ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка програмного забезпечення для визначення емоційного забарвлення відгуків про товар або послугу»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень.
Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Олександр ОМЕЛЯНЧУК
(підпис)

Виконав: здобувач вищої освіти групи ПД-41
Олександр ОМЕЛЯНЧУК

Керівник: Богдан ХУДІК
доктор філософії (PhD)

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Омелянчуку Олександру Володимировичу

1. Тема кваліфікаційної роботи: «Розробка програмного забезпечення для для визначення емоційного забарвлення відгуків про товар або послугу»
керівник кваліфікаційної роботи доктор філософії (PhD) Богдан ХУДІК,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 5

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про методи обробки природньої мови, технічна документація до бібліотек.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

1. Обґрунтування актуальності теми, мета та завдання роботи, об'єкт, предмет та методи дослідження, наукова новизна та практичне значення.
2. Огляд сентимент-аналізу, існуючих методів та інструментів, особливості аналізу україномовних текстів, огляд аналогічних програмних рішень.
3. Обґрунтування вибору засобів розробки (мова програмування, бібліотеки), проектування архітектури системи, опис алгоритмів попередньої обробки тексту та класифікації тональності.
4. Реалізація програмного забезпечення.
5. Опис тестового набору даних, методика проведення експериментів, аналіз отриманих результатів (оцінка точності, повноти, F1-міри), порівняння з аналогами.

5. Перелік графічного матеріалу: *презентація*

1. Блок-схеми алгоритмів роботи програмного забезпечення.
2. Діаграма архітектури розробленої системи.
3. Графіки результатів експериментів: наприклад, матриця плутанини, графіки точності та функції втрат під час навчання моделі, діаграми розподілу тональностей у наборі даних.
4. Скріншоти інтерфейсу користувача розробленого програмного забезпечення.
5. Приклади візуалізації результатів аналізу (наприклад, хмари слів для різних тональностей, стовпчасті діаграми).
6. Діаграми прецедентів (Use Case) для опису взаємодії користувача з системою.

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02.-04.03.2025	
2	Аналіз та дослідження існуючих аналогів	05.03-13.03.2025	
3	Провести аналіз актуальності задач сентимент-аналізу у сфері електронної комерції та сервісного обслуговування.	14.03-15.03.2025	
4	Дослідити аналогічні програмні рішення та системи.	16.03-18.03.2025	
5	Розробити функціональні та нефункціональні вимоги до системи.	19.03-20.03.2025	
6	Побудувати діаграми та моделі, що описують логіку роботи ПЗ.	21.03-25.03.2025	
7	Обрати технології для реалізації проєкту.	26.03-30.03.2025	
8	Зібрати корпус відгуків для навчання та тестування системи.	01.04-14.04.2025	
9	Реалізувати модуль обробки природної мови.	15.04-22.04.2025	
10	Побудувати інтерфейс користувача для взаємодії із системою.	23.04-30.04.2025	
11	Протестувати точність та швидкодію системи.	01.05-03.05.2025	
12	Провести тестування на реальних відгуках.	04.05-10.05.2025	
13	Порівняти результати із існуючими рішеннями.	11.05-15.05.2025	
14	Оформлення роботи: вступ, висновки, реферат	29.04-11.05.2025	
15	Розробка демонстраційних матеріалів	12.05-18.05.2025	
16	Попередній захист роботи	19.05-30.05.2025	

Здобувач вищої освіти _____ Олександр ОМЕЛЯНЧУК

(підпис)

Керівник

кваліфікаційної роботи _____ Богдан ХУДІК

(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 68 стор., 5 табл., 3 рис., 20 джерел.

Мета роботи – автоматизація обробки текстових відгуків.

Об'єкт дослідження – процес автоматизованого сентимент аналізу текстової інформації.

Предмет дослідження – методи та алгоритми визначення емоційного забарвлення на основі обробки природної мови.

Короткий зміст роботи: У роботі проаналізовано сучасні алгоритми та методи обробки природної мови в контексті задачі визначення емоційного забарвлення текстових відгуків про товари або послуги, включаючи підходи на основі машинного навчання та глибокого навчання. Досліджено інструментальні засоби мови програмування Python, зокрема бібліотеки NLTK, Transformers, PyTorch, PyQT6 для реалізації системи сентимент-аналізу. Розроблено архітектуру та алгоритм роботи програмного застосунку. Програмно реалізовані ключові функціональні можливості, зокрема: завантаження текстових відгуків, їх попередня обробка для української мови та класифікація відгуків за емоційним забарвленням (позитивне, негативне, нейтральне). Розроблено програмний інтерфейс для взаємодії з користувачем. Проведено експериментальну оцінку якості розробленого програмного забезпечення з використанням стандартних метрик. У роботі використано мову програмування Python, бібліотеки NLTK, Transformers, PyTorch, PyQT6 та Matplotlib для обробки текстів, реалізації алгоритмів машинного навчання та візуалізації результатів.

Сферою використання розробленого застосунку є автоматизований аналіз клієнтських відгуків для потреб бізнесу, маркетингових досліджень, моніторингу та покращення репутації бренду.

КЛЮЧОВІ СЛОВА: СЕНТИМЕНТ-АНАЛІЗ, ОБРОБКА ПРИРОДНОЇ МОВИ, PYTHON, NTKL, КЛАСИФІКАЦІЯ ТЕКСТІВ, ТОКЕНІЗАЦІЯ, ТОНАЛЬНІСТЬ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – Програмне Забезпечення

IDE – Integrated Development Environment

ML – Machine Learning

NLP – Natural Language Processing

BERT – Bidirectional Encoder Representations from Transformers

UBERT – Ukrainian Bidirectional Encoder Representations from Transformers

GUI – Graphical User Interface

API – Application Programming Interface

ВСТУП

У сучасному цифровому світі обсяг текстової інформації, генерованої користувачами, стрімко зростає. Особливе місце серед цих даних займають відгуки про товари та послуги, які публікуються на веб-сайтах, у соціальних мережах та на спеціалізованих платформах. Ці відгуки є надзвичайно цінним джерелом інформації для бізнесу, оскільки дозволяють зрозуміти ставлення споживачів до продуктів, виявити сильні та слабкі сторони, а також оперативно реагувати на потреби ринку. Щоденна поява величезної кількості нових відгуків робить їх ручний аналіз малоефективним та трудомістким. У зв'язку з цим виникає необхідність в автоматизованих інструментах для обробки та аналізу цих даних.

Аналіз емоційного забарвлення тексту, відомий також як сентимент-аналіз, є галуззю комп'ютерної лінгвістики та обробки природної мови (NLP), що фокусується на автоматичному визначенні емоційного тону (позитивного, негативного, нейтрального) або конкретних емоцій (радість, сум, гнів тощо), виражених у тексті. Такі системи дозволяють компаніям ефективно відстежувати репутацію бренду, аналізувати сприйняття маркетингових кампаній, покращувати якість обслуговування клієнтів та приймати обґрунтовані бізнес-рішення.

Актуальність даної роботи посилюється зростаючим попитом на інструменти аналізу текстів українською мовою. Незважаючи на значний прогрес у галузі NLP, кількість ресурсів, таких як розмічені корпуси текстів, спеціалізовані словники та готові програмні рішення для української мови, все ще обмежена порівняно з англійською та іншими поширеними мовами. Розробка програмного забезпечення для аналізу емоційного забарвлення саме україномовних відгуків сприятиме заповненню

цієї ніші та надасть українським підприємствам і дослідникам ефективний інструмент для роботи з текстовими даними.

Метою роботи є автоматизація обробки текстових відгуків.

Об'єктом дослідження є процес автоматизованого сентимент аналізу текстової інформації.

Предметом дослідження є методи та алгоритми визначення емоційного забарвлення на основі обробки природної мови.

Методи дослідження: для досягнення поставленої мети необхідно вирішити наступні завдання:

1. Провести аналіз існуючих рішень, для визначення емоційного забарвлення відгуків.
2. Провести аналіз науково-технічної літератури та існуючих підходів до визначення емоційного забарвлення тексту.
3. Дослідити особливості обробки та аналізу текстів українською мовою.
4. Скласти стек технологій реалізації застосунку, що будуть відповідати вимогам до ПЗ.
5. Розробити алгоритм та програмне забезпечення для збору, попередньої обробки та аналізу емоційного забарвлення україномовних відгуків.
6. Провести експериментальну оцінку ефективності розробленого ПЗ на наборі реальних відгуків.

Наукова новизна роботи полягає у розробці програмного продукту, адаптованого для аналізу емоційного забарвлення саме україномовних відгуків, що враховує специфіку мови та використовує сучасні підходи машинного навчання та обробки природної мови. Робота сприяє розвитку практичних інструментів для NLP українською мовою.

Практичне значення отриманих результатів полягає у практичному застосуванні даного ПЗ у різних сферах, зокрема: маркетинг та бізнес-аналітика, соціологічні дослідження та розробка систем підтримки клієнтів.

1 ТЕОРЕТИЧНІ ОСНОВИ АНАЛІЗУ ЕМОЦІЙНОГО ЗАБАРВЛЕННЯ ТЕКСТУ

1.1. Загальне поняття про аналіз емоційного забарвлення (сентимент-аналіз)

Аналіз емоційного забарвлення тексту, також відомий як сентимент-аналіз або аналіз тональності, є класом методів контент-аналізу в комп'ютерній лінгвістиці, призначених для автоматизованого виявлення в текстах емоційно забарвленої лексики та оцінки ставлення автора до об'єктів, про які йдеться мова. Це процес дослідження даних, що зберігаються в Інтернеті, з метою виявлення та категоризації думок, виражених у частині тексту. Основна мета сентимент-аналізу полягає у визначенні полярності вираженого в тексті настрою, який може бути класифікований як позитивний, негативний або нейтральний. Крім того, сентимент-аналіз може бути спрямований на виявлення конкретних почуттів та емоцій, таких як злість, радість, сум тощо.

Стрімкий розвиток сентимент-аналізу як галузі досліджень у комп'ютерних науках значною мірою пов'язаний із появою великої кількості суб'єктивних текстів в інтернет-середовищі. Щодня в мережі з'являється величезна кількість відгуків, коментарів та дописів у соціальних мережах, що робить ручний аналіз цієї інформації малоефективним або й зовсім неможливим. Автоматизований аналіз цих великих обсягів даних дозволяє організаціям приймати рішення на їх основі.

Сентимент-аналіз знаходить застосування в багатьох сферах, зокрема в бізнесі, політиці, медицині та освіті. Компанії використовують його для відстеження ставлення до бренду або продукту у відгуках клієнтів та дописах у соціальних мережах, що допомагає краще розуміти потреби споживачів та відповідно коригувати маркетингові стратегії. В економіці сентимент-аналіз стає все більш важливим,

пропонуючи цінні відомості та потенційно покращуючи економічне прогнозування та прийняття рішень.

1.2. Класифікація емоцій та моделі їх представлення в психології та лінгвістиці

Розуміння та класифікація емоцій є фундаментальним аспектом для розробки систем сентимент-аналізу. Психологія пропонує декілька моделей для систематизації емоційних станів.

Одним з поширених підходів є категоріальний підхід, який визначає невелику кількість базових емоцій, що вважаються універсальними для всіх людських культур. Психолог Пол Екман у 1970-х роках ідентифікував шість основних емоцій: щастя, смуток, відраза, страх, здивування та гнів. Пізніше він розширив цей перелік, включивши такі емоції, як гордість, сором, збентеження та хвилювання. У цьому підході кожен емоційний стан класифікується в окрему категорію. Деякі дослідження для виявлення емоцій з тексту обирають набір з шести емоцій, наприклад: радість, смуток, гнів, страх, любов та здивування. Інша класифікація, заснована на фундаментальних моделях, виділяє десять базових емоцій (гнів, очікування, недовіра, страх, щастя, радість, любов, смуток, здивування, довіра) та 56 вторинних емоцій.

Іншим важливим підходом є дименсійний підхід, який описує емоції за допомогою декількох загальних параметрів. Найчастіше використовуються два виміри: валентність (від приємного/позитивного до неприємного/негативного) та збудження (від високого рівня активації до низького/апатії). Прикладом такої моделі є колірне коло емоцій Роберта Плутчика, яке ілюструє первинні, вторинні та третинні діади, що описують відстань між емоціями. Циркумплексна модель емоцій Рассела також представляє базові емоції у двовимірному просторі відносно валентності та збудження, що дозволяє визначити бажану емоцію та оцінити її інтенсивність, аналізуючи лише ці два виміри.

Існує також компонентна модель оцінки, яка намагається виявити емоції на основі оцінок або інтерпретацій мови, тексту чи подій. Вона визначає емоції як збалансовану реакцію на події, агентів та об'єкти.

Для автоматичного аналізу та прогнозування емоцій у текстах, особливо коротких, таких як пости чи коментарі в соціальних мережах, найчастіше використовуються категоріальний та дименсійний підходи.

У лінгвістиці досліджується, як саме емоції вербалізуються, тобто знаходять своє вираження у мові. Аналізуються лексичні одиниці, фразеологізми, синтаксичні конструкції та інші мовні засоби, що передають емоційний стан мовця. Розуміння лінгвістичних маркерів емоцій є важливим для створення ефективних систем сентимент-аналізу.

1.3. Огляд існуючих підходів та методів сентимент-аналізу

Існують три основні підходи до аналізу тональності тексту: системи на основі правил, автоматичні системи (що використовують машинне навчання) та гібридні системи.

