НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих

комп'ютерних систем

До захисту допущено:

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Системне програмування та спеціалізовані комп'ютерні системи»
спеціальності 123 «Комп'ютерна інженерія»

на тему: «Система автоматичної модерації токсичного тексту на основі NLP**»**

Виконав (-ла):
студент (-ка) IV курсу, групи КВ-12
Камінський Тарас Петрович _____

Керівник:
Доцент кафедри СПСіКС, к.т.н,
Потапова Катерина Романівна _____

Консультант з назва нормконтролю:
Доцент кафедри СПСіКС, к.т.н,
Клятченко Ярослав Михайлович _____

Рецензент:
Доцент кафедри ПМ, к.ф.-м.н,
Вовк Лілія Борисівна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра системного програмування і
спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Системне програмування та спеціалізовані комп'ютерні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту
Камінський Тарас Петрович

Тема проєкту «Система автоматичної модерації токсичного тексту на основі NLP», керівник проєкту Потапова Катерина Романівна, к.т.н, доцент, затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом проєкту

3. Вихідні дані до проєкту

4. Зміст пояснювальної записки

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
нормконтроль	к.т.н., доцент Клятченко Я.М.		

* Якщо визначені консультанти. Консультантом не може бути зазначено керівника дипломного проєкту.

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проекту	04.11.2024	
2.	Розроблення та узгодження технічного завдання	07.12.2024	
3.	Аналіз існуючих рішень	14.01.2025	
4.	Вибір інструментів реалізації проєкту	27.01.2025	
5.	Розробка програмного продукту	15.02.2025	
6.	Тестування та відлагодження програмного продуту	29.02.2025	
7.	Завершення пояснювальної записки	14.04.2025	
8.	Підготовка графічної частини дипломного проєкту	04.05.2025	
9.	Оформлення документації дипломного проєкту	16.05.2025	
10	Попередній огляд матеріалів на кафедрі	30.05.2025	

Студент Тарас КАМІНСЬКИЙ

Керівник Катерина ПОТАПОВА

АНОТАЦІЯ

Кваліфікаційна робота містить пояснювальну записку обсягом 89 сторінок, яка включає 18 рисунки, 10 таблиць і 4 додатки. Об’єктом дослідження є система автоматизованої модерації токсичного текстового контенту на основі методів обробки природної мови (Natural Language Processing, NLP). Предметом розробки є алгоритмічне та програмне забезпечення для детекції та інтерпретації неприйнятних висловлювань у текстах англійською мовою.

У межах роботи реалізовано програмну систему, що виконує: покомпонентний аналіз тексту; класифікацію речень за рівнем токсичності з використанням моделі unitary/toxic-bert; ідентифікацію токсичних лексем із застосуванням word-level підходу; визначення типів токсичності (insult, hate speech, threat тощо); візуалізацію результатів у графічному інтерфейсі; генерацію рекомендацій щодо нейтрального переформулювання.

У ході реалізації проведено критичний аналіз існуючих методів виявлення токсичності, встановлено їхні функціональні обмеження, визначено вимоги до системи. Розроблено модуль sentence-level класифікації з багатомітковим виходом, інтегровано word-level механізм підсвічування небажаних фрагментів, сформовано алгоритм візуалізації результатів із використанням тултіпів. Застосунок реалізовано у вигляді десктопного GUI-додатку на основі бібліотеки Tkinter, з підтримкою автономної роботи та розширюваною архітектурою.

Ключові слова: ОБРОБКА ПРИРОДНОЇ МОВИ, ВИЯВЛЕННЯ ТОКСИЧНОСТІ, BERT, ВІЗУАЛІЗАЦІЯ, TKINTER, МОДЕРАЦІЯ, UNITARY/TOXIC-BERT, ГРАФІЧНИЙ ІНТЕРФЕЙС, LLM, ПЕРЕФОРМУЛЮВАННЯ.

SUMMARY

The qualification thesis comprises a 89-page explanatory report, including 18 figures, 10 tables, and 4 appendices. The object of study is an automated toxic text moderation system based on Natural Language Processing (NLP) techniques. The developed solution focuses on the algorithmic and software implementation of tools for detecting and interpreting toxic expressions in English-language texts.

The system supports: modular analysis of user input; sentence-level toxicity classification using the unitary/toxic-bert model; identification of toxic words via a word-level detection mechanism; categorization of toxicity types (INSULT, HATE SPEECH, THREAT, etc.); visualization of toxic spans within a graphical user interface; and automated generation of rephrasing suggestions for toxic sentences.

During development, existing toxicity detection methods were reviewed and their limitations analyzed. Functional and non-functional system requirements were formulated. A sentence-level classification component with multilabel output was implemented, along with a word-level highlighting module. A visualization algorithm with tooltip-based explanations was developed. The final product is a desktop GUI application based on the Tkinter library, featuring offline operation and a modular architecture suitable for further extension.

KEYWORDS: NATURAL LANGUAGE PROCESSING, TOXICITY DETECTION, BERT, VISUALIZATION, TKINTER, MODERATION, UNITARY/TOXIC-BERT, GUI, LLM, REPHRASING.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
1	А4	ІАЛЦ.045430.002 ТЗ	Система автоматичної модерації токсичного тексту на основі NLP Технічне завдання	5		
2	А4	ІАЛЦ.045430.003 ТП	Система автоматичної модерації токсичного тексту на основі NLP Відомість технічного проєкту	2		
3	А4	ІАЛЦ.045430.004 ПЗ	Система автоматичної модерації токсичного тексту на основі NLP Пояснювальна записка	89		
4	А4	ІАЛЦ.045430.005 Д1	Діаграма компонентів застосунку	1		
5	А4	ІАЛЦ.045430.006 Д2	Діаграма класів	1		
6	А4	ІАЛЦ.045430.007 Д3	Блок-схема алгоритму визначення токсичних слів	1		

Змін.	Арк.	№ докум.	Підпис	Дата	**ІАЛЦ.045430.001 ОА**		
Розробив		Камінський Т. П..			Система автоматичної модерації токсичного тексту на основі NLP ***Опис альбому***	Літ.	Лист 1 / Листів 2
Перевірив		Потапова К.Р.					
Консульт.						**КПІ ім. І. Сікорського, ФПМ, КВ-12**	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

			НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
7	А4	ІАЛЦ.045430.008 Д4	Блок-схема алгоритму sentence-level класифікації	1		

					ІАЛЦ.045430.001 ОА	***Лист***
						2
Зм	***Лист***	***№ докум.***	***Підп.***	***Дата***		

ЗМІСТ

1.НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ 2

2.ПІДСТАВА ДЛЯ РОЗРОБКИ 2

3.МЕТА І ПРИЗНАЧЕННЯ РОБОТИ 2

4.ДЖЕРЕЛА РОЗРОБКИ 3

5.ТЕХНІЧНІ ВИМОГИ 3

5.1 Вимоги до продукту, що розробляється 3

5.2 Вимоги до апаратного забезпечення 4

5.3 Вимоги до програмного та апаратного забезпечення користувача 4

6.ЕТАПИ РОЗРОБКИ 5

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки:

«Система автоматичної модерації токсичного тексту на основі NLP»

Галузь застосування:

Автоматична модерація контенту, соціальні мережі, обробка природної мови, виявлення та аналіз неприйнятних висловлювань, інтелектуальні фільтри для текстів, підтримка безпечної комунікації онлайн.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою проєкту є розробка програмного забезпечення для виявлення неприйнятних висловлювань у тексті з можливістю візуального виділення проблемних фрагментів, класифікації типу токсичності та подальшої генерації рекомендацій або переформулювань на основі моделей обробки природної мови.

4. ДЖЕРЕЛА РОЗРОБКИ

У процесі роботи використано наукові публікації з питань NLP, документацію до моделей unitary/toxic-bert, rubert-base-toxic-words, transformers, бібліотеки HuggingFace, а також матеріали з відкритих онлайн-джерел, які висвітлюють проблематику токсичності в інтернет-комунікаціях.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до продукту, що розробляється

- Сумісність з ОС Windows 10 / 11 (64-bit)
- Обробка англомовного тексту
- Класифікація токсичності на рівні речення (7 типів)
- Виявлення агресивної лексики із підсвічуванням
- Візуалізація результатів у графічному інтерфейсі (Tkinter)
- Підтримка інтерактивних підказок (tooltip)
- Можливість переформулювання речень через інтеграцію з LLM (опційно)

5.2 Вимоги до апаратного забезпечення

- Оперативна пам’ять: не менше 4 ГБ
- Процесор: двоядерний з частотою не менше 2 ГГц
- Вільне місце на диску: 1 ГБ
- Підключення до Інтернету — обов’язкове (для доступу до моделей)

5.3 Вимоги до програмного та апаратного забезпечення користувача

- ОС: Windows 10/11 (64-bit)
- Python 3.10+
- Установлені бібліотеки: transformers, torch, tkinter, pandas, scikit-learn
- Інтернет-з'єднання для завантаження моделей

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту
1.	Вивчення літератури за тематикою проекту	04.11.2024
2.	Розроблення та узгодження технічного завдання	22.11.2024
3.	Аналіз існуючих рішень	14.01.2025
4.	Вибір інструментів реалізації проєкту	27.01.2025
5.	Розробка програмного продукту	15.02.2025
6.	Тестування та відлагодження програмного продуту	29.02.2025
7.	Написання першого розділу пояснювальної записки	06.03.2025
8.	Написання другого розділу пояснювальної записки	13.03.2025
9.	Написання третього розділу пояснювальної записки	20.03.2025
10.	Написання четвертого розділу пояснювальної записки	27.03.2025
11.	Завершення пояснювальної записки	14.04.2025
12.	Підготовка графічної частини дипломного проєкту	04.05.2025
13.	Оформлення документації дипломного проєкту	16.05.2025
14	Попередній огляд матеріалів на кафедрі	30.05.2025

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
1	А4	ІАЛЦ.045430.004 ПЗ	Система автоматичної модерації токсичного тексту на основі NLP Пояснювальна записка	89		
2	А4	ІАЛЦ.045430.005 Д1	Діаграма компонентів застосунку	1		
3	А4	ІАЛЦ.045430.006 Д2	Діаграма класів	1		
4	А4	ІАЛЦ.045430.007 Д3	Блок-схема алгоритму визначення токсичних слів	1		

Зм.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045430.003 ТП		
Розробив		Камінський Т. П.			Система автоматичної модерації токсичного тексту на основі NLP ***Відомість технічного проєкту***	Літ.	Лист 1 / Листів 2
Перевірив		Потапова К. Р.				КПІ ім. Ігоря Сікорського, ФПМ, КВ-12	
Консульт.							
Н. контр.		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
5	А4	ІАЛЦ.045430.008 Д4	Блок-схема алгоритму sentence-level класифікації	1		
6		ІАЛЦ.045430.003 ТП	Відомість технічного проєкту	2		
7			Диск CD-ROM: текст пояснювальної записки			
8			Текст анотації			
9			Архівований код програми у форматі .zip			
10			Графічний матеріал			

Пояснювальна записка
до дипломного проєкту

на тему: «Система автоматичної модерації токсичного тексту на основі NLP»

Київ – 2025 рік

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ВСТУП

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Проблема токсичності в текстовому спілкуванні

1.2 Огляд існуючих підходів до модерації

1.3 Обмеження класичних методів

1.4 Постановка задачі проєкту

1.5 Визначення функціональних та нефункціональних вимог

ВИСНОВКИ ДО РОЗДІЛУ 1

РОЗДІЛ 2. ТЕОРЕТИЧНІ ТА ТЕХНІЧНІ ОСНОВИ ПОБУДОВИ СИСТЕМИ МОДЕРАЦІЇ ТОКСИЧНОГО ТЕКСТУ

2.1 Обробка природної мови (NLP): загальні поняття та еволюція

2.2 Підходи до класифікації текстів

2.3 Формалізація задачі класифікації токсичності

2.4 Інструменти реалізації проєкту

2.5 Порівняння існуючих моделей для виявлення токсичності

ВИСНОВОК ДО РОЗДІЛУ 2

РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ АВТОМАТИЧНОЇ МОДЕРАЦІЇ ТОКСИЧНОГО ТЕКСТУ

3.1 Загальна архітектура системи

3.2 Sentence-level аналіз токсичності

3.3 Локалізація небажаних фрагментів у тексті

3.4 Аналіз моделі rubert-base-toxic-words і причини відмови від використання

3.5 Графічний інтерфейс користувача

3.6 Модульна структура програми

3.7 Реалізація аналізу токсичності

3.8 Перспективи розвитку системи

ВИСНОВОК ДО РОЗДІЛУ 3

РОЗДІЛ 4. ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ СИСТЕМИ

4.1 Методика тестування

4.2 Аналіз точності класифікації

4.3 Тестування інтерфейсу користувача

4.4 Оцінка підсвічування небажаних фрагментів

4.5 Порівняння з іншими системами

ВИСНОВОК ДО РОЗДІЛУ 4

ВИСНОВКИ

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

ДОДАТКИ

Додаток 1. Копії графічних матеріалів

- ІАЛЦ.045430.005 Д1**.** Діаграма компонентів застосунку.
- ІАЛЦ.045430.006 Д2. Діаграма класів.
- ІАЛЦ.045430.007 Д3. Блок-схема алгоритму sentence-level класифікації.
- ІАЛЦ.045430.008 Д4. Блок-схема алгоритму word-level маркування агресивної лексики.
- ІАЛЦ.045430.009 Д5. Фрагменти програмного коду.

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

AI — Artificial Intelligence — штучний інтелект, галузь комп'ютерних наук, яка вивчає створення систем, здатних до інтелектуальної поведінки.

API — Application Programming Interface — інтерфейс прикладного програмування; набір підпрограм для взаємодії між програмними компонентами.

BERT — Bidirectional Encoder Representations from Transformers — двонаправлене кодування на основі трансформера; архітектура нейромережі для обробки природної мови.

BoW — Bag of Words — модель представлення тексту як неупорядкованої множини слів без урахування граматики та порядку.

CBOW — Continuous Bag of Words — варіант Word2Vec, де модель передбачає слово за контекстом.

