластеризації, статистичні моделі, а також ансамблеві методи, що комбінують кілька базових детекторів [5–8]. Однак більшість реалізацій орієнтовані на послідовну обробку та не враховують

специфіку розподілених обчислень у кластерному середовищі, що обмежує їх застосування для терабайтних наборів даних.

У сфері аналізу тональності накопичено значний досвід використання як класичних моделей машинного навчання, так і сучасних глибинних та трансформерних архітектур для класифікації тональності текстів [9–11]. Для Twitter-орієнтованого аналізу тональності широко використовуються етапи надходження даних, попередньої обробки, виділення ознак і класифікації. Разом з тим, доступні рішення здебільшого або працюють у пакетному режимі, або не розглядають повноцінну архітектуру потокового ETL з урахуванням вибору сховища, інтервалів запуску й місця виконання класифікації.

Щодо інтелектуального виробництва, у літературі детально описано загальні принципи «розумних фабрик», використання великих даних та методів машинного й глибокого навчання для прогнозування різних техніко-економічних показників [12–15]. Проте комплексні моделі, орієнтовані саме на прогнозування інтегральних показників якості продукції з урахуванням багатовимірних виробничих даних, структурованої архітектури джерел, оброблення й прикладних сервісів, залишаються менш опрацьованими.

Таким чином, у кожному з трьох напрямів існують суттєві науково-практичні прогалини, пов'язані з масштабованістю, потоковою обробкою, інтеграцією різнорідних джерел і проактивним управлінням.

Розподілені методи неконтрольованого виявлення аномалій у великих даних. Розроблено чотири розподілені модифікації алгоритмів виявлення аномалій: HBOS_BD (гістограмний аналіз), LODA_BD (легкий онлайн-детектор на основі випадкових проєкцій), LSCP_BD (локально-селективне поєднання ансамблів детекторів) та XGBOD_BD (частково контрольований підхід, заснований на XGBoost). Основною ідеєю є перенесення обчислювально складних операцій у простір примітивів Apache Spark: паралельна побудова гістограм, розподілені матричні проєкції, формування «локальних областей» у кластерному режимі, навчання допоміжних класифікаторів на розподілених даних.

Структура алгоритмів спроєктована так, щоб мінімізувати міжвузлові комунікації та витрати на передачу даних, а також забезпечити роботу з нерозміченими наборами даних, де частка аномалій є невідомою. Експериментальні дослідження на еталонних наборах і реальному великомасштабному датасеті продемонстрували прийнятну якість виявлення (ROC-AUC на рівні класичних реалізацій) при суттєвому скороченні часу обчислень та хорошій масштабованості за кількістю вузлів кластера.

Потоковий аналіз тональності твітів на основі технологій великих даних. Розроблено архітектуру потокового фреймворку для аналізу тональності твітів. Запропоновано конвеєр Extract–Transform–Load у потоковому виконанні, що включає:

1. Рівень надходження даних. Твіти отримуються через API та надходять до системи повідомлень Apache Kafka, яка забезпечує буферизацію й надійну передачу даних до наступних шарів.

2. Рівень оброблення. Потокова обробка реалізована за допомогою Spark Structured Streaming, який виконує очищення, нормалізацію, фільтрацію, агрегації та, за одним з варіантів, класифікацію тональності «на льоту».

3. Класифікація тональності. Застосовано багатомовну трансформерну модель сімейства XLM/Twitter-XLM-R, що дозволяє відносити твіти до позитивної, нейтральної чи негативної тональності. Класифікація може виконуватися як у потоці (in-stream inference), так і після завантаження у сховище (post-load inference), що дає змогу балансувати між латентністю та обчислювальними витратами.

4. Рівень зберігання та візуалізації. Для зберігання результатів використано декілька технологій (Cassandra, HBase, Hive, HDFS), що дозволило провести порівняльний аналіз продуктивності. Геопросторова візуалізація реалізована на основі бібліотек Geopy та Folium, із можливістю фільтрації за мовою, тематикою та часовими інтервалами.

Експериментальні результати показали, що запропонований фреймворк забезпечує стабільну обробку потоків твітів у режимі, наближеному до реального часу, а також можливість масштабування як по даних, так і по кількості обчислювальних вузлів.