1.3.1. Методи на основі правил та лексиконів

Системи на основі правил виконують аналіз тональності на основі набору правил, розроблених вручну. Ці правила можуть використовувати різні вхідні дані, включаючи класичні методи обробки природної мови (NLP), такі як токенізація, розмітка частин мови, синтаксичний розбір, а також лексикони (списки слів та виразів із заздалегідь визначеною тональністю). Прикладом такої системи є підрахунок кількості позитивних та негативних слів у тексті та визначення загальної тональності на основі їхнього співвідношення. Перевагою таких систем є їх прозорість та можливість точного налаштування. Однак, створення та підтримка таких систем може

бути дуже складним та трудомістким процесом, а додавання нових правил може мати небажані наслідки.

1.3.2. Методи машинного навчання

Автоматичні системи використовують методи машинного навчання (ML) для навчання на даних замість використання правил, створених вручну. Завдання аналізу тональності зазвичай моделюється як проблема класифікації, де класифікатору подається текст, і він повертає відповідну категорію (наприклад, позитивну, негативну або нейтральну).

Першим кроком у цьому підході є перетворення тексту в числове представлення, зазвичай вектор, де кожен компонент представляє частоту слова або виразу у попередньо визначеному словнику. Для цього можуть використовуватися різні методи, такі як "мішок слів" (Bag of Words) або TF-IDF (Term Frequency-Inverse Document Frequency).

На етапі класифікації використовуються статистичні моделі, такі як Наївний Баєсівський класифікатор (Naive Bayes), Логістична Регресія (Logistic Regression), Метод Опорних Векторів (Support Vector Machines, SVM) або дерева рішень. Ці алгоритми навчаються на великих наборах даних, де тексти вже розмічені за тональністю.

1.3.3. Методи глибокого навчання

Останнім часом методи глибокого навчання (Deep Learning, DL) набули значної популярності в аналізі сентименту завдяки їх здатності автоматично вивчати складні закономірності з сирих текстових даних. До найпоширеніших архітектур глибокого навчання, що використовуються для цієї задачі, належать:

- Згорткові нейронні мережі (Convolutional Neural Networks, CNN): CNN добре підходять для виявлення локальних патернів у тексті, таких як словосполучення, які можуть вказувати на певний сентимент.

- Рекурентні нейронні мережі (Recurrent Neural Networks, RNN) та їх варіанти (LSTM, GRU): RNN, особливо Довготривала Короткочасна Пам'ять (Long Short-Term Memory, LSTM) та Керовані Рекурентні Блоки (Gated Recurrent Unit, GRU), ефективно обробляють послідовності даних, такі як текст, враховуючи контекст слів у реченні.
- Трансформери (Transformers): Архітектура трансформерів, особливо такі моделі, як BERT (Bidirectional Encoder Representations from Transformers), RoBERTa, T5 та XLNet, досягли найвищих результатів у багатьох задачах обробки природної мови, включаючи аналіз сентименту. Ці моделі використовують механізми само-уваги (self-attention) для розуміння залежностей між словами в реченні та можуть бути попередньо навчені на великих обсягах текстових даних, а потім донавчені для конкретної задачі аналізу сентименту.

1.3.4. Гібридні підходи

Гібридні системи поєднують підходи на основі правил та автоматичні методи машинного навчання. Це дозволяє використовувати переваги обох підходів, наприклад, точність правил для певних випадків та здатність ML-моделей узагальнювати на нових даних.

1.4. Особливості сентимент-аналізу україномовних текстів

Аналіз тональності текстів українською мовою є актуальним напрямком досліджень, що має свої специфічні виклики та особливості. Однією з головних проблем є обмеженість ресурсів порівняно з англійською чи іншими поширеними мовами. Це стосується, зокрема, кількості та якості доступних розмічених корпусів текстів, спеціалізованих тональних словників для української мови. Розробка таких ресурсів є важливим завданням для покращення якості сентимент-аналізу.

Незважаючи на ці виклики, проводяться дослідження та створюються набори даних для аналізу тональності українською мовою. Наприклад, існують роботи, присвячені формуванню датасетів українською мовою та побудові моделей для аналізу емоційного забарвлення тексту. Також розробляються тональні словники для української мови.

Для аналізу україномовних текстів можуть використовуватися як багатомовні попередньо навчені моделі, так і моделі, спеціально навчені або донавчені на українських даних. Багатомовні моделі, такі як BERT (наприклад, bert-base-multilingual-uncased), показали свою ефективність для української мови. Водночас, ведуться роботи зі створення та донавчання монолінгвальних моделей для української мови, таких як Ukr-RoBERTa, Ukr-ELECTRA та LiBERTa Large, які можуть демонструвати кращі результати на специфічних україномовних завданнях.

Дослідники експериментують з різними алгоритмами машинного навчання для української мови, включаючи логістичну регресію, яка показала хороші результати, та різні архітектури нейронних мереж. Важливим аспектом є також попередня обробка тексту, що включає токенізацію, та видалення стоп-слів, адаптованих до української мови.

1.5. Огляд існуючих програмних засобів для сентимент-аналізу

Ринок програмних рішень для аналізу тональності тексту є не сильно насиченим, однак пропонує різноманітні інструменти, від потужних хмарних API до спеціалізованих платформ моніторингу та кастомізованих сервісів. Для розуміння поточного стану технологій та визначення місця розроблюваного в рамках даної роботи програмного забезпечення, доцільно провести огляд кількох провідних комерційних інструментів.

1.5.1 Google Cloud Natural Language API

Google Cloud Natural Language API є частиною широкого набору хмарних сервісів Google Cloud Platform. Він надає розробникам доступ до попередньо навчених моделей машинного навчання для виконання різноманітних завдань обробки природної мови, включаючи аналіз тональності, виявлення сутностей, синтаксичний аналіз, класифікацію контенту, тощо.

Переваги:

- Висока точність;
- Масштабованість та надійність.

Недоліки:

- Користувачі покладаються на попередньо навчені моделі Google, що обмежує можливості тонкого налаштування під специфічні домени або для врахування вузькоспеціалізованої лексики.
- Внутрішня робота моделей є непрозорою для користувача, що ускладнює розуміння причин конкретних класифікаційних рішень.
- Хоча підтримка української мови є, загальні моделі можуть не завжди ідеально вловлювати специфічні культурні або мовні нюанси, які міг би врахувати спеціалізований, кастомізований підхід.

1.5.2 YouScan

YouScan – це платформа для моніторингу соціальних медіа та онлайн-ЗМІ, яка використовує технології штучного інтелекту, включаючи аналіз тексту. Однією з ключових функцій платформи є автоматичне визначення тональності згадок про бренди, продукти, конкурентів чи будь-які інші ключові слова.

Переваги:

- Спеціалізація на соціальних медіа.
- Зручний інтерфейс.

- Виявлення трендів та інфлюенсерів.

Недоліки:

- Вузька спрямованість. Передусім, це інструмент моніторингу, а не універсальний API для аналізу довільних текстів поза контекстом зібраних згадок.
- Хоча аналіз тональності є, можливості для глибокої кастомізації саме NLP-моделей користувачем обмежені порівняно з платформами, що дозволяють тренувати власні моделі.

1.5.3 MonkeyLearn

MonkeyLearn – це платформа на основі штучного інтелекту, яка дозволяє користувачам аналізувати текст за допомогою попередньо навчених моделей або створювати власні кастомні моделі машинного навчання без необхідності глибокого програмування (no-code/low-code підхід). Аналіз тональності є одним із популярних застосувань платформи.

Переваги:

- Кастомізація.
- Доступність для нетехнічних користувачів.
- Інтеграції з популярними інструментами.

Недоліки:

Якість залежить від даних користувача.

Хоча кастомізація можлива, досягнення високої точності для мов зі складною морфологією та синтаксисом, як українська, все одно вимагає ретельної підготовки даних та розуміння процесу навчання моделей.

1.5.4 Порівняльний аналіз та висновки

Кожен із цих інструментів має свою нішу та цільову аудиторію. Для завдань, що потребують глибокої інтеграції, високої точності для широкого спектру тем українською мовою та контролю над процесом обробки, або ж у випадках, коли використання комерційних API є економічно невигідним чи обмежує дослідницькі можливості, розробка власного програмного забезпечення, як це передбачено даною дипломною роботою, є обґрунтованим та актуальним підходом. Власний проект дозволяє сфокусуватися на специфіці аналізу саме україномовних відгуків про товари та послуги, потенційно врахувати тонші мовні нюанси, культурний контекст та специфічну лексику, що може бути недосяжним для універсальних комерційних систем або вимагатиме значних зусиль для кастомізації на платформах типу MonkeyLearn. Таким чином, розробка спеціалізованого ПЗ для української мови заповнює важливу нішу, особливо з огляду на зростаючу потребу в аналітичних інструментах для локального ринку.

В таблиці 1.1 зведено результати аналізу існуючих застосунків для визначення емоційного забарвлення відгуків та їх порівняння.

Таблиця 1.1

Порівняння програм-аналогів з розробленим ПЗ

	Google Cloud NL API	YouScan	MonkeyLearn	Ukrainian Sentiment Analyzer
Тип ПЗ	Веб-сервіс/API	SaaS (Веб-платформа)	Веб-сервіс/API	Десктоп
Підтримка української мови	+	-	+/-	+

Продовження таблиці 1.1

Порівняння програм-аналогів з розробленим ПЗ

Завантаження файлів для аналізу	-	-	+	+
Локальна обробка даних	-	-	-	+
Візуалізація результатів	-	+	+	+
Можливість роботи оффлайн	-	-	-	+

2 ЛІНГВІСТИЧНА СПЕЦИФІКА УКРАЇНСЬКОЇ МОВИ ЯК ФУНДАМЕНТ ДЛЯ NLP

Обробка природної мови досягла значних успіхів, проте більшість передових розробок історично були зосереджені на англійській мові. При застосуванні цих технологій до інших мов, зокрема до української, виникає низка унікальних викликів, зумовлених її фундаментальними лінгвістичними властивостями.

Українська мова, на відміну від аналітичної англійської, є синтетичною флективною мовою, що докорінно змінює підходи до побудови NLP-пайплайнів. Розуміння її морфологічного багатства, синтаксичної гнучкості та лексико-семантичних особливостей є не просто теоретичною вправою, а критичною передумовою для створення ефективних та точних мовних моделей і застосунків. Цей звіт надає вичерпний аналіз специфіки української мови для задач NLP, окреслює основні труднощі та пропонує стратегічні напрямки для дослідників і розробників.

2.1 Морфологічне багатство та його наслідки

Найбільш визначальною рисою української мови, що відрізняє її від англійської в контексті NLP, є її надзвичайно багата та складна морфологія. Ця характеристика є першопричиною більшості викликів, з якими стикаються розробники, і вимагає фундаментально інших підходів до попередньої обробки тексту та побудови моделей.

2.1.1 Українська як флективна синтетична мова: порівняльний аналіз з аналітичною англійською

Українська мова належить до синтетичного типу флективних мов. Це означає, що граматичні відношення між словами в реченні виражаються переважно за допомогою внутрішніх змін самих слів, а саме через додавання або зміну закінчень (флексій). Наприклад, роль іменника в реченні (підмет, додаток, обставина) визначається його відмінком, який, у свою чергу, позначається відповідним закінченням. Цей "синтетичний" підхід докорінно відрізняється від "аналітичного" підходу англійської мови, де граматичні зв'язки передаються здебільшого за допомогою порядку слів та службових частин мови, таких як прийменники та артиклі.

В англійському реченні "The student reads a book" зміна порядку слів на "A book reads the student" кардинально змінює зміст. В українській мові речення "Студент читає книгу" зберігає своє основне значення навіть при перестановці слів ("Книгу читає студент" або "Читає студент книгу"), оскільки знахідний відмінок слова "книгу" чітко вказує, що воно є об'єктом дії, незалежно від його позиції. Ця фундаментальна типологічна розбіжність є джерелом більшості подальших труднощів в автоматичній обробці української мови.

Таблиця 2.1

Порівняння ключових лінгвістичних характеристик української та англійської мов

Характеристика	Українська мова	Англійська мова
Тип мови	Синтетична, флективна	Аналітична
Морфологія	Високорозвинена інфлекційна система: 7 відмінків, 2 числа, 3 роди; дієслівні види, часи, способи	Слаборозвинена інфлекційна система

Продовження таблиці 2.1

Порівняння ключових лінгвістичних характеристик української та англійської мов

Порядок слів	Вільний	Відносно фіксований
Відмінкова система	Багата (7 відмінків: називний, родовий, давальний, знахідний, орудний, місцевий, кличний)	Рудиментарна (збереглася переважно у займенників: I/me, he/him)
Артиклі	Відсутні	Використовуються (означені та неозначені)
Роль прийменників	Менш критична, значення часто передається відмінковими закінченнями	Критично важлива для вираження граматичних зв'язків

2.1.2 Висока інфлекційність: розгалужені парадигми та їхній вплив

Багатство української морфології проявляється у величезній кількості словоформ для кожної лексеми (словникової одиниці). Якщо в англійській мові дієслово "see" має лише 5 форм (see, sees, saw, seen, seeing), то його український відповідник "бачити" може мати до 26 різних форм залежно від часу, особи, числа, роду тощо. Типовий іменник може мати 17-19 форм, а прикметник — 32-43 форми.

