Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)

Кафедра *комп'ютерного моделювання та інтелектуальних технологій*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти *другий (магістерський)*

Дослідження методів автоматичної модерації контенту на платформі Discord на основі трансформерних моделей
(тема)

Виконав: студент *II* курсу, групи *СПРм-24-1*
Рехінкін М.А.
(прізвище, ініціали)

Спеціальність *122 Комп'ютерні науки*
(код і повна назва спеціальності)

Тип програми *освітньо-наукова*

Освітня програма *Системне проектування*
(повна назва освітньої програми)

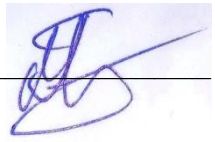
Керівник *доцент каф. КМІТ Ребезюк Л.М.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ *Гребеннік І.В.*
(підпис) (прізвище, ініціали)

2026 р.

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Студент  *Рехінкін М.А.*

Кваліфікаційна робота не містить відомостей заборонених до відкритого опублікування.

Керівник кваліфікаційної роботи  *доц. Ребезюк Л.М.*

Кваліфікаційна робота виконана у відповідності до діючих стандартів в Україні.

Керівник кваліфікаційної роботи  *доц. Ребезюк Л.М.*

Попередній захист проведено 13 травня 2026 р.

Керівник кваліфікаційної роботи  *доц. Ребезюк Л.М.*

Харківський національний університет радіоелектроніки

(назва закладу вищої освіти)

Факультет *Комп'ютерних наук*

Кафедра *комп'ютерного моделювання та інтелектуальних технологій*

Рівень вищої освіти *другий (магістерський)*

Спеціальність *122 Комп'ютерні науки*

(код і повна назва)

Тип програми *освітньо-наукова*

Освітня програма *Системне проектування*

(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:

Зав. кафедри *КМІТ*

проф. Гребеннік І.В.

" _____ " *2026 р.*

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Рехінкіну Максиму Андрійовичу*

(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження методів автоматичної модерації контенту на платформі Discord на основі трансформерних моделей*

затверджена наказом університету від " *06* **"** *квітня* **2026р. №** *268Ст*

2. Термін подання студентом роботи до екзаменаційної комісії *13 травня 2026 р.*

3. Вихідні дані до роботи*. Дослідити методи та моделі аналізу текстового та візуального користувальницького контенту в онлайн-середовищі з метою автоматичного виявлення токсичних текстових повідомлень і NSFW-контенту у зображеннях на основі сучасних підходів штучного інтелекту з використанням технологій машинного навчання. 3.1 Підходи, моделі та методи виявлення токсичних повідомлень. 3.2 Методи обробки зображень для виявлення NSFW-контенту. 3.3 Перелік використовуваних програмних засобів: мова програмування та бібліотеки для розробки Discord-бота.*

4. Перелік питань, що потрібно опрацювати в роботі *4.1 Вступ 4.2. Аналіз предметної галузі та постановка задачі дослідження 4.2.1 Поняття токсичності в онлайн-комунікації 4.2.2 Огляд існуючих рішень для виявлення та обробки токсичних повідомлень та зображень 4.2.3 Постановка задачі дослідження 4.3 Методи та моделі для аналізу токсичних повідомлень та NSFW-зображень 4.3.1. Аналіз підходів, моделей та методів виявлення токсичних повідомлень 4.3.2. Аналіз методів обробки зображень для виявлення NSFW-контенту (методи на основі комп'ютерного зору та мультимодальні моделі на основі трансформерів) 4.4 Вибір технологій та засобів розробки Discord-бота 4.4.1. Вибір мови програмування 4.4.2 Визначення бібліотек для розробки Discord-бота 4.5 Програмна реалізація та опис застосунку 4.5.1 Discord Developer Portal 4.5.2 Архітектура проекту 4.5.3 Функціонал застосунку 4.6 Висновки*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)
Титульний слайд кваліфікаційної роботи. Об'єкт, предмет дослідження, мета роботи (слайд 1). Актуальність роботи (слайд 2). Токсичність в онлайн середовищі (слайд 3). Методи виявлення тосичності (слайд 4). Аналіз тексту (RoBERTa) (слайд 5). Аналіз зображень (CLIP-based NSFW Detector) (слайд 6). Discord developer portal (слайд 7). Архітектура проекту (слайд 8). Функціонал застосунку (слайд 9). Команди бота (слайд 10). Висновки (слайд 11).

6. Дата видачі завдання *26 січня 2026 р.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1.	*Отримання завдання кваліфікаційної роботи*	*26.01.2026*	*виконано*
2.	*Аналіз завдання, літератури та аналогів з теми кваліфікаційної роботи*	*27.01 — 08.02.2026*	*виконано*
3.	*Аналіз предметної галузі та постановка задачі дослідження*	*09.02 — 19.02.2026*	*виконано*
4.	*Огляд існуючих рішень для виявлення та обробки токсичних повідомлень та зображень*	*20.02 — 01.03.2026*	*виконано*
5.	*Підходи, моделі та методи для аналізу токсичних повідомлень*	*02.03 — 13.03.2026*	*виконано*
6.	*Методи обробки зображень для виявлення NSFW-контенту*	*14.03 — 25.03.2026*	*виконано*
7.	*Вибір технологій та засобів розробки Discord-бота*	*26.03 — 08.04.2026*	*виконано*
8.	*Підготовка тез доповіді на конференцію*	*09.04 — 18.04.2026*	*виконано*
9.	*Програмна реалізація та опис застосунку*	*19.04 — 28.04.2026*	*виконано*
10.	*Оформлення пояснювальної записки*	*29.04 — 12.05.2026*	*виконано*
11.	*Створення комп'ютерної презентації та оформлення графічних матеріалів кваліфікаційної роботи*	*09.05 — 12.05.2026*	*виконано*
12.	*Представлення на рецензування*	*12.05.2026*	*виконано*
13	*Представлення кваліфікаційної роботи до ЕК*	*13.05.2026*	*виконано*

Студент _____ *Рехінкін М.А.*
(підпис)

Керівник роботи _____ *доц. Ребезюк Л.М.*
(підпис)

ABSTRACT

Master's thesis: *67* p., — tabl., *18* fig., *1* app., *9* sources.

COMPUTER VISION, CONTENT MODERATION, DISCORD, MACHINE LEARNING, NATURAL LANGUAGE PROCESSING, NSFW, TOXICITY OF TEXTUAL AND VISUAL CONTENT, TRANSFORMER MODELS

Object of research – process of automatic moderation of user-generated content in an online environment.

Subject of study – methods and models for analyzing textual and visual content in order to automatically detect toxic text messages and NSFW content in images based on modern artificial intelligence approaches using machine learning technologies.

The purpose of the work – develop an intelligent system in the form of a Discord bot that performs automatic analysis of user-generated content, detects toxic messages and inappropriate images, and provides an appropriate response to maintain a safe communication environment.

Research methods – systematic approach, methods of functional analysis, comparative analysis of web applications using different architectural approaches, as well as experimental research on ways to implement microservice and microfrontend architectures in web development

To achieve this goal, the following tasks were solved in the thesis: the subject area was analyzed and the main problems of toxicity in online communication were identified; existing methods and models for detecting toxic text and NSFW images were reviewed; approaches to system implementation were substantiated; text classification modules based on transformer models and image analysis modules using computer vision and multimodal models were developed; a Discord bot was implemented; system testing was conducted and its effectiveness was evaluated. The work uses modern methods of natural language processing, deep learning, and computer vision, including transformer models for text analysis and image classification models. The practical significance of the obtained results lies in the use of the developed system for automatic content moderation in online communities, which reduces the workload on administrators and increases safety in the digital environment.

РЕФЕРАТ

Кваліфікаційна робота: *67* с., — табл., *18* рис., *1* дод., *9* джерел.

МОДЕРАЦІЯ КОНТЕНТУ, ТОКСИЧНІСТЬ ТЕКСТОВОГО ТА ВІЗУАЛЬНОГО КОНТЕНТУ, ОБРОБКА ПРИРОДНОЇ МОВИ, КОМП'ЮТЕРНИЙ ЗІР, ТРАНСФОРМЕРНІ МОДЕЛІ, МАШИННЕ НАВЧАННЯ, NSFW, DISCORD

Об'єкт дослідження – процес автоматичної модерації користувальницького контенту в онлайн-середовищі.

Предмет дослідження – методи та моделі аналізу текстового та візуального контенту з метою автоматичного виявлення токсичних текстових повідомлень і NSFW-контенту у зображеннях на основі сучасних підходів штучного інтелекту з використанням технологій машинного навчання.

Метою роботи є розробка інтелектуальної системи у вигляді Discord-бота, який здійснює автоматичний аналіз користувальницького контенту, виявляє токсичні повідомлення та небажані зображення, а також забезпечує відповідну реакцію для підтримки безпечного середовища спілкування.

Для досягнення поставленої мети у роботі вирішено такі задачі: проведено аналіз предметної галузі та визначено основні проблеми токсичності в онлайн-комунікації; виконано огляд існуючих методів і моделей для виявлення токсичного тексту та NSFW-зображень; обґрунтовано вибір підходів до реалізації системи; розроблено модулі класифікації тексту на основі трансформерних моделей та аналізу зображень із використанням методів комп'ютерного зору і мультимодальних моделей; реалізовано програмний засіб у вигляді Discord-бота; проведено тестування системи та оцінено її ефективність. У роботі використано сучасні методи обробки природної мови, глибинного навчання та комп'ютерного зору, зокрема трансформерні моделі для аналізу тексту та моделі для класифікації зображень.

Практичне значення отриманих результатів – використання розробленої системи для автоматичної модерації контенту в онлайн-спільнотах, що дозволяє зменшити навантаження на адміністраторів і підвищити рівень безпеки цифрового середовища.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

API – Application Programming Interface;

BERT – Bidirectional Encoder Representations from Transformers;

BLIP – Bootstrapping Language-Image Pretraining;

CLIP – Contrastive Language–Image Pretraining;

CNN – Convolutional Neural Network;

GRU – Gated Recurrent Unit);

FLAN-T5 – Fine-tuned Language Net T5;

LSTM – Long Short-Term Memory;

ML – Machine Learning;

MLM – Masked Language Model;

NSFW – Not Safe For Work;

PIL – Python Imaging Library;

TF-IDF – Term Frequency–Inverse Document Frequency;

XLM – Cross-lingual Language Model.

ВСТУП

У сучасному цифровому середовищі онлайн-комунікація стала невід'ємною частиною повсякденного життя. Платформи для спілкування, зокрема Discord, активно використовуються для обміну повідомленнями, мультимедійним контентом та організації спільнот за інтересами. Разом із розвитком таких платформ зростає і кількість користувацького контенту, що створює нові виклики, пов'язані з контролем його якості та безпечності.

Однією з ключових проблем є поширення токсичних повідомлень та небажаного візуального контенту. Агресивні висловлювання, мова ворожнечі, образи, а також NSFW-зображення негативно впливають на атмосферу спілкування, можуть призводити до конфліктів між користувачами та знижувати загальний рівень довіри до онлайн-спільнот. У зв'язку з цим постає необхідність створення ефективних засобів автоматичної модерації, здатних оперативно виявляти та обмежувати поширення такого контенту.

Традиційні методи модерації, що базуються на ручній перевірці або простих правилах, не забезпечують достатньої ефективності в умовах великого обсягу даних і високої швидкості обміну інформацією. Це зумовлює актуальність застосування сучасних методів штучного інтелекту, зокрема моделей обробки природної мови та комп'ютерного зору. Особливе місце серед них займають трансформерні моделі, які демонструють високі результати у задачах аналізу тексту, а також мультимодальні підходи, що дозволяють обробляти як текстову, так і візуальну інформацію.

Актуальність теми дослідження полягає у необхідності створення інтелектуальних систем, здатних автоматично аналізувати різні типи користувацького контенту та забезпечувати ефективну модерацію в реальному часі. Це особливо важливо для платформ із великою кількістю активних користувачів, де ручна модерація є недостатньо ефективною або економічно невигідною.

Об'єктом дослідження є процес автоматичної модерації

користувальницького контенту в онлайн-середовищі.

Предметом дослідження є методи та моделі аналізу текстового та візуального контенту з метою автоматичного виявлення токсичних текстових повідомлень і NSFW-контенту у зображеннях на основі сучасних підходів штучного інтелекту з використанням технологій машинного навчання.

Метою роботи є розробка інтелектуальної системи у вигляді Discord-бота, який здійснює автоматичний аналіз користувальницького контенту, виявляє токсичні повідомлення та небажані зображення, а також забезпечує відповідну реакцію для підтримки безпечного середовища спілкування

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

Discord є сучасною онлайн-платформою, призначеною для організації текстового та голосового спілкування між користувачами. Вона була створена як відповідь на потребу у більш зручному та ефективному інструменті комунікації, насамперед для геймерської спільноти. Ініціатором розробки став Jason Citron, який прагнув усунути недоліки наявних на той момент сервісів для спілкування під час ігрового процесу.

Розробка платформи здійснювалася командою фахівців, які проаналізували слабкі сторони таких популярних рішень, як TeamSpeak та Skype. Основною метою було створення інструменту, що поєднував би стабільність роботи, високу якість зв’язку та зручний інтерфейс. У 2015 році сервіс було офіційно представлено широкій аудиторії, після чого він швидко здобув популярність завдяки своїй продуктивності та простоті використання.

Важливу роль у розвитку платформи відіграла активна спільнота користувачів, яка сприяла її поширенню та надавала зворотний зв’язок розробникам. Завдяки цьому Discord постійно вдосконалювався і з часом перетворився з вузькоспеціалізованого інструменту для геймерів на універсальну платформу для різних типів спільнот. Сьогодні Discord використовується як для особистого спілкування, так і для навчання, роботи та організації професійних об’єднань.

Популярність сервісу (див. рис.1.1) пояснюється не лише зручністю взаємодії, але й широкими можливостями налаштування. Користувачі можуть створювати власні сервери, організовувати канали різного типу та адаптувати інтерфейс відповідно до власних потреб. Особливістю платформи є можливість розширення її функціональності за допомогою ботів, які розробляються сторонніми користувачами та інтегруються у сервери.

Discord підтримує організацію голосових конференцій із можливістю налаштування каналів зв’язку, а також режиму «натисни, щоб говорити», що

дозволяє контролювати передачу звуку.