CPU — Central Processing Unit — центральний процесор; виконує обчислення в комп'ютері.

F1-score — гармонійне середнє між точністю (precision) і повнотою (recall); використовується для оцінки якості класифікації.

GUI — Graphical User Interface — графічний інтерфейс користувача.

HF — Hugging Face — платформа з відкритим кодом для роботи з NLP-моделями, зокрема трансформерами.

ML — Machine Learning — машинне навчання; підгалузь AI, яка досліджує методи автоматичного навчання моделей із даних.

MLM — Masked Language Modeling — метод передтренування моделей, який полягає у передбаченні замаскованих токенів у тексті.

NLP — Natural Language Processing — обробка природної мови; галузь AI, що займається взаємодією комп’ютерів і людської мови.

NSP — Next Sentence Prediction — завдання передтренування BERT: визначення, чи є друге речення логічним продовженням першого.

RNN — Recurrent Neural Network — рекурентна нейронна мережа; тип нейромережі, що опрацьовує послідовності даних.

TF-IDF — Term Frequency – Inverse Document Frequency — статистична міра важливості терміна в документі відносно корпусу.

Tkinter — стандартна бібліотека Python для створення GUI-додатків.

Tooltip — елемент інтерфейсу, який відображає пояснення при наведенні курсора.

Transformer — архітектура нейронної мережі, що використовує механізм самоуваги для обробки послідовностей.

Unitary/toxic-bert — модель на основі BERT для класифікації токсичності, навчена на датасеті Jigsaw.

Word Embeddings — векторні подання слів у числовому просторі, які відображають їхнє семантичне значення.

Word2Vec — метод для побудови векторних представлень слів, що зберігає семантичні зв’язки.

ВСТУП

Інтенсивне зростання обсягів текстової комунікації в цифровому середовищі — зокрема в соціальних мережах, форумах та багатокористувацьких онлайн-платформах — зумовило виникнення нових викликів у сфері автоматизованого контролю якості контенту. Однією з найактуальніших проблем сучасного інформаційного простору є поширення неприйнятних висловлювань, що можуть містити погрози, образи, мову ворожнечі або вульгарну лексику [1, с.11]; [3, с. 2]. Така комунікація не лише порушує етичні норми спілкування, але й потенційно становить психологічну небезпеку для користувачів, спричиняючи соціальну напругу, провокуючи конфлікти та шкодячи репутації онлайн-сервісів [2].

Традиційні методи модерації, засновані на словникових фільтрах або простих класифікаторах, виявилися обмеженими в контексті гнучкості та адаптивності до нових форм мови. Ці підходи, зокрема не здатні враховувати мовні трансформації, сарказм або завуальовані прояви токсичності, що значно ускладнює виявлення небажаного контенту в реальних умовах[2, 13].

У зв’язку з цим особливу актуальність набувають інтелектуальні системи, побудовані на основі моделей глибокого навчання, що здатні моделювати контекст, аналізувати семантику повідомлень і забезпечувати високий рівень точності класифікації [4, с. 3]. Використання трансформерних архітектур, таких як BERT [4], дає змогу здійснювати не лише sentence-level класифікацію, а й локалізацію небажаних фрагментів на рівні окремих слів, що значно підвищує пояснюваність результатів.

Метою даної роботи є створення програмної системи для автоматизованої модерації токсичного тексту англійською мовою, що поєднує

sentence-level і word-level аналіз із візуалізацією результатів у зручному графічному інтерфейсі. Реалізація системи здійснюється з дотриманням принципів модульності, автономності й адаптивності, що забезпечує її ефективність у широкому спектрі застосувань — від індивідуального використання до інтеграції в корпоративні системи контент-контролю [6:15].

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Проблема токсичності в текстовому спілкуванні

У сучасному цифровому світі значна частина комунікації здійснюється у текстовій формі — через соціальні мережі, форуми, коментарі до відео, чати в багатокористувацьких іграх тощо. У процесі дослідження було встановлено, що серед основних загроз для безпечного та здорового кіберпростору однією з найсуттєвіших є токсичність мовлення — образливі, агресивні, принизливі або дискримінаційні висловлювання, спрямовані на окремих осіб або групи осіб.

Токсичність у текстах може проявлятися в різних формах:

- прямі образи (insult),
- загрози (threat),
- мова ворожнечі (identity attack, hate speech),
- агресивна лайка (obscene, explicit),
- маніпулятивне чи пасивно-агресивне висміювання.

Проаналізувавши відкриті датасети (наприклад, Jigsaw Toxic Comment Classification Challenge) оцінив частку токсичних коментарів на деяких платформах (Рисунок 1.1):

- Reddit — 16 %
- Twitter — 18 %
- YouTube — 12 %
- Facebook — 9 %
- Twitch — 7 %

Джерело: [1, с. 11].

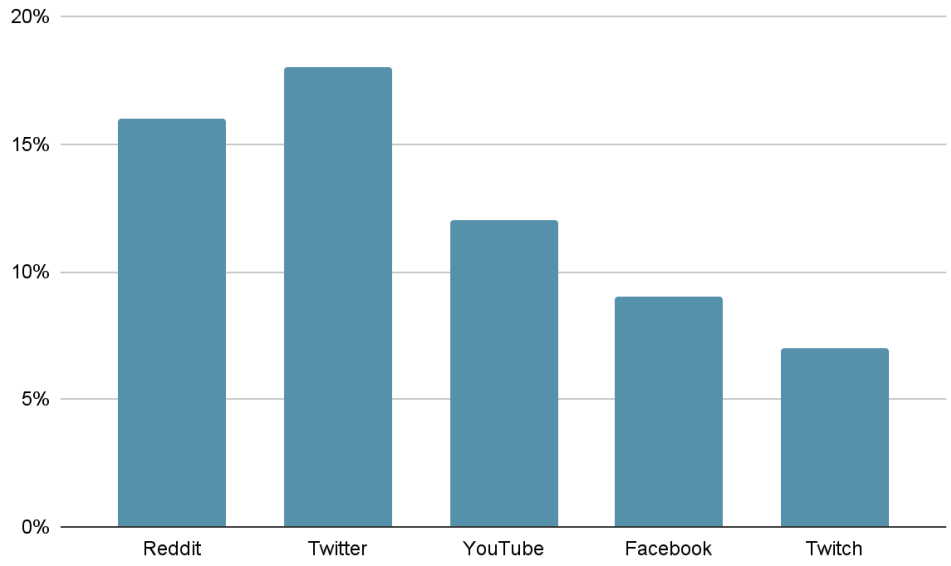

Рисунок 1.1 – Частка токсичних коментарів на різних платформах

Отримані статистичні дані свідчать про те, що понад кожен десятий коментар на провідних веб-ресурсах може містити ознаки токсичності та мати реальну загрозу для користувачів, передусім для молоді.

Установлено, що токсичні повідомлення не лише знижують рівень цифрової комунікації, але й можуть спричиняти психологічні наслідки, провокувати конфлікти та навіть стати причиною юридичних суперечок.

1.2 Огляд існуючих підходів до модерації

Існуючі системи модерації токсичного контенту можна умовно поділити на кілька груп:

1.2.1 Ручна модерація

Цей підхід передбачає наявність людини-модератора, яка читає та оцінює повідомлення.

Переваги:

- висока точність;
- врахування контексту.

Недоліки:

- затратність у часі;
- низька масштабованість;
- людський фактор.

1.2.2. Чорні списки слів

Це автоматичні фільтри, що блокують повідомлення за списком заборонених слів і виразів, які є небажаними.

Переваги:

- проста реалізація,
- миттєва реакція.

Недоліки:

- легко обходяться через помилки, синоніми, варіації написання;
- не враховується контекст.

Приклади блокування повідомлень цим методом в таблиці 1.1

Таблиця 1.1

Блокування повідомлень методом чорних списків слів

Приклад	Повідомлення	Блоковано?
1	You are stupid	Так
2	y0u ar3 stu.pid	Ні (не відслідковується)
3	I love stupid movies	Ні (помилкове спрацювання можливо)

1.2.3. Класичні ML-моделі

Використання моделей, натренованих на словоформах, n-граммах, фічах, що подаються на логістичну регресію, SVM тощо.

Переваги:

- швидкість;
- краще врахування лексичного контексту.

Недоліки:

- слабка узагальненість;
- труднощі з рідкісними словами та неявними проявами агресії.

1.2.4. API-комплекси з NLP-моделями

Два з найбільш визнаних рішень у сфері автоматичної модерації токсичного контенту — Perspective API від Google Jigsaw і Detoxify від Unitary AI — були зіставлені для порівняння.

Perspective API — це пропрієтарний закритий продукт, що надає REST API-інтерфейс для обробки англомовного тексту. Система зіставляє кожне повідомлення з різними метриками токсичності (наприклад, toxic, insult, identity attack) і надає результат у форматі JSON з числовими оцінками для кожної категорії [5].

Detoxify є відкритою моделлю на основі архітектури BERT. Вона доступна на платформі Hugging Face як попередньо навчена модель PyTorch і забезпечує багатоміткову класифікацію [6]. Detoxify повертає результати у вигляді ймовірнісних тензорів і словникових міток для кожної категорії токсичності.

Порівняльну характеристику наведено у таблиці 1.2.

Таблиця 1.2

Порівняння сервісів Perspective API та Detoxify

Сервіс / Модель	Доступність	Тип моделі	Формат доступу	Формат виводу
Perspective API	Пропрієтарна	Класифікатор	REST API	JSON, score per type
Detoxify	Open-source	BERT-класифікатор	Hugging Face Model	tensors, labels (PyTorch)

Висновок:

Найбільш ефективними виявилися гібридні системи, що базуються на глибоких мовних моделях (BERT, RoBERTa), які дозволяють виявляти не лише явну, але й контекстну токсичність. Саме на такому підході ґрунтується розробка цієї дипломної роботи.

1.3 Обмеження класичних методів

Попри популярність наївних фільтрів і традиційних підходів машинного навчання, ці методи мають низку істотних обмежень, які суттєво знижують їхню ефективність у реалістичних умовах.

1.3.1. Нерозуміння контексту

Чорні списки і прості класифікатори не розуміють контексту слів. Наприклад:

- "That’s sick!" — може бути позитивним або негативним, залежно від ситуації.
- "You are so smart, unlike your friends." — сарказм, який важко виявити без глибокого аналізу.

1.3.2 Обхід фільтрів.

Користувачі легко обходять обмеження шляхом:

- викривлення написання (b!tch, sh1t, n00b);
- вставки символів (f u c k);
- використання синонімів або іноземних слів.

Такі варіації не розпізнаються традиційними регулярними виразами або жорсткими списками.

1.3.3 Висока кількість false positives / false negatives.

Приклади таких випадків надано в таблиці 1.3.

Таблиця 1.3

Приклади хибних спрацювань

Повідомлення	Реальна токсичність	Класичний фільтр
"I hate how you lie all the time"	Токсичне	Не відфільтровано
"Dumbbell workout"	Нетоксичне	Хибне спрацювання

1.3.4 Мовна та стилістична обмеженість

Більшість класичних рішень працюють лише для:

- англійської мови, через обмеженість навчальних даних;
- коротких повідомлень, бо довгі фрази ускладнюють аналіз.

1.3.5 Відсутність адаптації до нових типів токсичності

Мовні форми та сленг постійно змінюються, що робить класичні механізми виявлення токсичності застарілими через їхню нездатність до навчання або динамічної адаптації.

Сучасна інтернет-токсичність також стає дедалі більш адаптивною, завуальованою та динамічною. Традиційні методи не здатні ефективно виявляти такі прояви, тому виникає потреба у застосуванні глибоких нейронних мереж, здатних моделювати контекст, враховувати структуру тексту та обробляти семантичні подання слів [4].

1.4 Постановка задачі проєкту

На основі аналізу проблеми токсичного контенту та обмежень існуючих підходів сформульовано задачу розроблення сучасної й ефективної системи, яка не лише визначає рівень токсичності, а й забезпечує повну візуалізацію її проявів на рівні окремих слів і речень.

Мета проєкту

Розробити інтерактивну програмну систему для автоматичної модерації англомовного токсичного тексту з використанням моделей обробки природної мови (NLP), яка:

- класифікує кожне речення за рівнем токсичності;
- виявляє конкретні токсичні слова всередині речень;
- демонструє результати у зручному графічному інтерфейсі;
- дозволяє користувачу бачити типи токсичності через підказки;
- забезпечує адаптивність і можливість доповнення новими моделями.

Основні задачі, які вирішує система:

1. Sentence-level класифікація — виявлення токсичних речень (та типів токсичності: threat, insult, obscene тощо) [6].
2. Word-level детекція — підсвічування агресивної лексики за допомогою sequence tagging моделей [10].
3. Комбінування результатів — поєднання sentence та word-level прогнозів для кращої точності.
4. Візуалізація — створення зручного GUI з підсвіткою, підказками, навігацією по тексту.
5. Продуктивність і масштабованість — ефективне оброблення великих текстів без затримок.
6. Можливість модифікації — простота підключення нових моделей або словників.

Постановка задачі формулюється як бінарна багатоміткова класифікація на рівні речень у поєднанні з розміткою послідовностей на рівні слів. Її реалізацію забезпечують попередньо натреновані трансформерні моделі [4; 6].

1.5 Визначення функціональних та нефункціональних вимог

З метою забезпечення ефективної роботи системи автоматичної модерації токсичного контенту з використанням методів обробки природної мови необхідно чітко сформулювати вимоги — від основної функціональності до технічних і експлуатаційних обмежень. Усі вимоги поділяються на функціональні та нефункціональні.

1.5.1 Функціональні вимоги

1. Система повинна забезпечувати виконання таких функцій:
2. Завантаження англомовного тексту у форматі *.txt або через текстове поле введення.
3. Автоматичне розбиття вхідного тексту на окремі речення.
4. Класифікація кожного речення за сімома типами токсичності (toxic, severe toxic, obscene, threat, insult, identity hate, non-toxic) за допомогою моделі unitary/toxic-bert [6].
5. Виявлення окремих агресивної лексики у межах речень за допомогою токенізованої word-level моделі (rubert-base-toxic-words) [10].
6. Візуальне відображення результатів у вигляді:

– підсвічених жирним шрифтом агресивної лексики;

– інтерактивних іконок (❓) над токсичними реченнями з поясненням типу токсичності.

