ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

УДК 004.9

doi: 10.32620/aktt.2026.2.13

В. С. ЧЕРНИЩУК, О. І. МОРОЗОВА

Національний аерокосмічний університет «Харківський авіаційний інститут», Харків, Україна

СИСТЕМА АВТОМАТИЗОВАНОГО АНАЛІЗУ СЕРВЕРНИХ ЛОГІВ ІЗ ЗАСТОСУВАННЯМ МЕТОДІВ NLP ТА АЛГОРИТМІВ СТИСНЕННЯ ТЕКСТОВОЇ ІНФОРМАЦІЇ

Предметом *вивчення в статті є процеси автоматизованого аналізу серверних логів програмних систем із використанням методів обробки природної мови.* ***Метою*** *є розробка підходу до підвищення ефективності аналізу логів шляхом застосування алгоритмів NLP та методів стиснення текстових даних.* ***Завдання****: дослідити особливості логів як джерела технічної інформації; проаналізувати сучасні методи обробки текстових даних, включаючи статистичні, векторні та нейромережеві підходи; обґрунтувати доцільність використання алгоритмів NLP для автоматизації аналізу логів; обґрунтувати застосування методів стиснення тексту для зменшення обсягу даних; розробити концепцію та архітектуру системи автоматизованого аналізу логів.* ***Використовуваними методами*** *є: методи обробки природної мови, зокрема TF-IDF, Word2Vec, FastText та трансформерні моделі; методи попередньої обробки тексту; підходи до стиснення логів і виділення шаблонів повідомлень; методи пошуку інформації у текстових масивах.* ***Отримані такі результати****. Розроблено концепцію системи автоматизованого аналізу серверних логів із застосуванням методів NLP та алгоритмів стиснення текстових даних. Запропоновано узагальнену архітектуру системи, що включає модулі збору, збереження, фільтрації, попередньої обробки, стиснення тексту, NLP-аналізу, формування подання інциденту та пошуку рішень. Розроблено алгоритм функціонування системи, який забезпечує поетапну обробку логів з урахуванням їх текстової природи та значного обсягу даних. Встановлено, що застосування методів NLP дозволяє підвищити точність виявлення помилок і класифікації інцидентів, а використання стиснення тексту сприяє зменшенню обчислювального навантаження та підвищенню продуктивності системи.* ***Висновки****. Запропонований підхід забезпечує підвищення ефективності діагностики програмних систем, скорочення часу виявлення та усунення помилок, а також зменшення навантаження на розробників.* ***Наукова новизна*** *отриманих результатів полягає в наступному: розроблено підхід до автоматизованого аналізу логів із поєднанням методів NLP та стиснення текстових даних; запропоновано узагальнену архітектуру системи аналізу логів, що враховує особливості неструктурованих текстових даних; отримали подальший розвиток методи обробки логів шляхом інтеграції сучасних моделей обробки природної мови з етапами оптимізації обсягу даних, що дозволяє підвищити ефективність аналізу в умовах великих інформаційних потоків.*

Ключові слова*: аналіз логів; обробка природної мови; NLP-алгоритми; стиснення тексту; інтелектуальні системи; діагностика програмних систем; обробка даних; відмовостійкість.*

Вступ

Робота сучасного додатку завжди супроводжується журналом логів системи. Загалом логи – це спеціальний файл, у якому накопичується зібрана службова та статистична інформація про події в системі/програмі. Операційні системи (особливо це стосується серверних ОС) та серверне програмне забезпечення зазвичай мають розвинуту систему ведення логів. За допомогою них можна змусити систему реєструвати у лог-файлах фактично будь-які події. Відповідно різні типи подій, різну інформацію можна зберігати у своїх спеціалізованих логах [1].

Актуальність аналізу логів особливо зростає у високонадійних технічних системах, зокрема в аерокосмічній галузі. Сучасні авіаційні та космічні комплекси, включаючи бортові обчислювальні системи, наземні центри управління та системи обробки телеметрії, генерують значні обсяги службової інформації у вигляді логів. Такі журнали подій виконують функцію «чорного ящика» цифрової інфраструктури, забезпечуючи фіксацію станів систем, відхилень у роботі та критичних подій. Умови обмеженого часу реакції, підвищених вимог

до надійності та складності взаємодії підсистем роблять ефективний аналіз логів ключовим фактором забезпечення безпеки та стабільності аерокосмічних систем.

Сучасний стан проблеми характеризується стрімким зростанням обсягів лог-даних у розподілених і мікросервісних системах. Навіть невеликі за масштабом програмні рішення можуть генерувати тисячі записів щодня, тоді як великі інформаційні системи – мільйони подій у реальному часі. Існуючі підходи до аналізу логів, що базуються на ручному перегляді або простих методах фільтрації та пошуку, часто не забезпечують достатньої ефективності в умовах таких обсягів даних. Це створює потребу у впровадженні автоматизованих інтелектуальних методів обробки текстової інформації.

Мотивацією дослідження стало у спробі впровадити технології ШІ в аналіз та автоматизацію пошуку рішень проблеми за інформацією в логах системи. Особливої актуальності це набуває у системах критичного призначення, де затримка у виявленні проблеми може призвести до значних втрат або відмов системи.

Логи аналізує не тільки розробник додатку, а ще DevOps, SRE інженер [2]. Розробники використовують їх для відстеження помилок та налагодження коду. Системні адміністратори – для моніторингу роботи серверів та інфраструктури. DevOps-фахівці покладаються на логи під час автоматизованого розгортання, оновлення та масштабування систем. Аналітики безпеки застосовують логи для виявлення підозрілої активності чи потенційних кіберзагроз. Усе це робить логи ключовим джерелом правди – своєрідним «чорним ящиком» будь-якої IT-системи.

Одним із перспективних напрямів у цьому контексті є застосування методів обробки природної мови (Natural Language Processing, NLP). На відміну від традиційних інструментів пошуку та фільтрації, NLP дозволяє системам «розуміти» зміст логів: виділяти ключові слова, класифікувати повідомлення за змістом, визначати схожі інциденти або навіть автоматично формувати запити до баз знань для пошуку рішень. Такий підхід є особливо актуальним для аерокосмічних систем, де аналіз великих обсягів телеметричних та службових повідомлень повинен виконуватися швидко, точно та з мінімальним втручанням людини.