Така розгалужена система словозміни створює дві серйозні проблеми для NLP-моделей:

1. Лексичний вибух та розрідженість даних: Для комп'ютерної моделі кожна унікальна словоформа (наприклад, рука, руки, руці, рукою, руці, рук, рукам, руками, руках) є окремим токеном. Це призводить до експоненційного зростання розміру словника, який модель має вивчити. Як наслідок, кожна окрема словоформа зустрічається в навчальних даних значно рідше, ніж її англійський відповідник. Це явище, відоме як розрідженість даних, ускладнює для моделі процес навчання узагальнених, надійних векторних представлень

для базового поняття (в даному випадку, "рука"), оскільки вона має недостатньо прикладів для кожної з його варіацій.

2. Обчислювальна складність морфологічного аналізу: Правильне визначення початкової форми слова (леми) та його граматичних характеристик (частина мови, відмінок, рід, число) перетворюється на складну, ресурсоємну задачу. Для її розв'язання необхідні великі електронні словники, детальні набори граматичних правил та складні алгоритми аналізу, здатні впоратися з численними винятками та особливостями мови.

2.1.3 Фундаментальна роль лематизації над стемінгом

В англомовному NLP для нормалізації слів часто використовується стемінг — процес обрізання суфіксів та закінчень для отримання "основи" слова. Наприклад, слова "thinking", "thinks" можуть бути зведені до основи "think". Однак для української мови такий підхід є не просто неефективним, а й руйнівним. Закінчення в українській мові несуть критично важливу граматичну інформацію, і їхнє видалення призводить до втрати змісту. Наприклад, стемінг слова "турботою" може дати безглузду основу "турбот", повністю стираючи інформацію про орудний відмінок.

Тому для української мови єдиним прийнятним методом нормалізації є лематизація — процес зведення словоформи до її канонічної, словникової форми (леми). Наприклад, словоформи іду, ідеш, ітимуть будуть коректно зведені до леми іти. Цей процес є обов'язковою передумовою для більшості подальших NLP-задач, таких як інформаційний пошук, класифікація текстів, машинний переклад та сентимент-аналіз. Без лематизації модель не зможе зрозуміти, що "книга" і "книгою" — це одне й те саме поняття.

Ефективність лематизації безпосередньо залежить від якості та повноти морфологічних словників. Саме тому створення та підтримка таких ресурсів, як ВЕСУМ (Великий електронний словник української мови), який містить інформацію

про сотні тисяч лем та їхні словоформи, є наріжним каменем для розвитку всієї екосистеми українського NLP. Навіть сучасні інструменти, що демонструють високу точність лематизації (понад 97%), іноді припускаються помилок, які доводиться виправляти за допомогою словників.

Таким чином, морфологічна складність є не просто однією з характеристик української мови, а центральним фактором, що визначає всю архітектуру та логіку NLP-пайплайнів. Якщо для англійської мови прогрес часто досягається шляхом збільшення обсягів даних та розміру моделей, то для української мови не менш важливим є інвестування у "розумну обробку" — розробку та вдосконалення фундаментальних лінгвістичних інструментів, таких як морфологічні аналізатори та лематизатори.

2.2 Синтаксична гнучкість та її виклики для моделей

Окрім морфологічного багатства, українська мова характеризується значною синтаксичною гнучкістю, що також створює серйозні виклики для традиційних NLP-архітектур. Ця гнучкість, що є прямим наслідком розвиненої системи відмінків, вимагає від моделей здатності розуміти граматичні зв'язки між словами незалежно від їхньої позиції в реченні.

2.2.1 Як відмінки та узгодження передають граматичне значення. Вільний порядок слів.

Як зазначалося раніше, в українській мові граматична роль слова (хто що робить і над ким) визначається не стільки його місцем у реченні, скільки його відмінковою формою та узгодженням з іншими словами (за родом, числом, відмінком). Це явище, відоме як вільний порядок слів, дозволяє переставляти члени речення для зміни акцентів чи стилістичного забарвлення без втрати основного змісту. Наприклад, речення "Дітям подобається морозиво" можна сформулювати як "Морозиво

подобається дітям" або "Дітям морозиво подобається", і в усіх випадках буде зрозуміло, що саме діти є суб'єктом сприйняття, а морозиво — об'єктом.

Для NLP-моделей, які історично розроблялися для мов із фіксованим порядком слів, таких як англійська, це становить величезну проблему. Моделі, що покладаються на позицію слова як на ключову ознаку для визначення його синтаксичної ролі, виявляються неефективними. Вони не можуть просто припустити, що слово на початку речення є підметом, а слово після дієслова — додатком. Модель повинна навчитися ідентифікувати та інтерпретувати морфологічні маркери (закінчення), щоб правильно встановити зв'язки між словами, які можуть бути рознесені на значну відстань одне від одного в реченні.

2.2.2 Переваги Трансформерів над RNN для української мови

Синтаксична гнучкість української мови безпосередньо впливає на вибір архітектури нейронної мережі.

1. Рекурентні нейронні мережі (RNN), включно з їхніми більш просунутими варіантами, такими як LSTM та GRU, обробляють текст послідовно, токен за токеном. Вони передають інформацію від попереднього кроку до наступного, що робить їх природно пристосованими для аналізу послідовностей. Однак їхня головна слабкість — це проблема довготривалих залежностей. Інформація про слова, що стоять на початку речення, може "затухати" або спотворюватися до того моменту, як модель дійде до кінця речення. У мові з вільним порядком слів, де підмет може стояти в кінці речення, а пов'язаний з ним додаток — на початку, RNN виявляються малоефективними.

2. Трансформери (Transformers), на противагу RNN, використовують архітектуру, що ідеально підходить для мов з вільним порядком слів. Замість послідовної обробки вони аналізують усі токени в реченні одночасно за допомогою механізму само-уваги (self-attention). Цей механізм дозволяє моделі для кожного

слова обчислити "оцінку важливості" всіх інших слів у реченні, незалежно від відстані між ними. Таким чином, модель може "зрозуміти", що прикметник на початку речення узгоджується з іменником у кінці, оскільки вона аналізує їхні граматичні форми, а не лише їхню близькість. Хоча Трансформери самі по собі не враховують порядок слів, ця проблема вирішується додаванням позиційних кодувань (positional encodings), які надають моделі інформацію про місце кожного токена в послідовності.

Поява архітектури Трансформерів стала не просто інкрементальним покращенням для українського NLP, а справжнім парадигмальним зсувом. Вона надала архітектурний фундамент, який набагато краще відповідає синтаксичній природі мови. Якщо для англійської мови з її фіксованим порядком слів RNN вже були досить ефективними, то для української саме Трансформери дозволили зробити якісний стрибок у розумінні синтаксичних зв'язків.

2.2.3 Роль синтаксичного аналізу залежностей

Щоб машина могла правильно інтерпретувати речення з вільним порядком слів, їй потрібно не просто ідентифікувати слова, а й побудувати структуру їхніх взаємозв'язків. Для цього використовується синтаксичний аналіз залежностей (dependency parsing). Цей підхід представляє речення у вигляді дерева, де кожне слово (крім кореневого, зазвичай дієслова-присудка) залежить від іншого, "головного" слова.

Наприклад, у реченні "Гарна дівчина швидко читає цікаву книгу" аналіз залежностей встановить такі зв'язки: "гарна" залежить від "дівчина" (означення), "дівчина" залежить від "читає" (підмет), "швидко" залежить від "читає" (обставина), а "цікаву" залежить від "книгу" (означення), яка, у свою чергу, залежить від "читає" (додаток). Побудова такого дерева дозволяє однозначно зрозуміти структуру речення, навіть якщо слова переставити місцями.

Якість синтаксичного аналізу є критично важливою для будь-яких завдань, що вимагають глибокого розуміння тексту, таких як видобування відношень (relation extraction), системи питань-відповідей (question answering) та складний сентимент-аналіз. Інструменти, що надають цю функціональність для української мови, як-от Stanza та NLP-Cube, зазвичай тренуються на спеціально анотованих корпусах, таких як Universal Dependencies (UD). Точність цих парсерів є прихованою, але фундаментальною передумовою для успіху багатьох високорівневих NLP-застосунків. Помилка на етапі синтаксичного аналізу (наприклад, неправильне приєднання означення до іменника) неминуче призведе до помилки в кінцевому результаті.

2.3 Лексико-семантичні та прагматичні особливості

Окрім структурних викликів на рівні морфології та синтаксису, обробка української мови стикається зі складними проблемами на рівні значення (семантики) та контексту (прагматики). Ці проблеми включають неоднозначність, непрямі форми висловлювання та поширеність нелітературних варіантів мови.

2.3.1 Лексична та синтаксична омонімія

Неоднозначність (ambiguity) є однією з центральних проблем в NLP для будь-якої мови, і українська не є винятком. Вона проявляється на кількох рівнях:

Лексична неоднозначність: Виникає, коли одна й та сама словоформа може мати кілька різних значень або належати до різних частин мови. Класичним прикладом є слово коса, яке може означати сільськогосподарське знаряддя, зачіску або географічний об'єкт. Інший тип лексичної неоднозначності — омоформи, коли словоформи різних лексем збігаються. Наприклад, слово «мати» може бути іменником (мама) або дієсловом (володіти). Для розв'язання такої неоднозначності модель повинна аналізувати контекст, у якому вжито слово.

Синтаксична неоднозначність (омонімія): Виникає, коли послідовність слів може бути розібрана синтаксично кількома різними способами, що призводить до різних інтерпретацій. Наприклад, у реченні "Він зустрів друга брата" незрозуміло, чи він зустрів друга, який є братом, чи він зустрів друга, що належить братові. Розв'язання таких синтаксичних омонімій є складним завданням, що вимагає не лише досконалих синтаксичних парсерів, а й іноді семантичного аналізу та знань про світ.

2.3.2 Виявлення сарказму, іронії та гумору

Виявлення непрямих форм висловлювання, таких як сарказм, іронія та гумор, є одним із найскладніших завдань сучасного NLP. Це завдання виходить за межі семантики (буквального значення слів) і належить до сфери прагматики — аналізу намірів мовця та контексту комунікації.

Сарказм часто полягає у використанні позитивно забарвлених слів для вираження негативної оцінки (наприклад, "Геніальне рішення!" у відповідь на очевидну дурницю). Для його розпізнавання модель повинна вміти ідентифікувати невідповідність між буквальним значенням висловлювання та реальним контекстом, тоном автора або загальновідомими фактами.

Для української мови ця проблема ускладнюється майже повною відсутністю великих, якісно анотованих публічних корпусів текстів із розміткою сарказму та іронії. Хоча існують загальні тональні словники, вони безсилі проти сарказму. Без таких спеціалізованих датасетів для тренування та тестування моделей розробка надійних систем детекції сарказму залишається відкритою дослідницькою проблемою.

2.3.3 Проблема нелітературної мови: обробка суржику та діалектів

Одним з найунікальніших і найскладніших викликів для українського NLP є обробка суржику — поширеної форми змішаного українсько-російського мовлення. Суржик не є стабільним діалектом чи єдиною мовною системою; це радше спектр

мовних практик, що виникають внаслідок тривалого мовного контакту та інтерференції. Він може проявлятися на всіх рівнях:

- Лексичному: Вживання російських слів в українському реченні (наприклад, пожалуста, конєшно).
- Фонетичному: Українські слова вимовляються з російською фонетикою.
- Морфологічному: Використання російських афіксів з українськими коренями (наприклад, префікс под- замість українського під-) або неправильне відмінювання українських слів за російським зразком.
- Синтаксичному: Побудова речень за російськими синтаксичними моделями (калькування).

Для NLP-систем суржик становить величезну проблему з кількох причин:

- Дефіцит даних: Суржик є переважно розмовною практикою, тому великих цифрових корпусів для його вивчення та моделювання практично не існує.
- Непослідовність: Прояви суржику значно варіюються залежно від регіону, віку, освіти та соціального статусу мовця, що унеможливлює створення єдиної моделі для його обробки.
- Відмова стандартних інструментів: Інструменти для токенізації, лематизації та синтаксичного аналізу, навчені на літературній українській мові, неминуче даватимуть збій при обробці суржику, оскільки його лексика та граматика не відповідають нормам.

Ця проблема створює значний розрив між академічними дослідженнями, які здебільшого фокусуються на стандартній мові, та практичними застосунками (наприклад, аналіз відгуків у соцмережах, чат-боти), які в реальному світі постійно стикаються з нелітературним мовленням користувачів. Ігнорування суржику призводить до значного, часто невиміряного, падіння точності NLP-систем у реальних

умовах експлуатації. Аналогічні, хоча й менш поширені, проблеми виникають при обробці регіональних діалектів.

2.4 Екосистема ресурсів та інструментів для українського NLP

Попри всі лінгвістичні виклики, екосистема ресурсів для українського NLP за останні роки зазнала стрімкого розвитку. Відбувся перехід від статусу "низькоресурсної" мови (low-resource language) до мови з великою кількістю сирих текстових даних та зростаючим набором спеціалізованих інструментів і анотованих датасетів. Ця частина звіту надає огляд ключових ресурсів, доступних для розробників та дослідників.

2.4.1 Корпуси та набори даних. Огляд ключових монолінгвальних корпусів

Наявність якісних та об'ємних даних є фундаментом для будь-якого сучасного NLP-проєкту, особливо для тренування глибоких нейронних мереж. За останні роки українська мова перестала бути низькоресурсною з точки зору обсягу сирих текстових даних. Існують декілька масштабних корпусів, що слугують основою для тренування великих мовних моделей та проведення лінгвістичних досліджень.