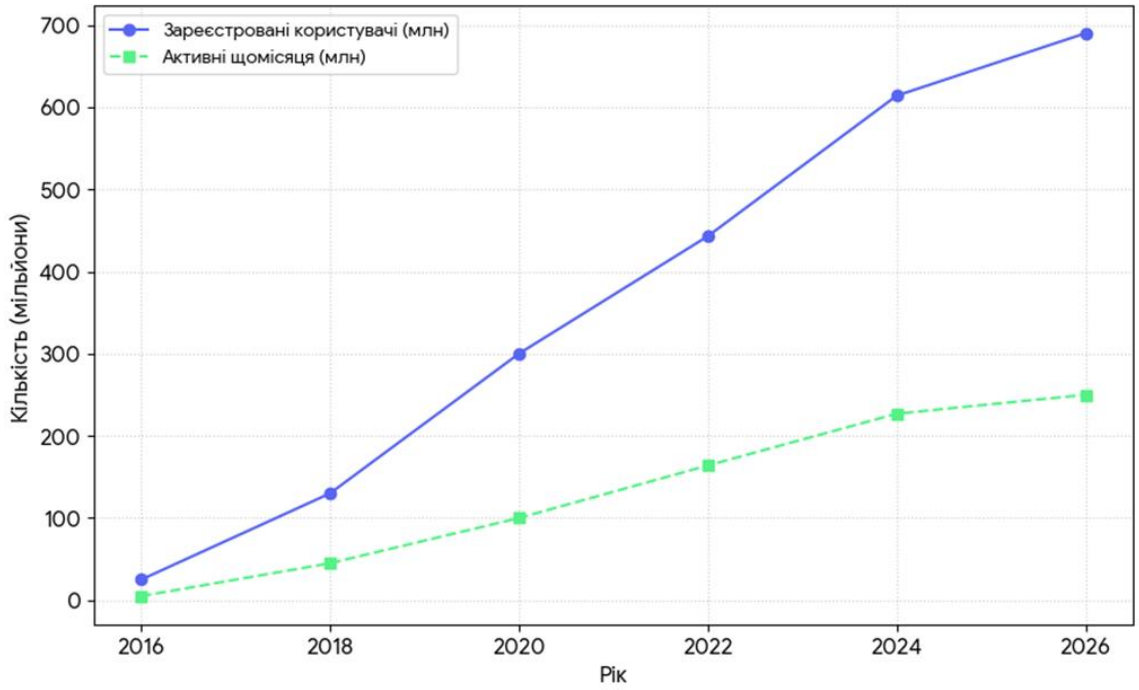


Рисунок 1.1 – Кількість користувачів Discord

Окрім цього, доступні текстові чати, які можуть бути як приватними, так і публічними. Наявність веб-версії робить платформу доступною навіть без встановлення окремого програмного забезпечення, хоча деякі функції, зокрема режим передачі голосу, можуть працювати з обмеженнями у браузері.

Сервіс також інтегрується з іншими популярними платформами, такими як YouTube, Spotify та Twitch, що дозволяє користувачам об’єднувати свої облікові записи та розширювати можливості взаємодії. Додаткові функції, такі як режим стрімера, забезпечують конфіденційність під час трансляцій, приховуючи особисті дані та вимикаючи небажані сповіщення.

Значною перевагою є наявність функції оверлею, яка дозволяє керувати каналами, звуком та іншими параметрами безпосередньо під час роботи в інших додатках. Крім того, користувачі можуть налаштовувати сповіщення, виконувати пошук по чатах та обмінюватися файлами з підтримкою форматування тексту та мультимедійних елементів.

У межах платформи передбачено кілька форматів взаємодії між

користувачами. Приватні чати забезпечують обмін повідомленнями між двома особами, тоді як групові чати дозволяють взаємодіяти більшій кількості учасників. Найбільш функціональним елементом є сервери, які виступають центром організації спільнот. Вони можуть містити численні текстові та голосові канали, а також систему ролей, що визначає рівень доступу користувачів до різних ресурсів.

Важливою складовою екосистеми Discord є боти — програмні модулі, які автоматизують різні процеси на сервері. Вони можуть виконувати функції модерації, контролюючи дотримання правил спілкування, забезпечувати відтворення музики, надавати інтерактивні розваги або виконувати допоміжні задачі, такі як переклад тексту чи пошук інформації. Завдяки цьому боти значно розширюють можливості платформи та спрощують управління спільнотами.

Таким чином, ефективне функціонування серверів Discord потребує як участі адміністраторів, так і використання автоматизованих інструментів. Поєднання людського контролю та програмних засобів, зокрема ботів, дозволяє підтримувати порядок, забезпечувати безпеку користувачів та створювати комфортне середовище для спілкування.

1.1 Поняття токсичності в онлайн-комунікації

У сучасному цифровому середовищі поняття токсичності використовується для опису деструктивної поведінки користувачів у процесі онлайн-комунікації. Під токсичністю розуміють повідомлення, що мають агресивний, образливий або провокативний характер і спрямовані на виклик негативних емоцій у співрозмовників. Особливо актуальною ця проблема є для платформ із активною взаємодією користувачів, таких як Discord, де обмін інформацією відбувається у режимі реального часу.

Токсичні повідомлення можуть проявлятися у різних формах — від відкритих образ і використання нецензурної лексики до більш прихованих проявів, таких як сарказм, іронія або маніпулятивні висловлювання. На відміну

від безпосереднього спілкування, онлайн-комунікація позбавлена невербальних сигналів, що значно ускладнює інтерпретацію емоційного змісту повідомлень. У результаті навіть нейтральні на перший погляд фрази можуть сприйматися як агресивні, що призводить до виникнення конфліктів.

Додатковим фактором поширення токсичності є відносна анонімність користувачів у мережі. Відсутність прямої відповідальності за власні висловлювання та обмежений соціальний контроль сприяють тому, що користувачі частіше дозволяють собі агресивну або некоректну поведінку. Це особливо характерно для великих онлайн-спільнот, де індивідуальна відповідальність розмивається, а рівень емоційної напруги може швидко зростати.

Окрім текстових повідомлень, важливою складовою проблеми є поширення небажаного візуального контенту. Зображення, що містять сцени сексуального характеру або інші неприйнятні матеріали, відносять до категорії NSFW. Такий контент може негативно впливати на користувачів та порушувати правила платформ, особливо якщо мова йде про відкриті спільноти з різною аудиторією.

Токсичність у цифровому середовищі має багатовимірний характер і може включати як явні, так і приховані форми агресії. Вона проявляється у вигляді образливих висловлювань, провокаційних повідомлень, навмисного створення конфліктних ситуацій, а також у формі мови ворожнечі, спрямованої проти окремих соціальних груп. Крім того, поширеним явищем є так званий тролінг — публікація повідомлень із метою викликати негативну реакцію інших користувачів.

Негативний вплив токсичності проявляється як на рівні окремих користувачів, так і на рівні спільнот у цілому. Постійна взаємодія з агресивним контентом може призводити до зниження психологічного комфорту, підвищення рівня стресу та втрати інтересу до участі в онлайн-обговореннях. У межах спільнот це спричиняє погіршення якості комунікації, зменшення активності користувачів та формування ворожої атмосфери.

З огляду на це, проблема токсичності потребує не лише теоретичного осмислення, а й практичних рішень для її ефективного виявлення. Важливим аспектом є класифікація різних проявів токсичної поведінки, що дозволяє створювати більш точні моделі аналізу контенту. Для сучасних систем модерації, зокрема реалізованих у вигляді ботів для платформ типу Discord, це є критично важливим, оскільки від точності визначення токсичності залежить ефективність усього процесу забезпечення безпечного середовища спілкування.

Таким чином, токсичність в онлайн-комунікації є складним явищем, що охоплює різні форми поведінки та типи контенту. Її поширення обумовлює необхідність застосування сучасних методів аналізу, зокрема з використанням технологій штучного інтелекту, що дозволяють автоматизувати процес модерації та підвищити його ефективність.

1.2 Огляд існуючих рішень для виявлення та обробки токсичних повідомлень та зображень

Різноманітні компанії та наукові установи розробляють системи модерації, що ґрунтуються на широкому спектрі підходів — від елементарних фільтрів за ключовими словами і жорстко заданих правил до складних алгоритмів машинного та глибинного навчання. Використання технологій обробки природної мови та зображень разом із нейронними мережами значно підвищує ефективність виявлення токсичного контенту. Водночас це породжує додаткові труднощі, зокрема необхідність коректного врахування контексту повідомлень і забезпечення стабільної роботи моделей у багатомовному середовищі.

Moderation API (OpenAI) — це спеціалізований інструмент автоматичної модерації текстового контенту, розроблений компанією OpenAI. Його основне призначення полягає у виявленні небажаних, небезпечних або таких, що порушують правила, повідомлень у цифрових системах [1].

Сервіс базується на попередньо навчених моделях машинного навчання

(див.рис.1.2), які аналізують текст і класифікують його за кількома категоріями, зокрема: мова ворожнечі (hate), переслідування (harassment), насильство (violence), самопошкодження (self-harm), сексуальний контент (sexual) та інші. Для кожної категорії система визначає ймовірність належності повідомлення до певного класу, що дозволяє гнучко налаштовувати рівень чутливості модерації.

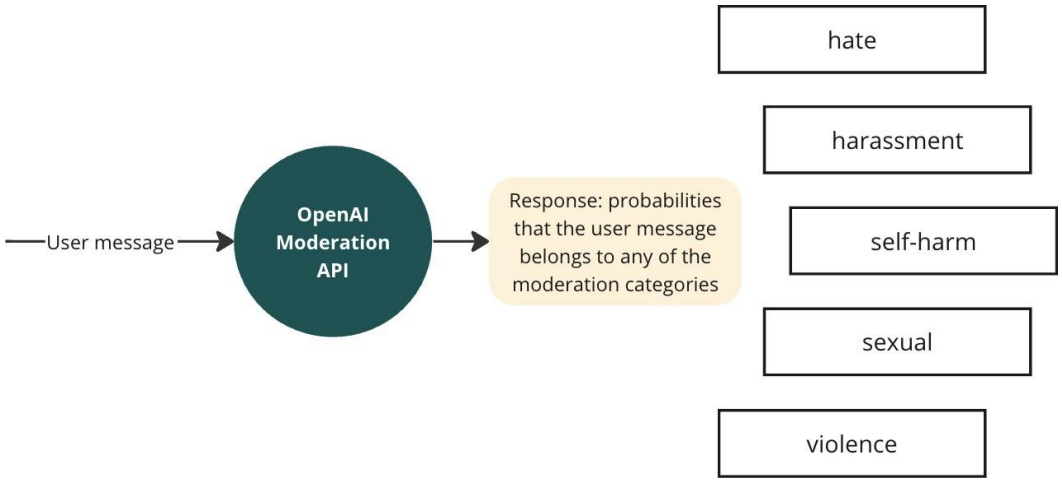


Рисунок 1.2 – Робота Moderation API

Moderation API широко застосовується для фільтрації користувацького контенту в чатах, форумах, соціальних мережах та інших онлайн-платформах. Також він використовується як додатковий рівень захисту при роботі з мовними моделями, забезпечуючи перевірку вхідних і вихідних повідомлень.

До основних переваг даного рішення належать висока швидкість роботи, простота інтеграції та відсутність необхідності самостійного навчання моделей. Водночас система має певні обмеження, зокрема закритість архітектури та неможливість повного налаштування під специфічні задачі. Крім того, у деяких випадках можливі помилкові спрацьовування через складність інтерпретації контексту.

GPT-3 (Generative Pre-trained Transformer 3) — це велика мовна модель, розроблена компанією OpenAI, яка належить до класу трансформерних моделей і призначена для обробки та генерації природної мови. Вона побудована на

архітектурі трансформера та містить сотні мільярдів параметрів, що дозволяє їй ефективно працювати з широким спектром текстових задач [2].

Модель була попередньо навчена на великих обсягах текстових даних з інтернету, що дало їй змогу засвоїти закономірності мови, граматику, стилістику та певні знання про світ. Завдяки цьому GPT-3 може виконувати різноманітні завдання без додаткового навчання, зокрема класифікацію тексту, переклад, узагальнення, відповіді на запитання та генерацію зв'язного тексту.

У задачах виявлення токсичності GPT-3 може використовуватися в режимі zero-shot або few-shot, коли модель отримує інструкцію у вигляді текстового запиту (промпту) і виконує класифікацію повідомлення без спеціального донавчання. Наприклад, їй можна поставити задачу визначити, чи є текст образливим, пояснити причину такого висновку або переформулювати повідомлення у нейтральному стилі.

Основною перевагою GPT-3 є її універсальність і гнучкість: змінюючи формулювання запиту, можна адаптувати модель до різних типів задач. Крім того, вона здатна враховувати контекст і виявляти складні або приховані форми токсичності, такі як сарказм чи непряма агресія.

Разом із тим, використання GPT-3 має і певні обмеження. До них належать значні обчислювальні ресурси, необхідність доступу до API, а також залежність якості результату від правильно сформульованого запиту. Крім того, модель не завжди забезпечує стабільну точність у специфічних доменах без додаткової адаптації.

FLAN-T5 (Fine-tuned Language Net T5) — це вдосконалена мовна модель, створена компанією Google на основі архітектури T5 (Text-to-Text Transfer Transformer). Вона належить до класу трансформерних моделей і спеціалізується на виконанні широкого спектра завдань обробки природної мови у форматі «текст-на-вхід — текст-на-вихід». Архітектура моделі FLAN-T5 представлена на рисунку1.3.

Ключовою особливістю FLAN-T5 є використання підходу instruction tuning — додаткового навчання на великій кількості завдань, сформульованих у

вигляді текстових інструкцій. Завдяки цьому модель краще розуміє запити користувача і здатна виконувати різноманітні задачі без потреби у спеціалізованому донавчанні [3].

У контексті виявлення токсичності FLAN-T5 може застосовуватись для класифікації повідомлень, пояснення причин токсичності, а також для переформулювання агресивних висловлювань у більш нейтральному стилі. Модель ефективно працює у режимах zero-shot і few-shot, що дозволяє використовувати її без підготовки власних датасетів.