Метою цієї статті є аналіз підходів до обробки логів програмних додатків у сучасних IT-системах, визначення ключових напрямів їхнього застосування, а також дослідження можливостей впровадження алгоритмів обробки природної мови для підвищення ефективності аналізу.

Запропонований підхід ґрунтується на інтеграції методів NLP у процес аналізу логів із попереднім етапом стиснення текстових повідомлень, що дозволяє зменшити обсяг даних та підвищити ефективність обробки. У роботі розглядається узагальнена архітектура системи автоматизованого аналізу логів та алгоритм її функціонування.

У першому розділі розглядаються особливості логів та формулювання задачі. У другому розділі запропоновано архітектуру системи автоматизованого аналізу логів та описано її основні компоненти. В кінці наведено висновки та перспективи подальших досліджень.

1. Огляд логів та постановка завдання

Поява логів як інструменту фіксації подій бере свій початок ще з перших поколінь обчислювальної техніки. У 1950–1960-х роках, коли програми запускалися на мейнфреймах із перфокарт або стрічок, результати виконання часто зберігалися у вигляді текстових повідомлень, які друкувалися на папері.[3] Ці повідомлення містили відомості про успішність виконання програм, помилки введення або обчислень. Уже тоді було зрозуміло, що фіксація внутрішніх станів системи допомагає виявити причини збоїв та покращити якість програмного забезпечення [4, 5]. З часом ці «друковані логи» трансформувалися у файли на дисках і стали невід'ємною частиною будь-якого програмного середовища. З розвитком операційних систем, розподілених обчислень і мережевих технологій роль логів суттєво зросла [6]. У 1990–2000-х роках з'явилися централізовані системи логування, такі як syslog у Unix-подібних системах, які дозволяли фіксувати події з різних джерел у єдиному форматі [7]. Паралельно з цим розвивалися перші бібліотеки для логування в прикладних мовах програмування (Log4j для Java, loguru для Python тощо) [8]. У міру ускладнення архітектури додатків – особливо з приходом мікросервісів і хмарних платформ – логування стало ще важливішим елементом моніторингу та спостережуваності .

Стандартно у середовищі програмістів використовують наступну класифікацію за логами. Вона не є визначеним стандартом, але є найбільш поширеною [9].

Рівні логів:

– Debug – запис масштабних переходів станів, наприклад, звернення до бази даних, старт/пауза сервісу, успішна обробка запису тощо.

– Warning – нештатна ситуація, потенційна проблема, може бути дивний формат запиту або

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

некоректний параметр виклику.

– Error – цей рівень використовується для запису помилок і проблем, які можуть призвести до некоректної роботи програми.

– Fatal – є найвищим рівнем критичності логів і вказує на найкритичніші помилки .

– Trace – покрокові записи процесу, що використовуються для детального аналізу виконання.

– Info – загальна інформація про роботу служби або сервісу.

За типами:

– системні – пов'язані зі системними подіями

– серверні – відображають процес обробки запитів;

– поштові – пов'язані з передачею повідомлень;

– логи баз даних – відображають операції доступу до даних;

– авторизаційні та аутентифікаційні – фіксують процеси входу та доступу користувачів.

Попри значну інформативність журналів подій, їх ефективне використання у великих програмних системах пов'язане з рядом труднощів. Сучасні програмні додатки можуть генерувати тисячі або навіть мільйони записів логів щодня [10]. У таких умовах ручний аналіз журналів стає складним і потребує значних витрат часу з боку розробників .

Додатковою проблемою є те, що повідомлення логів мають текстову форму та часто містять неструктуровану інформацію [11]. В одному записі можуть поєднуватися службові повідомлення системи, технічні терміни, назви класів або модулів програмного забезпечення, а також повідомлення про помилки або винятки. Це ускладнює автоматичну обробку таких даних традиційними методами аналізу [11].

У багатьох випадках для визначення причини помилки розробнику необхідно аналізувати stack trace, шукати додаткову інформацію у документації або на технічних форумах [11]. Цей процес може займати значний час, особливо у великих системах із мікросервісною архітектурою.

Одним із перспективних підходів до автоматизації аналізу логів є використання методів обробки природної мови (Natural Language Processing, NLP) [12]. NLP є галуззю штучного інтелекту, що досліджує методи автоматичного аналізу текстової інформації.

Методи NLP широко застосовуються у різних задачах, зокрема:

– пошук інформації у текстових документах ;

– класифікація текстів;

– аналіз тональності повідомлень;

– автоматичний переклад

– витягування ключових сутностей із тексту.

У контексті аналізу логів технології NLP плануються використовуватися для обробки текстових повідомлень журналів подій, визначення типів помилок, виділення ключових термінів та виявлення подібних повідомлень у великих масивах даних.

Застосування таких методів дозволить автоматизувати процес діагностики програмних систем. Зокрема, на основі аналізу тексту повідомлення про помилку можна визначити її основну причину, сформувати пошуковий запит та знайти можливі способи її вирішення у відкритих джерелах інформації [13].

Таким чином, виникає задача розробки методів автоматизованого аналізу логів програмних систем із використанням технологій NLP. Основною метою такого підходу є скорочення часу виявлення помилок, підвищення ефективності діагностики програмного забезпечення та зменшення навантаження на розробників під час аналізу журналів подій.

2. Стиснення тексту логів та зменшення обсягу даних перед NLP-аналізом

Однією з важливих задач у процесі автоматизованого аналізу логів можна виділити зменшення обсягу текстових даних перед виконанням подальшого аналізу. У програмних системах журнали подій можуть містити значну кількість повторюваних повідомлень, службових записів або технічної інформації, яка не несе безпосередньої аналітичної цінності. Обробка таких даних без попередньої оптимізації може призводити до збільшення обчислювального навантаження на систему та зниження ефективності роботи алгоритмів NLP. Тому логічним кроком буде використання методів для скорочення обсягу повідомлень, які дозволяють зменшити кількість даних, що передаються до модуля аналізу. Основна мета такого підходу полягає у збереженні змістовної інформації повідомлення при одночасному усуненні надлишкових або повторюваних елементів.