За останні роки українська мова перестала бути низькоресурсною з точки зору обсягу сирих текстових даних. Існують декілька масштабних корпусів, що слугують основою для тренування великих мовних моделей та проведення лінгвістичних досліджень:

- GRAC (General Regionally Annotated Corpus of Ukrainian): Це найбільший та найрепрезентативніший діахронічний корпус української мови, що охоплює тексти з 1816 року до сьогодення. Його обсяг перевищує 400 мільйонів токенів. Корпус містить тексти різних жанрів та стилів і має унікальну регіональну анотацію, що дозволяє досліджувати територіальні варіанти мови. GRAC є безцінним ресурсом для лінгвістів, лексикографів та істориків мови. Однак його

головним обмеженням є те, що він не доступний для повного завантаження, а робота з ним відбувається через веб-інтерфейс.

- PluG (Pluperfect GRAC): Це підкорпус GRAC, який містить тексти, що не захищені авторським правом (переважно до середини XX століття), і доступний для завантаження. Він є важливим ресурсом для досліджень історичного розвитку мови.
- Веб-корпуси (ukTenTen, Malyuk, UberText 2.0, OSCAR, CC-100): Це гігантські колекції текстів, зібраних шляхом автоматичного сканування інтернету (веб-кроулінгу).
- ukTenTen є одним з найбільших, його версія 2022 року налічує понад 9.5 мільярдів токенів.
- Malyuk — це компіляція з кількох джерел (UberText 2.0, OSCAR, новини) об'ємом 113 ГБ.
- UberText 2.0 містить понад 3.2 мільярда токенів з новин, художньої літератури, соцмереж та інших джерел.
- OSCAR та CC-100 — це українські частини великих багатомовних корпусів, створених на основі Common Crawl, об'ємом 28 ГБ та 10 ГБ відповідно. Ці корпуси є незамінними для попереднього навчання (pre-training) великих мовних моделей (LLM), але вони часто містять "шум" (дублікати, неякісний текст, помилки розмітки) і потребують ретельної фільтрації та очищення.
- Brown-UK (BRUK): Це збалансований корпус сучасної української мови (тексти 2010-2020 років) обсягом в 1 мільйон слів. Його створено за зразком класичного Браунівського корпусу для англійської мови. Завдяки збалансованості за жанрами, він є ідеальним для тестування та оцінки моделей, а також для навчання на менших, але якісних даних.

Таблиця 2.2

Огляд основних загальнодоступних корпусів української мови

Назва корпусу	Розмір (приблизний)	Тип контенту	Доступність	Основний сценарій використання
GRAC	> 437 млн токенів	Діахронічний, різноманітні жанри, регіональна анотація	Онлайн-інтерфейс пошуку	Лінгвістичні дослідження, лексикографія
PluG	Частина GRAC	Історичні тексти (1816-1954)	Для завантаження	Історичний NLP, дослідження мовних змін
Malyuk	113 ГБ	Веб-кроул, новини, UberText 2.0, OSCAR	Для завантаження	Попереднє навчання LLM
ukTenTen	> 9.5 млрд токенів	Веб-кроул, класифікація за жанрами та темами	Онлайн-інтерфейс, API	Попереднє навчання LLM, лінгвістичні дослідження
UberText 2.0	> 3.2 млрд токенів	Новини, художня література, соцмережі, юридичні тексти	Для завантаження	Попереднє навчання LLM
Brown-UK	1 млн слів	Збалансований за жанрами, сучасна мова	Для завантаження	Оцінка моделей, навчання на якісних даних

2.4.2 Спеціалізовані датасети для конкретних завдань

Якщо сирих даних для української мови багато, то якісних анотованих датасетів для конкретних задач (supervised learning) значно менше, хоча їхня кількість постійно зростає. Наявність таких ресурсів є критичною для розробки практичних застосунків.

- Граматична корекція помилок (GEC): UA-GEC — це перший публічний датасет для цієї задачі, випущений компанією Grammarly. Він містить 20,715 речень, анотованих для виправлення граматичних помилок та покращення стилю (fluency). Цей ресурс є ключовим для розробки українських аналогів систем перевірки граматики.
- Питання-відповіді (QA): UA-SQuAD — це українська версія знаменитого датасету Stanford Question Answering Dataset, що є стандартом для оцінки моделей розуміння прочитаного.
- Сентимент-аналіз: Спеціалізовані датасети для аналізу тональності часто створюються на основі відгуків користувачів. Прикладами є Yakaboo Book Reviews (відгуки на книги) та датасет відгуків з e-commerce платформ, на якому було навчено модель RoBERTa з точністю 92%. Окремо варто виділити датасет
- "Ukrainian Resilience", створений під час повномасштабного вторгнення, який містить понад 11,000 дописів із соцмереж, розмічених за ознакою "потребує допомоги / не потребує".
- Синтаксичний та семантичний аналіз: Корпус Universal Dependencies (UD) for Ukrainian містить тисячі речень, вручну анотованих морфологічними тегами та синтаксичними зв'язками (деревами залежностей). Це "золотий стандарт" для тренування синтаксичних парсерів, таких як Stanza.

2.4.3 Проблема "низькоресурсності": стратегії подолання

Попри наявність великих сирих корпусів, українська мова все ще може вважатися "низькоресурсною" в контексті анотованих даних для багатьох специфічних завдань (наприклад, розпізнавання сарказму, аналіз суржику, видобування відношень). Навчати складні моделі з нуля на невеликих датасетах неефективно.

Основною стратегією подолання цієї проблеми є трансферне навчання (transfer learning). Цей підхід полягає у використанні великої моделі, попередньо навченої на гігантських обсягах сирих текстів (наприклад, на корпусі Malyuk), а потім її "донавчанні" (fine-tuning) на невеликому, специфічному для задачі анотованому датасеті. Модель, яка вже "знає" загальні закономірності мови, швидко адаптується до нової задачі, навіть маючи лише кілька тисяч прикладів. Цей підхід є набагато ефективнішим і менш витратним, ніж тренування з нуля.

Екосистема українських даних демонструє певний "розрив": з одного боку, існують мільярдні веб-корпуси, а з іншого — невеликі, але дуже цінні, вручну створені анотовані датасети. Це означає, що головним вузьким місцем для розробки нових спеціалізованих застосунків є повільний та дорогий процес ручної анотації даних. Водночас, трагічні події повномасштабного вторгнення парадоксальним чином прискорили створення специфічних датасетів для аналізу військового та кризового дискурсу, що відкрило нові напрямки для досліджень.

2.5 Аналіз ключових бібліотек

Наявність готових до використання програмних бібліотек та інструментів значно спрощує та прискорює розробку NLP-застосунків. Для української мови існує декілька ключових інструментів, що вирішують базові задачі обробки тексту.

2.5.1 Морфологічний аналіз: порівняльна оцінка pymorphy2, Stanza та nlp-uk

Вибір інструменту для морфологічного аналізу є першим і одним з найважливіших рішень у проєкті. Три основні бібліотеки пропонують різні компроміси між швидкістю, функціональністю та точністю:

- pymorphy2: Це добре відомий морфологічний аналізатор для російської та української мов, який давно використовується в індустрії. Його головні переваги — висока швидкість роботи та простота у використанні. Він працює на основі великих, попередньо скомпільованих словників (для української мови використовується версія, що базується на старому випуску словника ВЕСУМ). pymorphy2 надає всі можливі морфологічні розбори для слова, але не виконує дизамбігуацію, тобто не визначає, який саме розбір є правильним у даному контексті. Це робить його ідеальним для завдань, де потрібна швидка лематизація без глибокого аналізу.
- Stanza: Це комплексний пакет для обробки природної мови від Стенфордського університету. На відміну від pymorphy2, Stanza пропонує повний NLP-пайплайн: токенізацію, лематизацію, тегування частин мови (POS-tagging) та, що найважливіше, синтаксичний аналіз залежностей. Українські моделі для Stanza тренуються на корпусі Universal Dependencies, що забезпечує високу якість синтаксичного розбору. Це потужний, але більш ресурсоємний інструмент, який є найкращим вибором для завдань, що вимагають глибокого синтаксичного розуміння тексту.
- nlp-uk: Це інструмент, розроблений спеціально для української мови. Він базується на актуальній версії словника ВЕСУМ та рушії LanguageTool. Його ключовою перевагою є підтримка базової дизамбігуації — спроби вибрати найбільш імовірний морфологічний розбір слова в контексті. Це робить його хорошим компромісом між швидкістю pymorphy2 та глибиною аналізу Stanza, особливо для завдань, де правильне визначення частини мови є критичним.

Вибір між цими інструментами залежить від конкретної задачі. Для простої та швидкої лематизації достатньо pymorphy2. Для завдань, що потребують розуміння структури речення, незамінним є Stanza. nlp-uk є хорошим вибором, коли потрібна якісна лематизація з урахуванням контексту, але без повного синтаксичного аналізу.

2.5.2 Комплексні рішення. LanguageTool

LanguageTool — це не просто бібліотека, а повноцінна система для перевірки граматики, стилю та орфографії, яка підтримує десятки мов, включно з українською. Вона використовує комбінацію тисяч правил, написаних лінгвістами, та статистичних моделей для виявлення широкого спектра помилок. Для українського NLP LanguageTool є важливим ресурсом не лише як готовий продукт, а і як джерело даних та правил. Наприклад, його використовують для автоматичної анотації даних у задачах граматичної корекції помилок.

2.5.3 ВЕСУМ та тональні лексикони. Роль словників

За багатьма інструментами стоять фундаментальні лексикографічні ресурси, без яких їхня робота була б неможливою:

- ВЕСУМ (Великий електронний словник української мови): Це наріжний камінь сучасної української комп'ютерної морфології. Словник містить понад 400,000 лем і надає для них повні парадигми словозміни з детальними граматичними тегами. Саме на основі ВЕСУМ працюють такі інструменти, як nlp-uk та LanguageTool, що забезпечує високу якість їхньої роботи.
- Тональні словники (Sentiment Lexicons): Це списки слів, яким вручну присвоєно оцінку тональності (позитивна, негативна, нейтральна). Хоча це простий підхід, він досі є важливим компонентом багатьох систем сентимент-аналізу, особливо в гібридних моделях. Для української мови існують загальнодоступні ресурси, такі як спеціалізований "Тональний

словник" та українська частина багатомовного словника Multilingualsentiment.

Таким чином, інструментарій для українського NLP демонструє чіткий поділ на дві категорії: спеціалізовані, керовані словниками інструменти для морфології (pymorphy2, nlp-uk) та універсальні, керовані корпусами інструменти для синтаксису (Stanza). Ця диференціація свідчить про зрілість екосистеми, яка пропонує розробникам вибір оптимального рішення для кожного рівня лінгвістичного аналізу.

2.6 Попередньо навчені моделі. Ембединги. LLM

Сучасний NLP неможливо уявити без попередньо навчених моделей, які дозволяють досягати високих результатів навіть з обмеженою кількістю даних для конкретної задачі. Еволюція таких моделей для української мови пройшла шлях від класичних векторних представлень до потужних трансформерних архітектур.

2.6.1 Класичні векторні представлення слів

До епохи Трансформерів стандартом для представлення слів у векторному просторі були статичні ембединги, такі як Word2Vec, GloVe та fastText. Для української мови існують готові, попередньо навчені моделі цих ембедингів, зазвичай натреновані на текстах Вікіпедії або Common Crawl. Вони добре вловлюють семантичну схожість між словами (наприклад, вектори слів "король" та "цар" будуть близькими у просторі). Однак їхній головний недолік — відсутність контекстуалізації. Слово «замок» матиме один і той самий вектор, незалежно від того, чи йдеться про споруду, чи про механізм.

2.6.2 Трансформерні моделі для української мови

Справжня революція в українському NLP, як і у світовому, пов'язана з появою Трансформерів. Ці моделі генерують контекстуалізовані ембединги, тобто векторне

представлення слова залежить від його оточення. Розвиток трансформерних моделей для української мови пройшов два ключові етапи.

1. Багатомовні моделі. Перші успіхи були досягнуті шляхом донавчання великих багатомовних моделей, які були попередньо навчені на текстах сотні мов одночасно, включно з українською. Найпопулярнішими з них є mBERT (multilingual BERT), XLM-RoBERTa та mT5. Ці моделі демонструють хороші результати і є чудовою відправною точкою, але їхня "ємність" розподілена між багатьма мовами, що може обмежувати їхню ефективність для кожної окремої мови.
2. Монолінгвальні українські моделі. Важливою віхою для української NLP-спільноти стало створення моделей, навчених виключно на українських текстах. Яскравим прикладом є LiBERTa Large — перша модель архітектури BERT Large, попередньо навчена з нуля лише на українських даних. Такі монолінгвальні моделі, як правило, перевершують багатомовні аналоги на україномовних задачах, оскільки весь їхній потенціал сфокусований на нюансах однієї мови. Це свідчить про зрілість екосистеми, яка накопичила достатньо даних (корпуси Malyuk, ukTenTen) та експертизи для створення власних фундаментальних моделей.