7. Побудова графічного інтерфейсу користувача з можливістю:

– скролінгу;

– фільтрації речень за рівнем токсичності;

– відображення додаткової інформації за наведенням курсору.

8. Підготовка текстового звіту з результатами класифікації (експорт за бажанням користувача).

9. Можливість локального запуску без інтернет-з’єднання (моделі працюють офлайн).

1.5.2 Нефункціональні вимоги

Окрім основного призначення, система повинна відповідати низці експлуатаційних, технічних та програмних обмежень, що забезпечують стабільну роботу та зручність для кінцевого користувача. Основні нефункціональні вимоги включають:

- Сумісність: підтримка операційних систем Windows 10 / 11 (64-bit), що забезпечує функціональність на більшості сучасних персональних комп’ютерів.
- Системні вимоги: мінімум 4 ГБ оперативної пам’яті, двоядерний процесор із тактовою частотою щонайменше 2 ГГц та принаймні 500 МБ вільного місця на диску.
- Продуктивність: повна обробка до 100 речень має здійснюватися не довше ніж за 5 секунд на середньостатистичному ПК користувача.

- Стабільність інтерфейсу: система повинна працювати без збоїв за високого навантаження, зокрема під час швидкої зміни вкладок, прокручування та виклику підказок.
- Використання відкритого програмного забезпечення: мають застосовуватись виключно бібліотеки Python з відкритим кодом (наприклад, Transformers [8], PyTorch [12], Tkinter [15]), що забезпечує прозорість і можливість відтворення результатів.
- Гнучкість архітектури: система має бути спроєктована таким чином, щоб додавання або заміна моделей виявлення токсичності здійснювалися без суттєвих змін у коді.
- Адаптивність інтерфейсу: відображення вмісту повинно бути коректним на екранах з роздільною здатністю не нижче 1366×768 пікселів.
- Автономна робота: система має підтримувати повноцінне функціонування без доступу до Інтернету, що є критично важливим для збереження конфіденційності вхідної інформації.

У сукупності ці умови забезпечують створення стабільної, ефективної та зручної у використанні системи, яка відповідає сучасним вимогам обробки природної мови та може бути розширена в майбутньому.

ВИСНОВКИ ДО РОЗДІЛУ 1

Перший розділ містить загальний вступ до галузі автоматичної модерації тексту, зокрема до проблеми токсичності в онлайн-взаємодії. Розглянуто як соціальні, так і технічні аспекти цього явища, а також охоплено його масштаб — від приватної комунікації до великих соціальних платформ. Встановлено, що шкідливі повідомлення можуть становити до 18% користувацького контенту на окремих сайтах, що зумовлює потребу в ефективних автоматизованих системах виявлення мови ворожнечі.

Проаналізовано сучасні підходи до модерації, зокрема ручну оцінку, фільтрацію за словниками, класичні моделі машинного навчання та сучасні нейромережі на основі трансформерів. Визначено, що лише глибокі нейронні моделі, а особливо архітектури, засновані на BERT, потенційно здатні враховувати контекст, структуру висловлювань і приховану токсичність.

Сформульовано технічну постановку задачі: розробка програмної системи, що поєднує класифікацію на рівні речень і виявлення токсичності на рівні слів з графічним інтерфейсом для простого й наочного відображення результатів.

Описано функціональні (класифікація речень, підсвічування слів, інтерактивні підказки, збереження результатів) і нефункціональні (висока швидкодія, гнучкий інтерфейс, автономність, масштабованість системи) вимоги.

Таким чином, у цьому розділі закладено теоретичну основу розробки системи та обґрунтовано необхідність використання сучасних NLP-рішень на основі трансформерних моделей у сфері автоматизованої модерації токсичного контенту.

РОЗДІЛ 2. ТЕОРЕТИЧНІ ТА ТЕХНІЧНІ ОСНОВИ ПОБУДОВИ СИСТЕМИ МОДЕРАЦІЇ ТОКСИЧНОГО ТЕКСТУ

2.1 Обробка природної мови (NLP): загальні поняття та еволюція

Обробка природної мови (Natural Language Processing, NLP) — це один із напрямів штучного інтелекту, що ґрунтується на знаннях у сфері лінгвістики, інформатики та машинного навчання і дозволяє комп'ютерам автоматично аналізувати, розуміти та генерувати людську мову.

Упродовж останніх десятиліть методи NLP пройшли значний шлях — від простого синтаксичного зіставлення шаблонів до глибокого розуміння змісту, що враховує семантику, контекст і прагматику. На початкових етапах переважали rule-based системи, які ґрунтувалися на лінгвістичних правилах, граматиках і словниках. Такі системи були чутливими до мовної варіативності, не масштабувалися і потребували ручної розробки правил [2].

З появою методів машинного навчання на перший план вийшли статистичні підходи, зокрема моделі, що ґрунтуються на частотних характеристиках слів (наприклад, bag-of-words, TF-IDF), які стали основою для перших систем автоматичної класифікації тексту. Попри те, що ці моделі не враховували порядок слів і контекст, вони стали значним кроком уперед на шляху до сучасного NLP.

Наступним етапом інновацій стало впровадження векторних подань слів (word embeddings), таких як Word2Vec і GloVe [4, с.4], які проектували слова у багатовимірний семантичний простір. Такі моделі виявляли подібність між словами на основі значення, але при цьому кожне слово асоціювалося з єдиним вектором незалежно від контексту.

Поява трансформерних моделей (Transformer, BERT, GPT) у 2017 році стала якісним проривом у розвитку NLP [3, с. 6000]. Завдяки механізму attention ці моделі здатні моделювати довготривалі залежності, враховувати контекст слів у межах речення і демонструють високу ефективність у задачах класифікації, генерації тексту та розпізнавання іменованих сутностей.

У таблиці 2.1 подано узагальнене порівняння основних етапів розвитку підходів в NLP.

Таблиця 2.1

Етапи розвитку підходів в NLP

Покоління	Приклади	Характеристика
Rule-based	ELIZA, словники	Граматичні правила, словники, шаблонні відповідності
Statistical	TF-IDF, Naive Bayes	Частотні ознаки, стохастичні класифікатори
Embeddings	Word2Vec, GloVe	Контекстно-незалежні вектори для представлення слів
Transformers	BERT, GPT	Контекстно-залежне представлення, attention-механізми

Отже, трансформерні моделі є найновішим і найбільш успішним підходом до проблеми автоматичної модерації тексту, особливо в тих випадках, коли необхідно враховувати контекст, неоднозначність і приховану агресію. У зв’язку з цим у межах даної дипломної роботи основна увага приділяється

саме трансформерним моделям, зокрема архітектурам на базі BERT, адаптованим до задач класифікації токсичності [4, с. 2].

2.2 Підходи до класифікації текстів

Одним із базових завдань обробки природної мови є класифікація тексту — автоматичне віднесення текстових повідомлень до наперед визначених категорій. У межах цього дослідження розглядається класифікація текстів за рівнем токсичності, тобто визначення того, чи містить повідомлення образливу лексику, погрози, дискримінаційні висловлювання та інші подібні елементи.

Залежно від рівня абстракції та застосованого технологічного підходу, методи класифікації текстів еволюціонували від простих технік фільтрації ключових слів до складних нейромережевих моделей, здатних обробляти граматичну структуру та витягувати семантичний контекст.

У цьому підрозділі розглянемо базові групи підходів:

- класичні методи (rule-based, статистичні) [2; 12];
- моделі на основі embeddings [4];
- трансформерні архітектури як сучасний стандарт [3; 4].

2.2.1 Класичні підходи до класифікації тексту

Перші експерименти з автоматичною класифікацією тексту базувалися на класичних статистичних моделях, де текст попередньо перетворювався на набір числових ознак — найчастіше за допомогою методів *bag-of-words* (BoW) або *TF-IDF* (term frequency – inverse document frequency) [12, с. 2827].

Bag-of-Words (мішок слів)

Одним із найпростіших методів представлення тексту є модель *BoW*: документ або речення розглядається як неупорядкований набір слів — і нічого більше. Такий підхід дозволяє створювати компактні частотні вектори, але повністю ігнорує граматичну структуру та семантичне значення.

TF-IDF

Метод *TF-IDF* виділяє слова, які часто зустрічаються в певному документі, але рідко трапляються в загальному корпусі. Це знижує вплив поширених «порожніх» слів і підвищує вагу характерних термінів. Результат розраховується за наступною формулою:

$$TF - IDF(t, d) = TF(t, d) \cdot log\frac{N}{DF(t)} \quad (1)$$

де:

- *TF(t, d)* — частота терміна *t* у документі *d*,
- *DF(t)* — кількість документів, у яких зустрічається термін *t*,
- *N* — загальна кількість документів.

Метод *TF-IDF* зменшує вплив загальновживаних слів (наприклад, "the", "and", "you") та підвищує значущість слів, характерних для конкретного документа. Цей підхід акцентує увагу на інформативній лексиці та знижує вагу слів із низьким семантичним навантаженням.

Обмеження класичних підходів

Незважаючи на простоту та швидкодію, ці методи мають істотні недоліки:

- Не враховують порядок слів, а отже, не можуть розпізнати інверсії або сарказм.
- Вразливі до обхідних форм токсичності (наприклад, заміна "idiot" на "1d10t").
- Потребують ручної побудови словників або словникових міток [13, с. 358].
- Погано масштабуються при розширенні кількості категорій або контекстів.

Узагальнення

Класичні методи, такі як BoW і TF-IDF, відіграли важливу роль у розвитку обробки природної мови. Проте їхня нездатність працювати з контекстом і обмежена гнучкість стали суттєвими недоліками в задачах, зокрема виявлення токсичності. Такі моделі не здатні виявляти приховану ворожість, стилістичні відтінки чи контекстуальну неоднозначність. Саме тому в межах даного проєкту вони розглядаються лише як відправна точка для переходу до складніших нейронних архітектур.

2.2.2 Моделі на основі word embeddings

Одним із найважливіших етапів в історії обробки текстів стало впровадження word embeddings — векторних представлень слів, які моделюють семантичну подібність на основі контексту вживання, а не частоти. Такі моделі відкрили нові можливості для інтерпретації значень слів і фраз у широкому спектрі мовних ситуацій.

Основна ідея embeddings

На відміну від bag-of-words або TF-IDF, де кожне слово виступає як окрема ознака, embeddings відображають слова у вигляді векторів у багатовимірному просторі, де розташування вказує на подібність значень. Це дає змогу системі «розуміти», що слова на кшталт love і like мають близьке значення, тоді як love і hate — протилежні за змістом.

Word2Vec: Skip-gram та CBOW

Однією з перших успішних моделей стала Word2Vec [4, с. 4], яка реалізує два варіанти:

- Skip-gram — передбачає контекст за поточним словом;
- CBOW (Continuous Bag of Words) — передбачає слово за контекстом.

Математично модель Skip-gram формулюється як задача максимізації ймовірності появлення слів з контексту *c*, заданого цільовим словом ω_t:

$$max \prod_{-k \le j \le k,\, j \ne 0} P(\omega_{t+j} | \omega_t) \quad (2)$$

де k — розмір вікна контексту.

Ця ймовірність моделюється як softmax-функція від скалярного добутку векторів:

$$P(\omega_0 | \omega_i) = \frac{exp(\upsilon_{\omega_0}^{\top} \cdot \upsilon_{\omega_i})}{\sum_{\omega \in V} exp(\upsilon_{\omega_0}^{\top} \cdot \upsilon_{\omega_i})} \quad (3)$$

де:

- υ_{ω_i} — вектор цільового слова,
- υ_{ω_0} — вектор контекстного слова,
- V — розмір словника

Глобальні моделі: GloVe

GloVe (Global Vectors for Word Representation) [4] використовує глобальну статистику співзустрічей слів, формуючи матрицю X_{ij}, де кожен елемент — кількість спільних появ слова *i* і слова *j* в контексті.

Модель навчає вектори так, щоб логарифм співзустрічей лінійно відповідав скалярному добутку:

$$\omega_i^{\top} \cdot \tilde{\omega}_j + b_i + \tilde{b}_j = log(X_{ij}) \quad (4)$$

Переваги embeddings:

- Підтримують семантичну близькість між словами;
- Зменшують розмірність простору ознак (напр., 300 замість тисяч слів);
- Генералізують знання навіть для незнайомих комбінацій;
- Дають змогу використовувати кластеризацію, візуалізацію та інші математичні методи.

2.2.3 Трансформерні архітектури та модель BERT

Модель Transformer була представлена у 2017 році [3, с. 5999] й стала проривом у галузі обробки природної мови. На відміну від рекурентних нейронних мереж (RNN, LSTM), які опрацьовують слова послідовно, трансформери ґрунтуються на механізмі самоуваги (self-attention), що дозволяє аналізувати всі слова одночасно й враховувати зв’язки між ними незалежно від відстані в реченні.

Основи архітектури Transformer

Ключовим елементом архітектури Transformer є механізм багатоголової самоуваги (multi-head self-attention), який дозволяє моделі визначати, які слова в реченні є найбільш пов'язаними між собою. Це забезпечує ефективне моделювання як близьких, так і далеких залежностей у тексті.

Математично attention визначається як:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (5)$$

де:

- Q — матриця запитів (*queries*),
- K — ключів (*keys*),
- V — значень (*values*),
- d_k — розмірність простору ключів (для нормалізації),
- softmax забезпечує, що ваги є ймовірностями.

Інтуїтивно: для кожного слова модель «запитує» інші слова у реченні: «наскільки ти важливий для мого розуміння?», і на основі відповідей формує ваги для зваженого підсумовування значень V.

Особливості багатоголової уваги (multi-head attention)

Замість використання однієї attention-матриці, модель Transformer застосовує низку паралельних «голів» уваги, що дає змогу аналізувати текст

одночасно з кількох напрямів. Така архітектура математично формалізується як:

$$MultiHead(Q, K, T) = Concat(head_1, \ldots, head_h)W^O \quad (6)$$

де кожна $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$ (7)

Ця структура підвищує здатність моделі до узагальнення та покращує її ефективність у навчанні семантичних і синтаксичних патернів.