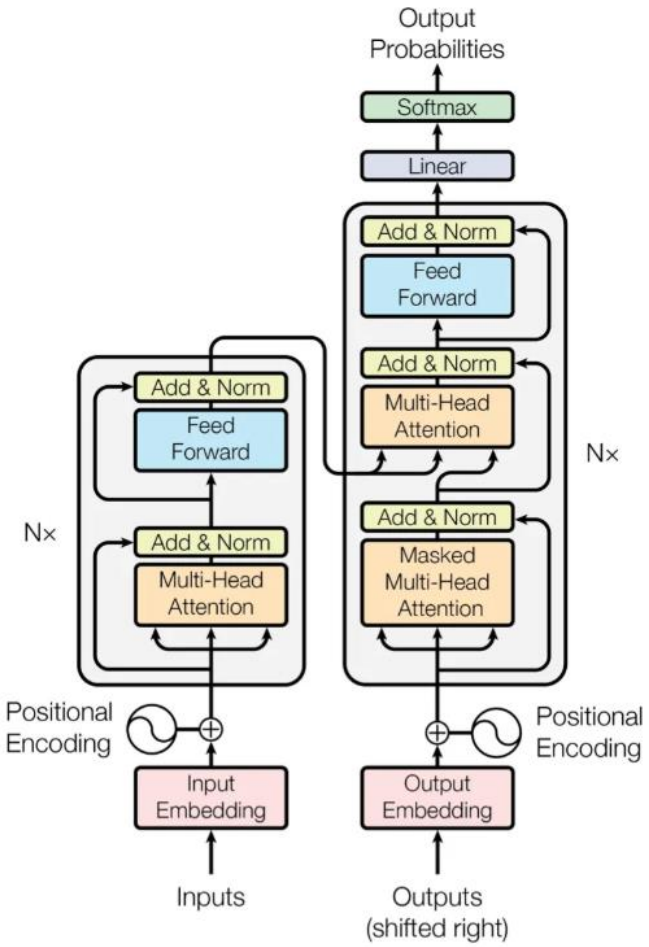


Рисунок 1.3 – Архітектура моделі FLAN-T5

Серед основних переваг FLAN-T5 варто відзначити високу гнучкість, здатність до узагальнення та хорошу інтерпретацію інструкцій. Вона демонструє стабільні результати на різних NLP-задачах і може бути адаптована

під різні сценарії використання лише за рахунок зміни формулювання запиту.

Разом із тим, модель має певні обмеження. Як і інші великі трансформери, вона потребує значних обчислювальних ресурсів, а якість результатів залежить від точності сформульованих інструкцій. Крім того, у деяких випадках можливі помилки при роботі зі складними або неоднозначними контекстами.

BERT (Bidirectional Encoder Representations from Transformers) — це трансформерна мовна модель, розроблена компанією Google, яка призначена для глибокого аналізу тексту з урахуванням контексту. Вона стала однією з ключових моделей у сфері обробки природної мови завдяки здатності одночасно враховувати як лівий, так і правий контекст слова в реченні.

Основою BERT є архітектура трансформера, зокрема енкодерна частина, яка використовує механізм self-attention. Це дозволяє моделі визначати взаємозв’язки між словами незалежно від їхнього розташування в тексті, що є критично важливим для розуміння змісту складних або неоднозначних висловлювань [4].

Модель проходить попереднє навчання на великих текстових корпусах із використанням спеціальних завдань, таких як masked language modeling (відновлення пропущених слів) та next sentence prediction (визначення логічної послідовності речень). Після цього BERT може бути донавчений (fine-tuning) для виконання конкретних задач, зокрема класифікації тексту, аналізу тональності та виявлення токсичних повідомлень. Схема навчання BERT показана на рисунку 1.4.

У задачах модерації контенту BERT демонструє високу ефективність, оскільки здатен враховувати контекст, виявляти приховані форми агресії, сарказм і складні мовні конструкції. Це робить його значно точнішим у порівнянні з класичними підходами, які базуються на окремих словах або простих статистичних ознаках.

До переваг моделі належать висока точність, гнучкість у застосуванні та можливість адаптації до різних мов і доменів. Водночас BERT має і певні

недоліки, серед яких значні обчислювальні витрати, необхідність наявності великого обсягу даних для донавчання та обмежена інтерпретованість результатів. Це може ускладнювати використання моделі у системах реального часу або у випадках, де важлива пояснюваність рішень.

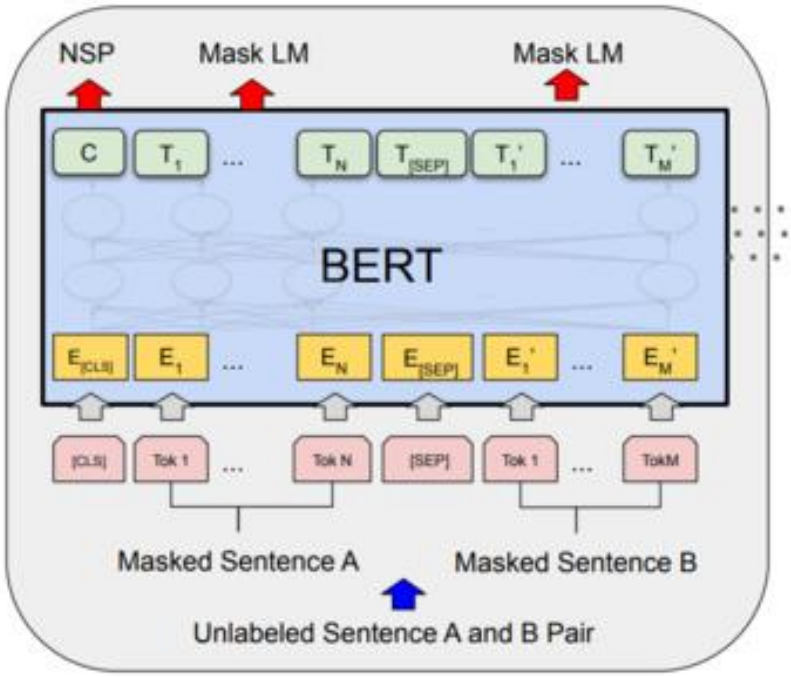


Рисунок 1.4 – Схема навчання BERT

Toxic-BERT — це спеціалізована модифікація моделі BERT, адаптована для задач виявлення токсичного контенту в текстових повідомленнях. Вона створена шляхом донавчання базової моделі BERT на спеціалізованих датасетах, що містять приклади агресивних, образливих або небажаних висловлювань.

Основна мета Toxic-BERT полягає у класифікації тексту за рівнем токсичності. Модель здатна визначати різні типи небажаного контенту, такі як мова ворожнечі, образи, погрози або провокативні повідомлення. Завдяки використанню трансформерної архітектури вона враховує контекст у межах усього речення, що дозволяє виявляти навіть непрямі або замасковані прояви токсичності.

Навчання Toxic-BERT зазвичай здійснюється на великих корпусах розмічених даних, наприклад, наборах коментарів із позначками токсичності. У процесі донавчання модель адаптується до специфіки задачі, що значно підвищує точність у порівнянні з універсальними мовними моделями.

До основних переваг Toxic-BERT належать висока точність класифікації, здатність працювати з контекстом і ефективність у виявленні складних форм токсичності, включаючи сарказм або приховану агресію. Вона добре узагальнює знання та може застосовуватися в різних системах модерації контенту.

Водночас модель має певні обмеження. Вона потребує значних обчислювальних ресурсів для навчання та використання, а також залежить від якості та збалансованості навчальних даних. Крім того, як і інші глибинні моделі, Toxic-BERT має обмежену пояснюваність, що може ускладнювати аналіз причин прийнятих рішень.

AngryBERT — це розширена модель на основі архітектури BERT, призначена для більш глибокого аналізу токсичних повідомлень. На відміну від класичних підходів, які обмежуються лише визначенням факту токсичності, дана модель вирішує кілька задач одночасно, зокрема визначає емоційний тон висловлювання та встановлює, на кого саме спрямована агресія.

Архітектурно AngryBERT поєднує базову трансформерну модель BERT із додатковими двонапрямленими рекурентними шарами, зокрема LSTM. Така комбінація дозволяє враховувати як глобальний контекст у тексті, так і послідовні залежності між словами. Для об’єднання отриманих ознак використовується спеціальний механізм інтеграції, що забезпечує узгоджене формування кінцевого результату. Алгоритм роботи моделі AngryBERT представлено на рисунку 1.5.

Модель реалізує багатозадачне навчання, у межах якого одночасно виконується класифікація токсичності, визначення емоційної тональності та ідентифікація цільової аудиторії або групи, на яку спрямоване висловлювання. Це дозволяє більш точно аналізувати складні або приховані форми агресії,

зокрема ті, що мають соціальний підтекст.

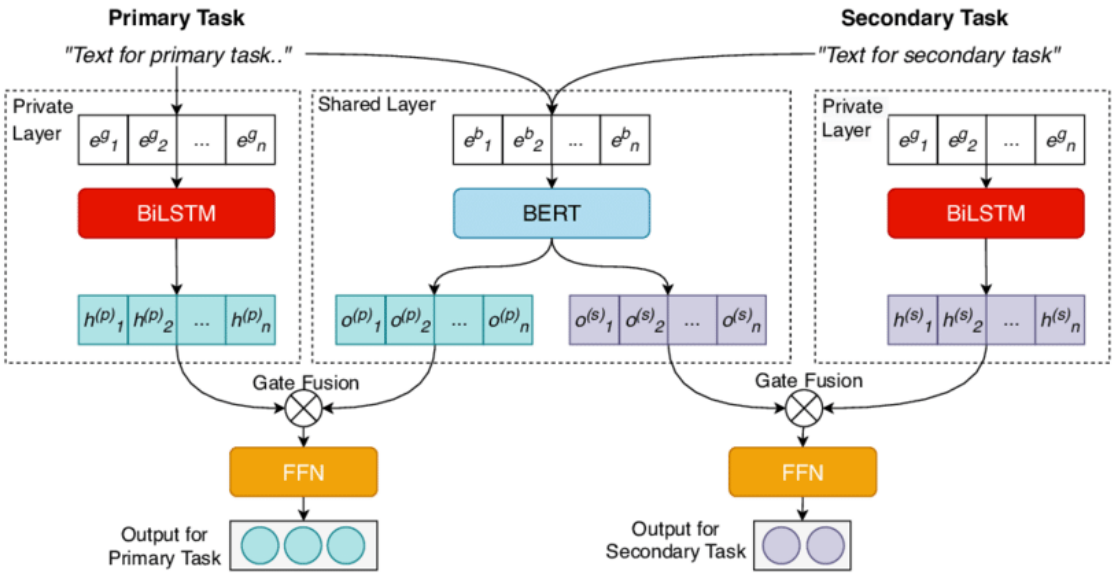


Рисунок 1.5 – Алгоритм роботи моделі AngryBERT

Серед переваг AngryBERT варто відзначити підвищену точність у порівнянні з одоотипними моделями, особливо у випадках латентної токсичності. Модель краще враховує контекст і зменшує кількість хибнопозитивних спрацьовувань.

Водночас підхід має і певні недоліки. Його реалізація є досить складною, оскільки потребує великих обсягів якісно розмічених даних для кількох задач одночасно. Крім того, модель є ресурсоємною, що може ускладнювати її використання в системах реального часу або у проєктах з обмеженими обчислювальними можливостями.

ViT Base NSFW Detector — це модель для виявлення небажаного або NSFW-контенту на зображеннях, побудована на основі архітектури Vision Transformer (ViT). Вона застосовується для автоматичної класифікації зображень з метою визначення їх безпечності для відображення у публічному середовищі.

Схематично робота ViT Base NSFW Detector представлено на рисунку 1.6. В основі моделі лежить підхід трансформерів, адаптований для задач

комп’ютерного зору.

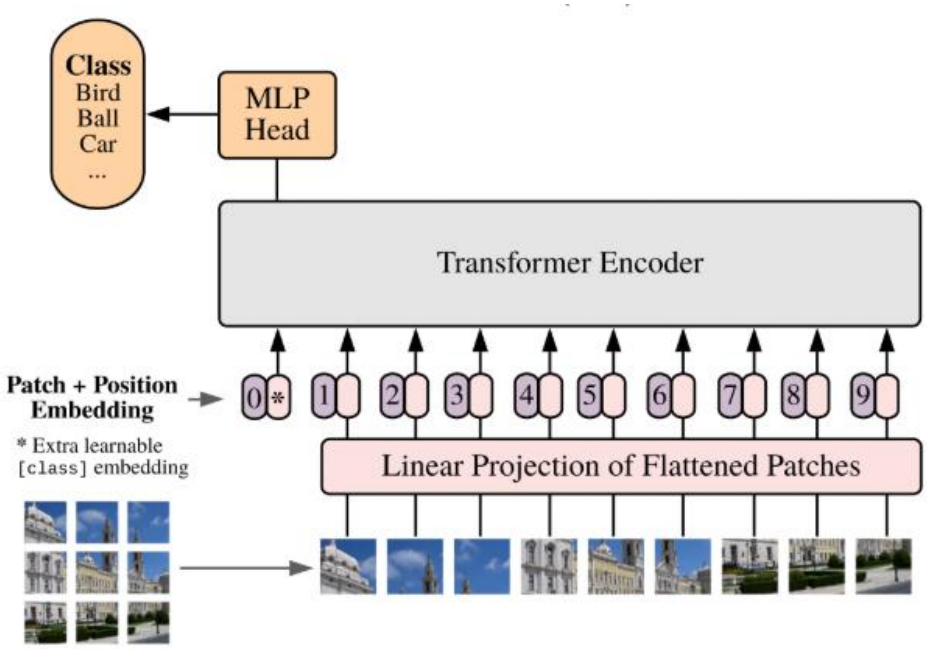


Рисунок 1.6 – Схема роботи ViT

Зображення розбивається на невеликі фрагменти (патчі), які перетворюються у векторні представлення та обробляються аналогічно до послідовностей у текстових моделях. Завдяки механізму self-attention модель здатна враховувати глобальний контекст у зображенні, що дозволяє більш точно виявляти складні візуальні патерни.

ViT Base NSFW Detector навчається на великих наборах даних, що містять як безпечні, так і небажані зображення. У результаті модель може класифікувати контент за категоріями, наприклад «safe» та «nsfw», або визначати рівень ризику. Це дозволяє використовувати її у системах автоматичної модерації, зокрема на платформах із користувацьким контентом [7].

Серед основних переваг моделі варто виділити високу точність, здатність враховувати глобальні залежності в зображенні та ефективність при роботі зі складними сценами. Вона перевершує класичні згорткові нейронні мережі у задачах, де важливий контекст усієї сцени.