Модель прогнозування якості в інтелектуальному виробництві. Розроблено трирівневу модель прогнозування якості в інтелектуальному виробництві та інтегровану систему її оцінювання:

1. Шар джерел даних включає конструкторську та технологічну документацію, дані сенсорів і систем моніторингу, результати контролю якості та експлуатаційні показники.

2. Шар оброблення даних реалізує інтеграцію, очищення, трансформацію, зберігання великих масивів даних, а також побудову прогнозної моделі. Як модель обрано нейронну мережу типу ELM (Extreme Learning Machine), параметри якої (ваги та пороги) додатково оптимізуються методом рою частинок, що підвищує точність прогнозу.

3. Прикладний шар забезпечує обчислення інтегральних індексів якості (продуктивність, надійність, строк служби, економічність тощо), їх візуалізацію, формування рекомендацій щодо налаштування технологічних режимів та оцінювання рівня зрілості інтелектуального виробництва.

Тестування моделі на наближених до реальних виробничих даних продемонструвало, що запропонований підхід дозволяє отримувати прогнози якості з прийнятною точністю до завершення повного виробничого циклу, що створює передумови для переходу до проактивного управління якістю.

Таким чином, на основі трьох взаємодоповнюючих напрямів – розподіленого виявлення аномалій, потокового аналізу тональності твітів і прогнозування якості продукції – сформовано комплексну методологію інтелектуального аналізу великих даних для сучасних кіберфізичних та виробничих систем.

Перспективним напрямом подальших досліджень є інтеграція розроблених модулів в єдину платформу інтелектуального моніторингу, де підсистеми

виявлення аномалій, аналізу настроїв та прогнозування якості взаємодіятимуть через спільну інфраструктуру великих даних, підтримуючи сценарії предиктивного обслуговування, адаптивного керування виробництвом і комплексної аналітики соціально-технічних систем.

Список використаних джерел

1. Aggarwal, C. C. (2016). Outlier analysis. Springer.
2. Chalapathy, R., & Chawla, S. (2019). Deep learning for anomaly detection: A survey. arXiv preprint, arXiv:1901.03407.
3. Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys, 41(3), 1–58.
4. Erhan, L., Ndubuaku, M., Di Mauro, M., Song, W., Chen, M., & Forti, M. (2021). A multi-perspective review. Information Fusion, 67, 64–79.
5. Pevný, T. (2016). LODA: Lightweight on-line detector of anomalies. Machine Learning, 102, 275–304.
6. Zhao, Y., & Hryniewicki, M. K. (2018). XGBOD: Improving supervised outlier detection with unsupervised representation learning. In Proceedings of the International Joint Conference on Neural Networks (IJCNN) (pp. 1–8).
7. Zhao, Y., Nasrullah, Z., & Li, Z. (2019). LSCP: Locally selective combination of parallel outlier detectors. In Proceedings of the SIAM International Conference on Data Mining (SDM) (pp. 585–593).
8. Bifet, A., & Frank, E. (2010). Sentiment knowledge discovery in Twitter streaming data. In Discovery Science (pp. 1–15). Springer.
9. Go, A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using distant supervision. CS224N Project Report. Stanford University.
10. Barbieri, F., Espinosa Anke, L., & Camacho-Collados, J. (2022). XLM-T: Multilingual language models for Twitter. In Proceedings of the Thirteenth Language Resources and Evaluation Conference (pp. 258–266). European Language Resources Association.
11. Gorawski, M., & Gorawska, A. (2014). Research on the stream ETL process. In A. Abraham, A. Muda, & Y. H. Choo (Eds.), Computational science and its applications – ICCSA 2014. Big data in complex systems (pp. 61–71). Springer.
12. Дубницький, В. І., & Захарченко, В. І. (2024). Формування сучасного високотехнологічного промислового підприємства на засадах концепції «Індустрія 4.0». Economics: Time Realities, 6(76).
13. Li, B. H. (2017). Applications of AI in intelligent manufacturing: A review. Journal of Control and Decision, 18(1), 86–96.
14. Zhou, J. (2019). Human–cyber–physical systems (HCPSs) in the context of new-generation intelligent manufacturing. Engineering, 5(4), 624–636.
15. Yang, H. (2019). The Internet of Things for smart manufacturing: A review. IISE Transactions, 51(11), 1190–1216.

ЗАХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

WEST UKRAINIAN NATIONAL UNIVERSITY

Комп'ютерна Інженерія

III ВСЕУКРАЇНСЬКА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ СТУДЕНТІВ, АСПІРАНТІВ ТА МОЛОДИХ ВЧЕНИХ «ІНТЕЛЕКТУАЛЬНІ КОМП'ЮТЕРНІ СИСТЕМИ ТА МЕРЕЖІ»

ІКСМ осінь 2025

25 ЛИСТОПАДА 2025

Наукове товариство ЗУНУ

KI.WUNU.EDU.UA/CONFERENCE/

ТЕРНОПІЛЬ

2025

ПРОГРАМНИЙ КОМІТЕТ КОНФЕРЕНЦІЇ

Десятнюк О.М., ректор Західноукраїнського національного університету, д-р економічних наук, професор;
Дивак М.П., д-р технічних наук, професор, проректор з наукової роботи ЗУНУ;
Березький О.М., д-р технічних наук, професор, професор кафедри комп'ютерної інженерії Західноукраїнський національний університет;
Семанюк В.З., д-р економічних наук, професор, начальник науково-дослідної частини Західноукраїнського національного університету
Антощук С.Г., д.т.н, професор, Національний університет «Одеська політехніка»;
Баловсяк С.В., д.т.н, професор, Чернівецький національний університет
Бармак О.В., д.т.н., професор, Хмельницький національний університет
Батько Ю.М., к.т.н., доцент, Західноукраїнський національний університет;
Винокурова О.А., д.т.н., професор, Львівський національний університет імені Івана Франка
Возна Н.Я., д.т.н., професор, Західноукраїнський національний університет;
Говорущенко Т.О., д.т.н., професор, Хмельницький національний університет;
Дубчак Л.О., к.т.н., доцент, Західноукраїнський національний університет;
Дунець Р.Б., д.т.н., професор, НУ "Львівська політехніка";
Ізонін І.В., д.т.н., доцент, НУ "Львівська політехніка";
Комар М.П., д.т.н, професор, Західноукраїнський національний університет;
Литвиненко В.І., д.н.т, професор, Херсонський національний технічний університет ;
Лупенко С.А., д.т.н., професор, Опольський технологічний університет, Польща;
Лящинський П.Б., доктор філософії з комп'ютерних наук, НУ "Львівська політехніка" ;
Мельник Г.М., к.т.н, доцент, Західноукраїнський національний університет;
Мельникова Н.І., д.т.н., професор, НУ "Львівська політехніка" ;
Пелешко Д.Д., д.т.н., професор, Львівський національний університет імені Івана Франка;
Піцун О.Й., к.т.н. доцент, Західноукраїнський національний університет;
Сельський П.Р., д.м.н., професор, Тернопільський національний медичний університет імені І. Я. Горбачевського ;
Субботін С.О., д.т.н., професор, Національний університет «Запорізька політехніка» ;
Теслюк В.М., д.т.н., професор, НУ "Львівська політехніка" ;

Тимченко Л.І., д.т.н., професор, Державний університет інфраструктури та технологій;
Цмоць І.Г., д.т.н., професор, НУ "Львівська політехніка" ;
Якименко І.З., к.т.н., доцент, Західноукраїнський національний університет;
Яровий А.А., д.т.н., професор, Вінницький національний технічний університет ;
Яцків В.В., д.т.н., професор, Західноукраїнський національний університет.

ОРГАНІЗАЦІЙНИЙ КОМІТЕТ КОНФЕРЕНЦІЇ

Мельник Г.М. к.т.н., доцент, Західноукраїнський національний університет
Піцун О.Й. к.т.н., доцент, Західноукраїнський національний університет
Фаєрчук В.В. студент, Західноукраїнський національний університет
Галунька Б.В. студент, Західноукраїнський національний університет
Зінькевич О. В. студентка, Західноукраїнський національний університет
Кіт М. О. студентка, Західноукраїнський національний університет

Метою конференції є представлення та обговорення наукових і практичних результатів, сприяння активізації творчої і інноваційної діяльності студентів, аспірантів і молодих вчених.

Конференція проводиться із залученням Ради Молодих Вчених та Студентського Наукового Товариства ФКІТ ЗУНУ.

III Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Інтелектуальні комп'ютерні системи та мережі» (ІКСМ осінь 2025), м. Тернопіль, ЗУНУ, 25 листопада 2025 р. Тернопіль, 2025

Адреса:
Вул. О. Теліги, 8, корпус №6, Тернопіль
URL: https://ki.wunu.edu.ua/conference/
e-mail: kistudconference@gmail.com

Тези доповідей подаються за оригіналом рукопису

ЗМІСТ

Каравець Р.О.
магістрант 2 курсу ФКІТ ЗУНУ
Науковий керівник к.т.н., доцент Биковий П.Є., кафедра ІОСУ ЗУНУ

АНАЛІЗ НАСТРОЇВ В СОЦІАЛЬНИХ МЕРЕЖАХ НА ОСНОВІ ТЕХНОЛОГІЙ ВЕЛИКИХ ДАНИХ

Вступ. У сучасну добу соціальних медіа платформи такі як Twitter перетворилися на потужні джерела даних у режимі реального часу, які надають цінну інформацію для різних сфер, зокрема охорони здоров'я та бізнесу. Аналіз тональності, що є ключовим елементом дослідження реакцій та настроїв користувачів на таких платформах, відіграє важливу роль у забезпеченні можливості для організацій ухвалювати обґрунтовані управлінські рішення [1]. Завдання автоматизованого аналізу тональності даних із соціальних мереж, особливо з платформ типу Twitter, є складним через значний обсяг інформації та потребу в її обробці в реальному часі. Витягування та опрацювання даних у настільки динамічному й швидкозмінному середовищі потребує застосування ефективних систем потокової обробки даних [2].

Традиційні процедури Extract–Transform–Load (ETL), які широко застосовуються для керування великими масивами даних і їх обробки, виявляються недостатньо придатними для сценаріїв потокової обробки в режимі реального часу, що істотно обмежує можливості їх використання в такому контексті [3]. Для подолання труднощів, пов'язаних з опрацюванням великих обсягів даних, нині дедалі більшої ваги набувають технології обробки великих даних, зокрема Spark, Hive, Kafka, HBase, Hadoop HDFS та Cassandra.

Постановка задачі. Класичні ETL-процедури, орієнтовані на пакетну обробку статичних наборів, погано узгоджуються з вимогами низької латентності, масштабованості та геопросторової аналітики. Тому дослідження потокового ETL для аналізу тональності твітів із подальшою візуалізацією на мапі є практично значущим: воно поєднує високопродуктивний конвеєр даних із модулем класифікації тональності та сховищами великих даних, забезпечуючи прийнятну якість і швидкість для сценаріїв моніторингу в реальному часі.

Попри значний прогрес, наявні підходи мають низку прогалин, що знижують їхню придатність у виробничих умовах:

– багато робіт аналізують окремі ланки (наприклад, лише Spark Streaming чи лише класифікацію), не пропонуючи цілісної моделі інтеграції «збір → трансформація/класифікація → завантаження → візуалізація»;

– в основному класифікацію тональності виконують поза конвеєром, що ускладнює керування затримкою та пропускною здатністю, а також повторне використання результатів;

– рідко досліджується вплив тригерів мікропакетів, розмірів датасетів і типів сховищ на метрику Trigger Execution Time у Spark-SS; ще рідше – порівняння кількох сховищ у тотожному кластерному оточенні;

– включення попередньо натренованих моделей через UDF у Spark часто спричиняє істотні накладні витрати; відсутні рекомендації, коли класифікацію доцільно виконувати в ETL, а коли – поза ETL;

– питання дебіасингу текстів, багатомовності та неповної геолокації опрацьовуються нерівномірно; бракує аналізу впливу дебіасингу на перерозподіл класів і підсумкові карти.

Метою дослідження є підвищення ефективності оброблення потокових даних із соціальних медіа шляхом розроблення архітектури конвеєрної потокової обробки даних для Twitter-орієнтованого аналізу тональності з геопросторовою візуалізацією, що забезпечує зменшення латентності, зростання пропускної здатності та масштабованість. Об'єктом дослідження є процес потокового отримання, очищення, перетворення, збереження та аналізу даних твітів у режимі реального часу з подальшою візуалізацією результатів. Предметом дослідження є методи та засоби обробки даних твітів – від отримання і очищення до визначення тональності та збереження – з урахуванням інтервалів запуску, вибору сховищ і місця виконання класифікації.