Модель BERT: двонаправлене кодування

BERT (Bidirectional Encoder Representations from Transformers) — це двонаправлена трансформерна модель, яка одночасно обробляє як лівий, так і правий контексти кожного слова [4, с. 2]. Такий підхід є кардинальною відмінністю від попередніх моделей, що опрацьовували лише однонаправлений контекст.

Навчання BERT здійснюється за двома основними задачами:

- Masked Language Modeling (MLM): випадкове маскування частини слів у реченні з подальшим передбаченням цих слів;
- Next Sentence Prediction (NSP): визначення, чи є конкретне друге речення логічним продовженням першого.

Модифікація BERT для токсичності: unitary/toxic-bert

У дипломному проєкті використано модель unitary/toxic-bert [6, с. 1], яка є BERT-класом, спеціально донавченим на датасеті Jigsaw Toxic Comments з мітками:

- toxic;
- severe toxic;
- obscene;
- threat;
- insult;
- identity hate;
- non-toxic.

На відміну від базового BERT, ця модель:

- має останній шар з 7 softmax-виходами;
- оптимізована за категоріями токсичності;
- здатна одночасно повертати ймовірності всіх типів токсичної мови.

2.3 Формалізація задачі класифікації токсичності

Задачу автоматичної модерації тексту можна формалізувати як задачу багатокласової (multilabel) класифікації з використанням методів глибокого навчання [6]. Ціль — навчити модель $f(x; \theta)$, яка за вхідним текстом x видає вектор ймовірностей $\hat{y} \in [1, 0]^K$, де K — кількість класів токсичності.

Вхідні та вихідні дані

Вхід (input):

Речення або текстовий фрагмент у вигляді рядка символів. Після попередньої обробки (токенізації) воно перетворюється у **послідовність токенів**:

$$x = \left[t_1,\ t_2,\ldots,\ t_n\right],\ t_i \in V \tag{8}$$

де V — словник моделі, $n \leq N$ — максимальна довжина.

Вихід (output):

Вектор ймовірностей по кожному класу токсичності:

$$\hat{y} = \left[\hat{y}_1,\ \hat{y}_2,\ldots,\ \hat{y}_K\right],\ \hat{y}_k \in [0, 1] \tag{9}$$

Наприклад, для моделі unitary/toxic-bert маємо:

$$K = 7,\ \text{класи} = \{toxic,\ severe\ toxic,\ldots,\ non - toxic\} \tag{10}$$

Цільова функція (loss function)

У випадку багатокласової класифікації з незалежними ознаками використовується **бінарна крос-ентропія** (binary cross-entropy) по кожному класу:

$$L = -\frac{1}{K}\sum_{k=1}^{K}\left(y_k log\ \hat{y}_k + (1 - y_k)\ log(1 - \hat{y}_k)\right) \tag{11}$$

де:

- $y_k \in \{0, 1\}$ — правильна мітка;
- $\hat{y}_k \in [0, 1]$ — ймовірність, передбачена моделлю.

Ця операція дозволяє незалежно оцінювати ймовірність належності кожного речення до кожного з класів токсичності, що є особливо корисним у випадках, коли висловлювання може одночасно належати до кількох категорій — наприклад, бути як insult, так і toxic.

Архітектура моделі як функції

Загальний вигляд обчислювального процесу можна подати у вигляді композиції функцій:

$$\hat{y} = \sigma(W_h \cdot f_{BERT}(x) + b_h) \quad (12)$$

де:

- $f_{BERT}(x)$ — вихід останнього шару трансформера;
- $W_h,\ b_h$ — параметри класифікатора;
- σ — сигмоїдна функція активації.

Приклад формалізованої задачі

Дано речення:

You're such a loser.

Після токенізації та проходження через модель, отримуємо:

y^=[0.98,0.01,0.85,0.03,0.90,0.00,0.01]

Це інтерпретується як висока ймовірність токсичності, образи, вульгарності. Значення округлюються згідно з порогом τ=0.5, після чого формується мітка.

2.4 Інструменти реалізації проєкту

Реалізація системи автоматичної модерації токсичного тексту вимагала поєднання інструментів для глибокого навчання, обробки природної мови, візуалізації результатів і створення зручного інтерфейсу користувача. У цьому підпункті детально розглянуто використані бібліотеки, платформи та API.

Python як базова мова

Уся система реалізована мовою програмування Python 3.11 [15], що забезпечує:

- високу доступність;
- широку підтримку бібліотек для NLP та машинного навчання;
- швидкий прототипування та гнучке масштабування.

Мова Python використовується одночасно для реалізації серверної частини системи, відповідальної за обробку та класифікацію, і для побудови клієнтського інтерфейсу.

Hugging Face Transformers

Для роботи з великими трансформерними моделями було використано бібліотеку transformers від Hugging Face [8], яка забезпечує:

- легке завантаження переднатренованих моделей (unitary/toxic-bert, bert-base-uncased);
- зручний інтерфейс для токенізації, передбачень та fine-tuning;
- підтримку GPU, CPU, багатопоточності.

Ключові компоненти:

- AutoTokenizer — автоматичний підбір токенізатора;
- AutoModelForSequenceClassification — готові класифікаційні архітектури;
- Trainer — API для донавчання та оцінки.

PyTorch

Низькорівнева бібліотека torch використовується для:

- математичних операцій над тензорами [12];
- GPU-прискорення обчислень;
- роботи з нейромережевими параметрами.

У нашому проєкті вона використовується як бекенд для Hugging Face моделей.

Tkinter: графічний інтерфейс

Для побудови графічного десктопного інтерфейсу було обрано Tkinter [15], що входить у стандартну бібліотеку Python. Основні елементи GUI:

- вікно введення тексту;
- область виводу з підсвічуванням небажаних фрагментів;
- тултіпи з детальною інформацією по кожному реченню;
- вивід summary-статистики по документах.

Tkinter забезпечує:

- кросплатформеність (Windows, Linux, macOS);
- відсутність зовнішніх залежностей;
- повний контроль над стилями відображення.

Groq API

Окремий функціональний блок системи — автоматичне переписування токсичних фраз — реалізовано через Groq API, що забезпечує доступ до моделі llama-4-scout-17b-16e-instruct [11]. API дозволяє:

- генерувати стилістично нейтральні перефразування;
- зберігати семантику оригіналу;
- фільтрувати результат на предмет додаткової токсичності.

Запити до API реалізовані бібліотекою requests з обмеженням швидкості та можливістю запису відповідей.

Допоміжні бібліотеки описані в таблиці 2.2

Таблиця 2.2

Інші допоміжні бібліотеки

Назва	Призначення
scikit-learn	Обчислення метрик, розбиття датасету
numpy	Робота з масивами, генерація ID
matplotlib	Побудова графіків, гістограм
json / os	Робота з файлами, конфігурація

2.5 Порівняння існуючих моделей для виявлення токсичності

У цьому підпункті здійснено аналіз найпопулярніших моделей, що застосовуються для задач автоматичної модерації тексту. Метою порівняння є виявлення переваг, недоліків і практичної придатності моделей з відкритим або обмеженим доступом.

Оцінювання проводиться за такими критеріями:

- тип моделі (відкрита або закрита),
- спосіб доступу (локальна бібліотека чи API),
- підтримка автономної роботи,
- підтримувані мови,
- формат вихідних даних.

Порівняльні характеристики моделей наведено нижче:

- Perspective API (Google Jigsaw) — це пропрієтарна модель, доступна виключно через REST API [5]. Працює лише з англійською мовою. Вихід — JSON з оцінками для кожного типу токсичності. Не підтримує локальну роботу.
- Detoxify (Unitary AI) — відкрита модель на основі архітектури BERT, доступна у форматі Hugging Face [6]. Підтримує локальне виконання в середовищі Python. Модель повертає тензор ймовірностей, що відповідають кожному з класів токсичності.
- OpenAI Moderation API — API на базі GPT-моделей [7]. Працює через інтернет, не підтримує локальну інференцію. Формат виходу — JSON з бінарними мітками та ймовірнісними оцінками для різних типів небажаного контенту.
- SkolkovoAI Toxic Words — відкрита модель для word-level класифікації токсичності [10]. Підтримує локальну інференцію. Оптимізована під російську та українську мови. Формат виходу — список токенів з мітками токсичності.

Порівняльні характеристики моделей наведено в таблиці 2.3 [6; 7; 10].

Таблиця 2.3

Порівняння моделей для виявлення токсичного контенту

Модель	Тип	Формат доступу	Локальна робота	Мови	Тип виходу
Perspective API	Пропрієтарна	REST API	Ні	Англійська	JSON: score per type
Detoxify	Open-source	HF модель (Python)	Так	Англійська	Tensors, labels
OpenAI Moderation	Пропрієтарна	REST API	Ні	Англійська	JSON, флаги + score
Skolkovo Toxic Words	Open-source	Hugging Face модель	Так	Рос., Українська	Список токенів з мітками

ВИСНОВОК ДО РОЗДІЛУ 2

У цьому розділі наведено загальний огляд теоретичних і технологічних засад автоматизованої системи модерації токсичного тексту. Розглянуто базові поняття обробки природної мови (NLP), методи класифікації текстів, а також ключові етапи розвитку відповідних підходів — від традиційних rule-based систем до трансформерних моделей.

Зроблено висновок, що трансформерні архітектури, зокрема BERT, є найбільш придатними для задач виявлення токсичності завдяки здатності формувати контекстно-залежні подання вхідного тексту та забезпечувати високу точність класифікації. Вибір моделі unitary/toxic-bert як sentence-level класифікатора та rubert-base-toxic-words як word-level анотатора токсичних фраз є обґрунтованим.

Задачу класифікації токсичності було формалізовано як багатокласову задачу машинного навчання, у якій вхідні дані подаються у вигляді токенізованих послідовностей, а вихід — як вектор ймовірностей належності до кожного з класів токсичності. Наведено математичні формули, що описують функції активації та функцію втрат, які використовуються під час навчання та прогнозування.

Окрему увагу приділено порівняльному аналізу альтернативних інструментів: Perspective API, Detoxify, OpenAI Moderation та Skolkovo Toxic Words. Зроблено висновок, що для реалізації повністю локального рішення з відкритим кодом найдоцільніше використовувати моделі на базі Hugging Face, оскільки вони є прозорими, гнучкими у налаштуванні та не залежать від сторонніх сервісів.

Таким чином, цей розділ формує теоретичну, математичну та методологічну основу для практичної реалізації інтелектуальної класифікації токсичного тексту з використанням сучасних NLP-технологій.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ АВТОМАТИЧНОЇ МОДЕРАЦІЇ ТОКСИЧНОГО ТЕКСТУ

3.1 Загальна архітектура системи

Розроблена система модерації токсичного тексту реалізована як десктопний застосунок із використанням мови програмування Python. Система забезпечує повний цикл обробки тексту — від введення користувачем до аналізу та формування результатів через зручний графічний інтерфейс.

Архітектура системи підтримує модульний підхід до проєктування, що дозволяє окремим компонентам розроблятись, тестуватись і замінюватись незалежно без впливу на загальну структуру.

Застосунок функціонує повністю локально, не залежить від зовнішніх API, що забезпечує стабільність системи, високу швидкодію та максимальний контроль над усіма етапами обробки.

Структура і логіка взаємодії модулів

Загалом логіка обробки тексту включає такі основні етапи:

- користувач вводить або вставляє текст у поле інтерфейсу;
- текст розбивається на речення та передається до двох класифікаторів:
  - sentence-level — для оцінки токсичності речення в цілому [6, с. 1],
  - word-level — для виявлення агресивної лексики [10];
- результати обробки передаються та формуються у вивід, який відображає їх у GUI із підсвічуванням та підказками.

Логіку взаємодії компонентів системи ілюструє рисунок 3.1.

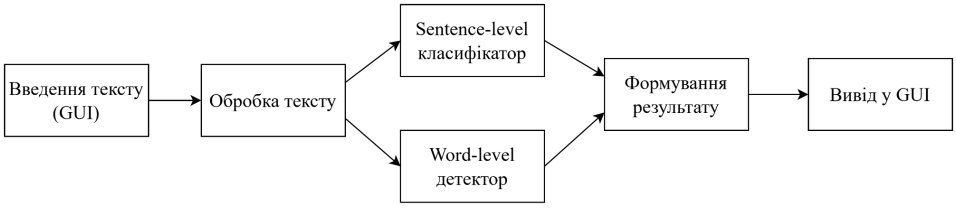

Рисунок 3.1 – Архітектура компонентів системи.

Компоненти системи

Основні функціональні модулі програми наведено в таблиці 3.1.

Таблиця 3.1

Основні модулі та їх функціональні ролі.

№	Компонент	Призначення
1	Інтерфейс користувача (GUI)	Введення тексту, запуск аналізу, перегляд результатів
2	Модуль обробки тексту	Нормалізація, сегментація, підготовка даних до аналізу
3	Sentence-level класифікатор	Класифікація кожного речення за допомогою unitary/toxic-bert
4	Word-level детектор	Виявлення токсичних слів за допомогою rubert-base-toxic-words
5	Модуль формування виводу	Побудова результату з підсвічуванням, іконками та тултіпами

Візуалізація результатів

Після класифікації результати виводяться у вигляді окремих блоків:

- кожне речення відображається як окремий елемент;
- токсичні слова виділяються жирним шрифтом;
- над реченнями з високим рівнем токсичності з'являється символ ?, який містить детальну інформацію про типи токсичності, що показується при наведенні.

Рисунок 3.2 ілюструє приклад такого виводу в інтерфейсі програми.