2.6.3 Сучасний стан та майбутнє LLM

Наразі сфера NLP стрімко рухається в напрямку великих мовних моделей (Large Language Models, LLM). Для української мови цей процес включає кілька напрямків:

1. Донавчання (Fine-tuning) відкритих LLM. Оскільки тренування LLM з нуля є надзвичайно дорогим, спільнота зосереджується на адаптації існуючих потужних відкритих моделей, таких як Llama. Проєкти на кшталт UAlpaca донавчають такі моделі на наборах інструкцій українською мовою, щоб навчити їх виконувати завдання та вести діалог.

2. Спільні завдання (Shared Tasks). Для стимулювання інновацій та об'єктивної оцінки прогресу проводяться змагання. Наприклад, UNLP 2024 Shared Task був присвячений саме донавчанню LLM для української мови, що дозволило порівняти різні підходи та методики.
3. Основними перешкодами на шляху розвитку українських LLM є величезні обчислювальні потреби для тренування та донавчання, а також необхідність у створенні великих, якісних та різноманітних наборів даних з інструкціями (instruction datasets) українською мовою.

Розвиток попередньо навчених моделей для української мови є яскравим прикладом того, як локальна NLP-спільнота пройшла шлях від "запозичення" багатомовних технологій до створення власних, високопродуктивних, мовно-специфічних ресурсів. Цей перехід від статусу споживача до статусу виробника фундаментальних моделей є критично важливим для досягнення найвищих результатів у задачах обробки української мови.

3 МЕТОДОЛОГІЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Обґрунтування вибору мови програмування Python та необхідних бібліотек

Вибір мови програмування та відповідних інструментів є ключовим етапом у розробці будь-якого програмного забезпечення. Для даної дипломної роботи було обрано мову програмування Python. Це рішення ґрунтується на низці вагомих переваг, які Python пропонує для вирішення поставлених завдань.

Мова здобула широку популярність у галузях штучного інтелекту, машинного навчання та обробки природної мови завдяки своєму простому синтаксису, великій спільноті розробників та наявності численних потужних бібліотек. Для задачі сентимент-аналізу існує низка ефективних інструментів та бібліотек на Python.

3.1.1 NLTK (Natural Language ToolKit)

Одна з найстаріших та найповніших бібліотек для NLP. NLTK надає інструменти для токенізації, стемінгу, лематизації, тегування частин мови, синтаксичного аналізу та класифікації текстів. Вона також включає модуль VADER (Valence Aware Dictionary and sEntiment Reasoner), який є ефективним інструментом для сентимент-аналізу, особливо для текстів із соціальних мереж, оскільки він враховує емодзі, інтенсифікатори та пунктуацію.

3.1.2 Transformers

Бібліотека, що надає доступ до тисяч попередньо навчених моделей трансформерів, таких як BERT, RoBERTa, GPT та інші, для широкого спектру завдань NLP, включаючи сентимент-аналіз. Вона дозволяє легко завантажувати та використовувати ці моделі, а також донавчати їх на власних даних.

3.1.3 PyTorch

Потужний фреймворк для глибокого навчання, що дозволяє створювати та навчати складні нейронні мережі, такі як CNN, RNN (LSTM, GRU) та трансформери, для задач сентимент-аналізу.

3.1.4 PyQt6

Набір Python-зв'язок для розробки графічних інтерфейсів користувача (GUI). PyQt6 дозволяє створювати сучасні та інтерактивні настільні додатки з використанням широкого набору віджетів, інструментів для компонування елементів, діалогових вікон та можливостей для обробки подій.

3.1.5 Matplotlib

Популярна бібліотека для створення статичних, анімованих та інтерактивних візуалізацій у Python. Matplotlib надає широкий спектр інструментів для побудови різноманітних графіків, таких як гістограми, стовпчасті діаграми, кругові діаграми, точкові діаграми тощо. У даному проєкті Matplotlib використовується для візуалізації результатів сентимент-аналізу.

3.1.6 Google BERT

Також, була використана модель обробки природної мови Google BERT (Bidirectional Encoder Representations from Transformers). Модель, що революціонізувала підхід до розуміння контексту в тексті. На відміну від попередніх моделей, BERT обробляє текст двонаправлено, що дозволяє йому краще вловлювати зв'язки між словами та їхнє значення залежно від оточення.

3.2 Середовище розробки

Інтегроване середовище розробки (IDE) – це програмне забезпечення для розробки програмного коду. IDE зазвичай включають текстовий редактор, інструменти для автоматизації збірки коду, інтерфейс для системи контролю версій, консоль, тощо.

3.2.1 Visual Studio Code

Visual Studio Code (VS Code) – це безкоштовний редактор вихідного коду, розроблений компанією Microsoft, який швидко здобув величезну популярність серед розробників завдяки своїй легкості, потужності та гнучкості. Він підтримує безліч мов програмування та має розгалужену екосистему розширень, що дозволяє адаптувати його під будь-які потреби розробки.

3.3 Вибір та обґрунтування методів машинного навчання або глибокого навчання для визначення емоційного забарвлення

Для визначення емоційного забарвлення тексту існує декілька підходів. Вибір конкретного методу залежить від низки факторів, таких як обсяг та якість доступних даних, необхідна точність, обчислювальні ресурси та специфіка мови.

Методи на основі лексиконів та правил, такі бібліотеки, як TextBlob та VADER (в NLTK), використовують словники слів з попередньо визначеною тональністю та набори правил для визначення загального сентименту тексту. Ці методи є відносно простими в реалізації та інтерпретації, але їх ефективність може бути обмеженою для складних текстів, що містять сарказм, іронію або специфічну для домену лексику.

Класичні методи машинного навчання, такі алгоритми, як Наївний Баєсівський класифікатор (Naive Bayes), Логістична Регресія (Logistic Regression) та Метод Опорних Векторів (Support Vector Machines, SVM). Ці методи навчаються на розмічених даних (тексти з відомим емоційним забарвленням) і можуть показувати

хороші результати, особливо якщо дані належним чином попередньо оброблені та векторизовані (наприклад, за допомогою TF-IDF). Дослідження показують, що, наприклад, логістична регресія може бути ефективною для аналізу тональності українських текстів.

Моделі глибокого навчання, зокрема згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN, LSTM, GRU) та трансформери (BERT, RoBERTa), продемонстрували найсучасніші результати в багатьох завданнях NLP, включаючи сентимент-аналіз. Трансформерні моделі, такі як BERT, особливо ефективні, оскільки вони здатні враховувати контекст слів у реченні та можуть бути попередньо навчені на величезних обсягах текстових даних. Для української мови існують як багатомовні попередньо навчені моделі BERT, так і монолінгвальні моделі (наприклад, Ukr-RoBERTa, LiBERTa Large), які можуть бути донавчені для конкретного завдання сентимент-аналізу.

Для даної дипломної роботи, враховуючи потенційну складність відгуків та прагнення до високої точності, пріоритетними є методи машинного навчання та глибокого навчання. Для досягнення вищої точності, особливо при наявності достатнього обсягу навчальних даних, буде розглянуто використання попередньо навчених трансформерних моделей з подальшим донавчанням (fine-tuning) на цільовому наборі даних відгуків. Цей підхід дозволяє використовувати потужні можливості великих мовних моделей, адаптуючи їх до специфіки аналізованих текстів.

3.4. Опис процесу збору та підготовки даних

Якість та репрезентативність даних є критично важливими для успішного навчання моделей машинного навчання та отримання достовірних результатів сентимент-аналізу. Процес збору та підготовки даних включає два основні етапи:

визначення джерел та методів збору даних, а також попередню обробку текстових даних.

3.4.1. Джерела даних та методи збору

Відгуки про товари та послуги можна збирати з різноманітних джерел в Інтернеті. Основними джерелами можуть бути:

- Веб-сайти інтернет-магазинів та маркетплейсів.
- Спеціалізовані сайти з відгуками.
- Соціальні мережі.
- Форуми та обговорення.

Для збору даних можуть використовуватися наступні методи:

- API (Application Programming Interface): Багато платформ надають API для доступу до їхніх даних. Наприклад, Twitter API дозволяє збирати твіти за певними ключовими словами або хештегами. Це є переважним методом, оскільки він є структурованим та легальним.
- Веб-скрейпінг (Web Scraping): У випадках, коли API недоступний або не надає необхідної інформації, можна використовувати техніки веб-скрейпінгу для автоматичного вилучення даних безпосередньо з HTML-сторінок веб-сайтів.
- Використання існуючих наборів даних: Якщо доступні публічні набори даних україномовних відгуків, їх також можна використовувати або доповнювати власними зібраними даними.

Зібрані дані будуть зберігатися у структурованому форматі для подальшої обробки.

3.4.2. Етапи попередньої обробки текстових даних

Сирі текстові дані, зібрані з Інтернету, зазвичай містять багато "шуму" та нерелевантної інформації, яка може негативно вплинути на якість моделі. Тому

попередня обробка тексту є надзвичайно важливим етапом. Для україномовних відгуків вона включатиме наступні кроки:

1. Очищення тексту:

- Видалення HTML-тегів.
- Видалення URL-адрес, електронних пошт та іншої нерелевантної контактної інформації.
- Видалення спеціальних символів, емодзі (якщо вони не несуть важливої семантичної інформації для конкретної моделі), та зайвих пробілів.

2. Приведення до нижнього регістру (наприклад, "Товар" і "товар" будуть розглядатися як одне слово),

3. Розбиття тексту на окремі слова або токени. Для цього можна використовувати інструменти з бібліотек NLTK.

4. Видалення пунктуації: Видалення розділових знаків, якщо вони не є важливими для аналізу.

5. Видалення стоп-слів: Стоп-слова – це поширені слова, які не несуть значного семантичного навантаження (наприклад, "і", "в", "на", "але", "був"). Їх видалення допомагає зменшити розмірність даних та зосередитися на значущих словах. Необхідно використовувати список стоп-слів, специфічний для української мови. Такі списки можуть бути доступні в бібліотеках або створені вручну.

6. Обробка чисел та інших специфічних токенів.

Після попередньої обробки дані будуть готові для векторизації та подальшого використання в моделях машинного навчання.

3.5 Опис етапів розробки програмного забезпечення

Розробка програмного забезпечення для визначення емоційного забарвлення відгуків буде проходити через декілька послідовних етапів, що забезпечують структурований підхід до створення функціонального та надійного продукту:

Необхідно:

1. провести аналіз актуальності задач сентимент-аналізу у сфері електронної комерції та сервісного обслуговування;
2. дослідити аналогічні програмні рішення та систем;
3. провести дослідження літературних джерел для ознайомлення з предметною галуззю;
4. розробити функціональні та нефункціональні вимоги до системи;
5. побудувати діаграми та моделі, що описують логіку роботи ПЗ;
6. обрати технології для реалізації проєкту;
7. зібрати корпус відгуків для навчання та тестування системи;
8. реалізувати модуль обробки природної мови;
9. побудувати інтерфейс користувача для взаємодії із системою;
10. протестувати точність та швидкодію системи;
11. провести тестування на реальних відгуках;
12. порівняти результати із існуючими рішеннями;
13. узагальнити отримані результати, виявити сильні та слабкі сторони ПЗ.

4 РОЗРОБКА ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИЗНАЧЕННЯ ЕМОЦІЙНОГО ЗАБАРВЛЕННЯ ВІДГУКІВ

4.1 Теоретичні Основи Архітектури BERT

Для глибокого розуміння застосованої методології необхідно розглянути теоретичні основи архітектури BERT (Bidirectional Encoder Representations from Transformers), яка стала революційним кроком у галузі обробки природної мови (NLP).

4.1.1 Походження від Архітектури Трансформер

Архітектура BERT базується виключно на енкодерній частині моделі Трансформер, яка була вперше представлена у 2017 році. На відміну від повних моделей Трансформер, що складаються з енкодера та декодера і застосовуються для задач секвенс-ту-секвенс (наприклад, машинний переклад), BERT використовує лише стек енкодерів для побудови глибоких контекстуалізованих векторних представлень слів (ембедингів). Кожен енкодер у стеку складається з двох основних підшарів: механізму багатошарової уваги (multi-head self-attention) та повнозв'язної нейронної мережі прямого поширення (feed-forward neural network).

4.1.2 Механізм Двоспрямованості

Ключовою інновацією BERT є його глибока двоспрямованість. Попередні state-of-the-art моделі, такі як рекурентні нейронні мережі (RNN) та їх варіації (LSTM, GRU), обробляли текстові послідовності в одному напрямку: або зліва-направо, або справа-наліво. Навіть моделі, що поєднували два напрямки (bi-LSTM), робили це шляхом конкатенації результатів двох незалежних односпрямованих проходів. Натомість BERT, завдяки механізму уваги, аналізує всю вхідну послідовність

одночасно. Це дозволяє кожному токену в реченні "звертати увагу" на всі інші токени, незалежно від їхньої позиції, таким чином враховуючи повний контекст — як лівий, так і правий. Такий підхід забезпечує значно глибше розуміння мовних нюансів, зокрема омонімії, складних синтаксичних конструкцій та семантичних зв'язків у реченні.