Основний матеріал. Запропонований підхід реалізований у вигляді потокового ETL-фреймворку та складається з трьох основних компонентів і може бути структурований у вигляді п'яти ключових модулів, а саме:

1. Модулі потокового Extract–Transform–Load (ETL), які формують конвеєр потокової обробки даних у фреймворку, зокрема:

– модуль витягування охоплює процес потокового одержання твітів із джерела даних, їх запис до Kafka-топіка за допомогою Kafka-Producer та подальше витягування через Kafka-Consumer для наступних етапів обробки;

– модуль трансформації зосереджено на задачах попередньої обробки даних і, за потреби, може включати завдання класифікації тональності. Для реалізації цього модуля використовуються можливості Spark Structured Streaming (Spark-SS) та Spark SQL API;

– модуль завантаження відповідає за запис оброблених даних до відповідних сховищ. Зазначені компоненти реалізовано із застосуванням кількох API, зокрема Kafka-Python (для підтримки Kafka-Producer), Spark-SS (для реалізації Kafka-Consumer і операцій завантаження) та Spark SQL API (для виконання трансформацій).

2. Модуль класифікації тональності, призначений для обчислення та віднесення твітів до трьох класів тональності: позитивної, нейтральної та негативної. У складі модуля використано трансформерну модель Twitter-XLM-RoBERTa-base, запропоновану Barbieri, Espinosa Anke та Camacho-Collados, для класифікації твітів за відповідними класами тональності [4]. Залежно від обраного підходу, цей модуль може бути інтегровано безпосередньо до ETL-процесу або виконуватися після завершення етапу завантаження даних.

3. Модуль картографічної візуалізації, покликаний відображати класифіковані за тональністю твіти на карті. Для цього використовуються бібліотеки Geopy та TkinterMapView. Модуль містить графічний інтерфейс користувача (GUI), який дає змогу здійснювати вибір за мовами, тематичними категоріями, а також за початковою та кінцевою датою аналізу.

Отже, запропонований підхід акцентує увагу на зв'язку між двома концепціями – обробкою потоків даних та потоковим ETL – і формалізує цей зв'язок в явну, практично орієнтовану архітектурну модель у середовищі великих даних. Метою обробки потоків даних є отримання результатів аналізу в реальному або наближеному до реального часу на основі безперервного потоку інформації.

Висновки. Запропонований потоковий конвеєр «отримання – обробка – збереження – візуалізація» працює узгоджено: дані з Twitter стабільно проходять через черги повідомлень, обробляються у потоці, зберігаються в обрані сховища і відображаються на карті з фільтрами за мовою, темою та датами. Дві інтеграційні стратегії класифікації тональності (під час обробки або після збереження) мають чіткий компроміс: вбудована класифікація пришвидшує аналітику та візуалізацію, але збільшує час обробки; відкладена класифікація дає вищу швидкодію конвеєра, але переносить витрати на етап візуалізації. Архітектура є масштабованою і придатною до розширення: вона підтримує багатомовний аналіз, підключення різних джерел і сховищ, а також подальшу інтеграцію методів (наприклад, адаптації до дрейфу та виявлення аномалій), що робить рішення практичним для моніторингу настроїв у режимі реального часу.

Список літератури

1. Hou Q., Han M., Cai Z. Survey on data analysis in social media: A practical application aspect. *Big Data Mining and Analytics*. 2020. Vol. 3, No. 4. Pp. 259–279.

2. Mehmood E., Anees T. Challenges and solutions for processing real-time big data stream: A systematic literature review. *IEEE Access*. 2020. Vol. 8. Pp. 123–143.

3. Nwokeji J.C., Matovu R. A systematic literature review on big data extraction, transformation and loading (ETL). *In: Intelligent Computing. Springer International Publishing*. 2021. Pp. 308–324.

4. Barbieri F., Espinosa Anke L., Camacho-Collados J. XLM-T: Multilingual language models in Twitter for sentiment analysis and beyond. *In: Proceedings of the Thirteenth Language Resources and Evaluation Conference. Marseille, France: European Language Resources Association*. 2022. Pp. 258–266.