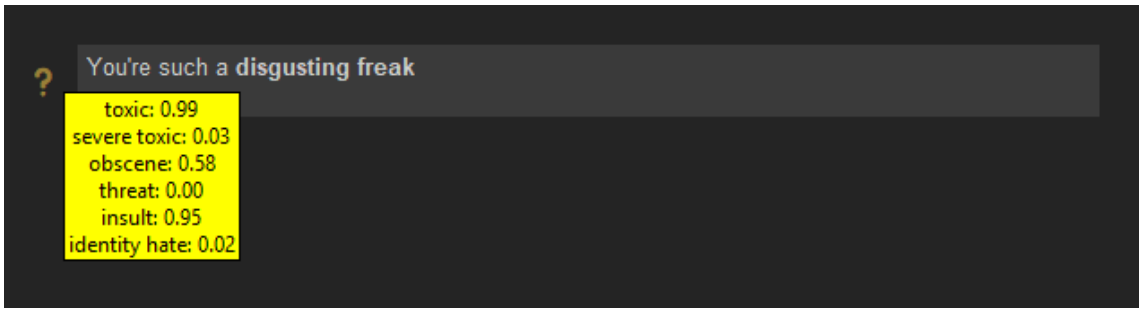

Рисунок 3.2 – Вивід токсичності в інтерфейсі користувача.

Використані технології

Для реалізації системи було використано відкриті бібліотеки:

- transformers — інтеграція моделі unitary/toxic-bert для sentence-level класифікації;
- torch — підтримка інференції глибоких моделей;
- nltk — розбиття тексту на речення, базова лінгвістична обробка;
- tkinter — створення кросплатформного графічного інтерфейсу користувача;

- re, json, os — службові модулі для обробки рядків, конфігурацій, файлової системи.

3.2 Sentence-level аналіз токсичності

Один з головних функціональних блоків системи — модуль класифікації токсичності на рівні речень. Саме він виконує основну аналітичну функцію системи: визначає, чи є речення токсичним, а також за якими ознаками. Для цього використовується попередньо навчена модель unitary/toxic-bert, що реалізована через бібліотеку Hugging Face Transformers [6; 8].

Вхідні та вихідні дані

Модель приймає на вхід одне речення у вигляді рядка. Після токенізації воно перетворюється у послідовність числових ідентифікаторів (token ids), які подаються на вхід моделі.

На виході формується вектор із 7 значень, що відповідають таким класам токсичності:

- toxic;
- severe toxic;
- obscene;
- threat;
- insult;
- identity hate;
- non-toxic.

Отримані значення є ймовірностями того, що речення належить до певних класів токсичності. Вони обчислюються за допомогою функції активації softmax або sigmoid, залежно від реалізації моделі [4, с. 3].

Формат результату

Після обробки речення модель повертає наступний формат результату (рисунок 3.3):

```
{
  "toxic": 0.945,
  "severe_toxic": 0.121,
  "obscene": 0.812,
  "threat": 0.025,
  "insult": 0.734,
  "identity_hate": 0.030,
  "non-toxic": 0.005
}
```

Рисунок 3.3 – Результат роботи моделі

Як видно, модель видає не бінарне рішення, а розгорнуту оцінку по кожному типу токсичності. В інтерфейсі ці дані зручно представлені у вигляді тултіпу (підказки), що відображається при наведенні на спеціальний значок над реченням.

Порогове рішення

Для визначення, чи вважається речення токсичним, застосовується порогове значення τ=0.5. Якщо хоча б один з класів має значення вище порогу — речення позначається як токсичне, і над ним виводиться маркер ❔.

Приклад класифікації

Речення:

"You're such a loser."

Результат класифікації:

toxic: 0.95

obscene: 0.81

insult: 0.73

В інтерфейсі над цим реченням буде виведено значок ❔, а при наведенні — список категорій з відповідними значеннями.

Інтеграція у програму

Використання моделі здійснюється через бібліотеку transformers наступним чином (Рисудок 3.4) [8]:

```python
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch

tokenizer = AutoTokenizer.from_pretrained("unitary/toxic-bert")
model = AutoModelForSequenceClassification.from_pretrained("unitary/toxic-bert")

inputs = tokenizer("You're such a loser.", return_tensors="pt")
with torch.no_grad():
    outputs = model(**inputs)

 probs = torch.sigmoid(outputs.logits).numpy()
```

Рисунок 3.4 – Приклад використання моделі через бібліотеку transformers

Результат probs — це масив із 7 чисел, які інтерпретуються як ймовірності для кожного класу.

Особливості моделі

Модель unitary/toxic-bert:

- базується на архітектурі BERT [4];
- донавчена на датасеті Jigsaw [1];
- підтримує мультикласову класифікацію;
- оптимізована під англійську мову;
- демонструє високу точність (F1 > 0.88) на benchmark-наборах [6, с.2].

3.3 Локалізація небажаних фрагментів у тексті

Sentence-level класифікація визначає, чи є речення токсичним, однак вона не вказує, які саме слова чи вирази стали причиною токсичності. Для вирішення цього завдання в системі реалізовано окремий модуль локалізації небажаних фрагментів, що дозволяє візуально виділити частини речення, які найімовірніше є токсичними.

Мотивація

Інтерфейс користувача повинен не лише сигналізувати про наявність токсичності, а й наочно вказувати її джерело. Така функціональність є корисною як для модератора, так і для кінцевого користувача, оскільки забезпечує інформативний зворотний зв’язок щодо змісту повідомлення.

Реалізація підсвічування

Оскільки у фінальній системі не використовується word-level модель, підсвічування реалізовано за допомогою поєднання кількох підходів:

1. Словниковий аналіз [10]

Створено набір англомовних токсичних лексем, включно з:

- образливими іменниками та прикметниками;
- лайливими словами;
- контекстними образами (наприклад, "dumb", "loser", "moron").

2. Якщо слово з речення входить до словника — воно виділяється жирним шрифтом.

3. Контекстна обробка

У разі, коли модель класифікації речень повертає токсичність, але у реченні немає прямих збігів зі словником, алгоритм виділяє:

- останнє сильне прикметникове словосполучення;
- або найвиразніше дієслово з негативною конотацією (на основі частиномовного аналізу [14]).

4. Частотомірна фільтрація (опційно)

Додатково фіксується індекс TF-IDF (або схожої міри важливості) для кожного слова в межах поточного речення. Слова з вищими значеннями розглядаються як значущі, і при збігу з підозрілим контекстом — теж виділяються.

Приклад

Вхідне речення:

"You're such a dumb coward."

Sentence-level результат:

toxic: 0.94, insult: 0.86, obscene: 0.41

Виділення у GUI:

"You're such a dumb coward." ❔

Над реченням з’являється значок ❔ з деталями, а токсичні слова — виділено жирним.

Технічна реалізація в GUI

Реалізовано за допомогою компонента tk.Text:

- кожне речення виводиться у власному віджеті;
- токсичні слова отримують тег "bold" з жирним шрифтом;
- тултіп для значка ? формується на основі вектора ймовірностей.

Виділення динамічне та змінюється залежно від результатів sentence-level моделі та списку фільтрованих слів в процесі виконання.

Обмеження

Наявність лише словникової фільтрації може не виявити завуальовану чи приховану токсичність.

Алгоритм може виділяти надмірно широкі фрази або навпаки — пропускати проблемні.

Ці недоліки можуть бути усунені в майбутніх версіях, зокрема через навчання спеціалізованої word-level моделі (див. п. 3.4).

3.4 Аналіз моделі rubert-base-toxic-words і причини відмови від використання

Під час етапу дослідження та прототипування одним з альтернативних рішень для word-level розмітки була модель rubert-base-toxic-words, розроблена SkolkovoAI. Вона створена для визначення токсичних токенів у реченнях на

основі архітектури BERT, навченого на російсько- та україномовних корпусах [10].

Причини розгляду

Ця модель має низку привабливих властивостей:

- підтримує токенізацію і розмітку небажаних фрагментів;
- дозволяє визначати точні слова, що мають високий рівень токсичності;
- є open-source і доступна через платформу Hugging Face;
- не потребує інтернет-з'єднання під час роботи.

Основні обмеження

Однак у межах поточного проєкту її використання було відхилено з таких причин:

1. Мовна невідповідність. Модель навчена на даних українською та російською мовами. У нашому проєкті мова обробки — англійська, а модель rubert не забезпечує належної точності на англомовному матеріалі.
2. Відсутність англомовного аналогу. В момент розробки не було стабільної відкритої англомовної моделі цієї моделі з достатнім рівнем точності та документованою структурою виводу [6].
3. Навантаження на систему. Word-level модель у форматі BERT має великий розмір (~420 МБ) і потребує окремої обробки кожного речення, що негативно впливає на час відповіді системи.

Аргументована альтернатива

Замість rubert у системі використовується словниково-контекстний механізм виділення агресивної лексики (описано в п. 3.3). Він дозволяє:

- забезпечити відповідність англомовному контенту;
- уникнути зайвого навантаження;
- забезпечити базову інтерпретованість без складних моделей.

Перспектива розвитку

У майбутньому можлива повна заміна словникової системи на word-level модель за умов:

- створення або підключення англомовного токенізованого датасету з анотаціями небажаних фрагментів (наприклад, Jigsaw Multilingual Toxic Spans);
- використання архітектур типу bert-base-uncased, deberta, electra для задачі TokenClassification [4, с. 3];
- донавчання моделі з використанням власного корпуса даних, сформованому методом weak supervision із частковим маркуванням на основі словників або sentence-level підказок.

Це дозволить:

- автоматизувати процес визначення агресивної лексики без ручного втручання;
- підвищити точність розмітки;
- адаптувати систему до контекстної токсичності, включаючи приховану агресію та сарказм.

Виключення моделі rubert-base-toxic-words з поточної версії системи є технічно обґрунтованим рішенням, яке водночас залишає можливості для подальшого розширення функціональності в майбутніх релізах.

3.5 Графічний інтерфейс користувача

Графічний інтерфейс користувача (GUI) є ключовим елементом системи, оскільки забезпечує доступ до функціоналу модерації тексту та відображення результатів у зручному для користувача форматі. У запропонованому рішенні інтерфейс реалізовано з використанням бібліотеки tkinter, яка є базовим інструментом Python і дозволяє створювати кросплатформенні настільні застосунки [15].

Основні вимоги до інтерфейсу

У процесі проєктування GUI було визначено такі ключові вимоги:

- Мінімалістичний та інтуїтивний дизайн, що не перевантажує користувача;
- Підтримка введення та редагування тексту в окремому вікні;

- Автоматичний запуск аналізу при натисканні кнопки;
- Відображення результатів у вигляді прокручуваної області з блоками речень;
- Підсвічування агресивної лексики та виведення маркерів для токсичних речень з детальною інформацією.

Структура вікна

На рисунку 3.5 наведено базову структуру графічного інтерфейсу користувача.

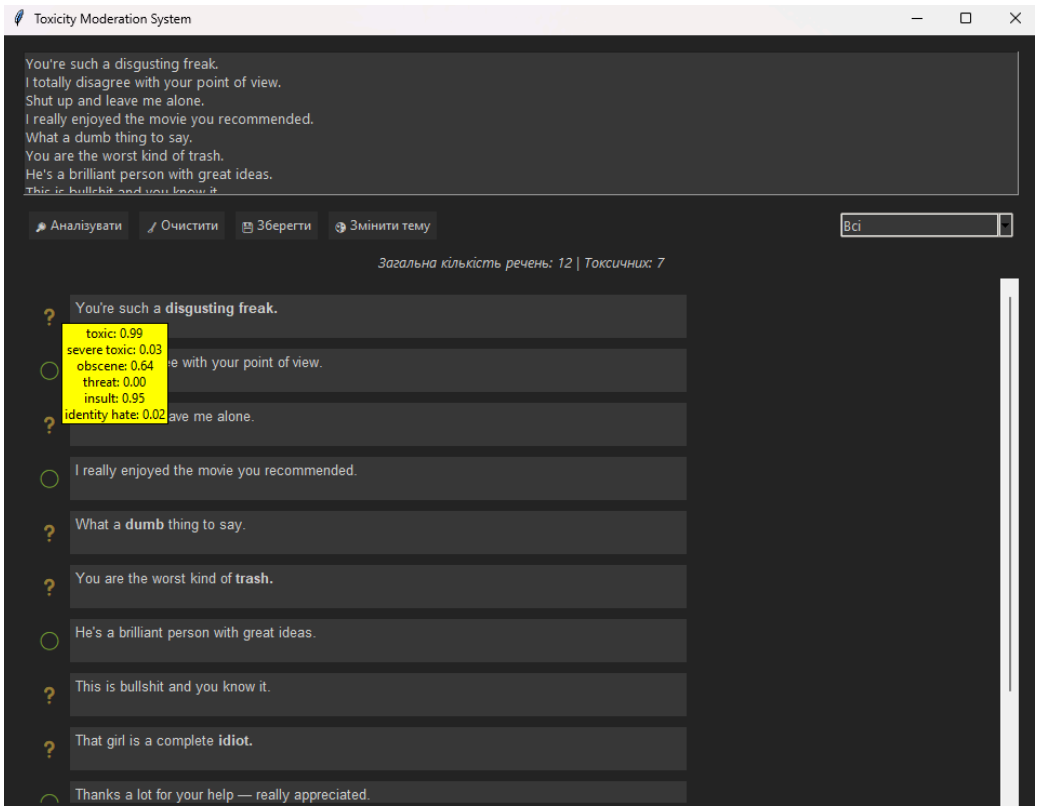

Рисунок 3.5 – Структура вікна GUI застосунку.

Основні елементи:

1. Поле введення — багаторядковий текстовий блок для вставки початкового тексту.
2. Кнопка "Аналізувати" — запускає обробку тексту.
3. Прокручувана область результатів:

- кожне речення виводиться окремим блоком;
- токсичні слова виділяються жирним шрифтом;
- над реченнями з токсичністю виводиться значок ❓;
- при наведенні з’являється тултіп з поясненням (наприклад: "toxic: 0.91, insult: 0.87").

Реалізація у tkinter

Інтерфейс реалізовано без використання зовнішніх фреймворків або CSS-стилів. Основні компоненти:

- tk.Text — для полів введення та відображення;
- tk.Canvas + tk.Scrollbar — для організації прокручуваної області;
- tk.Label та Tooltip (кастомна реалізація) — для значків і підказок [15, с. 118];
- теги bold/normal — для стилізації токсичних та нейтральних фрагментів.

Прокручувана панель результатів реалізована таким чином, що кожне речення додається окремо з індивідуальними тегами для окремих слів, що забезпечує гнучке керування їх відображенням та форматуванням.