4.1.3 Задачі Попереднього Навчання

Потужність BERT походить від процесу попереднього навчання на величезних масивах нерозмічених текстових даних. Цей процес є формою самокерованого навчання (self-supervised learning), оскільки мітки для навчання генеруються автоматично із самих даних, що дозволяє використовувати практично необмежені текстові корпуси (наприклад, Вікіпедія та книжкові колекції). Попереднє навчання BERT включає дві основні задачі:

1. Модель Маскованої Мови (Masked Language Model, MLM): У цій задачі 15% токенів у вхідній послідовності випадковим чином маскуються. Завдання моделі — передбачити оригінальні ідентифікатори цих замаскованих токенів на основі навколишнього двоспрямованого контексту. Процедура маскування є нетривіальною: з обраних 15% токенів, 80% замінюються на спеціальний токен, 10% — на випадковий токен зі словника, а решта 10% залишаються незмінними. Така стратегія змушує модель покладатися не на наявність самого токена , а на семантичний та синтаксичний контекст для відновлення інформації, що сприяє вивченню глибоких мовних патернів.
2. Прогнозування Наступного Речення (Next Sentence Prediction, NSP): Для цієї задачі модель отримує на вхід пару речень (А і Б) і повинна визначити, чи є речення Б логічним продовженням речення А в оригінальному тексті. У 50% випадків пара є справжньою, а в інших 50% речення Б є випадковим реченням з корпусу. Це завдання дозволяє BERT розуміти

зв'язки між реченнями, що є критично важливим для задач, які вимагають аналізу дискурсу, таких як питально-відповідні системи та узагальнення тексту.

Саме ця комбінація двоспрямованості та самокерованого навчання на гігантських корпусах дозволяє BERT вивчати універсальні мовні представлення, які потім можна ефективно адаптувати до широкого спектра специфічних NLP-задач за допомогою процесу донавчання.

4.2 Вибір Попередньо Навченої Моделі для Задач Української Мови

Вибір правильної попередньо навченої моделі є критичним етапом, який значною мірою визначає потенційну якість кінцевого рішення. Для роботи з українською мовою існує стратегічний вибір між використанням багатомовних моделей, розроблених великими міжнародними дослідницькими групами, та спеціалізованих мономовних моделей, створених українською NLP-спільнотою.

Багатомовні моделі (Multilingual Models) є зручною та доступною відправною точкою. Яскравим представником є модель bert-base-multilingual-cased від Google, яка була навчена на текстових корпусах 104 мов, включаючи українську. Вона використовує спільний словник (vocabulary) розміром 110,000 токенів для всіх мов. Перевагою такого підходу є універсальність та здатність до міжмовного перенесення знань (cross-lingual transfer), що може бути корисним для мов з обмеженими ресурсами. На базі таких моделей існують вже донавчені рішення для сентимент-аналізу, як-от tabularisai/multilingual-sentiment-analysis, що прямо підтримує українську мову. Однак, головним недоліком є те, що спільний словник та "розподілені" параметри моделі можуть неоптимально відображати унікальні морфологічні та синтаксичні особливості української мови, що потенційно обмежує максимальну досяжну точність.

Мономовні українські моделі створюються для подолання цих обмежень. Прикладом є модель LiBERTa Large — перша модель архітектури BERT Large, навчена "з нуля" виключно на українських текстах. Такі моделі мають спеціалізований словник і параметри, повністю налаштовані на граматику, семантику та стилістику української мови. Це створює потенціал для досягнення значно вищої точності на україномовних задачах. Проте їх розробка та навчання вимагають величезних обчислювальних ресурсів та доступу до великих національних текстових корпусів, що робить їх менш доступними для широкого кола дослідників.

Ця ситуація формує своєрідну "українську NLP-дилему": вибір між швидкою інтеграцією "достатньо хорошого" глобального інструменту та довгостроковою інвестицією у створення суверенних, більш точних, але й значно дорожчих мовних ресурсів. Для цілей даної дипломної роботи було прийнято прагматичне рішення використовувати багатомовну модель bert-base-multilingual-cased. Цей вибір дозволяє продемонструвати всю методологію донавчання на доступних обчислювальних ресурсах, водночас усвідомлюючи, що використання мономовних моделей, як-от LiBERTa Large, є перспективним напрямком для подальшого підвищення якості системи.

Таблиця 4.1

Порівняльний аналіз попередньо навчених моделей для NLP українською мовою

Назва моделі	Базова архітектура	Мови навчання	Переваги	Недоліки	Приклад використання
bert-base-multilingual-cased	BERT Base	104 мови, вкл. українську	Висока доступність, легкість інтеграції, підтримка Hugging Face.	Неспеціалізований словник, можлива нижча точність порівняно з мономовними аналогами.	Базова модель для донавчання на широкому спектрі задач.

Продовження таблиці 4.1

Порівняльний аналіз попередньо навчених моделей для NLP українською мовою

tabularisai/multilingual-sentiment-analysis	DistilBERT Multilingual Cased	20+ мов, вкл. українську	Готове рішення для сентимент-аналізу, легка у використанні, оптимізована (DistilBERT).	Обмежена 5-класовою шкалою, "чорна скринька" щодо даних для донавчання.	Швидке прототипування систем аналізу відгуків.
LiBERTa Large	BERT Large	Тільки українська	Спеціалізована для української мови, потенційно найвища точність, великий розмір моделі.	Високі вимоги до обчислювальних ресурсів, менша доступність порівняно з Hugging Face Hub.	Фундаментальні дослідження української мови, побудова state-of-the-art систем.

4.3 Парадигма Донавчання (Fine-Tuning) для Трансферного Навчання

Концепція донавчання є центральною для сучасного NLP і являє собою застосування трансферного навчання. Аналогічно до комп'ютерного зору, де моделі, попередньо навчені на гігантських наборах даних, як-от ImageNet, адаптуються для специфічних завдань (наприклад, медичної діагностики), моделі BERT, що вивчили загальні закономірності мови, донавчаються для конкретних прикладних задач.

Процес донавчання архітектурно полягає у додаванні одного або кількох нових, нетренованих шарів поверх попередньо навченої моделі BERT. Для задачі класифікації тексту це, як правило, один повнозв'язний (лінійний) шар, який приймає на вхід векторне представлення спеціального токена і видає на виході вектор логітів, розмірність якого дорівнює кількості цільових класів (наприклад, 3 для "позитивний", "негативний", "нейтральний"). Вектор, що стоїть на початку кожної послідовності, агрегує інформацію про все речення і слугує його узагальненим представленням для класифікаційних завдань.

Під час донавчання оновлюються ваги не лише новоствореного класифікаційного шару, а й усіх або частини шарів базової моделі BERT. Це дозволяє моделі не просто навчити новий класифікатор, а тонко налаштувати свої внутрішні представлення під специфіку лексики, синтаксису та семантики цільового набору даних. Цей процес є надзвичайно ефективним з точки зору використання ресурсів. Попереднє навчання BERT з нуля є обчислювально дорогим і тривалим процесом, доступним лише великим корпораціям та дослідницьким центрам. Натомість донавчання вимагає значно меншого, розміченого для конкретної задачі, набору даних (тисячі або десятки тисяч прикладів) і може бути виконане за відносно короткий час (зазвичай 2-4 епохи) на одному сучасному графічному процесорі (GPU). Таким чином, парадигма "pre-train, then fine-tune" демократизувала доступ до передових NLP-технологій, дозволивши широкому колу розробників та дослідників створювати високоточні моделі для своїх завдань.

4.4 Практична Реалізація Конвеєра Донавчання з Використанням PyTorch та Hugging Face

Сучасна розробка в галузі NLP є не роботою з одним ізольованим інструментом, а майстерним володінням інтегрованою екосистемою, де кожен компонент виконує свою специфічну роль. У даному дослідженні було використано такий екосистемний

підхід, де інтегроване середовище розробки (IDE) Visual Studio Code, бібліотека transformers від Hugging Face та фреймворк глибокого навчання PyTorch утворюють єдиний потужний конвеєр для донавчання моделі. Нижче наведено покроковий опис цього процесу.

4.5 Токенізація та Кодування Вхідних Даних

Токенізація є процесом перетворення сирого тексту на послідовність чисел (ідентифікаторів токенів), які модель може зрозуміти. Для цього було завантажено токенізатор, що відповідає обраній моделі — BertTokenizer.from_pretrained('bert-base-multilingual-cased').

Процес кодування кожного текстового прикладу за допомогою токенізатора включає кілька важливих кроків:

1. Додавання спеціальних токенів: На початок кожної послідовності додається токен (classification). Його фінальне векторне представлення після проходження через всі шари BERT використовується як агрегований ембединг усього речення для задачі класифікації. Токен (separator) додається в кінці послідовності для позначення її завершення.
2. Падинг (Padding): Усі послідовності в межах одного батчу повинні мати однакову довжину. Коротші послідовності доповнюються спеціальними токенами `` до максимальної довжини в батчі (або до фіксованої максимальної довжини max_length).
3. Обрізання (Truncation): Послідовності, довші за встановлений ліміт max_length (який не може перевищувати максимальну довжину, що підтримується моделлю, зазвичай 512 токенів), обрізаються.
4. Створення маски уваги (Attention Mask): Генерується бінарний тензор такої ж розмірності, як і послідовність input_ids. Він містить 1 на позиціях

реальних токенів та 0 на позиціях ``-токенів. Ця маска дозволяє механізму уваги ігнорувати доповнюючі токени і фокусуватися лише на змістовній частині вхідних даних.

Результатом цього етапу для кожного тексту є словник, що містить тензори input_ids та attention_mask, готовий для подачі в модель.

4.6 Конфігурація Моделі, Оптимізатора та Планувальника Швидкості Навчання

На цьому етапі конфігуруються ключові компоненти тренувального процесу. З бібліотеки transformers було завантажено клас BertForSequenceClassification. Цей клас є зручною обгорткою, яка складається з попередньо навченої моделі BERT та доданого поверх неї лінійного класифікаційного шару з нетренованими вагами. Кількість вихідних нейронів у цьому шарі відповідає кількості класів у задачі.

Для оновлення ваг моделі було обрано оптимізатор AdamW (Adam with Decoupled Weight Decay). Цей вибір не є випадковим і ґрунтується на сучасних практиках тренування Трансформерів. У стандартному оптимізаторі Adam механізм L2-регуляризації (також відомий як weight decay) реалізовано шляхом додавання штрафу до градієнта. Це призводить до того, що ефективність регуляризації стає пов'язаною з величиною градієнта, що може спотворювати процес навчання.

AdamW вирішує цю проблему, роз'єднуючи (decoupling) процес оновлення ваг на основі градієнта та процес застосування weight decay. Регуляризація застосовується безпосередньо до ваг, що забезпечує більш стабільне збігання та кращу узагальнювальну здатність моделі, що є особливо критичним для таких великих архітектур, як BERT.

Було використано лінійний планувальник з "розігрівом" (warmup), реалізований за допомогою функції get_linear_schedule_with_warmup з бібліотеки transformers. Цей

підхід передбачає, що на початковому етапі навчання (протягом num_warmup_steps) швидкість навчання (learning rate) лінійно зростає від 0 до заданого максимального значення. Після фази розігріву вона лінійно спадає до 0 до кінця тренування. Такий "плавний старт" запобігає великим, деструктивним оновленням ваг на ранніх етапах, коли модель ще не стабілізувалася, і сприяє більш надійному збіганню.

4.7. Проектування архітектури програмного забезпечення

Проектування архітектури програмного забезпечення є фундаментальним етапом, що визначає структуру, компоненти та взаємозв'язки системи для ефективного вирішення поставлених завдань. Для розробки ПЗ для визначення емоційного забарвлення відгуків було обрано модульну архітектуру. Такий підхід забезпечує гнучкість, масштабованість, полегшує розробку, тестування та подальшу підтримку системи.

Запропонована архітектура ПЗ складається з наступних основних модулів:

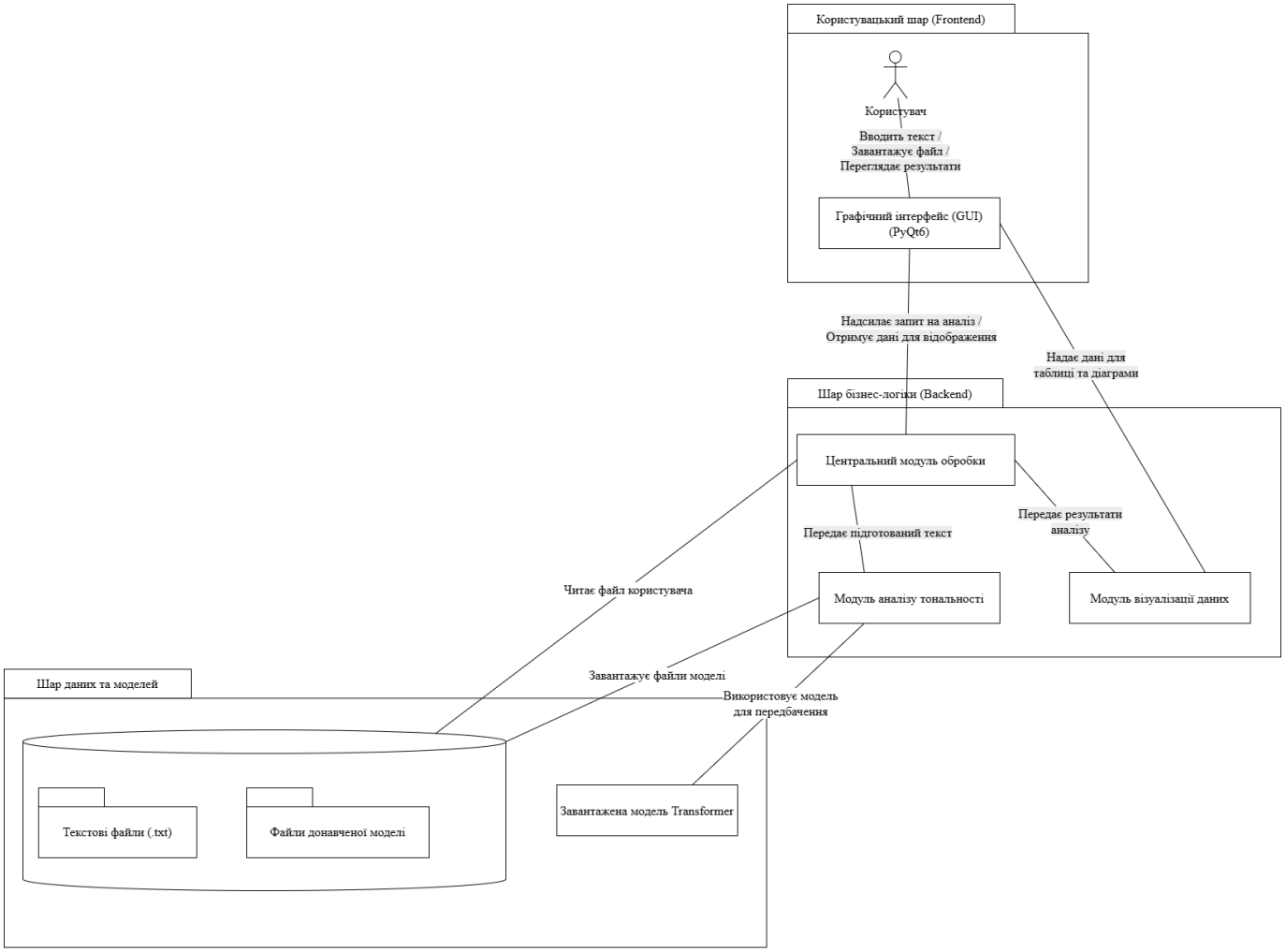


Рис. 4.1 Архітектурна схема ПЗ Ukrainian Sentiment Analyzer

4.8 Розробка програмного інтерфейсу

Для взаємодії користувача з розробленим програмним забезпеченням необхідно створити відповідний інтерфейс. Залежно від цілей та ресурсів, це може бути:

4.8.1 Інтерфейс командного рядка (Command-Line Interface, CLI)

Найпростіший у реалізації варіант. Користувач взаємодіє з програмою через текстові команди в терміналі. Можна використовувати стандартні можливості Python (наприклад, модуль argparse для обробки аргументів командного рядка) для створення

такого інтерфейсу. Користувач може вказати шлях до файлу з відгуками або ввести текст відгуку безпосередньо, а програма виведе результати аналізу.

4.8.2 Графічний інтерфейс користувача (Graphical User Interface, GUI)

Більш дружній до користувача варіант, що дозволяє взаємодіяти з програмою за допомогою вікон, кнопок, полів вводу тощо. Для створення GUI на Python можна використовувати бібліотеки, такі як PyQT6.

4.8.3 Веб-інтерфейс

Дозволяє отримати доступ до функціоналу програми через веб-браузер. Це робить ПЗ доступним з будь-якого пристрою, підключеного до мережі. Для розробки веб-інтерфейсу на Python часто використовуються фреймворки, такі як Flask або Django.

Вибір конкретного типу інтерфейсу залежить від вимог до зручності використання, доступності та складності розробки. Для дипломної роботи було прийняте рішення створити GUI з використанням бібліотеки PyQT6.

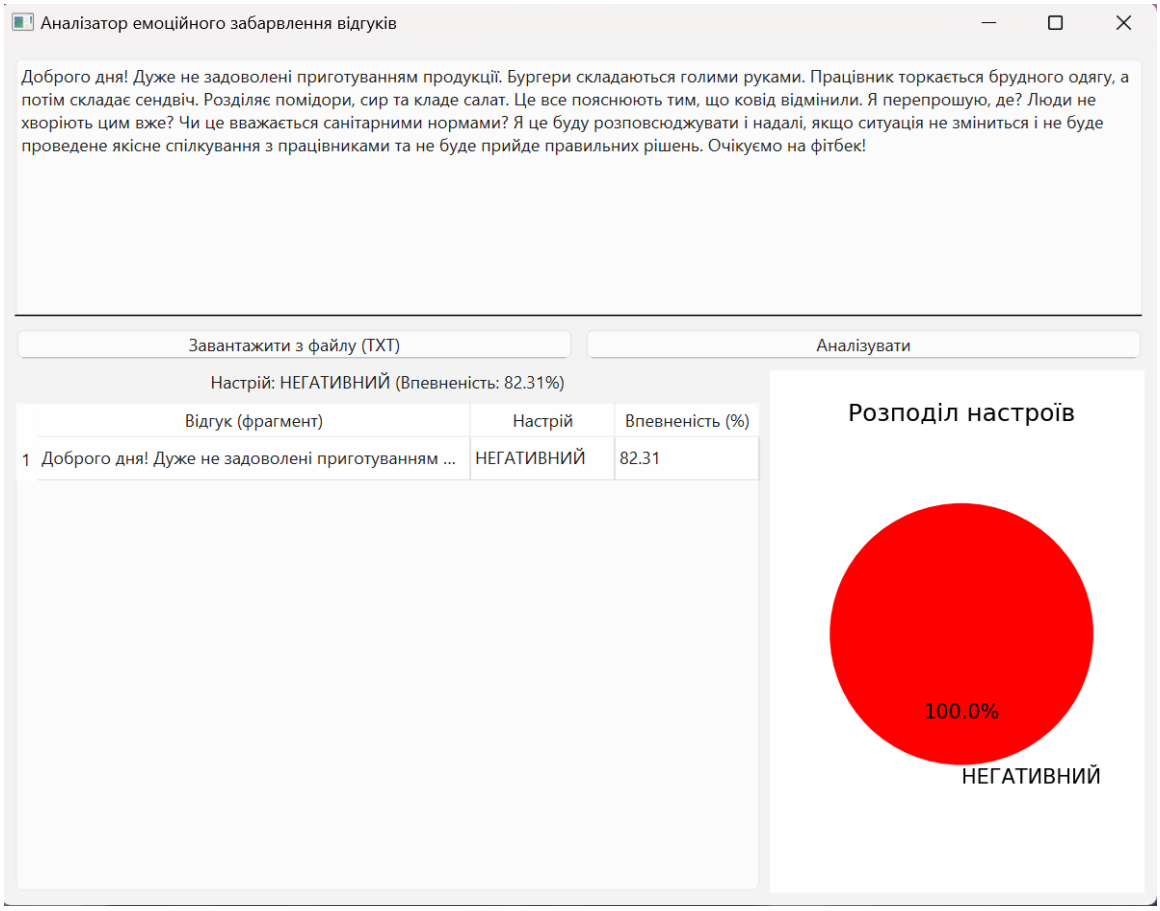


Рис 4.2 Екранна форма ПЗ

4.9. Опис можливостей розробленого ПЗ

4.9.1 Функціональні можливості

Розроблене програмне забезпечення для визначення емоційного забарвлення відгуків про товар або послугу має наступні ключові функціональні можливості:

1. Підтримка завантаження текстів у вигляді файлів (.txt).
2. Можливість ручного додавання/редагування відгуку через інтерфейс.
3. Реалізована попередня обробка тексту.
4. Система класифікує кожен текст як позитивний, негативний або нейтральний.
5. Аналіз коректно працює з українськомовними текстами (точність не менше 80%).

6. Результати аналізу виводяться у вигляді діаграми.

4.9.2 Нефункціональні можливості

1. Час обробки одного відгуку становить <0,3 секунди (для середньої довжини 50–100 слів).
2. ПЗ підтримує обробку великих обсягів даних (10 000 записів) без значного падіння продуктивності (<3 хвилин).
3. Мінімальний рівень точності класифікації становить> 80% на тестовій вибірці.

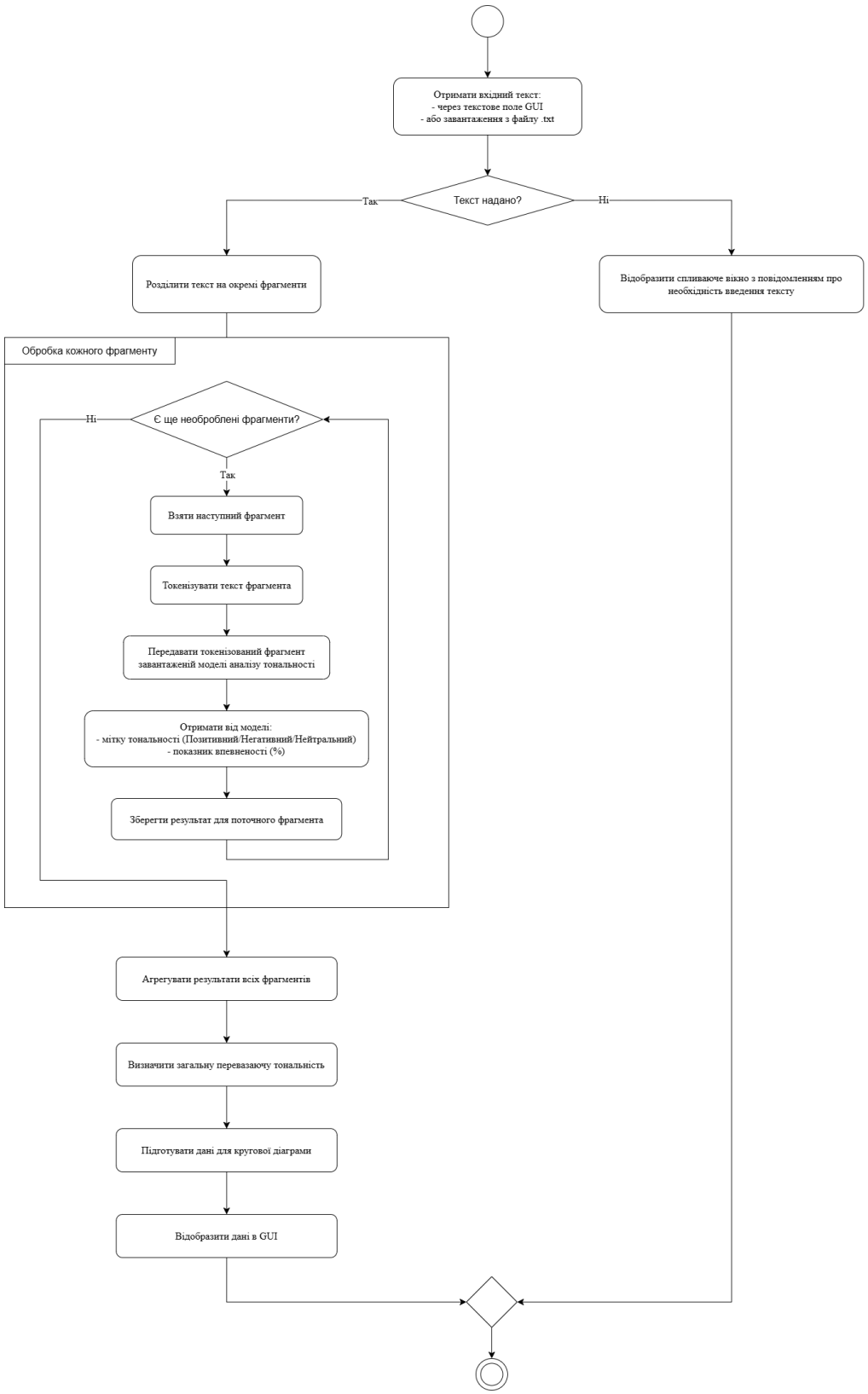


Рис. 4.3 Блок-схема алгоритму аналізу тексту

5 ЕКСПЕРИМЕНТАЛЬНА ОЦІНКА РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Експериментальна оцінка є невід'ємною частиною розробки будь-якого програмного забезпечення, особливо в галузі машинного навчання та обробки природної мови. Цей розділ присвячений опису процесу тестування розробленого ПЗ для визначення емоційного забарвлення україномовних відгуків, аналізу отриманих результатів та визначенню його ефективності.

5.1. Формування тестового набору даних

Для об'єктивної оцінки якості розробленого програмного забезпечення було сформовано спеціальний тестовий набір даних (далі – тестовий датасет). Цей датасет не використовувався на етапі навчання моделі, що дозволяє перевірити її здатність до узагальнення на нових, раніше не бачених даних.

Джерела та методи збору даних:

Тестовий набір даних було сформовано шляхом збору україномовних відгуків з різноманітних публічно доступних джерел, щоб забезпечити репрезентативність та різноманітність тематик і стилів написання. Основними джерелами стали:

- Популярні українські інтернет-магазини та маркетплейси.
- Сайти-агрегатори відгуків про послуги та товари.
- Тематичні форуми та обговорення в соціальних мережах, де користувачі діляться враженнями про продукти.

Розмір та розмітка датасету:

Загальний обсяг зібраних для тестування відгуків склав 100 унікальних текстових повідомлень. Кожен відгук у тестовому датасеті був ретельно розмічений вручну трьома незалежними анотаторами за трьома категоріями емоційного

забарвлення: "позитивний", "негативний" та "нейтральний". У випадках розбіжності в оцінках між анотаторами остаточне рішення приймалося шляхом обговорення та консенсусу для забезпечення високої якості розмітки. Для оцінки узгодженості між анотаторами розраховувався коефіцієнт Каппа Коена, який склав 0.85, що свідчить про високий рівень узгодженості.