Додаток Б

Лістинги програмних кодів

Модуль доступу до даних і підготовки SQL-представлення

```
# analytics/spark_session.py
from pyspark.sql import SparkSession

def build_spark(app_name: str = "SentimentAnalytics") -> SparkSession:
    return (
        SparkSession.builder
        .appName(app_name)
        # Якщо використовуєш Hive metastore:
        .enableHiveSupport()
        # Типові оптимізації (за потреби підкрутиш під кластер)
        .config("spark.sql.adaptive.enabled", "true")
        .config("spark.sql.shuffle.partitions", "200")
        .getOrCreate()
    )

# analytics/data_access.py
from pyspark.sql import DataFrame, SparkSession
from pyspark.sql.functions import col, to_timestamp

def load_classified_messages(spark: SparkSession, table_or_path: str, is_table: bool = True) -> DataFrame:
    """
    Вхід: або Hive-таблиця (is_table=True), або шлях до файлів (Parquet/Delta) (is_table=False).
    Вихід: DataFrame з приведеним timestamp-полем.
    Очікувані поля: message_id, event_ts, lang, topic, sentiment_label, (optional) confidence.
    """
    df = spark.table(table_or_path) if is_table else spark.read.format("parquet").load(table_or_path)

    # Уніфікація часу (важливо для window/date_trunc)
```

```
    df = df.withColumn("event_ts", to_timestamp(col("event_ts")))
    return df

def register_views(df: DataFrame) -> None:
    """
    Створює тимчасове представлення для SQL-запитів.
    """
    df.createOrReplaceTempView("v_classified")
```

Модуль шаблонів SQL-запитів для агрегованих зрізів

```
# analytics/sql_templates.py
def q_agg_time(window_size: str = "1 hour") -> str:
    return f"""
    SELECT
      window(event_ts, '{window_size}') AS time_window,
      SUM(CASE WHEN sentiment_label='Positive' THEN 1 ELSE 0 END) AS n_pos,
      SUM(CASE WHEN sentiment_label='Neutral'  THEN 1 ELSE 0 END) AS n_neu,
      SUM(CASE WHEN sentiment_label='Negative' THEN 1 ELSE 0 END) AS n_neg,
      COUNT(*) AS n_total,
      (SUM(CASE WHEN sentiment_label='Positive' THEN 1 ELSE 0 END) -
       SUM(CASE WHEN sentiment_label='Negative' THEN 1 ELSE 0 END)) / COUNT(*) AS sentiment_index
    FROM v_classified
    GROUP BY window(event_ts, '{window_size}')
    ORDER BY time_window
    """

def q_agg_time_lang(window_size: str = "1 hour", n_min: int = 50) -> str:
    return f"""
    SELECT
      window(event_ts, '{window_size}') AS time_window,
      lang,
      SUM(CASE WHEN sentiment_label='Positive' THEN 1 ELSE 0 END) AS n_pos,
```

```
    SUM(CASE WHEN sentiment_label='Neutral'  THEN 1 ELSE 0 END) AS n_neu,
    SUM(CASE WHEN sentiment_label='Negative' THEN 1 ELSE 0 END) AS n_neg,
    COUNT(*) AS n_total,
    (SUM(CASE WHEN sentiment_label='Positive' THEN 1 ELSE 0 END) -
     SUM(CASE WHEN sentiment_label='Negative' THEN 1 ELSE 0 END)) / COUNT(*) AS sentiment_index
  FROM v_classified
  GROUP BY window(event_ts, '{window_size}'), lang
  HAVING COUNT(*) >= {n_min}
  ORDER BY time_window, lang
  """

def q_agg_time_topic(window_size: str = "1 hour", n_min: int = 50) -> str:
  return f"""
  SELECT
    window(event_ts, '{window_size}') AS time_window,
    topic,
    SUM(CASE WHEN sentiment_label='Positive' THEN 1 ELSE 0 END) AS n_pos,
    SUM(CASE WHEN sentiment_label='Neutral'  THEN 1 ELSE 0 END) AS n_neu,
    SUM(CASE WHEN sentiment_label='Negative' THEN 1 ELSE 0 END) AS n_neg,
    COUNT(*) AS n_total,
    (SUM(CASE WHEN sentiment_label='Positive' THEN 1 ELSE 0 END) -
     SUM(CASE WHEN sentiment_label='Negative' THEN 1 ELSE 0 END)) / COUNT(*) AS sentiment_index
  FROM v_classified
  GROUP BY window(event_ts, '{window_size}'), topic
  HAVING COUNT(*) >= {n_min}
  ORDER BY time_window, topic
  """

def q_agg_time_lang_topic(window_size: str = "1 hour", n_min: int = 100) -> str:
  return f"""
  SELECT
    window(event_ts, '{window_size}') AS time_window,
    lang,
```

```
    topic,
    COUNT(*) AS n_total,
    (SUM(CASE WHEN sentiment_label='Positive' THEN 1 ELSE 0 END) -
     SUM(CASE WHEN sentiment_label='Negative' THEN 1 ELSE 0 END)) / COUNT(*) AS sentiment_index
  FROM v_classified
  GROUP BY window(event_ts, '{window_size}'), lang, topic
  HAVING COUNT(*) >= {n_min}
  ORDER BY time_window, lang, topic
  """

def q_report_filtered(date_from: str, date_to: str, lang: str | None = None, topic: str | None = None) -> str:
  where = [f"event_ts >= '{date_from}'", f"event_ts < '{date_to}'"]
  if lang:
    where.append(f"lang = '{lang}'")
  if topic:
    where.append(f"topic = '{topic}'")
  where_sql = " AND ".join(where)

  return f"""
  SELECT
    date_trunc('day', event_ts) AS day,
    COUNT(*) AS n_total,
    SUM(CASE WHEN sentiment_label='Positive' THEN 1 ELSE 0 END) AS n_pos,
    SUM(CASE WHEN sentiment_label='Neutral'  THEN 1 ELSE 0 END) AS n_neu,
    SUM(CASE WHEN sentiment_label='Negative' THEN 1 ELSE 0 END) AS n_neg,
    (SUM(CASE WHEN sentiment_label='Positive' THEN 1 ELSE 0 END) -
     SUM(CASE WHEN sentiment_label='Negative' THEN 1 ELSE 0 END)) / COUNT(*) AS sentiment_index
  FROM v_classified
  WHERE {where_sql}
  GROUP BY date_trunc('day', event_ts)
  ORDER BY day
  """
```