У наукових дослідженнях, присвячених аналізу логів, зазначається, що значна частина повідомлень журналів подій має подібну або ідентичну структуру. Наприклад, у роботах Xu та ін. показано, що більшість логів складається з шаблонів повідомлень, у яких змінюються лише окремі параметри або значення змінних [4]. Це означає, що значну частину тексту можна представити у більш компактному

вигляді шляхом виділення структурного шаблону повідомлення.

Існує кілька підходів до зменшення обсягу текстових логів:

1. Видалення службових елементів повідомлення. На цьому етапі з тексту логів вилучаються технічні елементи, які не впливають на зміст помилки, наприклад:
   - часові мітки;
   - ідентифікатори потоків;
   - службові маркери форматування.
2. Виділення шаблонів логів (log template extraction). Цей підхід полягає у визначенні структури повідомлення та заміні змінних параметрів узагальненими маркерами. Подібні методи активно застосовуються у сучасних системах аналізу логів і дозволяють суттєво зменшити кількість унікальних текстових повідомлень [5].
3. Усунення повторюваних повідомлень. У великих інформаційних системах одна й та сама помилка може генеруватися багаторазово протягом короткого проміжку часу. У таких випадках доцільно зберігати лише один запис події та лічильник повторень.
4. Лексичне скорочення тексту. На цьому етапі застосовуються стандартні методи обробки тексту:
   - видалення стоп-слів;
   - лематизація;
   - нормалізація термінів.

Застосування цих методів дозволяє значно скоротити розмір тексту та підвищити ефективність подальшої обробки.

Згідно з результатами досліджень у галузі аналізу логів, попередня обробка та стиснення повідомлень дозволяють зменшити обсяг даних на 40–70% без суттєвої втрати змістовної інформації [6]. Це, у свою чергу, дозволяє значно підвищити швидкість роботи алгоритмів NLP та зменшити використання обчислювальних ресурсів.

У запропонованій системі етап стиснення тексту виконується після первинної фільтрації логів і перед передачею повідомлення до NLP-модуля. На цьому етапі видаляються службові елементи, нормалізується текст повідомлення та формується компактне представлення помилки. Отриманий результат передається до модуля аналізу, що дозволяє підвищити ефективність обробки логів та зменшити навантаження на систему.

Таким чином, використання методів стиснення тексту є важливим етапом у процесі автоматизованого аналізу логів, оскільки воно дозволяє зменшити обсяг даних, підвищити швидкість обробки повідомлень і покращити якість подальшого NLP-аналізу.

3. Запропоноване рішення для автоматизованого аналізу логів

На основі проведеного аналізу існуючих підходів до обробки тексту та специфіки журналів подій доцільним є створення узагальненої системи автоматизованого аналізу логів, призначеної для виявлення помилок, виділення ключових ознак інцидентів і пошуку можливих способів їх усунення. Основна ідея запропонованого рішення полягає у поєднанні централізованого збору журналів подій, засобів попередньої обробки тексту, алгоритмів NLP та механізмів пошуку релевантної інформації у зовнішніх або внутрішніх джерелах знань.

У традиційному підході аналіз логів виконується вручну. Фахівець має знайти потрібний запис серед великої кількості повідомлень, визначити тип помилки, проаналізувати контекст її виникнення та самостійно сформулювати запит для пошуку можливого рішення. Такий процес потребує значного часу, залежить від досвіду спеціаліста та ускладнюється у випадку роботи з великими розподіленими системами, де журнали подій генеруються багатьма компонентами одночасно.

Запропоноване рішення спрямоване на автоматизацію цього процесу. Система повинна приймати повідомлення журналів подій із різних джерел, зберігати їх, відбирати найбільш значущі записи, аналізувати текстовий зміст повідомлень і формувати структуроване подання помилки. На основі цього подання може виконуватися подальший пошук схожих інцидентів або можливих варіантів усунення проблеми.

Таким чином, система виконує роль інтелектуального середовища підтримки аналізу логів. Її використання дозволяє зменшити час первинної діагностики, знизити навантаження на спеціалістів та підвищити ефективність роботи з великими обсягами журналів подій.

3.1. Загальна концепція системи

У загальному вигляді система автоматизованого аналізу логів повинна реалізовувати повний цикл обробки повідомлень – від моменту надходження логів до формування рекомендацій або результатів пошуку для користувача (рис. 1). Основу такого підходу становить поетапне перетворення неструктурованих текстових повідомлень у структуровані дані, придатні для подальшого аналізу.

Робота системи передбачає виконання таких основних етапів:

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

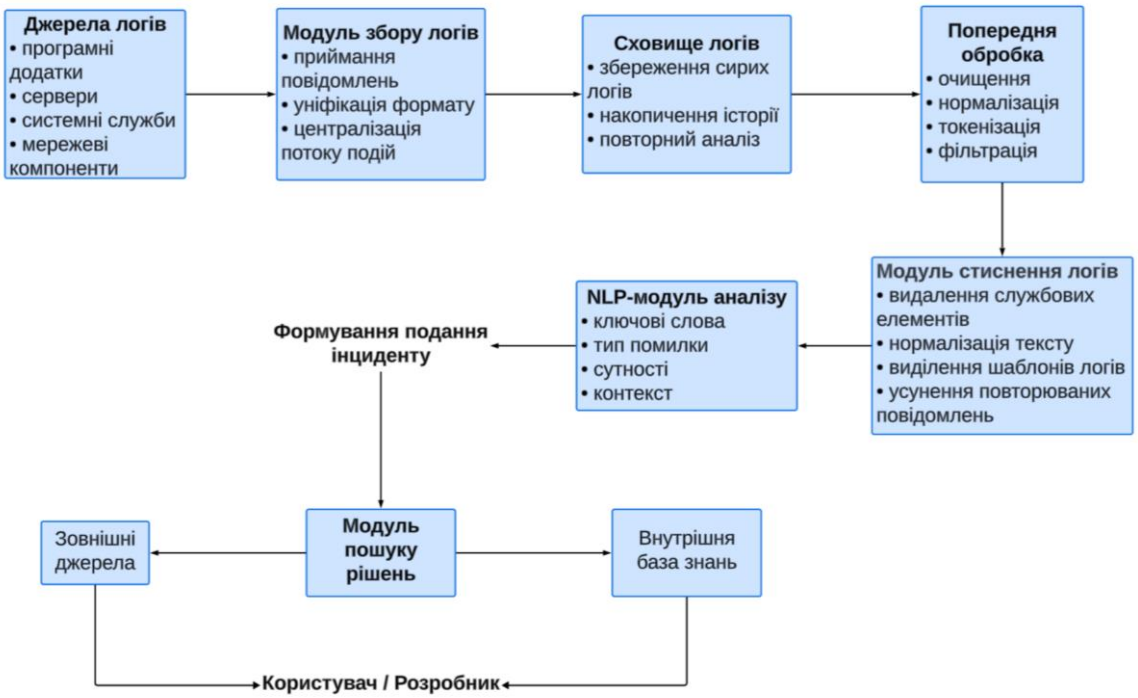

Рис. 1. Загальна концепція системи автоматизованого аналізу логів

1) отримання журналів подій із зовнішніх джерел;