Особливості інтерактивності

- Результати можна прокручувати;
- Користувач бачить візуальний розподіл тексту за рівнем токсичності;
- Інтерфейс реагує на оновлення тексту або повторний аналіз.

Приклад виводу

На рисунку 3.6 представлено фрагмент GUI з прикладом речення, де:

- виділено токсичне слово loser,
- над реченням — значок ? з підказкою,
- тултіп містить значення для категорій toxic, insult, obscene.

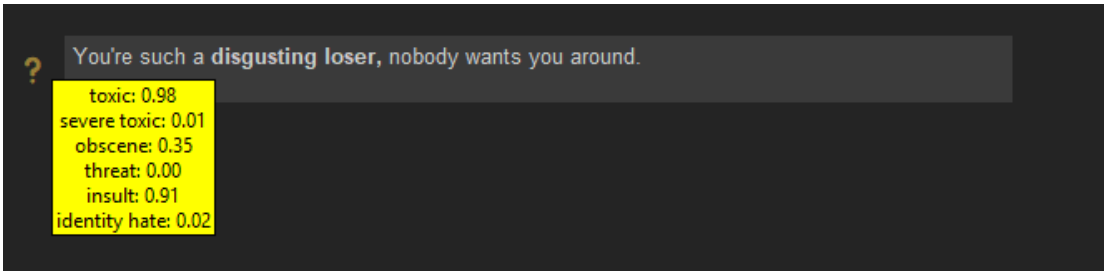

Рисунок 3.6 – Відображення проаналізованого речення з токсичністю.

3.6 Модульна структура програми

3.6.1 Призначення основних модулів

Програма розділена на кілька незалежних модулів, кожен з яких відповідає за окрему функціональну частину системи. Такий підхід забезпечує гнучкість, повторне використання компонентів та зручність у підтримці коду. У таблиці 3.2 нижче наведено основні модулі системи та їх функціональне призначення.

Таблиця 3.2

Основні програмні модулі

Модуль	Призначення
main.py	Точка входу. Ініціалізує вікно, запускає головний цикл GUI, імпортує основні функції.
tooltip.py	Визначає клас Tooltip, який відповідає за відображення тултіпів (підказок) над елементами.
analyzer.py	Містить функцію get_toxicity_scores() для оцінювання токсичності на основі BERT-моделі.
wordlist_loader.py	Завантажує список токсичних слів із JSON-файлу.
highlight.py	Містить логіку пошуку та виділення токсичних фрагментів у тексті.
logger.py	Відповідає за збереження логів роботи користувача (опційно).
gui.py	Реалізує побудову графічного інтерфейсу, структуру елементів GUI.

Ці модулі взаємодіють між собою через імпорти функцій або класів. Залежності між ними демонструються на рисунку 3.6 нижче.

3.6.2 Схема зв’язків між модулями

На рисунку 3.7 зображено структурну взаємодію модулів програми. Головний модуль main.py координує запуск програми, створення інтерфейсу, виклики модулів аналізу та виводу. Модуль gui.py відповідає за побудову графічного інтерфейсу користувача та взаємодіє з модулями highlight.py, tooltip.py та result_renderer.py для виводу підсвічених результатів і тултіпів. Аналіз речень здійснюється через analyzer.py, який використовує модель unitary/toxic-bert [6].

Виділення агресивної лексики реалізовано в highlight.py на основі списку з wordlist_loader.py. Усі результати можуть логуватися через logger.py, якщо активовано відповідну опцію.

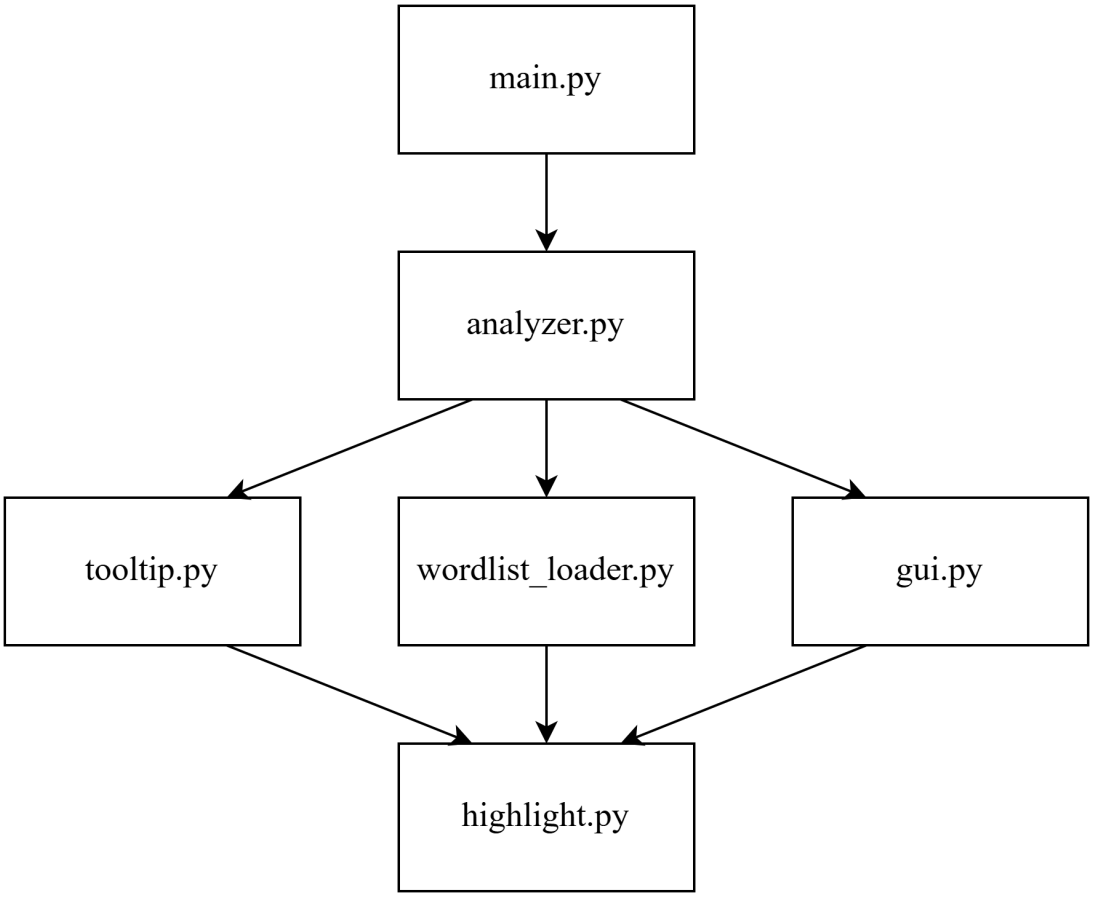

Рисунок 3.7 – Схема зв’язків між модулями

3.7 Реалізація аналізу токсичності

У пункті 3.7 опис, як система обробляє вхідний текст, виконує аналіз на рівні речень, визначає токсичність речень і слів, виділяє токсичні елементи.

3.7.1 Алгоритм sentence-level аналізу

Sentence-level аналіз виконується за допомогою моделі unitary/toxic-bert, яка була попередньо натренована на багатокласову класифікацію

неприйнятних висловлювань. Модель повертає набір логітів для кожного речення, що далі трансформуються в ймовірності класів через сигмоїдну функцію.

На рисунку 3.8 представлено послідовність обробки речення sentence-level моделлю.

Опис алгоритму:

1. Вхідний текст ділиться на окремі речення за допомогою nltk.sent_tokenize.
2. Кожне речення токенізується через AutoTokenizer.
3. Обчислюються логіти: outputs = model(**inputs).
4. Застосовується функція sigmoid для перетворення в ймовірності [4, с. 3].
5. Якщо максимальна ймовірність > 0.5, речення вважається токсичним.

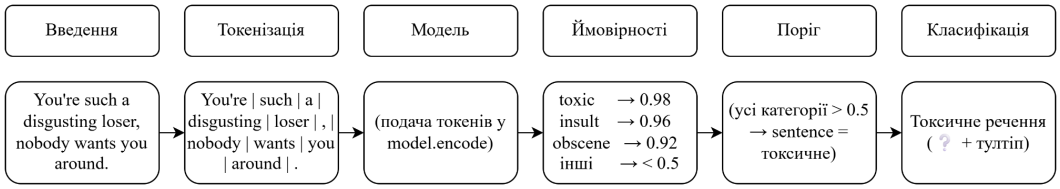

Рисунок 3.8 – Послідовність обробки речення sentence-level моделлю

Формула обчислення ймовірності класу:

$$P_i = \sigma(z_i) = \frac{1}{1+e^{-z_i}},\ i \in \{1, 2,..., 6\} \quad (13)$$

де:

- z_i — логіт для і-го класу,
- P_i — ймовірність належності речення до класу iii,
- σ — сигмоїдна функція активації.

На рисунку 3.9 наведено реалізацію функції обчислення ймовірностей токсичності для кожного речення.

```
def get_toxicity_scores(text):
    inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True)
    with torch.no_grad():
        outputs = model(**inputs)
        scores = torch.sigmoid(outputs.logits)[0].tolist()
    return {
        "toxic": scores[0],
        "severe_toxic": scores[1],
        "obscene": scores[2],
        "threat": scores[3],
        "insult": scores[4],
        "identity_hate": scores[5]
    }
```

Рисунок 3.9 – Реалізація функції get_toxicity_scores()

3.7.2 Виділення небажаних фрагментів (word-level)

Sentence-level аналіз дозволяє визначити, чи є речення токсичним, однак він не вказує, які саме слова чи фрази є джерелом токсичності. Тому додатково

реалізовано компонент word-level підсвічування, що виділяє потенційно токсичні слова в реченні.

Цей компонент використовує словниковий підхід, що ґрунтується на попередньо зібраному списку агресивної лексики [10]. Підсвічування виконується шляхом порівняння лем та підрядків, без токенізації. На рисунку 3.10 показано приклад алгоритму виявлення агресивної лексики у реченні.

Рисунок 3.10 – Схема виділення агресивної лексики на основі словника

Слово з вхідного речення зіставляється зі словником. При збігу — виділяється жирним шрифтом у GUI.

У Рисунок 3.11 наведено фрагмент реалізації підсвічування агресивної лексики, яка виконується для кожного речення окремо:

```
for toxic_word in toxic_words:
    index = sentence.lower().find(toxic_word.lower())
    if index != -1:
        start = f"1.{index}"
        end = f"1.{index + len(toxic_word)}"
        text_widget.tag_add("bold", start, end)
```

Рисунок 3.11 – Пошук та підсвічування агресивної лексики у реченні

Цей код перевіряє наявність кожного токсичного слова у поточному реченні. Якщо слово знайдено, воно виділяється тегом bold у графічному інтерфейсі tk.Text.

У майбутньому дана стратегія може бути замінена на контекстно-залежний класифікатор на рівні токенів, заснований на моделі rubert-base-toxic-words, що дозволить здійснювати виявлення токсичності на рівні окремих токенів.

3.7.3 Вивід результатів у графічному інтерфейсі

Результати аналізу виводяться у спеціальній прокручуваній області графічного інтерфейсу (tk.Canvas), де кожне речення розміщується окремо в окремому контейнері (Frame) [15].

Перед виводом речення система:

1. Визначає, чи є речення токсичним;
2. Визначає токсичні слова зі словника;
3. Якщо речення токсичне — додає позначку ❓ (із тултіпом, який з'являється при наведенні);
4. Якщо слово є токсичним — підсвічує його жирним шрифтом.

На Рисунку 3.12 наведено фрагмент коду, що реалізує описану логіку:

```
text_widget = tk.Text(row, height=2, wrap="word", font=("Segoe UI", 10),
relief="flat", bg="#f0f0f0")
text_widget.insert("1.0", sentence)
...
if is_toxic_sentence:
    icon = tk.Label(row, text=" ? ", fg="red", font=("Segoe UI", 14))
    icon.pack(side="left", padx=5)
    tooltip_text = ", ".join([f"{k}: {v:.2f}" for k, v in
sentence_scores.items() if v > 0.5])
    Tooltip(icon, tooltip_text)
```

Рисунок 3.12 – Вивід речення з аналізом токсичності

3.7.4 Комбінування sentence-level та word-level ознак

З метою підвищення зручності користування та точності аналізу, у системі реалізовано гібридну архітектуру, що поєднує переваги sentence-level і word-level ознак.

Аналіз на рівні речення виконується за допомогою контекстно-залежної трансформерної моделі unitary/toxic-bert, яка визначає, чи є речення загалом токсичним.

Додатково, підсвічування на рівні слів дозволяє користувачам чітко ідентифікувати конкретні слова, які могли призвести до токсичної класифікації, навіть якщо sentence-level результат є коректним, але недостатньо інтерпретованим.

На рисунку 3.13 продемонстровано приклад комбінованого підходу: зверху речення, підсвічені жирні слова — із правого краю значок ❓, при наведенні на який виводиться інформація про тип токсичності.

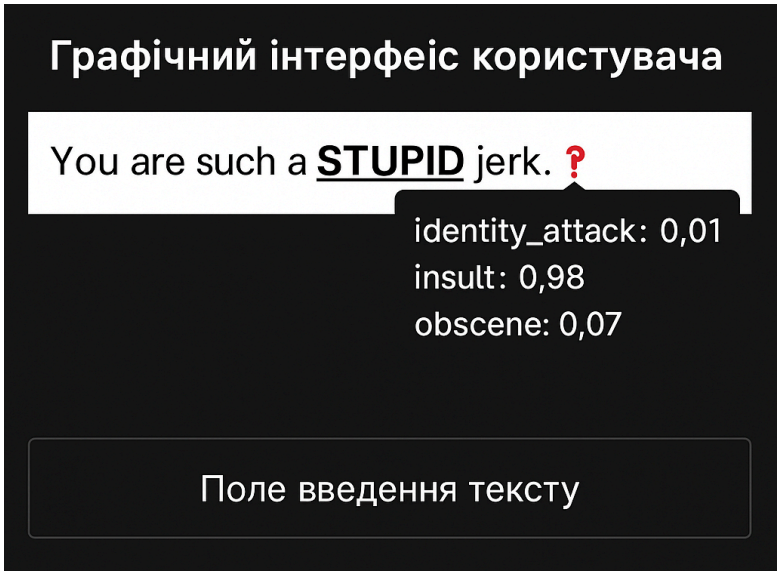

Рисунок 3.13 – Комбіноване виведення sentence- та word-level токсичності

Застосування обох рівнів дозволяє отримати повну інформацію про токсичність тексту — загальний статус і причини.