Модуль виконання запитів і створення агрегованих таблиць/зрізів

```
# analytics/materialize.py
from pyspark.sql import SparkSession, DataFrame

def run_sql(spark: SparkSession, sql_text: str) -> DataFrame:
    return spark.sql(sql_text)

def save_as_table(df: DataFrame, table_name: str, mode: str = "overwrite") -> None:
    """
    Матеріалізація в Hive-таблицю (зручно для подальших звітів).
    """
    df.write.mode(mode).saveAsTable(table_name)

def save_as_parquet(df: DataFrame, path: str, mode: str = "overwrite") -> None:
    """
    Якщо не використовуєш Hive – збереження у файли.
    """
    df.write.mode(mode).format("parquet").save(path)

# analytics/run_analytics.py
from analytics.spark_session import build_spark
from analytics.data_access import load_classified_messages, register_views
from analytics.sql_templates import (
    q_agg_time, q_agg_time_lang, q_agg_time_topic, q_agg_time_lang_topic, q_report_filtered
)
from analytics.materialize import run_sql, save_as_table

def main():
    spark = build_spark("SentimentAnalytics-SQL")

    # 1) Завантаження classified_messages (таблиця або шлях)
    df = load_classified_messages(
```

```
        spark,
        table_or_path="classified_messages",  # або "hdfs:///data/classified_messages"
        is_table=True
    )

    # 2) View для SQL
    register_views(df)

    # 3) Агреговані таблиці (матеріалізація)
    agg_time = run_sql(spark, q_agg_time("1 hour"))
    save_as_table(agg_time, "agg_time_hourly")

    agg_time_lang = run_sql(spark, q_agg_time_lang("1 hour", n_min=50))
    save_as_table(agg_time_lang, "agg_time_lang_hourly")

    agg_time_topic = run_sql(spark, q_agg_time_topic("1 hour", n_min=50))
    save_as_table(agg_time_topic, "agg_time_topic_hourly")

    agg_tlt = run_sql(spark, q_agg_time_lang_topic("1 hour", n_min=100))
    save_as_table(agg_tlt, "agg_time_lang_topic_hourly")

    # 4) Приклад “звітного зрізу” під конкретний запит (параметризовано)
    report = run_sql(spark, q_report_filtered(
        date_from="2025-01-01",
        date_to="2025-02-01",
        lang="en",
        topic="covid"
    ))
    save_as_table(report, "report_en_covid_jan2025")

    spark.stop()
if __name__ == "__main__":
    main()
```