2) збереження отриманих повідомлень у централізованому сховищі;

3) відбір повідомлень, що містять ознаки помилки або аномальної поведінки;

4) попередня обробка тексту логів, що включає і стиснення;

5) застосування методів NLP для виділення ключових сутностей і контекстних ознак;

6) формування пошукового або аналітичного подання інциденту;

7) пошук можливих рішень або схожих випадків;

8) передача результатів користувачу.

У такій концепції система не обмежується лише функцією зберігання журналів подій. Вона виконує інтелектуальний аналіз змісту повідомлень і допомагає перейти від простого фіксування факту помилки до інтерпретації її причин та можливих шляхів усунення.

Для схеми доцільно показати такий узагальнений ланцюг: Джерела логів → Модуль збору → Сховище даних → Модуль аналізу → Модуль пошуку рішень → Користувач.

3.2. Загальна архітектура системи

Архітектура запропонованої системи повинна будуватися за модульним принципом. Такий підхід дозволяє розділити функціональність на окремі логічні компоненти, кожен із яких виконує власну задачу в межах загального процесу аналізу логів. Модульна структура (рис. 2) також забезпечує гнучкість системи, можливість заміни окремих алгоритмів і подальше масштабування рішення.

У складі загальної архітектури доцільно виділити такі основні компоненти.

3.3. Модуль збору логів

Модуль збору логів забезпечує отримання повідомлень журналів подій із різних програмних або системних джерел. Такими джерелами можуть бути прикладні системи, сервери, служби, мережеві компоненти або інші інформаційні підсистеми.

Основне призначення цього модуля полягає у централізації надходження логів та приведенні повідомлень до єдиного формату. На цьому етапі може виконуватися первинна перевірка структури повідомлень, фіксація часу надходження, а також прив’язка запису до конкретного джерела.

3.4. Сховище логів та результатів аналізу

Після надходження повідомлень вони повинні зберігатися у централізованому сховищі. Це дозволяє не лише виконувати поточний аналіз, але й накопичувати історію інцидентів для повторного використання, статистичного аналізу та формування бази знань.

У сховищі можуть зберігатися:

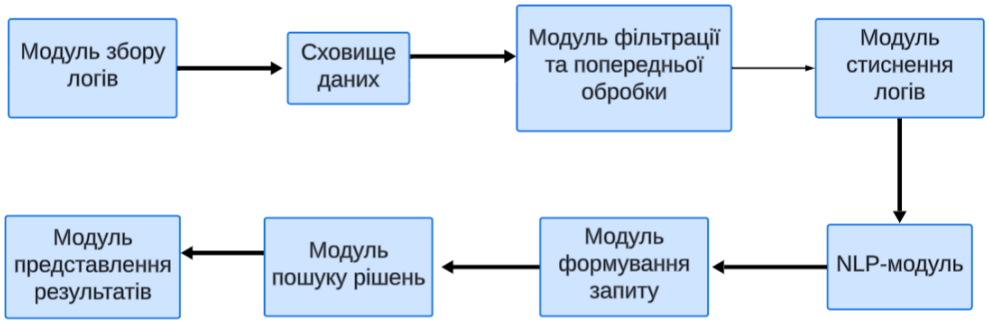

Рис. 2. Загальна архітектура системи автоматизованого аналізу логів

– сирі повідомлення логів;
– очищені або нормалізовані тексти;
– результати NLP-аналізу;
– виділені ключові слова та сутності;
– сформовані пошукові запити;
– знайдені рішення або посилання на джерела.

Таким чином, сховище виступає не лише як архів, а і як основа для подальшого інтелектуального аналізу.

3.5. Модуль фільтрації та попередньої обробки

Оскільки журнали подій можуть містити велику кількість службової або низькорівневої інформації, перед повним аналізом доцільно виконувати фільтрацію повідомлень. На цьому етапі відбираються записи, які мають найбільшу цінність для діагностики: повідомлення про помилки, попередження, критичні події або аномальні записи.

Після фільтрації виконується попередня обробка тексту. Вона включає нормалізацію символів, очищення від технічного шуму, токенізацію, виділення інформативних фрагментів та підготовку тексту до подальшого застосування алгоритмів NLP.

3.6. Модуль стиснення логів

Перед передачею повідомлень до NLP-модуля виконується етап стиснення тексту логів. Основною задачею цього етапу є зменшення обсягу текстових даних шляхом видалення службових елементів, виділення шаблонів повідомлень та усунення повторюваних записів. Такий підхід дозволяє скоротити кількість тексту, що передається до модуля аналізу, та зменшити обчислювальне навантаження на систему.

На цьому етапі можуть застосовуватися методи:

– видалення службових елементів;
– нормалізація тексту;
– виділення шаблонів логів;
– усунення повторюваних повідомлень.

Результатом роботи модуля є компактне представлення лог-повідомлення, яке зберігає ключову інформацію про помилку та передається до NLP-модуля для подальшого аналізу.

3.7 Модуль NLP-аналізу

Цей модуль є центральним елементом системи, оскільки саме він забезпечує змістовний аналіз повідомлень. На його основі неструктурований текст логів перетворюється на набір формалізованих ознак.

У межах роботи цього модуля можуть виконуватися:

– виділення типів помилок;
– пошук ключових термінів;
– розпізнавання технічних сутностей;
– визначення контекстних ознак повідомлення;
– побудова векторного представлення тексту;
– оцінювання подібності між інцидентами.