Водночас модель має і певні обмеження. Вона потребує значних обчислювальних ресурсів для навчання та інференсу, а також великого обсягу розмічених даних. Крім того, у деяких випадках можливі помилки класифікації через неоднозначність візуального контенту або культурні відмінності у сприйнятті.

CLIP-based NSFW Detector — це модель для виявлення небажаного контенту на зображеннях, яка базується на архітектурі CLIP (Contrastive Language–Image Pretraining), розробленій компанією OpenAI. Її особливість полягає у поєднанні аналізу зображень і тексту в єдиному просторі представлення. Схематично принцип роботи CLIP-based NSFW Detector представлено на рисунку 1.7.

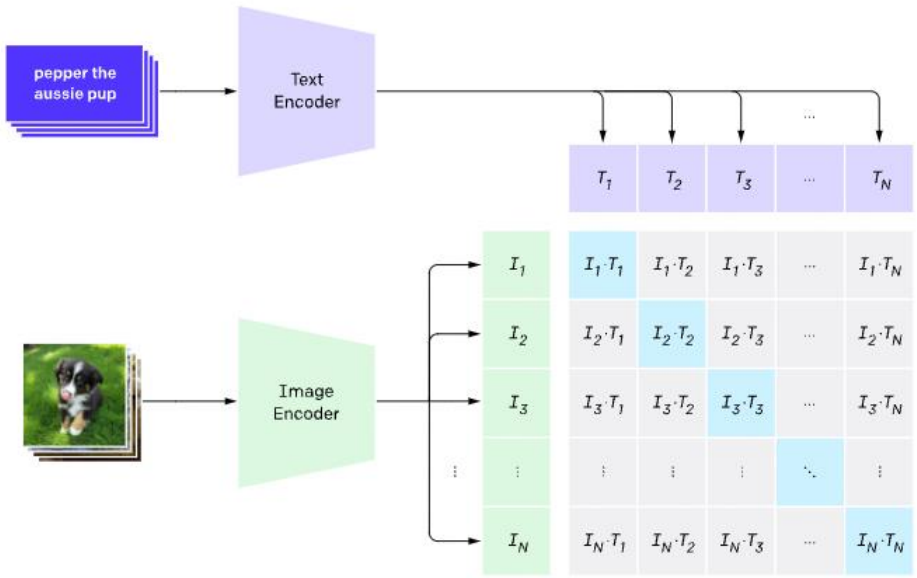


Рисунок 1.7 – Принцип роботи CLIP

Принцип роботи моделі ґрунтується на порівнянні зображення з текстовими описами. CLIP навчається встановлювати відповідність між візуальним контентом і текстовими підписами, що дозволяє використовувати його для класифікації без необхідності окремого навчання під кожну задачу. У випадку NSFW-детекції модель порівнює зображення з набором текстових запитів, таких як «safe content», «explicit content», «nudity» тощо, і визначає, до

якої категорії воно ближче [8].

Однією з ключових переваг CLIP-based підходу є його гнучкість. Модель може працювати у режимі zero-shot, тобто без додаткового навчання на спеціалізованих датасетах. Це дозволяє швидко адаптувати систему до нових типів контенту лише шляхом зміни текстових описів.

Крім того, CLIP ефективно враховує як глобальний контекст зображення, так і семантичні зв’язки, що підвищує точність виявлення складних або неоднозначних випадків. Такий підхід є особливо корисним для платформ, де контент може мати різні форми та стилі.

Водночас існують і певні обмеження. Результати роботи моделі значною мірою залежать від якості сформульованих текстових запитів. Також можливі помилки у випадках, коли зображення є контекстно неоднозначним або виходить за межі навчального розподілу. Крім того, як і інші великі моделі, CLIP потребує значних обчислювальних ресурсів для ефективного використання.

1.3 Постановка задачі дослідження

З урахуванням проведеного аналізу можна зробити висновок, що проблема автоматизованої модерації контенту є однією з актуальних задач у сфері сучасних інформаційних технологій. Особливо це стосується платформ онлайн-комунікації, таких як Discord, де користувачі активно обмінюються текстовими повідомленнями та мультимедійним контентом. Наявність токсичних висловлювань та небажаних зображень негативно впливає на якість взаємодії між користувачами та потребує ефективних засобів контролю.

Метою даної роботи є розробка інтелектуальної системи у вигляді Discord-бота, яка забезпечує автоматичний аналіз користувацького контенту, виявлення токсичних повідомлень і NSFW-зображень, а також формування відповідних реакцій для підтримки безпечного середовища спілкування.

Для досягнення поставленої мети необхідно вирішити такі задачі:

– провести аналіз поняття токсичності в онлайн-комунікації та дослідити особливості поширення небажаного контенту у цифрових середовищах;

– виконати огляд сучасних методів і моделей для виявлення токсичних текстових повідомлень і NSFW-зображень;

– визначити функціональні вимоги до системи автоматичної модерації контенту у середовищі Discord;

– обґрунтувати вибір моделей машинного навчання, зокрема трансформерних моделей, для аналізу тексту та методів комп’ютерного зору для обробки зображень;

– реалізувати модуль класифікації текстових повідомлень на основі попередньо навчених моделей для визначення рівня токсичності;

– реалізувати модуль аналізу зображень для виявлення NSFW-контенту з використанням сучасних моделей комп’ютерного зору;

– розробити механізм реагування системи на виявлений небажаний контент (попередження, видалення або інші дії);

– інтегрувати всі компоненти у вигляді Discord-бота, здатного працювати в реальному часі;

– провести тестування системи на реальних або тестових даних та оцінити її ефективність, точність і швидкодію.

Таким чином, поставлена задача спрямована на створення комплексної системи модерації, яка поєднує аналіз текстового та візуального контенту з використанням сучасних методів штучного інтелекту.

2 МЕТОДИ ТА МОДЕЛІ ДЛЯ АНАЛІЗУ ТОКСИЧНИХ ПОВІДОМЛЕНЬ ТА NSFW-ЗОБРАЖЕНЬ

Створення систем, здатних автоматично виявляти токсичний контент і відповідним чином реагувати на нього, базується на комбінуванні різних підходів до аналізу даних. У таких системах використовуються як моделі класифікації, що визначають наявність агресивних або небажаних елементів у текстових повідомленнях і зображеннях, так і генеративні моделі, які дозволяють формувати коректні відповіді або дії у відповідь на виявлений контент.

Кожен із цих компонентів спирається на різні методологічні підходи — від простих правил і статистичних методів до сучасних трансформерних архітектур і моделей комп'ютерного зору. У контексті розробки системи модерації для Discord особливо важливо враховувати як текстовий, так і візуальний контент, що потребує поєднання методів обробки природної мови та аналізу зображень.

Перед переходом до практичної реалізації системи необхідно детально розглянути існуючі алгоритми, моделі та підходи, що застосовуються для виявлення токсичності та NSFW-контенту, оцінити їх ефективність у реальних умовах та визначити ключові відмінності між ними. Це дозволить обґрунтувати вибір оптимальних рішень для побудови ефективного Discord-бота автоматичної модерації.

2.1 Аналіз методів виявлення токсичних повідомлень

Виявлення токсичності у текстових повідомленнях є однією з ключових задач при створенні систем автоматичної модерації контенту. Ефективність розв'язання цієї задачі залежить від багатьох факторів, зокрема складності мовного матеріалу, обсягу доступних даних та вимог до точності роботи системи. Для аналізу тексту можуть застосовуватися різні підходи —

починаючи від простих правил і фільтрів та закінчуючи сучасними трансформерними моделями.

Кожен із цих підходів має свої особливості, переваги та обмеження, які впливають на їх практичне використання. Вибір конкретного методу залежить від середовища застосування, наприклад таких платформ як Discord, а також від характеру комунікації та типу контенту, що підлягає аналізу.

2.1.1 Rule-based підходи

Rule-based підходи належать до перших методів, які почали застосовуватися для автоматизованого виявлення небажаного контенту в онлайн-комунікації. Їх принцип роботи полягає у використанні заздалегідь визначених правил, таких як списки заборонених слів, шаблони типових висловлювань або регулярні вирази, що сигналізують про наявність агресії чи образливого змісту. Подібні системи не потребують навчання, характеризуються високою швидкістю роботи та прозорістю, оскільки результат напряму залежить від сформованого набору правил.

У простих випадках такі підходи демонструють достатню ефективність, зокрема при виявленні явної нецензурної лексики, прямих образ або закликів до насильства. Проте їх основним недоліком є відсутність урахування контексту. Система не здатна коректно інтерпретувати іронію, метафори чи неоднозначні висловлювання. У результаті нейтральні на перший погляд фрази можуть бути класифіковані як токсичні, тоді як образливі висловлювання, замасковані під жарт або мем, можуть залишатися непоміченими.

Ще одним суттєвим обмеженням є те, що користувачі швидко пристосовуються до таких систем. Токсичний зміст часто маскується шляхом зміни написання слів, додавання символів або використання альтернативних формулювань, які відсутні у словниках. Через це ефективність rule-based рішень знижується, якщо правила не оновлюються регулярно, що потребує постійного втручання людини та ускладнює масштабування на великих

платформах, зокрема таких як Discord.

Для підвищення якості роботи таких систем у дослідженнях пропонувалося поєднувати автоматичне формування шаблонів токсичних висловлювань із їх подальшим ручним доопрацюванням. Було встановлено, що використання додаткових маркерів, наприклад повторення слів великими літерами або специфічної образливої лексики, дозволяє значно підвищити точність окремих правил.

У процесі створення правил використовувалися як окремі слова (уніграми), так і їх поєднання (біграми), отримані з попередньо розмічених корпусів текстів. Проте навіть при такому підході виявилося, що покриття залишається обмеженим, оскільки значна частина токсичних висловлювань не потрапляє під задані шаблони. Це зумовлює необхідність використання комбінованих рішень.

Одним із ефективних напрямів є поєднання rule-based підходів із моделями глибокого навчання, наприклад BERT. У таких системах правила виконують роль початкового фільтра, який відсіює очевидно неприйнятний контент, тоді як нейронні моделі здійснюють більш глибокий контекстуальний аналіз.

Незважаючи на наявні недоліки, rule-based підходи залишаються актуальними завдяки своїй простоті, швидкодії та низьким вимогам до ресурсів. Вони ефективно використовуються як перший етап обробки, дозволяючи зменшити навантаження на більш складні моделі та оперативно реагувати на явні порушення.

Крім того, важливою перевагою таких систем є їхня гнучкість у налаштуванні. Кожне правило є зрозумілим і контрольованим, що дає змогу швидко вносити зміни, виправляти помилки або доповнювати систему новими шаблонами відповідно до потреб конкретн

2.1.2 ML-моделі

Класичні алгоритми машинного навчання стали наступним етапом розвитку після rule-based підходів у задачах модерації контенту, оскільки дозволяють автоматично знаходити закономірності у тексті без необхідності ручного визначення правил. На відміну від статичних словників, такі моделі працюють із числовими ознаками, сформованими на основі статистичних характеристик тексту, що робить їх більш гнучкими та здатними адаптуватися до різних варіантів формулювань.

Ключовою ідеєю цих підходів є перетворення текстових повідомлень у векторний простір, де кожне повідомлення представлено у вигляді набору числових характеристик. Одним із найпоширеніших методів такого подання є TF-IDF (Term Frequency–Inverse Document Frequency). Цей підхід враховує як частоту появи слова в окремому повідомленні, так і його рідкісність у межах усього корпусу текстів. Слова, які часто зустрічаються в конкретному повідомленні, але є рідкісними загалом, отримують більшу вагу, оскільки вони більш інформативні для класифікації.

Завдяки цьому TF-IDF дозволяє ефективніше відрізняти нейтральні тексти від токсичних. Наприклад, слова з негативним емоційним забарвленням можуть виступати сильними індикаторами агресивного контенту, тоді як загальновживані слова мають менший вплив на результат. Представлення у вигляді TF-IDF реалізується як розріджена матриця, що дозволяє ефективно працювати навіть із великими словниками, оскільки кожен текст містить лише невелику частину всіх можливих слів.

Для покращення якості аналізу часто застосовуються n-грамні ознаки, які враховують послідовності слів. Наприклад, біграми або триграми дозволяють фіксувати типові фрази, що можуть мати токсичний характер лише в поєднанні слів. Це особливо корисно у випадках, коли окремі слова не є агресивними, але разом утворюють образливий вислів. Крім того, n-грамний підхід допомагає виявляти замасковану токсичність, коли користувачі змінюють написання слів.

Водночас збільшення довжини n-грам призводить до значного зростання кількості ознак, що може спричинити перенавчання, особливо при обмеженому обсязі даних. Тому на практиці зазвичай використовуються біграми або триграми, а словниковий простір додатково обмежується за частотністю. Комбінація TF-IDF та n-грам створює більш інформативне представлення тексту і є стандартним підходом у класичних задачах NLP.

Отримані векторні представлення використовуються як вхідні дані для алгоритмів класифікації. Найбільш поширеними серед них є логістична регресія, метод опорних векторів (SVM) та наївний байєсівський класифікатор. Ці моделі добре працюють із розрідженими високорозмірними даними та дозволяють ефективно виділяти ознаки, що характеризують токсичний контент.

Логістична регресія є одним із найпростіших, але ефективних методів, який оцінює ймовірність належності повідомлення до певного класу. Вона добре підходить для задач класифікації тексту, забезпечує стабільні результати та дозволяє інтерпретувати вплив окремих ознак на рішення моделі.

Метод опорних векторів орієнтований на знаходження оптимальної межі між класами у просторі ознак. Для текстових задач зазвичай використовується лінійне ядро, що дозволяє ефективно працювати з великими векторними просторами. SVM демонструє високу точність, особливо у випадках, коли дані є частково перекривними або містять складні приклади.

Наївний байєсівський класифікатор базується на ймовірнісному підході та припущенні незалежності ознак. Незважаючи на спрощення, він часто показує хороші результати, особливо на невеликих вибірках, і відзначається високою швидкістю роботи та низькими обчислювальними витратами.

Кожен із цих алгоритмів має свої особливості застосування залежно від характеристик даних, таких як обсяг корпусу, баланс класів та рівень шуму. У задачах виявлення токсичності вони часто використовуються як базові моделі або як частина більш складних систем.