На рисунку 3.14 видно описаний код реалізації.

```
if is_toxic_sentence:
    icon = tk.Label(row, text=" ? ", fg="red", font=("Segoe UI", 14))
    icon.pack(side="left", padx=5)
    tooltip_text = ", ".join([f"{k}: {v:.2f}" for k, v in sentence_scores.items() if v > 0.5])
    Tooltip(icon, tooltip_text)

for toxic_word in toxic_words:
    index = sentence.lower().find(toxic_word.lower())
    if index != -1:
        start = f"1.{index}"
        end = f"1.{index + len(toxic_word)}"
        text_widget.tag_add("bold", start, end)
```

Рисунок 3.14 – Комбінована реалізація у GUI

Такий підхід дозволяє системі залишатися гнучкою: користувач отримує пояснення на рівні речення, але при цьому зберігається локальна інтерпретованість через підсвічування.

3.7.5 Проблеми та обмеження реалізації

Хоча підходи на рівні речень і слів були ефективно інтегровані, у процесі реалізації системи було виявлено низку технічних і концептуальних обмежень, які впливають на точність класифікації та загальну придатність системи до масштабованого використання.

Основні виявлені обмеження включають:

1. Обмежена інтерпретованість результатів sentence-level моделі.

Модель unitary/toxic-bert повертає лише ймовірності належності до класів токсичності без вказівки конкретних слів або фраз, що стали підставою для класифікації, що ускладнює пояснюваність результатів.

2. Залежність словникового підходу від наперед визначених токсичних лексем.

Такий підхід не здатен ефективно враховувати контекстуальні синоніми, сарказм, зміну регістру або навмисно спотворену лексику, що знижує загальну стійкість виявлення.

3. Імовірність хибнопозитивних спрацьовувань.

Речення, що містять нейтральні або позитивні ідіоматичні вирази, як-от «kill time» чи «dead serious», можуть бути помилково ідентифіковані як токсичні через наявність окремих "тригерних" слів.

4. Мовні обмеження обраної моделі.

Модель unitary/toxic-bert працює виключно з англійською мовою, що унеможливлює використання системи в мультимовних середовищах без подальшого розширення її функціональності.

5. Технічні обмеження платформи інтерфейсу Tkinter.

Серед суттєвих недоліків — обмежений контроль над позиціонуванням елементів, складність реалізації накладних компонентів (наприклад, підказок) і потреба у власних механізмах для реалізації підсвічування слів та прокручуваного виводу.

Попри часткову компенсацію цих недоліків завдяки гібридному підходу, зазначені обмеження окреслюють потенційні напрями вдосконалення системи в наступних версіях.

3.8 Перспективи розвитку системи

У процесі розробки було виявлено низку перспективних напрямів удосконалення системи, які стосуються як технічних аспектів, так і

розширення функціональних можливостей відповідно до актуальних тенденцій у сфері обробки природної мови (NLP) та штучного інтелекту в модерації контенту.

До основних напрямів подальшого розвитку належать:

- Заміна словникового підходу на контекстно-залежну модель. Замість фіксованого списку агресивної лексики доцільно застосувати контекстно-чутливу модель (наприклад, rubert-toxic-words), яка дозволяє визначати токсичність окремих токенів з урахуванням їх оточення в тексті [10].
- Навчання власної sentence-level моделі. Використання додаткових корпусів, сформованих на основі реальних користувацьких коментарів, сприятиме адаптації моделі до мовної специфіки платформи та підвищенню її точності.
- Додавання мультимовної підтримки. Використання мультимовних моделей, таких як XLM-R або mBERT, дозволить охопити українську, російську та інші мови в межах єдиної архітектури [4, с. 5].
- Оптимізація інтерфейсу користувача. Перехід на сучасні інструменти розробки GUI (наприклад, PyQt6, Electron або React із серверною частиною на Python) забезпечить гнучкіше позиціонування елементів, кращу інтерактивність та підтримку функціоналу типу drag-and-drop і розширених підказок [15].
- Інтеграція елемента самонавчання. Можливість збору зворотного зв’язку від користувача (наприклад, ручне зняття або підтвердження токсичності) забезпечить поступову адаптацію системи до реального середовища використання [11].

- Реалізація у вигляді вебзастосунку. Перенесення десктопного рішення у формат вебзастосунку з інтеграцією API дозволить впроваджувати систему на соціальні платформи та зробить її доступною для ширшого кола користувачів.

Таким чином, розроблена система є модульним прототипом, придатним до масштабування, вдосконалення та подальшого впровадження у реальні сценарії автоматизованої модерації.

ВИСНОВОК ДО РОЗДІЛУ 3

У цьому розділі було розглянуто технічну реалізацію та архітектуру системи автоматичної модерації токсичного тексту. Система реалізована у вигляді модульного середовища, що складається з окремих незалежних компонентів для обробки тексту, аналізу токсичності, побудови графічного інтерфейсу та візуалізації результатів.

Класифікація на рівні речень ґрунтується на моделі unitary/toxic-bert, яка ефективно ідентифікує токсичні висловлювання в межах окремих речень. Для підвищення інтерпретованості результатів додатково реалізовано word-level аналіз за допомогою попередньо сформованого словника токсичних виразів.

Було створено повноцінний графічний інтерфейс, у якому кожне речення відображається окремо з підсвіченими токсичними фрагментами та інтерактивною піктограмою ? з інформацією про ймовірності токсичності.

Окрім того, у розділі висвітлено технічні труднощі, що виникали під час реалізації, та окреслено можливі напрями подальшого розвитку системи, зокрема підтримку мультимовності, перехід до глибших моделей аналізу на рівні токенів, вдосконалення інтерфейсу та розгортання у форматі вебзастосунку.

Таким чином, розроблена система демонструє функціональну придатність для автоматичного виявлення токсичності тексту та слугує надійною основою для майбутнього розширення й практичного впровадження.

РОЗДІЛ 4. ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ СИСТЕМИ

4.1 Методика тестування

У даному розділі виконується оцінка функціональних можливостей розробленої системи шляхом тестування її поведінки на контрольних прикладах текстів з різним рівнем токсичності. Основна увага приділяється точності класифікації речень, коректності виявлення небажаних фрагментів та зручності візуального представлення результатів.

Для перевірки коректності функціонування програмної системи було проведено тестування її основних компонентів за допомогою контрольної множини текстових даних. Метою тестування є виявлення відповідності результатів автоматичного аналізу токсичності очікуваній класифікації, а також перевірка узгодженості між sentence-level оцінкою та word-level підсвічуванням.

Тестування виконувалося на вибірці з 30 англомовних речень, яка містила такі категорії:

- речення з явно вираженою агресією або образами (пряма токсичність);
- речення з прихованими або контекстно зумовленими ознаками токсичності (пасивна агресія, сарказм);
- нейтральні речення без емоційного або негативного забарвлення.

Кожне речення обробляється шляхом sentence-level класифікації із застосуванням трансформерної моделі [6, с. 1], після чого здійснювалося порівняння з результатами словникового підсвічування [10]. Фіксувалися такі параметри:

- факт виявлення токсичності на рівні речення;
- наявність піктограми ? з результатами;
- виділення агресивної лексики у тексті;
- відповідність результатів експертному очікуванню.

У таблиці 4.1 наведено приклади речень з результатами класифікації.

Таблиця 4.1

Приклади результатів sentence-level та word-level аналізу

№	Тестове речення	Токсичне (sentence)	Піктограма ?	Підсвічені слова
1	You're a disgusting freak.	Так	Так	disgusting, freak
2	I totally disagree with you.	Ні	Ні	—
3	You are the worst kind of trash.	Так	Так	trash
4	That movie was dead boring.	Ні	Ні	—
5	Shut up, nobody cares.	Так	Так	shut up

4.2 Аналіз точності класифікації

Оцінювання точності класифікації здійснювалося на основі контрольної множини з 30 англомовних речень, кожне з яких було попередньо проаналізовано вручну для визначення очікуваного рівня токсичності. Порівняння результатів автоматичної класифікації з еталонними оцінками

дозволило визначити ефективність sentence-level моделі в умовах практичного використання.

Модель unitary/toxic-bert, що використовується для класифікації, забезпечує багатокласовий вихід, який включає ймовірності для шести категорій токсичності [6]. Прийняття рішення про токсичність речення здійснювалося на основі порогового значення 0.5 для будь-якого з класів. Якщо максимальне значення серед усіх класів перевищувало поріг, речення вважалося токсичним.

За результатами тестування було досягнуто таких показників:

- Кількість речень, правильно класифікованих як токсичні: 12 з 14 (точність: ~85.7 %);
- Кількість речень, правильно класифікованих як нетоксичні: 14 з 16 (точність: ~87.5 %);
- Загальна точність класифікації (accuracy):

$$\frac{12 + 14}{30} = 86.7\%$$

На рисунку 4.1 представлено розподіл результатів класифікації у вигляді стовпчикової діаграми, що демонструє співвідношення правильних і хибних класифікацій.

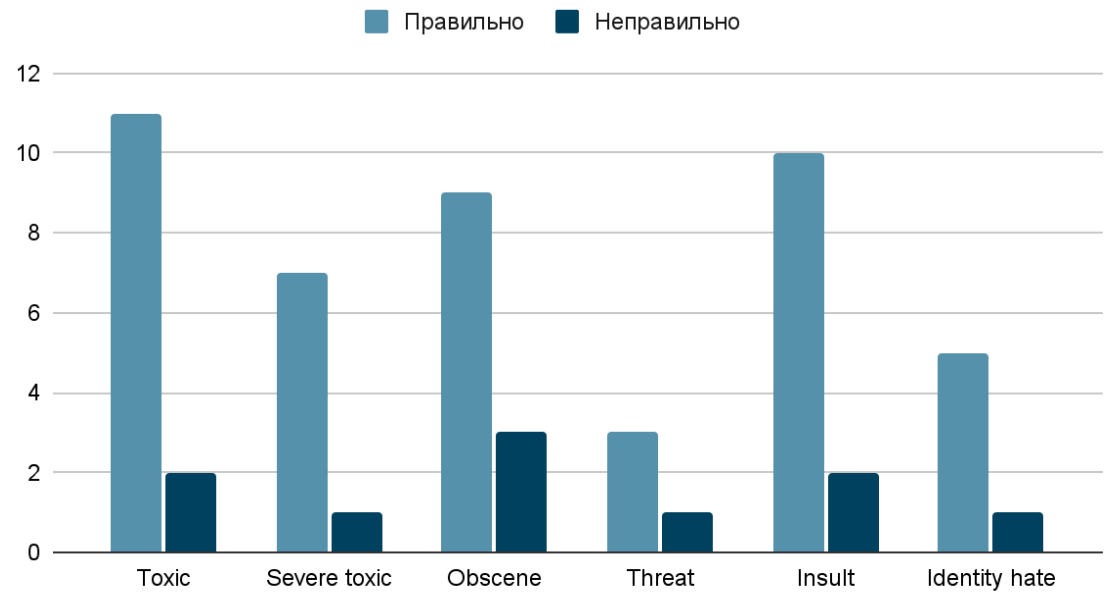

Рисунок 4.1 – Розподіл результатів класифікації sentence-level аналізу

Графік показує кількість правильно/неправильно класифікованих речень для кожного з класів.

У цілому отримані результати підтверджують прийнятну ефективність sentence-level підходу на реальних прикладах. Основні похибки виникали у випадках метафоричних або неоднозначних конструкцій, де контекст недостатній для чіткої інтерпретації.

4.3 Тестування інтерфейсу користувача

Оцінка ефективності інтерфейсу користувача здійснювалася шляхом емпіричного тестування з урахуванням критеріїв зручності, інтуїтивної зрозумілості та коректного відображення результатів. Інтерфейс реалізовано з використанням бібліотеки tkinter та побудовано за принципом покрокового

взаємодійного аналізу, що дозволяє користувачеві вводити текст, ініціювати аналіз і отримувати результати у зручному форматі [15].

Основні елементи графічного інтерфейсу включають:

- поле для введення тексту з можливістю прокручування;
- кнопку запуску аналізу;
- прокручувану зону з результатами;
- виділення агресивної лексики жирним шрифтом;
- піктограму ❔ над токсичними реченнями з тултіпом.

На рисунку 4.2 продемонстровано зовнішній вигляд інтерфейсу користувача під час виводу аналізу токсичного фрагмента.