На практиці цей модуль може використовувати як класичні методи представлення тексту, так і сучасні трансформерні моделі, залежно від складності задачі та наявних обчислювальних ресурсів.

3.8. Модуль формування аналітичного запиту

Після завершення NLP-аналізу необхідно сформувати структуроване подання інциденту, придатне для подальшого пошуку. Саме цю задачу виконує модуль формування аналітичного запиту.

На основі виділених ознак створюється короткий змістовний опис помилки, який може використовуватися:

– для пошуку у зовнішніх джерелах;

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

– для порівняння з раніше збереженими випадками;
– для класифікації інцидентів;
– для побудови статистики повторюваних помилок.

Цей етап є важливим, оскільки саме від правильності побудови подання інциденту залежить якість пошуку релевантних рішень.

3.9. Модуль пошуку рішень та знань

Після формування аналітичного подання система виконує пошук релевантної інформації. Пошук може здійснюватися як у внут рішньому сховищі попередніх інцидентів, так і у зовнішніх джерелах знань.

Узагальнено даний модуль може виконувати:
– пошук схожих логів у базі історичних записів;
– пошук уже відомих рішень;
– пошук зовнішніх матеріалів, пов’язаних із виявленою помилкою;
– ранжування знайдених результатів за релевантністю.

Цей компонент дозволяє системі перейти від етапу виявлення проблеми до етапу підтримки її розв’язання.

3.10. Модуль представлення результатів

Завершальним компонентом є модуль представлення результатів. Його задача полягає у передачі користувачу результатів аналізу у зрозумілому та зручному вигляді.

Результат може містити:
– короткий опис помилки;
– виділені ключові ознаки;
– рівень критичності;
– схожі інциденти;
– можливі варіанти рішень;
– посилання на джерела або записи з бази знань.

Такий підхід дозволяє значно спростити сприйняття результатів аналізу та скоротити час прийняття рішення користувачем.

3.11. Логіка функціонування системи

Функціонування системи можна подати як послідовний процес переходу від сирого повідомлення журналу подій до отримання аналітичного результату. На першому етапі повідомлення надходить від зовнішнього джерела до модуля збору. Далі воно зберігається у централізованому сховищі.

Після цього система визначає, чи є повідомлення релевантним для подальшого аналізу. Якщо запис містить ознаки помилки або попередження, він передається до модуля попередньої обробки, де очищується та нормалізується.

На наступному етапі виконується NLP-аналіз тексту. Модель або набір алгоритмів виділяє основні ознаки інциденту: тип помилки, ключові слова, назви сутностей, контекстні фрази. Отриманий набір ознак використовується для побудови структурованого аналітичного подання.

Після цього запускається пошуковий етап, у межах якого система шукає подібні інциденти або можливі рішення. Завершальним кроком є подання результатів користувачу (рис. 3).

3.12. Переваги запропонованої архітектури

Запропонована архітектура має низку переваг. Насамперед, модульний принцип побудови дозволяє легко змінювати окремі компоненти системи без повного перепроєктування рішення. Наприклад, за потреби можна замінити алгоритм NLP-аналізу або змінити джерела пошуку рішень.

Крім того, система поєднує централізоване накопичення логів із інтелектуальною обробкою їхнього змісту. Це дозволяє не лише виявляти окремі помилки, а й формувати історію інцидентів, виконувати повторний аналіз та накопичувати базу знань.

Ще однією важливою перевагою є універсальність архітектури. Вона не залежить від конкретної платформи, мови програмування або типу прикладної системи, а тому може бути адаптована до різних середовищ, де використовується логування подій.

Нарешті, використання NLP як центрального елемента архітектури дозволяє перейти від простого збереження логів до аналізу їхнього змісту. Саме це робить систему більш корисною у практичному аспекті, оскільки вона орієнтована не лише на фіксацію подій, а й на підтримку прийняття рішень під час діагностики помилок.

4. Обговорення результатів

Отримані результати дослідження підтверджують доцільність використання методів обробки природної мови для автоматизованого аналізу логів програмних систем. Запропонована архітектура дозволяє інтегрувати різні підходи до