Основною перевагою класичних підходів є їхня простота, швидкодія та інтерпретованість. Вони дозволяють будувати ефективні рішення навіть за

обмежених ресурсів, що є важливим, наприклад, при розробці систем модерації для Discord. Такі моделі можуть використовуватися як початковий рівень аналізу або як базова точка порівняння для більш складних методів.

Разом із тим, класичні моделі мають суттєві обмеження. Вони не враховують повноцінний контекст, погано обробляють складні мовні конструкції та не здатні ефективно виявляти іронію чи приховану агресію. Крім того, їхня здатність до узагальнення є обмеженою, що ускладнює обробку нових або нестандартних формулювань.

Незважаючи на це, класичні методи залишаються важливим інструментом у задачах аналізу тексту. Вони широко застосовуються для побудови прототипів, у ресурсно обмежених системах або як частина комбінованих підходів, де виконують роль попереднього фільтра перед використанням більш складних моделей глибокого навчання.

2.1.3 Глибокі нейронні мережі

Глибокі нейронні мережі стали наступним етапом розвитку методів виявлення токсичного контенту, коли можливостей класичних підходів виявилося недостатньо. На відміну від моделей, що базуються на частотних характеристиках, таких як TF-IDF або n-грамні представлення, глибинні архітектури дозволяють працювати з текстом як із послідовністю, враховуючи взаємозв’язки між словами. Це дає змогу ефективніше виявляти семантичні та синтаксичні залежності, а також розпізнавати складні мовні явища, зокрема приховану агресію, іронію чи неоднозначні висловлювання.

Однією з ключових архітектур для обробки послідовних даних стали рекурентні нейронні мережі, зокрема LSTM (Long Short-Term Memory). Вони були розроблені для подолання проблеми втрати інформації при роботі з довгими послідовностями. Основною ідеєю LSTM є використання внутрішнього стану пам’яті, який дозволяє зберігати важливу інформацію протягом тривалого часу. Завдяки механізму керованих «воріт» модель

визначає, яку інформацію необхідно зберегти, оновити або використати на поточному кроці.

Така структура дозволяє ефективно враховувати як короткі, так і довгі залежності у тексті, що є важливим при аналізі повідомлень, де токсичність може проявлятися поступово або залежати від попереднього контексту. Водночас LSTM мають певні недоліки: вони потребують значних обчислювальних ресурсів, а їх послідовна природа ускладнює паралельну обробку даних, що знижує швидкодію при роботі з великими обсягами інформації.

Як більш компактна альтернатива була запропонована архітектура GRU (Gated Recurrent Unit). Вона спрощує структуру LSTM, об’єднуючи окремі механізми керування інформацією та зменшуючи кількість параметрів. Завдяки цьому GRU працює швидше та потребує менше ресурсів, при цьому зберігаючи високу якість результатів. У багатьох випадках такі моделі демонструють порівнянну або навіть кращу ефективність, особливо при роботі з обмеженими наборами даних.

Вибір між LSTM та GRU зазвичай визначається компромісом між точністю та продуктивністю. У задачах модерації контенту, зокрема в системах на кшталт Discord, де важлива швидкість обробки повідомлень у реальному часі, GRU часто є більш доцільним варіантом.

Ще одним підходом до аналізу тексту є використання згорткових нейронних мереж (CNN). Хоча спочатку ці моделі були розроблені для обробки зображень, вони успішно застосовуються і в задачах обробки тексту. Основною перевагою CNN є здатність виявляти локальні шаблони — характерні фрази або поєднання слів, які можуть сигналізувати про токсичний зміст [6].

Типова архітектура CNN для тексту включає перетворення слів у векторні представлення, застосування згорткових фільтрів для виявлення локальних патернів, а також використання операцій підвибірки для виділення найбільш значущих ознак. Такий підхід дозволяє ефективно знаходити ключові фрази незалежно від їх позиції в тексті.

Згорткові мережі мають ряд практичних переваг. Вони здатні швидко обробляти дані завдяки паралельним обчисленням, що є важливим для систем реального часу. Крім того, вони добре працюють із короткими текстами, такими як повідомлення в чатах або коментарі, де токсичність часто виражається у компактній формі.

Попри значні переваги, глибокі нейронні мережі мають і певні обмеження. Вони потребують великої кількості розмічених даних для ефективного навчання, а також характеризуються низькою інтерпретованістю результатів. Це ускладнює аналіз причин прийнятих рішень і може бути критичним у системах модерації.

У сучасних рішеннях такі моделі часто замінюються трансформерними архітектурами або використовуються як їх складові. Проте LSTM, GRU та CNN відіграли важливу роль у розвитку методів обробки природної мови і досі залишаються актуальними у випадках, коли необхідно досягти балансу між точністю, швидкістю та обчислювальними витратами.

2.1.4 Моделі на основі трансформерів

Моделі, побудовані на основі трансформерної архітектури, стали важливим етапом розвитку методів обробки природної мови, зокрема у задачах автоматичного виявлення токсичного контенту. На відміну від рекурентних підходів, де обробка відбувається послідовно, трансформери дозволяють аналізувати всю текстову послідовність одночасно. Це стало можливим завдяки механізму self-attention, який забезпечує врахування взаємозв’язків між словами незалежно від їхнього розташування у реченні. Такий підхід особливо ефективний у випадках, коли токсичність має прихований характер або залежить від контексту.

Однією з найбільш відомих моделей цього класу є BERT (Bidirectional Encoder Representations from Transformers). Її ключова особливість полягає у двонаправленому аналізі тексту, коли кожен елемент розглядається з

урахуванням як попереднього, так і наступного контексту. Це дозволяє моделі точніше інтерпретувати складні мовні конструкції та краще розпізнавати смислові відтінки висловлювань.

Разом із тим, використання BERT пов’язане з високими обчислювальними витратами, оскільки модель містить значну кількість параметрів. Для зменшення ресурсоємності було розроблено ряд оптимізованих варіантів (див.рис.2.1). Зокрема, модель ALBERT використовує повторне застосування параметрів між шарами та оптимізоване представлення ембедінгів, що дозволяє зменшити обсяг пам’яті без суттєвої втрати якості [5].

Подальшим розвитком цієї ідеї стала модель ELBERT, яка доповнює оптимізовану архітектуру механізмом раннього завершення обчислень. У такому підході модель може припинити обробку на проміжному рівні, якщо вже досягнуто достатньої впевненості у результаті.

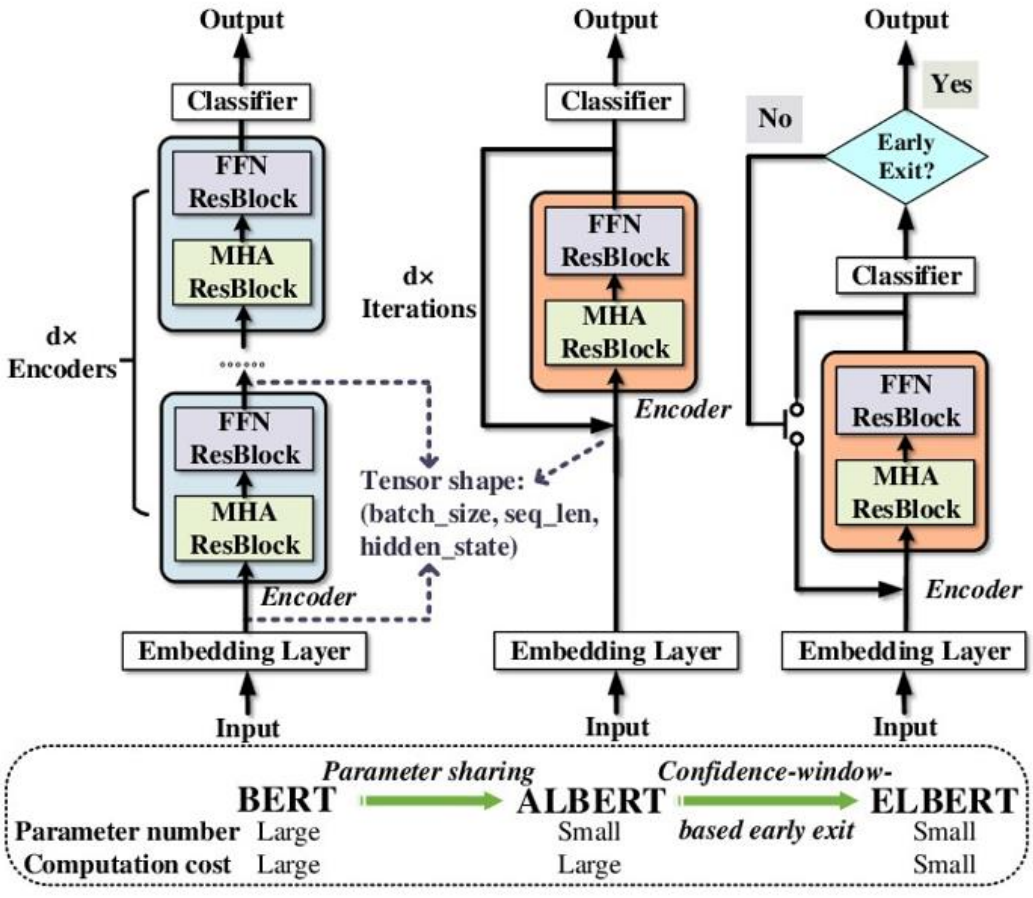


Рисунок 2.1 – Архітектури BERT, ALBERT і ELBERT

Це дає змогу адаптувати глибину обчислень до складності вхідного тексту та підвищити ефективність роботи в системах реального часу.

У задачах виявлення токсичності трансформерні моделі зазвичай застосовуються у вигляді попередньо навчених рішень із додатковим класифікаційним шаром або шляхом донавчання на спеціалізованих наборах даних, що містять приклади токсичних повідомлень.

Серед удосконалених варіантів варто відзначити RoBERTa, яка є оптимізованою версією BERT із покращеною процедурою навчання. Вона демонструє вищу точність у задачах класифікації тексту, зокрема при виявленні непрямих або складних форм агресії. Модель DistilBERT, у свою чергу, орієнтована на зменшення обчислювальних витрат і забезпечує баланс між швидкодією та якістю, що робить її придатною для використання в онлайн-системах, наприклад у ботах для Discord.

Окрім універсальних моделей, існують також спеціалізовані рішення, адаптовані безпосередньо до задач модерації. До них належать моделі, донавчені на тематичних датасетах токсичних коментарів, які здатні класифікувати різні типи агресивного контенту. Також існують готові сервіси з API-доступом, що дозволяють інтегрувати функціональність виявлення токсичності без необхідності самостійного навчання моделей. Для багатомовних сценаріїв використовуються моделі на кшталт XLM-RoBERTa, які підтримують роботу з різними мовами, включаючи українську.

Завдяки високій точності, здатності враховувати контекст та гнучкості у використанні, трансформерні моделі стали сучасним стандартом у задачах аналізу токсичності. Вони ефективно працюють як із простими, так і зі складними текстами, забезпечують хорошу узагальнювальну здатність і можуть застосовуватись навіть за умов обмеженої кількості розмічених даних.

2.2 Аналіз методів обробки зображень для виявлення NSFW-контенту

Автоматичне розпізнавання NSFW-контенту у зображеннях є важливою

складовою сучасних систем модерації цифрового середовища. Складність цієї задачі зумовлена великою варіативністю візуальних даних, різними форматами подання контенту та неоднозначністю його інтерпретації. Якість роботи таких систем залежить від багатьох чинників, зокрема характеристик вхідних зображень, обсягу навчальних даних і вимог до точності класифікації.

Для вирішення цієї задачі застосовується широкий спектр методів — від класичних алгоритмів комп'ютерного зору, що базуються на аналізі кольорів і текстур, до сучасних підходів глибокого навчання та трансформерних моделей, здатних враховувати складні візуальні закономірності. Кожен із підходів має власну сферу ефективного застосування, а також певні обмеження, які необхідно враховувати під час проєктування системи.

Вибір конкретного рішення значною мірою визначається умовами використання, наприклад при інтеграції у платформи на кшталт Discord, де важливими є швидкість обробки, масштабованість і здатність працювати з різнорідним користувацьким контентом.

2.2.1 Методи на основі комп'ютерного зору

Методи на основі комп'ютерного зору стали одним із перших підходів до автоматичного виявлення NSFW-контенту в зображеннях. Вони базуються на аналізі візуальних характеристик зображення, таких як колір, текстура, форма об'єктів та їх просторове розташування. На відміну від сучасних глибинних моделей, ці підходи не потребують великих обсягів даних для навчання, однак значною мірою залежать від якості вручну визначених ознак.

Одним із найпростіших і водночас поширених підходів є аналіз кольорових характеристик, зокрема виявлення ділянок шкіри. Ідея полягає у визначенні пікселів, що відповідають певному діапазону кольорів, характерному для людської шкіри, після чого оцінюється їхня кількість і розподіл у зображенні. Якщо частка таких пікселів перевищує заданий поріг, зображення може бути класифіковане як потенційно небажане. Проте цей

підхід є доволі обмеженим, оскільки не враховує контекст: велика кількість тілесних кольорів може бути присутня і в цілком допустимих зображеннях, наприклад, на фотографіях пляжу або спортивних подій.

Більш складні методи комп’ютерного зору використовують аналіз текстур та форм об’єктів. У таких підходах застосовуються дескриптори ознак, які дозволяють виявляти характерні патерни, притаманні певним типам зображень. Наприклад, можуть використовуватися гістограми орієнтованих градієнтів або локальні бінарні шаблони для опису структури зображення. Отримані ознаки передаються до класифікаторів, які визначають належність зображення до певного класу.

Ще одним напрямом є використання методів сегментації, які дозволяють виділяти окремі області зображення та аналізувати їх окремо. Це дає змогу точніше визначати, які частини зображення можуть містити небажаний контент. У деяких випадках також застосовується детекція об’єктів, коли система намагається знайти конкретні елементи, пов’язані з людським тілом або певними сценами.

Перевагою класичних методів комп’ютерного зору є їхня відносна простота та низькі обчислювальні витрати. Вони можуть працювати в режимі реального часу та не потребують значних ресурсів, що робить їх придатними для використання у простих системах або як початковий етап обробки. Наприклад, у системах модерації для платформ на кшталт Discord такі методи можуть виконувати роль швидкого попереднього фільтра.

Водночас ці підходи мають суттєві обмеження. Вони погано узагальнюють нові або складні типи зображень, не враховують глибокий контекст сцени та можуть давати велику кількість хибних спрацювань. Крім того, їхня ефективність значною мірою залежить від якості ручного налаштування ознак, що ускладнює адаптацію до нових умов.