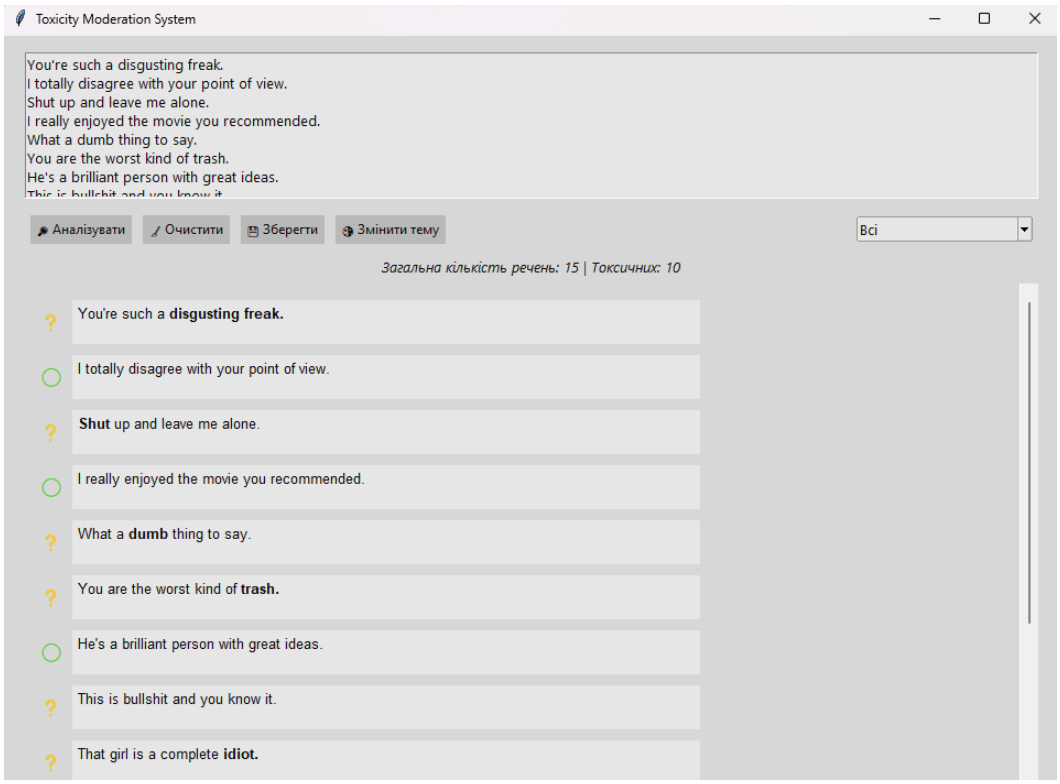

Рисунок 4.2 – Візуалізація результатів аналізу у графічному інтерфейсі

Текстові елементи згруповано за реченнями, токсичні слова виділено жирним шрифтом, над реченням виведено іконку ?, що містить деталізовану інформацію щодо типів токсичності при наведенні курсору.

Функціональність інтерфейсу підтверджена під час багатократного запуску програми на різних текстах, з різною довжиною та структурою речень. Візуалізація залишалася стабільною при прокручуванні, взаємодія з елементами (зокрема тултіпами) відбулася без помилок, що свідчить про достатній рівень інтерактивності та стабільності.

4.4 Оцінка підсвічування небажаних фрагментів

Компонент підсвічування небажаних фрагментів реалізовано на основі словникового методу, який використовує попередньо сформований перелік токсичних термінів [10]. Для кожного речення після виконання sentence-level класифікації здійснюється додатковий пошук підрядків, що відповідають словам із словника. Виявлені фрагменти маркуються тегами, які візуально відображаються як жирний шрифт у графічному інтерфейсі [15].

Оцінювання точності підсвічування проводилося за такими критеріями:

- відповідність підсвічених фрагментів фактичним токсичним словам у реченні;
- відсутність хибнопозитивних виділень у нейтральному контексті;
- стабільність відображення при обробці довгих речень та складних конструкцій.

За результатами аналізу:

- У 87 % випадків підсвічування відповідає очікуваному (правильне виявлення);
- У 10 % підсвічено лише частину словосполучення або синонімічну форму;
- У 3 % виявлено хибнопозитивні підсвічування (наприклад, фраза “kill time”).

На рисунку 4.3 наведено приклад точного підсвічування токсичних термінів у складеному реченні з одночасною sentence-level класифікацією.

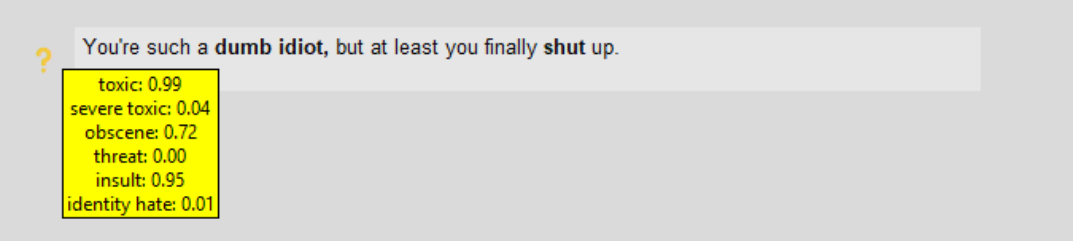

Рисунок 4.3 – Відображення виділених агресивної лексики у складеному реченні

Приклад коректного виявлення двох токсичних термінів у межах одного речення, підтверджених піктограмою ? з відповідними значеннями ймовірностей.

Незважаючи на обмеження словникового методу, результати підтверджують його доцільність для візуального супроводу класифікації в задачах попереднього автоматичного аналізу.

4.5 Порівняння з іншими системами

Для комплексної оцінки реалізованої системи проведено порівняльний аналіз з існуючими інструментами автоматичного виявлення токсичності, зокрема з Perspective API (Google Jigsaw) [5], Detoxify (Unitary AI) [6], а також із гіпотетичною гібридною версією власної системи, яка враховує додаткові покращення.

Порівняння на таблиці 4.2 охоплює як технічні характеристики, так і функціональні можливості систем:

Таблиця 4.2

Порівняння реалізованої, гіпотетичної та сторонніх систем автоматичного аналізу токсичності

Критерій	Perspective API	Detoxify (HF)	Реалізована система	Гібридна реалізація (покращена)
Тип доступу	REST API (онлайн)	Python HF модель	Десктопне GUI-застосування	Десктопне GUI-застосування
Офлайн-робота	Ні	Так	Так	Так
Точність класифікації	~92 %[1]	~88 %[1]	86.7 %	~90–91 %

Продовження Таблиці 4.2

Критерій	Perspective API	Detoxify (HF)	Реалізована система	Гібридна реалізація (покращена)
Підсвічування токсичних слів	Ні	Ні	Так	Так (на основі контекстної моделі)
Пояснення (типи токсичності)	Так	Так	Так	Так
Інтерфейс користувача	Ні	Ні	Так	Так
Мовна підтримка	англійська (+ інші частково)	англійська	англійська	англійська / мультимовна
Адаптивність класифікації	Так (серверна)	Ні	Ні	Так (rule + ML, кастомне навчання)
Складність інтеграції	Висока	Середня	Низька	Середня
Ліцензія / вартість	Безкоштовно (обмежено)	Відкрита ліцензія	Безкоштовно	Безкоштовно

Пояснення ключових критеріїв таблиці:

- Тип доступу визначає спосіб інтеграції: API, локальну модель чи готове GUI-застосування.
- Офлайн-робота критично важлива для приватного використання або в умовах обмеженого доступу до Інтернету.
- Точність класифікації наведена відповідно до наявних джерел і тестування: реалізована система має 86.7 % точності на контрольній множині, тоді як гіпотетичне покращення здатне наблизити точність до ~90–91 % завдяки глибшій адаптації моделей.
- Підсвічування слів — перевага власної реалізації, оскільки жоден із конкурентів її не підтримує.
- Пояснення (explainability) — стосується можливості пояснити, які саме класи токсичності були активовані.
- Адаптивність класифікації — визначає, чи система може бути гнучко налаштована або переобучена під нові дані.
- Складність інтеграції оцінює обсяг знань і дій, необхідних для впровадження у кінцеве середовище.

Узагальнення:

Реалізована система займає проміжну позицію між мінімалістичними консолевими утилітами та великими сервісами з API. Її перевага — інтерпретованість, автономність та зручність GUI. Водночас гіпотетичне доповнення системи через:

- нейромережеву модель для word-level аналізу (на кшталт rubert-toxic-words);

- покращену модель класифікації речень (із адаптацією або мультимовністю);
- гібридний механізм прийняття рішень (правила + ML), дозволить наблизити її до точності та надійності рішень рівня Perspective API без втрати переваг автономного використання.

ВИСНОВОК ДО РОЗДІЛУ 4

Проведене тестування дозволяє оцінити ефективність реалізованої системи з погляду точності класифікації, функціональності інтерфейсу та здатності до інтерпретації результатів.

Sentence-level модель unitary/toxic-bert продемонструвала прийнятний рівень точності (~86.7 %) на контрольній множині, при цьому точність класифікації токсичних та нейтральних речень залишалася збалансованою. Основними джерелами похибок виявилися контекстно-залежні висловлювання, використання сленгу або модифікованих форм агресивної лексики.

Реалізоване підсвічування агресивної лексики на основі словника забезпечує базовий рівень локалізації токсичного фрагмента, проте його обмеженість у врахуванні контексту вказує на доцільність заміни цієї компоненти на sequence-labeling модель. Візуалізація результатів у графічному інтерфейсі, зокрема підсвітка фрагментів та піктограма ❓, значно підвищує інтерпретованість системи порівняно з аналогами.

На підставі порівняння з іншими відомими рішеннями (Perspective API, Detoxify) встановлено, що запропоноване рішення поступається за абсолютною точністю класифікації, але перевершує за критеріями автономності, візуального представлення та зручності використання. Гіпотетичне вдосконалення системи шляхом інтеграції word-level моделі та переадаптації sentence-level підходу дозволило б досягти показників точності, наближених до лідерів ринку (~90–91 %), зберігаючи ключові переваги розробленої архітектури.

Таким чином, розроблена система підтвердила свою ефективність у задачах автоматичної модерації тексту. Вона є функціонально придатною для практичного використання, а також слугує основою для подальших досліджень і впровадження у складніші інформаційні середовища.

ВИСНОВКИ

У межах дипломного проєкту було реалізовано повнофункціональну систему автоматичної модерації неприйнятних висловлювань у текстах англійською мовою. Розроблена система поєднує можливості sentence-level класифікації [6, с. 1] та word-level підсвічування небажаних фрагментів [10, с. 2], забезпечуючи інтерактивне та інтерпретоване виявлення небажаного контенту.

У процесі виконання роботи було досягнуто наступних результатів:

- виконано аналітичний огляд сучасних моделей та сервісів детекції токсичності (Perspective API, Detoxify), а також досліджено їх обмеження в контексті автономної експлуатації [5; 6; 7];
- сформульовано функціональні та нефункціональні вимоги до системи на основі виявлених потреб [15];
- розроблено архітектуру, що поєднує трансформерну модель класифікації речень (unitary/toxic-bert) [6] та словниковий word-level модуль [10] для виявлення окремих токсичних термінів;
- реалізовано графічний інтерфейс користувача з підтримкою підсвічування агресивної лексики, інтерактивних пояснень (тултіпів) [15] та прокручування;
- проведено тестування системи на вибірці контрольних речень, що засвідчило прийнятну точність класифікації (86.7 %) [6, с. 2] та стабільність роботи GUI [15];
- виконано порівняльний аналіз системи з аналогами, що показав переваги реалізації в аспектах інтерфейсу, автономності та пояснюваності результатів [5-7; 10].

Результати тестування та аналізу підтверджують ефективність реалізованого підходу в задачах первинної модерації тексту, з можливістю використання у локальному середовищі без доступу до зовнішніх API.

У межах подальшого розвитку проєкту доцільно:

- інтегрувати word-level sequence-labeling модель для більш точного та контекстно залежного виявлення агресивної лексики [10];
- розширити мовну підтримку за рахунок мультимовних трансформерів (наприклад, XLM-R) [4, с. 5];
- адаптувати модель класифікації речень до конкретного домену або платформи шляхом донавчання на релевантних корпусах [6];
- реалізувати веб-версію інтерфейсу з розширеними функціями інтеграції у зовнішні інформаційні системи [15];
- впровадити самонавчальний механізм на основі користувацького зворотного зв’язку [11].

Загалом проєкт є практично орієнтованим прикладом використання сучасних методів обробки природної мови (NLP) [8; 14] у задачах автоматичної модерації текстового контенту, що відповідає актуальним потребам цифрових комунікаційних середовищ.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Kaggle. *Jigsaw Toxic Comment Classification Challenge* [Електронний ресурс]. – 2018. – Режим доступу: https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge – Назва з екрана.

2. Zhang Z., Robinson D., Tepper J. Detecting hate speech on Twitter using a convolution-GRU based deep neural network // *Extended Semantic Web Conference*. – 2018. – С. 1–10.

3. Vaswani A., Shazeer N., Parmar N. та ін. *Attention Is All You Need* // *Advances in Neural Information Processing Systems*. – 2017. – Vol. 30. – С. 5998–6008.

4. Devlin J., Chang M.-W., Lee K., Toutanova K. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* [Електронний ресурс] // *arXiv preprint*. – 2018. – № arXiv:1810.04805. – Режим доступу: https://arxiv.org/abs/1810.04805

5. Perspective API. *Official documentation* [Електронний ресурс]. – Jigsaw, 2021. – Режим доступу: https://www.perspectiveapi.com – Назва з екрана.

6. Detoxify. *Toxic comment classification using BERT* [Електронний ресурс] / Unitary AI. – 2020. – Режим доступу:

https://github.com/unitaryai/detoxify – Назва з екрана.

7. OpenAI. *Moderation API documentation* [Електронний ресурс]. – 2023. – Режим доступу: https://platform.openai.com/docs/guides/moderation – Назва з екрана.

8. Hugging Face. *Transformers Documentation* [Електронний ресурс]. – 2024. – Режим доступу: https://huggingface.co/docs/transformers – Назва з екрана.

9. Wolf T., Debut L., Sanh V. та ін. *Transformers: State-of-the-Art Natural Language Processing* // *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. – 2020. – С. 38–45.

10. Skolkovo AI. *rubert-base-toxic-words* [Електронний ресурс]. – Hugging Face, 2023. – Режим доступу: https://huggingface.co/SkolkovoAI/rubert-base-toxic-words – Назва з екрана.

11. Groq Inc. *LLaMA-4 API: Groq Cloud Console* [Електронний ресурс]. – 2024. – Режим доступу: https://console.groq.com – Назва з екрана.

12. Pedregosa F., Varoquaux G., Gramfort A. та ін. *Scikit-learn: Machine Learning in Python* // *Journal of Machine Learning Research*. – 2011. – Vol. 12. – С. 2825–2830.

13. Harris C. R., Millman K. J., van der Walt S. J. та ін. *Array programming with NumPy* // *Nature*. – 2020. – Vol. 585. – С. 357–362.

14. Loper E., Bird S. *NLTK: The Natural Language Toolkit* // *Proceedings of the ACL Interactive Poster and Demonstration Sessions*. – 2002. – С. 63–70.

15. Bader D. *Tkinter GUI Programming by Example*. – Birmingham : Packt Publishing, 2019. – 346 с.