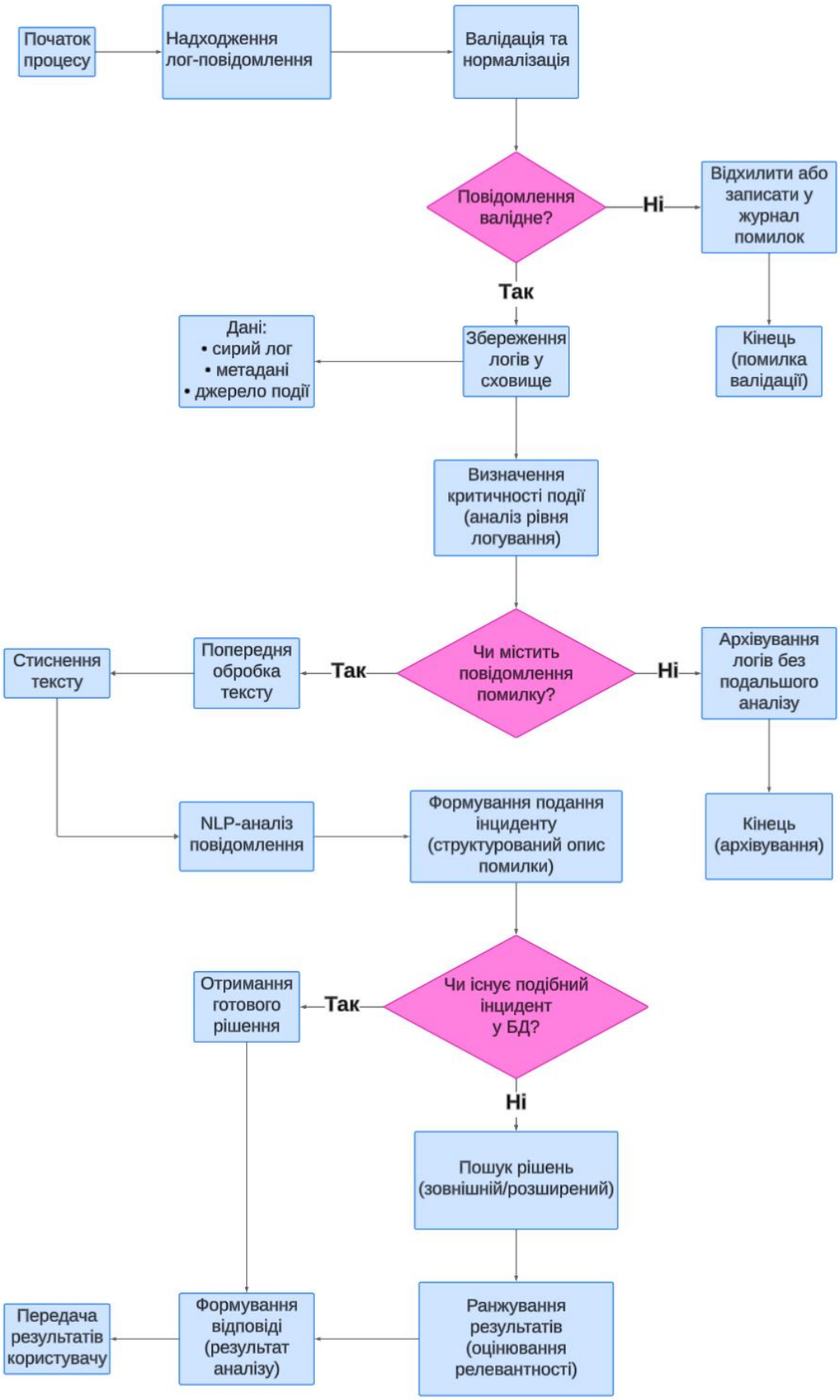

Рис. 3. Послідовність функціонування системи автоматизованого аналізу логів

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

обробки тексту та забезпечує поетапний аналіз лог-повідомлень – від їх отримання до формування рекомендацій для користувача.

У порівнянні з традиційними методами аналізу логів, що базуються на ключових словах або регулярних виразах, запропонований підхід має ряд переваг. Зокрема, використання NLP дозволяє враховувати контекст повідомлення, що значно підвищує точність виявлення помилок і класифікації інцидентів [14, 15]. Крім того, застосування алгоритмів стиснення тексту дозволяє зменшити обсяг оброблюваних даних, що позитивно впливає на продуктивність системи.

Разом з тим, результати дослідження показують, що ефективність запропонованого підходу значною мірою залежить від якості вхідних даних. Логи, які містять недостатньо структуровану або неповну інформацію, можуть ускладнювати процес аналізу навіть при використанні сучасних моделей NLP. Це особливо актуально для систем, де відсутня стандартизація форматів логування.

Окремо слід відзначити, що використання складних моделей обробки природної мови, зокрема трансформерних архітектур, може потребувати значних обчислювальних ресурсів. Сучасні дослідження показують, що моделі на основі трансформерів та попередньо навчених мовних моделей демонструють високу ефективність у задачах аналізу логів, однак їх використання пов’язане з підвищеними вимогами до обчислювальних ресурсів [16]. У реальних умовах це може обмежувати можливість їх застосування у системах із високим навантаженням або вимогами до обробки даних у реальному часі. У таких випадках доцільним є використання комбінованого підходу, який поєднує прості методи попереднього аналізу з більш складними моделями лише на критичних етапах.

Порівняння з існуючими підходами до аналізу логів показує, що сучасні системи часто орієнтовані або на статистичний аналіз, або на використання глибокого навчання. Запропоноване рішення поєднує ці підходи, що дозволяє досягти балансу між точністю аналізу та швидкістю обробки даних. Додатковою перевагою є можливість інтеграції системи з базами знань, що відкриває перспективи для накопичення досвіду та повторного використання рішень.

Разом з тим, слід враховувати, що автоматизовані системи аналізу логів не можуть повністю замінити експертну оцінку фахівця. У складних випадках, особливо при виникненні нетипових або нових помилок, остаточне рішення все ще потребує участі людини. Таким чином, запропонований підхід слід розглядати як інструмент підтримки прийняття рішень, а не як повністю автономну систему.

Перспективним напрямом подальших досліджень є підвищення адаптивності системи до різних типів логів, удосконалення методів виділення технічних сутностей, а також інтеграція з системами моніторингу та управління інцидентами. Особливий інтерес становить застосування великих мовних моделей для аналізу логів і виявлення аномалій, що активно досліджується у сучасних роботах. Також перспективним є застосування таких підходів у системах критичного призначення, зокрема в аерокосмічній галузі, де вимоги до точності та швидкості аналізу є особливо високими.

Таким чином, результати дослідження демонструють, що використання методів NLP у поєднанні з оптимізацією обробки даних є перспективним напрямом розвитку систем аналізу логів і може суттєво підвищити ефективність діагностики програмних систем.

Висновки

У статті було розглянуто проблему аналізу логів програмних систем та можливості використання методів обробки природної мови для автоматизації цього процесу. Проведений аналіз показав, що журнали подій є одним із ключових джерел інформації про роботу програмного забезпечення. Вони містять відомості про стан системи, виконання операцій, виникнення помилок та інші технічні події. Водночас із розвитком складних програмних архітектур, зокрема мікросервісних систем і хмарних платформ, обсяг журналів подій значно зріс, що ускладнює їх ручний аналіз.

У роботі було досліджено сучасні підходи до обробки текстових даних у рамках технологій NLP. Зокрема, розглянуто статистичні методи представлення тексту, такі як Bag-of-Words та TF-IDF, векторні моделі слів (Word2Vec, GloVe, FastText), а також сучасні підходи на основі нейронних мереж і трансформерних архітектур. Проведений аналіз показав, що різні методи мають різну ефективність залежно від типу текстових даних та задачі, яка вирішується.

Статистичні методи є простими у реалізації та не потребують значних обчислювальних ресурсів, однак вони не враховують контекст появи слів. Векторні моделі слів дозволяють враховувати семантичні зв’язки між словами, проте формують статичні представлення слів без урахування конкретного контексту. Найбільш сучасні підходи, засновані на трансформерних архітектурах, дозволяють аналізувати текст із урахуванням

контексту та демонструють високі результати у задачах класифікації та пошуку інформації.