У сучасних системах модерації такі методи рідко використовуються як самостійне рішення, проте можуть застосовуватися у поєднанні з більш складними моделями, виконуючи допоміжну роль або забезпечуючи первинну

обробку зображень перед передачею їх до глибинних нейронних мереж.

2.2.2 Мультимодальні моделі на основі трансформерів

Мультимодальні моделі на основі трансформерів є сучасним підходом до виявлення NSFW-контенту, який поєднує аналіз зображень і текстової інформації в єдиній архітектурі. На відміну від класичних методів комп’ютерного зору, що працюють лише з візуальними ознаками, такі моделі здатні враховувати семантичний контекст, співставляючи зображення з текстовими описами або запитами. Це значно розширює можливості системи при обробці складного або неоднозначного контенту.

Основою подібних підходів є трансформерна архітектура, яка застосовується як для обробки тексту, так і для представлення зображень. Зображення попередньо перетворюється у набір векторів, наприклад шляхом поділу на патчі, після чого ці представлення обробляються аналогічно до текстових послідовностей. Завдяки механізму self-attention модель може встановлювати взаємозв’язки між різними частинами зображення, а також між візуальними та текстовими ознаками.

Одним із ключових підходів у мультимодальних моделях є спільне навчання на парах «зображення–текст». У процесі такого навчання модель формує єдиний простір представлень, у якому близькі за змістом зображення та текстові описи мають схожі вектори. Це дозволяє виконувати класифікацію NSFW-контенту шляхом порівняння зображення з відповідними текстовими категоріями, наприклад «безпечний контент» або «небажаний контент», без необхідності окремого навчання під конкретну задачу.

До найбільш відомих представників таких моделей належить CLIP (Contrastive Language–Image Pretraining), розроблений компанією OpenAI. Він дозволяє порівнювати зображення з текстовими описами та ефективно використовуватися для zero-shot класифікації NSFW-контенту. Іншим прикладом є BLIP (Bootstrapping Language-Image Pretraining), який поєднує

задачі генерації підписів до зображень і їх розуміння, що дозволяє глибше аналізувати візуальний зміст. Також варто згадати моделі на основі Vision Transformer у поєднанні з текстовими енкодерами, які формують основу сучасних мультимодальних систем.

Перевагою мультимодальних трансформерів є їхня гнучкість і здатність працювати в умовах обмежених даних. Вони можуть використовуватися у режимах zero-shot або few-shot, що дозволяє адаптувати систему до нових типів контенту без повного перенавчання. Крім того, такі моделі краще справляються з контекстно складними випадками, коли рішення неможливо прийняти лише на основі візуальних ознак.

Разом із тим, ці підходи мають і певні недоліки. Вони є обчислювально складними та потребують значних ресурсів для ефективного використання. Також якість результатів може залежати від правильно сформульованих текстових запитів або категорій, що додає додатковий рівень складності при налаштуванні системи.

У контексті розробки систем модерації, зокрема для платформ на кшталт Discord, мультимодальні трансформерні моделі відкривають можливість створення більш універсальних і точних рішень, які здатні одночасно аналізувати різні типи контенту та враховувати їх взаємозв’язок. Це робить їх перспективним напрямом для побудови сучасних систем автоматичної модерації.

3 ВИБІР ТЕХНОЛОГІЙ ТА ЗАСОБІВ РОЗРОБКИ DISCORD-БОТА

Під час розробки Discord-бота для автоматичної модерації повідомлень і зображень було обрано набір технологій, орієнтованих на створення масштабованої, гнучкої та придатної до практичного використання системи. Важливими критеріями вибору стали підтримка сучасних ML-моделей, можливість інтеграції з Discord API, простота розгортання, продуктивність та доступність бібліотек для роботи з трансформерними архітектурами.

3.1 Вибір мови програмування

Мовою програмування було обрано Python 3.14. Вибір саме цієї мови зумовлений її широким застосуванням у сфері розробки систем штучного інтелекту, обробки природної мови та комп’ютерного зору. Python є однією з найпоширеніших мов у галузі машинного навчання завдяки простому синтаксису, великій кількості готових бібліотек і високій швидкості розробки програмних рішень.

Однією з головних переваг Python є його зрозуміла та лаконічна структура. Це дозволяє реалізовувати складну логіку програмних систем із меншою кількістю коду порівняно з багатьма іншими мовами програмування. Такий підхід спрощує підтримку проєкту, полегшує модифікацію окремих компонентів та забезпечує кращу масштабованість системи в майбутньому.

Python активно використовується у сфері машинного навчання та аналізу даних, що робить його особливо придатним для створення систем автоматичної модерації контенту. Для цієї мови існує велика кількість готових бібліотек і фреймворків, які дозволяють інтегрувати сучасні моделі штучного інтелекту без необхідності реалізації складних алгоритмів з нуля. Завдяки цьому суттєво скорочується час розробки та спрощується інтеграція моделей аналізу тексту й зображень у програмну систему.

Додатковою перевагою Python є підтримка асинхронного програмування, що є важливим для роботи Discord-бота в режимі реального часу. Система повинна одночасно отримувати повідомлення від користувачів, аналізувати контент, взаємодіяти з API платформи та виконувати модераційні дії без значних затримок. Механізми асинхронного виконання дозволяють ефективно обробляти велику кількість подій та підвищують загальну продуктивність програмного засобу.

Важливим фактором також є активна спільнота розробників Python та наявність великої кількості документації й прикладів реалізації різноманітних систем. Це значно полегшує процес розробки, тестування та подальшого вдосконалення програмного продукту. Завдяки поєднанню простоти, гнучкості та широких можливостей інтеграції сучасних AI-технологій Python є доцільним вибором для створення інтелектуальної системи автоматичної модерації контенту.

Під час розробки програмної системи використовувалось віртуальне середовище Python, призначене для ізоляції залежностей проєкту та забезпечення стабільної роботи програмного забезпечення. Використання virtual environment дозволяє встановлювати необхідні бібліотеки окремо для конкретного проєкту без впливу на глобальні налаштування Python у системі.

Для розробки програмної системи було використано інтегроване середовище розробки PyCharm. Вибір цього середовища зумовлений його орієнтованістю на роботу з мовою Python та наявністю широкого набору інструментів для створення, тестування й налагодження програмних застосунків. PyCharm забезпечує зручну роботу зі структурою проєкту, підтримує автоматичне керування віртуальним середовищем та дозволяє ефективно працювати з великою кількістю модулів і бібліотек.

Однією з важливих переваг PyCharm є вбудовані засоби аналізу коду, автодоповнення та пошуку помилок, що значно спрощує процес розробки та підвищує якість програмного забезпечення. Середовище також підтримує

інтегроване керування залежностями, запуск і тестування програм без необхідності використання сторонніх інструментів.

3.2 Визначення бібліотек для розробки бота

В першу чергу, для розробки бота потрібен інструмент для робоити з даними Discord. Для забезпечення взаємодії розроблюваної системи з платформою Discord необхідно буде використовувати Discord API — набір програмних інтерфейсів, що надає можливість отримувати доступ до функціоналу серверів, каналів, повідомлень і користувачів. Використання API дозволить реалізувати автоматичне отримання повідомлень у режимі реального часу, аналіз вмісту текстових повідомлень та вкладених зображень, а також виконання дій модерації без участі адміністратора.

Для спрощення роботи з Discord API планується використання бібліотеки discord.py, яка забезпечує високорівневий інтерфейс для створення Discord-ботів мовою Python. Завдяки цій бібліотеці буде реалізовано обробку подій сервера, взаємодію з користувачами, отримання вкладень, видалення неприйнятного контенту та виконання адміністративних команд.

Важливою перевагою discord.py є підтримка асинхронної моделі виконання, що дозволить системі одночасно обробляти кілька подій без суттєвих затримок. Це є особливо важливим для задач автоматичної модерації, де швидкість реагування на токсичний або NSFW-контент безпосередньо впливає на ефективність роботи системи.

Крім базової взаємодії з повідомленнями, Discord API та discord.py надаватимуть можливість працювати з ролями користувачів, системою прав доступу та механізмами модерації платформи. Завдяки цьому у програмному засобі можна буде реалізувати функціонал автоматичних попереджень, тимчасового обмеження доступу до сервера та блокування користувачів у разі систематичних порушень правил спільноти.

Для вирішення задачі аналізу текстових повідомлень буде необхідно використовувати бібліотеку Hugging Face Transformers, яка є одним із найпоширеніших інструментів для роботи з сучасними трансформерними моделями. Бібліотека надає готові засоби для інтеграції попередньо навчених моделей машинного навчання та підтримує широкий спектр задач обробки природної мови, зокрема класифікацію тексту, аналіз тональності, генерацію тексту та виявлення токсичного контенту.

Однією з головних переваг Hugging Face Transformers є можливість використання pretrained-моделей без необхідності їх навчання з нуля. Це дозволить значно скоротити час розробки системи та використовувати вже готові моделі, навчені на великих текстових корпусах. Для задач автоматичної модерації контенту така бібліотека забезпечуватиме можливість інтеграції моделей типу BERT, RoBERTa або XLM-RoBERTa для аналізу токсичності повідомлень.

Найбільш зручною буде модель XLM-RoBERTa — багатомовну трансформерну архітектуру, побудовану на основі моделі RoBERTa. Особливістю XLM-RoBERTa є підтримка великої кількості мов, зокрема української та англійської, що дозволяє використовувати одну модель для аналізу повідомлень різними мовами без необхідності автоматичного перекладу тексту.

Бібліотека також забезпечує високорівневий API через механізм pipeline, що спрощує процес взаємодії з трансформерними моделями. Завдяки цьому система зможе отримувати текстові повідомлення від користувачів Discord та автоматично передавати їх на аналіз моделі для визначення рівня токсичності або належності до певної категорії неприйнятного контенту.

Додатковою перевагою Hugging Face Transformers є підтримка багатомовних моделей, що є важливим для роботи з англомовними та україномовними повідомленнями. Це дозволить створити більш гнучку систему модерації, здатну працювати в багатомовному середовищі без необхідності окремої реалізації моделей для кожної мови.

PyTorch є одним із найпоширеніших інструментів для побудови та виконання нейронних мереж і широко застосовується у сучасних AI-системах, пов’язаних з обробкою тексту та зображень.

Він забезпечує ефективну роботу з багатовимірними масивами даних, які є основою обчислень у нейронних мережах. Завдяки цьому фреймворк дозволяє швидко виконувати операції над великими обсягами даних та підтримує використання апаратного прискорення через GPU. Це є важливим для задач аналізу токсичних повідомлень та NSFW-зображень, оскільки трансформерні моделі та моделі комп’ютерного зору потребують значних обчислювальних ресурсів.

Однією з ключових особливостей PyTorch є динамічний обчислювальний граф, який спрощує процес розробки та налагодження моделей. Такий підхід дозволяє більш гнучко працювати зі складними архітектурами нейронних мереж і спрощує інтеграцію сучасних AI-моделей у програмну систему. Крім того, PyTorch активно використовується більшістю сучасних бібліотек машинного навчання, що забезпечує сумісність із трансформерними моделями та засобами комп’ютерного зору.

У межах системи автоматичної модерації PyTorch буде використовуватись як базова платформа для виконання моделей аналізу тексту та зображень, забезпечуючи швидке виконання inference-процесів і стабільну роботу компонентів штучного інтелекту.

Для реалізації модуля аналізу зображень планується використання бібліотеки OpenCLIP, яка є відкритою реалізацією архітектури CLIP (Contrastive Language–Image Pretraining). Основне призначення цієї технології полягає у спільному аналізі текстової та візуальної інформації, що дозволяє моделі встановлювати семантичний зв’язок між зображенням і текстовими описами.

На відміну від класичних моделей комп’ютерного зору, які виконують класифікацію лише за фіксованим набором категорій, OpenCLIP використовує мультимодальний підхід. Модель формує векторні представлення як для

зображення, так і для текстових описів, після чого порівнює їхню схожість у спільному просторі ознак. Завдяки цьому система зможе визначати, наскільки зображення відповідає категоріям на кшталт «NSFW content», «nudity», «explicit image» або іншим текстовим описам неприйнятного контенту.

Важливою перевагою OpenCLIP є підтримка pretrained-моделей, навчених на великих мультимодальних датасетах. Це дозволяє використовувати вже готові моделі для виявлення NSFW-зображень без необхідності повного навчання нейронної мережі з нуля. Такий підхід спрощує інтеграцію системи аналізу зображень у Discord-бота та зменшує обчислювальні витрати під час розробки.

Крім цього, OpenCLIP підтримує сучасні трансформерні архітектури комп'ютерного зору, зокрема Vision Transformer (ViT), що забезпечує високу якість аналізу графічного контенту та дозволяє виявляти складні або частково приховані прояви NSFW-контенту. Завдяки поєднанню мультимодального підходу та трансформерних моделей OpenCLIP є ефективним рішенням для побудови систем автоматичної модерації зображень у Discord-середовищі.

Також для роботи із графічним контентом у системі автоматичної модерації планується використання бібліотеки Pillow (PIL — Python Imaging Library). Вона забезпечує базові засоби обробки зображень і дозволяє виконувати підготовку графічних файлів перед їх передачею до моделей комп'ютерного зору.

За допомогою Pillow система зможе відкривати зображення різних форматів, виконувати конвертацію кольорових просторів, змінювати розмір зображень та здійснювати інші операції попередньої обробки. Це є важливим етапом перед аналізом NSFW-контенту, оскільки більшість моделей машинного навчання потребує стандартизованого формату вхідних даних.

Перевагою Pillow є простота інтеграції з Python та сумісність із бібліотеками машинного навчання. Підготовлені за допомогою Pillow зображення можуть безпосередньо передаватися у моделі OpenCLIP або інші системи комп'ютерного зору для подальшого аналізу.

Інформацію про користувачів та їх порурушення потрібно буде зберігати для коректної модерації серверо, тому потрібно реалізувати базу даних. SQLite забезпечуватиме збереження даних про користувачів, типи порушень, результати аналізу моделей, час фіксації інцидентів та виконані дії модерації. Завдяки цьому система зможе вести історію порушень кожного користувача, аналізувати повторні випадки токсичної поведінки та реалізовувати механізм накопичення санкцій, таких як warn, timeout або ban.