На основі проведеного аналізу було запропоновано узагальнену архітектуру системи автоматизованого аналізу логів. Запропоноване рішення передбачає централізований збір журналів подій, їх збереження у сховищі даних, попередню обробку текстових повідомлень, застосування методів NLP для виділення ключових ознак помилки та формування структурованого подання інциденту. Отримані результати можуть використовуватися для пошуку схожих випадків або визначення можливих способів усунення проблеми.

Запропонований підхід може використовуватися у різних типах інформаційних систем, де активно застосовується журналювання подій. Насамперед це стосується серверних програмних систем, веб-застосунків, розподілених сервісів та хмарних платформ.

Використання систем автоматизованого аналізу логів дозволяє:

– прискорити процес виявлення помилок у програмному забезпеченні;

– зменшити навантаження на розробників під час аналізу журналів подій;

– автоматизувати первинну діагностику програмних інцидентів;

– накопичувати базу знань про типові помилки та способи їх усунення;

– підвищити ефективність роботи систем моніторингу.

Особливо актуальним застосування такого підходу є у великих програмних системах, де обсяг журналів подій може становити тисячі або навіть мільйони записів на добу.

Основними перевагами запропонованої системи

є:

– автоматизація процесу аналізу текстових повідомлень логів;

– можливість обробки великих обсягів журналів подій;

– виділення ключових елементів повідомлень про помилки;

– підвищення швидкості пошуку причин програмних збоїв;

– можливість інтеграції з системами моніторингу та технічної підтримки.

Крім того, модульна архітектура системи дозволяє гнучко адаптувати її до різних типів програмних систем та розширювати функціональність за рахунок підключення нових алгоритмів аналізу.

Попри переваги, запропонований підхід має певні обмеження. Насамперед ефективність системи залежить від якості логів та інформативності повідомлень. Якщо записи журналів подій містять недостатньо контекстної інформації, це може ускладнювати аналіз.

Крім того, використання сучасних моделей обробки тексту, особливо трансформерних архітектур, може потребувати значних обчислювальних ресурсів. Це може обмежувати можливість їх використання у системах з високим навантаженням або у середовищах із обмеженими ресурсами.

Також слід враховувати, що автоматичний аналіз логів не завжди дозволяє однозначно визначити причину виникнення помилки. У складних випадках система може лише запропонувати можливі варіанти рішень або напрямки подальшого дослідження.

Подальший розвиток досліджень у цій сфері може бути пов’язаний із застосуванням спеціалізованих моделей аналізу логів, удосконаленням методів виділення технічних сутностей та створенням систем накопичення знань про типові інциденти у програмних системах.

Також перспективним напрямом є інтеграція систем аналізу логів із інструментами моніторингу, системами управління інцидентами та платформами DevOps, що дозволить створювати комплексні рішення для автоматизації діагностики програмного забезпечення.

Внесок авторів: огляд логів, архітектура системи автоматизованого аналізу логів та її основні компоненти – **Владислав Чернищук**; постановка завдання та пошук літератури – **Ольга Морозова**; редагування та аналіз результатів – **Владислав Чернищук, Ольга Морозова**.

Конфлікт інтересів

Автори заявляють, що у них немає конфлікту інтересів щодо цього дослідження, фінансового, особистого, авторського чи іншого, який міг би вплинути на дослідження та його результати, представлені в цій статті.

Фінансування

Дослідження проводилося без фінансової підтримки.

Доступність даних

Рукопис не має пов'язаних даних.

Використання штучного інтелекту

Під час підготовки початкової чернетки рукопису були використані генеративні інструменти штучного інтелекту для допоміжного мовного редагування та структуризації тексту; перевірка фактів, інтерпретація результатів і фінальне наукове редагування мають бути виконані автором перед поданням.

Усі автори прочитали і погодили остаточну версію рукопису.

Література

1. A Survey on Automated Log Analysis for Reliability Engineering [Text] / P He., J. Zhu, S. He, & et al. // ACM Computing Survey, 2020, vol. 1, iss. 1, pp. 1–37. DOI: 10.48550/arXiv.2009.07237.

2. What is Site Reliability Engineering (SRE)? Amazon Web Services [Electronic resource]. – Available at: https://aws.amazon.com/what-is/sre/ (accessed 15.01.2026).

3. Salz, P. Logging in the Java Platform [Electronic resource] / P. Salz. – Available at: https://docs.oracle.com (accessed 15.01.2026).

4. Sommerville, I. Software Engineering [Text] / I. Sommerville. – Harlow, Pearson, 2016. – 816 p.

5. Tanenbaum, A. S. Modern Operating Systems [Text] / A. S. Tanenbaum, & H. Bos. – Boston, Pearson, 2015. 1136 p.

6. Silberschatz, A. Operating System Concepts [Text] / A. Silberschatz, P. B. Galvin, & G. Gagne. – Hoboken, Wiley, 2018. – 976 p.

7. Gerhards R. The Syslog Protocol (RFC 5424) [Text] / R. Gerhards. – IETF, 2009. – 37 p.

8. Log4j Documentation. Apache Software Foundation [Electronic resource]. – Available at: https://logging.apache.org/log4j/ (accessed 15.01.2026).

9. Jurafsky, D. Speech and Language Processing[Text] / D. Jurafsky, & J. H. Martin. – 2023. –600 p.

10. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [Text] /J. Devlin, M.-W. Chang, K. Lee, & K. Toutanova // Proceedings of NAACL-HLT. – 2019. – P. 4171–4186.

11. Detecting Large-Scale System Problems by Mining Console Logs [Text] / W. Xu, L. Huang, A. Fox, D. Patterson, & M. Jordan // Proceedings of SOSP. – 2009. – P. 117–132.

12. He P., Zhu J., Zheng Z. & Lyu M. R. Drain: An Online Log Parsing Approach with Fixed Depth Tree [Text] /. IEEE ICWS, 2017. – P. 33–40.

13. Du, M. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning [Text] / M. Du, & F. Li. – ACM CCS, 2017. – P. 1285–1298.

14. Zhang, T. Log Analysis Using Large Language Models for Anomaly Detection [Text] / T. Zhang, Y. Liu, & J. Wang // Proceedings of the ACM Symposium on Cloud Computing (SoCC). *– 2024. – P. 98–110.*

15. Chen, X. A Survey on Deep Learning for Log Analysis in Distributed Systems [Text] / X. Chen, Y. Li, & H. Zhang // Future Generation Computer Systems. *– 2023. – Vol. 140. – P. 452–468.*

16. Wang, Z. Log Parsing and Anomaly Detection Using Pre-trained Language Models [Text] / Z. Wang, Q. Xu & L. Chen // IEEE Transactions on Network and Service Managemen. *– 2024. – Vol. 21, iss. 1. – P. 112–125.*

References

1. He, P., Zhu, J., He, S., Li, J., & Lyu, M. R. A Survey on Automated Log Analysis for Reliability Engineering. *ACM Computing Survey*, 2020, vol. 1, iss. 1, pp. 1–37. DOI: 10.48550/arXiv.2009.07237.

2. *What is Site Reliability Engineering (SRE)? Amazon Web Services.* Available at: https://aws.amazon.com/what-is/sre/ (accessed 15.01.2026).

3. Salz, P. *Logging in the Java Platform.* Available at: https://docs.oracle.com (accessed 15.01.2026).

4. Sommerville, I. *Software Engineering*. Harlow, Pearson, 2016. 816 p.

5. Tanenbaum, A. S., & Bos, H. *Modern Operating Systems*. Boston, Pearson, 2015. 1136 p.

6. Silberschatz, A., Galvin, P. B., & Gagne, G. *Operating System Concepts*. Hoboken, Wiley, 2018. 976 p.

7. Gerhards, R. *The Syslog Protocol (RFC 5424)*. IETF, 2009. 37 p.

8. *Log4j Documentation. Apache Software Foundation*. Available at: https://logging.apache. org/log4j/ (accessed 15.01.2026).

9. Jurafsky, D. & Martin, J. H. *Speech and Language Processing*, 2023. 600 p.

10. Devlin, J., Chang, M.-W., Lee, K., & Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

11. Xu, W., Huang, L., Fox, A., Patterson, D., & Jordan, M. Detecting Large-Scale System Problems by Mining Console Logs. *Proceedings of SOSP*, 2009, pp. 117–132.

12. He, P., Zhu, J., Zheng, Z., & Lyu, M. R. Drain: An Online Log Parsing Approach with Fixed Depth Tree. *IEEE ICWS*, 2017, pp. 33–40.

13. Du, M., & Li, F. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. *ACM CCS*, 2017, pp. 1285–1298.

14. Zhang, T., Liu, Y., & Wang, J. Log Analysis Using Large Language Models for Anomaly Detection. *Proceedings of the ACM Symposium on Cloud Computing (SoCC)*, 2024, pp. 98–110.

15. Chen, X., Li, Y., & Zhang, H. A Survey on Deep Learning for Log Analysis in Distributed Systems. *Future Generation Computer Systems*, 2023, vol. 140, pp. 452–468.

16. Wang, Z., Xu, Q., & Chen, L. Log Parsing and Anomaly Detection Using Pre-trained Language Models. *IEEE Transactions on Network and Service Managemen*, 2024, vol. 21, iss. 1, pp. 112–125.

Отримано 15.01.2026, отримано у доопрацьованому вигляді 05.03.2026
Дата ухвалення 15.04.2026, дата публікації 22.04.2026

THE SYSTEM FOR THE AUTOMATED ANALYSIS OF SERVER LOGS USING NLP METHODS AND TEXT COMPRESSION ALGORITHMS

Vladyslav Chernyshchuk, Olga Morozova

The subject of the article is the processes of the automated analysis of server logs of software systems using Natural Language Processing methods. The goal is to develop an approach to improve the efficiency of log analysis by applying NLP algorithms and text compression methods. **The tasks to be solved are**: to study the features of logs as a source of technical information; to analyze modern methods of text data processing, including statistical, vector-based, and neural network approaches; to justify the feasibility of using NLP algorithms for automating log analysis; to substantiate the use of text compression methods to reduce data volume; and to develop a concept and architecture of an automated log analysis system. **The methods used include:** Natural Language Processing techniques such as TF-IDF, Word2Vec, FastText, and transformer-based models; text preprocessing methods; log compression and log template extraction approaches; and information retrieval methods applied to text corpora. **The following results were obtained**. A concept of an automated server log analysis system based on NLP methods and text compression algorithms was developed. A generalized system architecture was proposed, including modules for log collection, storage, filtering, preprocessing, text compression, NLP analysis, incident representation, and solution search. An algorithm for system operation was developed, providing step-by-step log processing while taking into account the textual nature and large volume of data. It was established that the use of NLP methods improves the accuracy of error detection and incident classification, while text compression reduces the computational load and increases system performance. **Conclusions**. The proposed approach improves the efficiency of software diagnostics, reduces the time required for error detection and resolution, and decreases the workload on developers. **The scientific novelty** of the obtained results lies in the following: an approach to automated log analysis combining NLP methods and text compression techniques was developed; a generalized architecture of a log analysis system that considers the specifics of unstructured textual data was proposed; existing log processing methods were further developed through the integration of modern NLP models with data volume optimization stages, which improves analysis efficiency in conditions of large-scale information flows.

Keywords: log analysis; natural language processing; NLP algorithms; text compression; intelligent systems; software diagnostics; data processing; fault tolerance.

Чернищук Владислав Сергійович – асп. каф. кібербезпеки та інтелектуальних інформаційних технологій, Національний аерокосмічний університет «Харківський авіаційний інститут», Харків, Україна

Морозова Ольга Ігорівна – д-р техн. наук, проф., проф. каф. кібербезпеки та інтелектуальних інформаційних технологій, Національний аерокосмічний університет «Харківський авіаційний інститут», Харків, Україна.

Vladyslav Chernyshchuk – PhD Student at the Department of Cybersecurity and Intelligent Information Technologies, National Aerospace University «Kharkiv Aviation Institute», Kharkiv, Ukraine,

e-mail: v.chernyshchuk@student.csn.khai.edu, ORCID: 0009-0003-1327-5783.

Olga Morozova – Doctor of Technical Science, Professor, Professor at the Department of Cybersecurity and Intelligent Information Technologies, National Aerospace University «Kharkiv Aviation Institute», Kharkiv, Ukraine, e-mail: o.morozova@csn.khai.edu, ORCID: 0000-0001-7706-3155, Scopus Author ID: 57194517520.