Однією з головних переваг SQLite є простота інтеграції з Python та невеликі вимоги до ресурсів. Для роботи з базою даних не потрібно окремо налаштовувати сервер або систему адміністрування, що спрощує розгортання програмного засобу та зменшує складність архітектури проєкту.

SQLite забезпечує достатню швидкодію для задач Discord-бота, оскільки обсяг даних у системі модерації є відносно невеликим, а більшість операцій полягає у швидкому записі та отриманні інформації про порушення користувачів. Це робить SQLite практичним рішенням для локальної системи автоматичної модерації контенту.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ОПИС ЗАСТОСУНКУ

4.1 Discord Developer Portal

Першим етам для розробки буде створення Discord-бота на сайті Discord Developer Portal, де потрібно авторизуватися у власному обліковому записі Discord. Після цього створюється новий застосунок за допомогою кнопки New Application. На цьому етапі вказується назва застосунку, яка надалі буде відображатися як назва бота або пов’язаного з ним програмного продукту.

Після створення застосунку необхідно перейти до розділу Bot і додати до застосунку бота. У цьому розділі формується токен доступу, який використовується програмним кодом для підключення до Discord API. Токен є конфіденційним параметром, тому його не можна публікувати або зберігати у відкритому вигляді в коді. Сторінка з налаштуваннями бота представленно на рисунку 4.1.

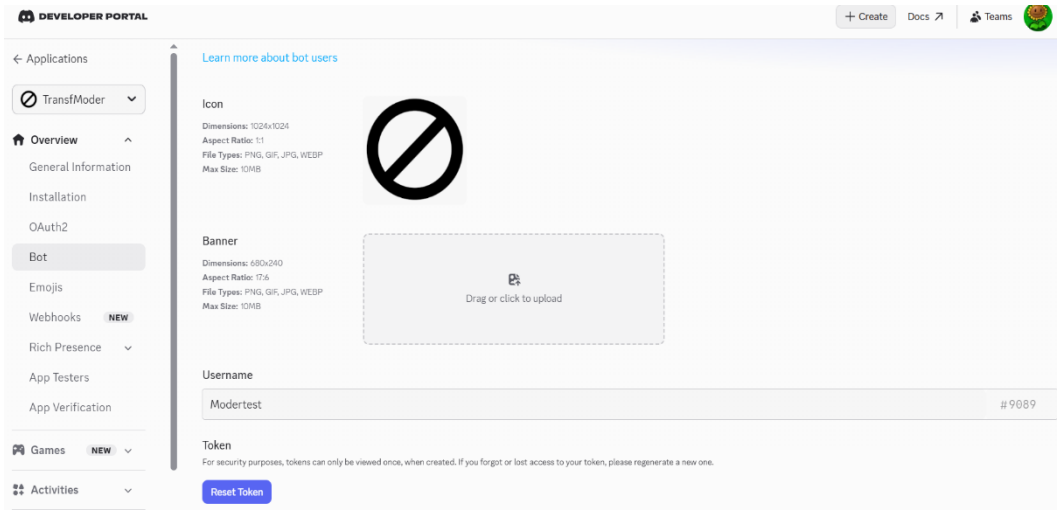


Рисунок 4.1 – Сторінка з налаштуваннями бота

Далі потрібно налаштувати права доступу бота. У розділі Bot необхідно активувати потрібні intent-параметри, зокрема Message Content Intent, щоб бот міг отримувати текст повідомлень, а також Server Members Intent, якщо система повинна працювати з інформацією про користувачів сервера. Це важливий

етап, оскільки без відповідних дозволів бот не зможе повноцінно виконувати функції автоматичної модерації.

Наступним кроком є створення посилання для додавання бота на сервер. Для цього використовується розділ OAuth2, далі URL Generator. У полі Scopes потрібно обрати пункт bot, після чого вказати необхідні права доступу, наприклад перегляд каналів, читання історії повідомлень, надсилання повідомлень, керування повідомленнями, тимчасове обмеження користувачів або їх блокування. Після вибору прав система генерує спеціальне посилання, за яким бот може бути доданий на потрібний Discord-сервер.

Після переходу за згенерованим посиланням обирається сервер, на який буде додано бота, і підтверджується надання йому необхідних дозволів. Після цього бот з’являється на сервері, однак для його повноцінної роботи необхідно запустити програмний код локально або на сервері. Саме код забезпечує обробку повідомлень, аналіз токсичності, перевірку зображень і виконання модераційних дій.

Завершальним етапом є перевірка прав бота безпосередньо на Discord-сервері. Його роль повинна мати достатні дозволи для видалення повідомлень, надсилання попереджень, застосування timeout або ban. Також бажано розмістити роль бота вище за ролі користувачів, до яких він має застосовувати модераційні дії. Після цього бот буде готовий до інтеграції з програмною частиною системи автоматичної модерації контенту. Створений бот, якого додали на сервер зображено на рисунку 4.2.

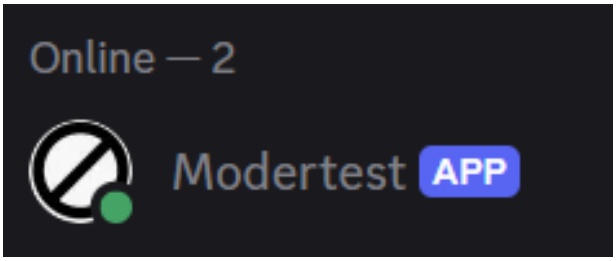


Рисунок 4.2 – Створений бот

4.2 Архітектура проекту

Архітектура програмної реалізації Discord-бота побудована за модульним принципом, тому кожна частина системи відповідає за окрему функцію: взаємодію з Discord, аналіз тексту, аналіз зображень, прийняття рішення, роботу з базою даних та логування. Такий підхід спрощує розробку, тестування й подальше розширення системи. На рисунку 4.3 зображено структуру проекту.

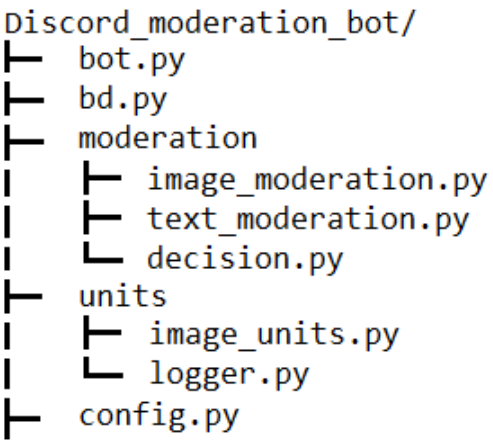


Рисунок 4.3 – Структура проекту

Основним елементом програмної системи є файл bot.py, який виконує функцію центрального керуючого модуля Discord-бота. Саме через цей файл здійснюється запуск усієї системи, ініціалізація компонентів модерації та організація взаємодії між окремими модулями проєкту. У ньому створюється екземпляр бота, налаштовуються права доступу (intents), підключаються модулі аналізу тексту, зображень і база даних, а також визначаються події та команди, на які реагуватиме бот.

Однією з основних задач bot.py є обробка подій Discord-сервера у режимі реального часу. Після запуску програми бот підключається до Discord API та починає відстежувати нові повідомлення в текстових каналах. Коли користувач надсилає повідомлення, Discord API генерує відповідну подію, яка автоматично

обробляється ботом через асинхронний механізм on_message. У межах цієї події система аналізує структуру повідомлення: перевіряє наявність текстового вмісту, вкладених файлів або зображень, після чого передає дані до відповідних модулів аналізу.

Файл bot.py також відповідає за координацію роботи всієї системи модерації. Після отримання результатів від моделей аналізу тексту та зображень він приймає рішення щодо подальших дій: видалення повідомлення, логування інциденту, запису порушення в базу даних або застосування санкцій до користувача. Завдяки такому підходу bot.py фактично виступає ядром програмної системи, яке об’єднує всі інші компоненти в єдину інформаційну систему автоматичної модерації контенту. Приклад ініціалізації Discord-бота на рисунку 4.4.

```python
import discord
from discord.ext import commands

intents = discord.Intents.default()
intents.message_content = True
intents.members = True

bot = commands.Bot(
    command_prefix="!",
    intents=intents
)

@bot.event
async def on_ready():
    print(f"Bot connected as {bot.user}")

bot.run(DISCORD_TOKEN)
```

Рисунок 4.4 – Ініціалізації Discord-бота

У цьому фрагменті коду створюється екземпляр Discord-бота, активуються необхідні дозволи для отримання текстових повідомлень та інформації про користувачів сервера, а також реалізується обробка події успішного підключення до Discord API.

Модуль text_moderation.py реалізує функціонал автоматичного аналізу текстових повідомлень користувачів Discord-сервера. Основне призначення цього модуля полягає у виявленні токсичних, агресивних або образливих висловлювань за допомогою трансформерних моделей обробки природної мови. Саме цей компонент системи відповідає за взаємодію між текстовими повідомленнями користувачів та моделлю машинного навчання.

У межах модуля виконується підключення попередньо навченої трансформерної моделі через бібліотеку Hugging Face Transformers. Для аналізу повідомлень використовується механізм pipeline, який спрощує взаємодію з моделлю та дозволяє виконувати класифікацію тексту без необхідності ручної реалізації процесу токенізації або обробки тензорів. Після отримання повідомлення від Discord-бота текст передається у модель, яка повертає набір категорій токсичності та відповідні значення впевненості (score). Реалізації модуля аналізу тексту наведено на рисунку 4.5.

```python
from dataclasses import dataclass
from transformers import pipeline

from config import TEXT_MODEL_NAME


@dataclass  3+ usages
class TextModerationResult:
    is_toxic: bool
    score: float
    label: str


class TextModerator:  2+ usages
    def __init__(self):
        self.classifier = pipeline(
            task: "text-classification",
            model=TEXT_MODEL_NAME,
            tokenizer=TEXT_MODEL_NAME,
            top_k=None,
            truncation=True,
        )

    def analyze(self, text: str) -> TextModerationResult:
        if not text.strip():
            return TextModerationResult(False, 0.0, "empty")
```

Рисунок 4.5 – Реалізації модуля аналізу тексту

Отримані оцінки використовуються для визначення рівня токсичності повідомлення. Система аналізує результати класифікації та порівнює отримане значення score із заданим порогом токсичності. Якщо значення перевищує встановлений threshold, повідомлення вважається неприйнятним і надалі може бути видалене або використане для застосування санкцій до користувача.

Окремою перевагою реалізації є підтримка багатомовних трансформерних моделей, зокрема XLM-RoBERTa, що дозволяє виконувати аналіз повідомлень українською та англійською мовами без необхідності попереднього перекладу тексту. Завдяки контекстному аналізу модель здатна виявляти не лише пряму нецензурну лексику, а й приховані форми агресії, сарказму або токсичних висловлювань.

Модуль image_moderation.py відповідає за автоматичний аналіз зображень, які користувачі надсилають у Discord-канали як вкладення до повідомлень. Його основним завданням є визначення того, чи містить зображення небажаний або NSFW-контент. На відміну від текстового модуля, цей компонент працює з візуальними даними та використовує мультимодальний підхід, тобто поєднує аналіз зображення з текстовими описами можливих класів.

У цьому модулі реалізується робота з бібліотекою OpenCLIP, яка використовується як основа для CLIP-based NSFW Detector. Спочатку зображення відкривається за допомогою бібліотеки Pillow, конвертується у формат RGB та проходить попередню обробку відповідно до вимог CLIP-моделі. Після цього модель формує векторне представлення зображення, яке відображає його зміст у спільному просторі ознак.

Окремо формуються текстові описи класів, з якими буде порівнюватися зображення. До таких описів можуть належати безпечні категорії, наприклад a safe image, a normal photo, а також описи небажаного контенту, наприклад nudity, explicit sexual content, nsfw image. Модель також перетворює ці текстові описи у векторні представлення. Далі виконується порівняння вектора зображення з векторами текстових описів, після чого система визначає, до якої

групи зображення ближче за змістом.

Результатом роботи модуля є числова оцінка NSFW-ризику. Якщо значення цієї оцінки перевищує встановлений поріг, зображення класифікується як неприйнятне, а повідомлення передається до модуля прийняття рішення для подальшої модерації. Такий підхід є гнучким, оскільки дозволяє змінювати або розширювати набір текстових описів без повного перенавчання моделі. Приклад фрагменту коду, для форматування зображення, наведено на рисунку 4.6.

```python
def analyze(self, image_bytes: bytes) -> ImageModerationResult:
    image = Image.open(BytesIO(image_bytes)).convert("RGB")
    image_tensor = self.preprocess(image).unsqueeze(0).to(self.device)

    with torch.no_grad():
        image_features = self.model.encode_image(image_tensor)
        image_features /= image_features.norm(dim=-1, keepdim=True)

        similarity = (100.0 * image_features @ self.text_features.T).softmax(dim=-1)
        scores = similarity[0].detach().cpu().tolist()

    prompt_scores = list(zip(self.prompts, scores))
    best_prompt, best_score = max(prompt_scores, key=lambda x :{__getitem__} : x[1])
    nsfw_score = sum(score for prompt, score in prompt_scores if prompt in self.nsfw_prompt_set)

    return ImageModerationResult(
        is_nsfw=nsfw_score > 0.0,
        score=float(nsfw_score),
        best_prompt=best_prompt,
    )
```

Рисунок 4.6 – Форматування зображення

Для спрощення структури проєкту та уникнення дублювання коду допоміжний функціонал системи винесено до окремої папки utils. Такий підхід дозволяє логічно відокремити другорядні службові компоненти від основних модулів аналізу тексту та зображень. Крім цього, використання окремих utility-модулів підвищує читабельність коду, спрощує підтримку системи та дозволяє повторно використовувати допоміжні функції в різних частинах проєкту.

Одним із таких компонентів є файл image_utils.py, який відповідає за первинну роботу із вкладеннями Discord-повідомлень. Перед передачею файлу

до NSFW-модуля система повинна перевірити, чи є вкладення дійсно зображенням. Для цього аналізується MIME-тип або розширення файлу. Якщо файл належить до підтримуваних графічних форматів, система завантажує його у пам’ять для подальшого аналізу моделлю OpenCLIP. Це дозволяє уникнути помилок при обробці інших типів вкладень, наприклад архівів або документів.

Окрему роль у системі виконує файл logger.py, який реалізує механізм журналювання роботи Discord-бота. Логування є важливим елементом будь-якої інформаційної системи, оскільки дозволяє контролювати її стан, виявляти помилки та аналізувати поведінку користувачів. У межах проєкту журнал використовується для фіксації запуску бота, підключення до Discord API, виявлення токсичних повідомлень або NSFW-контенту, а також виконання модераційних дій, таких як видалення повідомлень, timeout чи ban.

Ведення журналу роботи системи також спрощує процес тестування та налагодження програмного забезпечення. У разі виникнення помилки розробник може швидко визначити її причину, переглянувши відповідні записи у логах. Крім того, логування дозволяє відстежувати статистику роботи системи модерації та аналізувати ефективність моделей машинного навчання.

Для збереження інформації про роботу системи модерації та обліку порушень користувачів у проєкті використовується модуль db.py, розташований у папці database. Цей компонент відповідає за взаємодію Discord-бота з базою даних SQLite та реалізує основні операції збереження, оновлення й отримання інформації про порушення користувачів.

Одним із головних завдань модуля є створення таблиці порушень під час першого запуску системи. У таблиці зберігаються дані про кожен зафіксований інцидент: ідентифікатор користувача Discord, тип порушення (токсичне повідомлення або NSFW-зображення), оцінка моделі, дата й час події, а також модераційна дія, яка була застосована системою. Такий підхід дозволяє формувати історію поведінки користувачів та реалізувати накопичувальну систему санкцій.

Після виявлення неприйнятного контенту Discord-бот звертається до

db.py для додавання нового запису про порушення. Збереження інформації у базі даних дає змогу не лише фіксувати окремі інциденти, а й аналізувати кількість попередніх порушень конкретного користувача. Саме на основі цієї інформації система визначає, яку санкцію необхідно застосувати: попередження, timeout або ban.

Окрім запису нових даних, модуль також реалізує функції підрахунку кількості порушень користувача та очищення історії за потреби. Це дозволяє гнучко керувати системою модерації та підтримувати актуальність інформації в базі даних. Використання окремого модуля для роботи з БД забезпечує відокремлення логіки збереження даних від основної логіки Discord-бота, що покращує структуру проєкту та спрощує його подальшу підтримку.

Файл config.py використовується для централізованого зберігання основних параметрів роботи системи автоматичної модерації. У ньому розміщуються конфігураційні змінні, які визначають поведінку Discord-бота та параметри взаємодії з іншими компонентами системи. Такий підхід дозволяє відокремити службові налаштування від основної програмної логіки та спрощує подальше адміністрування проєкту.

У конфігураційному файлі зберігаються ключові параметри системи, зокрема токен Discord-бота, порогові значення токсичності тексту та NSFW-контенту, назви моделей машинного навчання, ліміти порушень для застосування санкцій, а також шлях до файлу бази даних SQLite. Наявність окремого конфігураційного модуля дозволяє швидко змінювати поведінку системи без необхідності редагування основних модулів проєкту.

Особливо важливими є параметри порогових значень (threshold). Саме вони визначають, при якому рівні токсичності або NSFW-ризику повідомлення буде вважатися неприйнятним. Наприклад, зменшення порогу токсичності робить систему більш суворою, тоді як збільшення — зменшує кількість хибних спрацьовувань. Аналогічно налаштовується чутливість NSFW-модуля для аналізу зображень.

Окремо у файлі конфігурації задаються назви моделей, які

використовуються для аналізу тексту та зображень. Завдяки цьому систему можна адаптувати до інших моделей без зміни основного програмного коду. Подібний підхід підвищує гнучкість архітектури та спрощує подальше масштабування або модернізацію Discord-бота.

4.3 Функціонал застосунку

Після запуску бот підключається до Discord API та починає відстежувати повідомлення у текстових каналах сервера. Кожне нове повідомлення автоматично аналізується системою. Якщо повідомлення містить текст, або зображення, воно передається до модуля модерації, яки використовує трансформерні моделі та визначає рівень токсичності повідомлення. Після завершення аналізу бот приймає рішення щодо допустимості контенту. Якщо повідомлення або зображення визначається як токсичне чи NSFW, система автоматично видаляє його з каналу Discord. Одночасно інформація про порушення записується до бази даних SQLite, де зберігається історія поведінки користувачів. На рисунку 4.7 зображено повідомлення від бота про порушення правил. Вони автоматично відправляются в спеціалізований текстовий канал, для зручного доступу до логів бота. Завдяки цьому можна модератори серверу мають змогу швитко та зручно подивитись інформацію про порушення та міри, що прийняв бот.

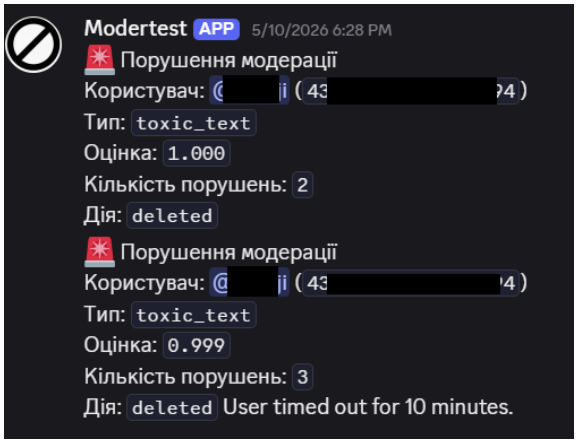


Рисунок 4.7 – Повідомлення бота про порушення

Також бот вітправляє приватне повідомлення для іфорармування ппорушника. Користувач серверу бачить причину порушення та дії бота. На рисунку 4.8 зображено приклад повідомлення.

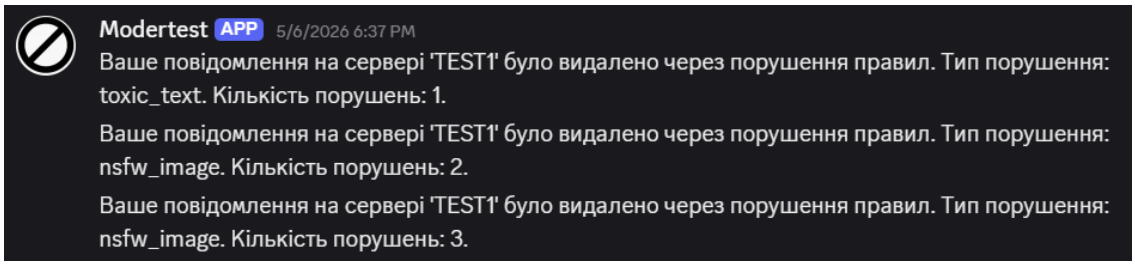


Рисунок 4.8 – Повідомлення учасника серверу про порушення

Також вбот дозволяє продивитись інфомацію про конкретного користувача. Було створенно сеціальні команди для бота, щоб користувачі серверу могли продивитись свою історію порушень, а модератори в разі необхідності швидко очистити історію порушень конкретної людини. На рисунках 4.9 та 4.10 зображено приклади використання команд бота для різних команд.

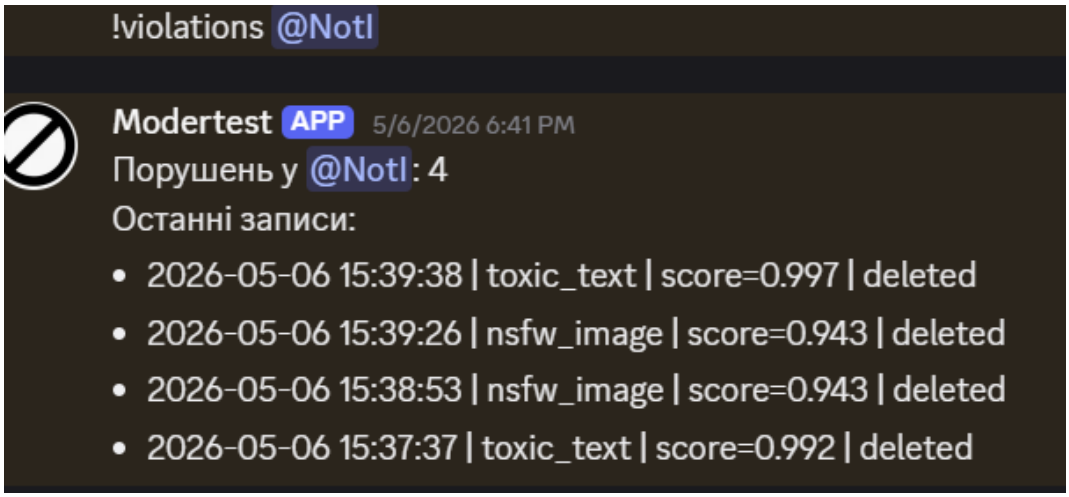


Рисунок 4.9 – Використання команди !violations @user

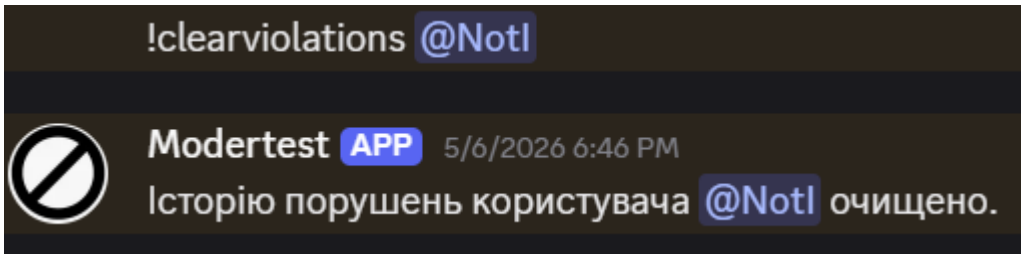


Рисунок 4.10 – Використання команиди !clearviolations @user

Таким чином, розроблений Discord-бот використовує трансформерні моделі для інтелектуального аналізу повідомлень користувачів. В разі виявлення порушення він автоматично приймає рішення, та інформує користувача, а також зберігає інформацію про інцедент. Завдяки командам модерація серверу може легко керуфати ботом та інформацією про порушення користувачів. Бот значно спрощює задачу модерації серверу для допомагає підтримувати порядок в текстових каналах, що позитивно впливає на комунікацію людей під час спілкування.

ВИСНОВКИ

У процесі виконання кваліфіеаційної роботи було проведено згідно завданню комплексне дослідження методів та моделей автоматичної модерації контенту в онлайн-середовищі, зокрема на платформі Discord. Актуальність дослідження зумовлена стрімким зростанням обсягів користувацького контенту та необхідністю забезпечення безпечного і комфортного середовища спілкування. Основна увага була зосереджена на вирішенні задач виявлення токсичних текстових повідомлень і NSFW-контенту у зображеннях із використанням сучасних підходів штучного інтелекту.

У ході роботи було виконано аналіз предметної галузі, який дозволив визначити ключові проблеми, пов’язані з поширенням агресивного та небажаного контенту в онлайн-комунікації. Дослідження показало, що традиційні підходи до модерації, засновані на ручній перевірці або простих правилах, не забезпечують достатньої ефективності в умовах масштабних цифрових платформ. Це обґрунтувало доцільність використання методів машинного навчання, глибинного навчання та трансформерних моделей.

Важливим етапом стало вивчення існуючих рішень для аналізу тексту та зображень, що дозволило визначити їхні переваги та обмеження. Було встановлено, що найбільш ефективними у задачах виявлення токсичності є моделі на основі трансформерної архітектури, які здатні враховувати контекст і виявляти складні форми агресії. У свою чергу, для аналізу зображень доцільним є використання методів комп’ютерного зору та мультимодальних моделей, які забезпечують більш глибоке розуміння візуального контенту.

У межах роботи було обґрунтовано вибір підходів до побудови системи автоматичної модерації та сформовано вимоги до її функціональності. Особливу увагу приділено поєднанню текстових і візуальних модулів аналізу, що дозволяє створити комплексне рішення для обробки різних типів контенту. Такий підхід забезпечує більш високий рівень точності та універсальності порівняно з одновимірними системами.

Результатом роботи стала розробка інтелектуальної системи у вигляді Discord-бота, здатного автоматично аналізувати повідомлення користувачів і зображення, виявляти токсичний або небажаний контент та реагувати відповідно до заданих правил. Запропонована архітектура передбачає модульну побудову, що дозволяє гнучко змінювати або вдосконалювати окремі компоненти без порушення цілісності системи.

Практична цінність отриманих результатів полягає у можливості використання розробленого підходу для автоматизації процесів модерації в онлайн-спільнотах. Це дозволяє суттєво знизити навантаження на адміністраторів, підвищити швидкість реагування на порушення та покращити загальну якість комунікації.

Таким чином, у роботі було досягнуто поставленої мети та закладено основу для створення ефективної системи автоматичної модерації контенту, що відповідає сучасним вимогам цифрового середовища.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. OpenAI. Moderation API Documentation. URL: https://platform.openai.com/docs/guides/moderation (дата звернення: 25.02.2026).

2. Brown T. et al. Language Models are Few-Shot Learners // NeurIPS. – 2020. URL: https://arxiv.org/abs/2005.14165 (дата звернення: 02.03.2026).

3. Raffel C. et al. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer // JMLR. – 2020. URL: https://arxiv.org/abs/1910.10683 (дата звернення: 17.10.2026).

4. Devlin J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // NAACL. – 2019. URL: https://arxiv.org/abs/1810.04805 (дата звернення: 04.04.2026).

5. Vaswani A. et al. Attention Is All You Need // NeurIPS. – 2017. URL: https://arxiv.org/abs/1706.03762 (дата звернення: 07.03.2026).

6. He K. et al. Deep Residual Learning for Image Recognition // CVPR. – 2016. URL: https://arxiv.org/abs/1512.03385 (дата звернення: 11.03.2026).

7. Dosovitskiy A. et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale // ICLR. – 2021. URL: https://arxiv.org/abs/2010.11929 (дата звернення: 17.03.2026).

8. Radford A. et al. Learning Transferable Visual Models From Natural Language Supervision (CLIP) // ICML. – 2021. URL: https://arxiv.org/abs/2103.00020 (дата звернення: 25.03.2026).

9. Рехінкін М.А., Ребезюк Л.М. Розробка інтелектуальної системи автоматичної модерації контенту на платформі DISCORD на основі трансформерних моделей // Матеріали 2-ї Міжнародної науково-практичної конференції «СУЧАСНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ MIT&AIS-2026». Харків – Яремче: Харківський національний університет радіоелектроніки, 2026. С.118-119.