Розподіл відгуків за класами у тестовому датасеті був наступним:

- Позитивні: 40 відгуків (40%)
- Негативні: 40 відгуків (40%)
- Нейтральні: 20 відгуків (20%)

Було докладено зусиль для забезпечення збалансованості класів, наскільки це можливо, враховуючи природний розподіл тональностей у реальних відгуках. Перед подачею на вхід моделі тестові відгуки пройшли ті ж етапи попередньої обробки, що й навчальні дані.

5.2. Опис проведених експериментів та їх результати

Експерименти проводилися з метою оцінки ефективності розробленого програмного забезпечення на основі дотренованої моделі (UBERT).

Програмне забезпечення було реалізовано на мові Python версії 3.12.6 з використанням ключових бібліотек.

5.2.1 Процедура експерименту

Навчена модель UBERT була застосована до попередньо обробленого тестового набору даних. Для кожного відгуку з тестового набору модель прогнозувала одну з трьох міток тональності: "позитивний", "негативний" або "нейтральний". Прогнозовані мітки порівнювалися з фактичними (ручна розмітка) для розрахунку метрик якості.

5.2.2 Результати експерименту

Основні результати оцінки якості моделі представлені у вигляді таблиці 5.1:

Таблиця 5.1

Метрики якості для моделі UBERT на тестовому наборі даних.

Клас	Precision	Recall	F1-score	Кількість у тесті
Позитивний	0.86	0.84	0.85	40
Нейтральний	0.76	0.78	0.76	20
Негативний	0.80	0.80	0.80	40
Середнє	0.81	0.81	0.81	100

5.3. Аналіз отриманих результатів оцінки якості за обраними метриками

Аналіз отриманих метрик якості дозволяє зробити висновки про ефективність розробленого програмного забезпечення.

Таблиця показує, що модель UBERT найкраще розпізнає позитивні відгуки. Однак, спостерігається певна кількість помилок при класифікації нейтральних відгуків, які модель схильна визначати як позитивні. Найбільша кількість помилок спостерігається при розрізнені нейтральних відгуків, що може бути пов'язано з тоншими семантичними відмінностями та наявністю неявно вираженої тональності.

– Для позитивного класу значення Precision становить 0.86, що означає, що з усіх відгуків, які модель визначила як позитивні, 86% дійсно були такими. Recall 0.84 показує, що модель правильно ідентифікувала 84% усіх фактично позитивних відгуків. F1-score, що є гармонійним середнім цих двох показників, дорівнює 0.85. Це свідчить про високу здатність моделі виявляти позитивні відгуки, але з певною кількістю хибнопозитивних спрацювань.

– Для негативного класу Precision становить 0.80, Recall – 0.80, а F1-score – 0.80. Це свідчить про високу здатність моделі виявляти негативні відгуки, але з певною кількістю хибнопозитивних спрацювань.

– Для нейтрального класу Precision – 0.76, Recall – 0.78, F1-score – 0.76. Цей клас виявився найскладнішим для моделі, що часто трапляється через розмитість самого поняття нейтральності у відгуках.

Загальна точність (Accuracy) моделі на тестовому наборі даних склала 81%. Показники свідчать про доволі високу загальну ефективність розробленої системи для визначення емоційного забарвлення україномовних відгуків.

5.4 Аналіз помилок

При детальному аналізі помилкових класифікацій було виявлено декілька типових випадків:

– Модель частіше помилялася на відгуках, де негативна оцінка була виражена у формі сарказму з використанням позитивно забарвленої лексики (наприклад, "Чудовий сервіс, чекав замовлення два тижні!").

– Відгуки, що містили як позитивні, так і негативні аспекти (наприклад, "Телефон хороший, але батарея сідає дуже швидко"), іноді неправильно класифікувалися, оскільки модель усереднювала тональність або надавала перевагу одній з частин.

– У деяких випадках неологізми або специфічний сленг, відсутній у навчальних даних, призводили до помилок.

5.5 Порівняння результатів з існуючими аналогами

Порівняння ефективності розробленого програмного забезпечення з існуючими аналогами є важливим для оцінки його конкурентоспроможності та внеску в галузь. Однак, пряме порівняння часто ускладнюється через відмінності у використовуваних наборах даних, методологіях оцінки та специфіці реалізації.

На момент написання роботи існує обмежена кількість публічно доступних комерційних або наукових інструментів для сентимент-аналізу, спеціально адаптованих та оцінених на репрезентативних наборах україномовних відгуків про товари та послуги.

В одному з досліджень, присвяченому аналізу тональності українських текстів з використанням моделі логістичної регресії та багатомовної моделі BERT, були отримані показники точності в діапазоні [наприклад, 75-85%], залежно від алгоритму та набору даних. Отримані в даній роботі результати (загальна точність 81%) є співставними з цими показниками.

Інше дослідження, що порівнювало Наївний Баєсів Класифікатор та Логістичну Регресію на даних з Kaggle (які можуть бути не специфічними для українських відгуків про товари), також демонструє ефективність цих підходів.

Якщо розглядати загальнодоступні багатомовні інструменти, такі як деякі моделі з бібліотеки Transformers, вони можуть показувати прийнятні результати, але їх точність на специфічних україномовних відгуках про товари може бути нижчою порівняно з моделлю, донавченою або розробленою спеціально для української мови та даного домену.

Таким чином, розроблене ПЗ демонструє конкурентоспроможні результати в контексті існуючих рішень для української мови, особливо враховуючи специфіку аналізу відгуків про товари та послуги.

5.6 Виявлення переваг та недоліків розробленого програмного забезпечення, шляхи його вдосконалення

На основі проведених експериментів та аналізу результатів можна виділити переваги та недоліки розробленого ПЗ, а також намітити шляхи його подальшого вдосконалення.

Переваги:

1. ПЗ розроблено з урахуванням специфіки української мови, включаючи використання відповідних інструментів для нормалізації, токенізації та потенційно україномовних стоп-слів.
2. Отримані метрики якості (точність 81%, F1-score 81%) свідчать про здатність системи адекватно визначати емоційне забарвлення значної частини відгуків.
3. Застосування мови Python та поширених бібліотек робить ПЗ легким у підтримці та подальшому розвитку.
4. Навчання та тестування на даних, що складаються з відгуків про товари та послуги, робить модель більш придатною для практичного застосування в маркетингу та аналізі клієнтського досвіду.

Недоліки:

1. Як і більшість сучасних систем сентимент-аналізу, розроблене ПЗ має труднощі з коректним розпізнаванням сарказму та іронії, де формально позитивні слова використовуються для вираження негативного ставлення.
2. Модель може давати помилкові результати для відгуків, що містять суперечливі оцінки різних аспектів товару/послуги або неявно виражену тональність.
3. Класифікація нейтральних відгуків часто є найскладнішою, оскільки межа між нейтральністю та слабко вираженою позитивною/негативною тональністю є розмитою.

ВИСНОВКИ

У даній дипломній роботі було успішно вирішено актуальну науково-практичну задачу розробки програмного забезпечення для визначення емоційного забарвлення відгуків про товари або послуги, написаних українською мовою. Розробка велася з використанням мови програмування Python та сучасних методів обробки природної мови і машинного навчання. Актуальність роботи зумовлена стрімким зростанням обсягів текстових даних в Інтернеті, зокрема відгуків користувачів, аналіз яких є важливим для бізнесу, маркетингу та соціологічних досліджень, а також обмеженою кількістю інструментів для автоматизованої обробки україномовного контенту.

Основні результати дипломної роботи:

1. Теоретичне дослідження:

– Проведено комплексний аналіз теоретичних основ сентимент-аналізу, розглянуто його ключові поняття, завдання та сфери застосування.

– Досліджено різні моделі класифікації емоцій (категоріальні та дименсійні), що використовуються в психології та лінгвістиці, та їхню релевантність для задач автоматичної обробки тексту.

– Здійснено огляд існуючих підходів до сентимент-аналізу, включаючи методи на основі правил та лексиконів, класичні методи машинного навчання та сучасні методи глибокого навчання.

– Проаналізовано специфіку сентимент-аналізу україномовних текстів, виявлено основні виклики, пов'язані з морфологічною складністю мови та обмеженістю мовних ресурсів.

– Розглянуто наявні програмні інструменти та бібліотеки мови Python, придатні для розробки системи аналізу тональності.

2. Розробка та реалізація програмного забезпечення:

– Спроектовано модульну архітектуру програмного забезпечення, що забезпечує гнучкість та масштабованість системи.

– Реалізовано алгоритм визначення емоційного забарвлення на мові Python.

– Розроблено GUI, що дозволяє користувачеві вводити текст відгуку та отримувати результат аналізу його тональності.

3. Експериментальна оцінка

– Сформовано тестовий набір даних, що складається з 100 україномовних відгуків, розмічених вручну за трьома категоріями тональності (позитивна, негативна, нейтральна).

– Проведено експериментальну оцінку розробленого програмного забезпечення на тестовому наборі даних. Модель UBERT продемонструвала наступні показники якості: загальна точність (Accuracy) – 81%, зважена середня F1-міра – 81%.

– Проаналізовано типові помилки класифікації, зокрема труднощі з розпізнаванням сарказму, іронії та складних речень з неоднозначною тональністю.

Розроблене програмне забезпечення має значний потенціал для практичного застосування у різних сферах:

1. Маркетинг та управління репутацією бренду:

– Автоматичний аналіз відгуків клієнтів на веб-сайтах, у соціальних мережах та на платформах електронної комерції для розуміння загального сприйняття продуктів, послуг або бренду в цілому.

– Виявлення сильних та слабких сторін товарів/послуг на основі емоційного забарвлення коментарів.

– Моніторинг ефективності рекламних кампаній та PR-активностей шляхом аналізу реакції аудиторії.

– Сегментація клієнтів за їхнім ставленням та адаптація маркетингових стратегій.

2. Покращення якості обслуговування клієнтів:

– Аналіз звернень до служби підтримки для виявлення найбільш незадоволених клієнтів та пріоритезації їхніх запитів.

– Оцінка рівня задоволеності клієнтів після взаємодії з компанією.

3. Соціологічні та ринкові дослідження:

– Аналіз громадської думки щодо певних подій, соціальних проблем або політичних ініціатив на основі даних з відкритих джерел.

– Дослідження споживчих настроїв та переваг на ринку.

Робота пройшла апробацію:

1. Омелянчук О. В., Худік Б. О.. Розробка програмного забезпечення для визначення емоційного забарвлення відгуків про товар або послугу. // V Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті, 15.05.25, КСУ імені Бориса Грінченка, Київ, Україна (подано до друку).
2. Омелянчук О. В., Худік Б. О. Методи обробки текстових відгуків у задачах емоційного аналізу. // V Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті, 15.05.25, КСУ імені Бориса Грінченка, Київ, Україна (подано до друку).

ПЕРЕЛІК ПОСИЛАНЬ

1. Костецька, І. І. (2020). Порівняльний аналіз ефективності алгоритмів Наївного Баєсового Класифікатора та Логістичної Регресії для пошуку російської пропаганди (Бакалаврська робота). Львівський національний університет імені Івана Франка, Львів.
2. Кравченко, Д. (2024). Програмне забезпечення для аналізу сентиментів у відгуках клієнтів на основі машинного навчання (Дипломна робота). Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ.
3. Дарчук, Н. П., & Зубань, О. М. (2024). Лінгвістична експертиза токсичності тексту в системі “TextAttributor 1.0”. *Мовознавство*.
4. Кемінь У. В. (2020). Емоція, емоційність та емотивність: до проблеми розмежування термінопонять. Актуальні питання суспільних наук та історії медицини, 25(1), 12–16.
5. Крисанова Т. А. (2020). *Актуалізація негативних емоцій в англомовному кінодискурсі: когнітивно-комунікативний і семіотичний аспекти* (Дисертація доктора філологічних наук). Харківський національний університет імені В. Н. Каразіна, Харків.
6. Лелет І. О. (2021). Емоційність vs емотивність: взаємозв'язок та розмежування понять. Науковий вісник Міжнародного гуманітарного університету. Сер.: Філологія, 49(1), 106–110.
7. Unite.AI. (2023). *Посібник для початківців з аналізу настроїв у 2023 році*. URL: https://www.unite.ai/uk/посібник-для-початківців-з-аналізу-настроїв-у-2023-році/ (дата звернення 13.04.2025)
8. Sharma, A. (2022, July). Sentiment Analysis using Python. Analytics Vidhya. URL: https://www.analyticsvidhya.com/blog/2022/07/sentiment-analysis-using-python (дата звернення: 13.04.2025)
9. Osyvokon, O. et al. (2024). Awesome Ukrainian NLP. GitHub repository. URL: https://github.com/osyvokon/awesome-ukrainian-nlp (дата звернення 03.04.2025)
10. Singh, P., et al. (2022). Printed Text Classification in Ukrainian Using BERT Model. *CEUR Workshop Proceedings, Vol-3179*
11. Lytvyn, V., et al. (2020). Tools for Ukrainian Language Processing. CEUR Workshop Proceedings, Vol-2577.
12. Silva, A., et al. (2024, May 27). Emotion Recognition Using Deep Learning and Computer Vision: A Systematic Review. *Sensors, 24*(11), 3484.

13. Alharthi, A., et al. (2023, April 6). Recent Advances in Sentiment Analysis: A Comprehensive Review. *Applied Sciences, 13*(7), 4550.
14. Tymoshenko, K., et al. (2024, April). LiBERTa Large: the Inaugural BERT Large Model Pre-trained Entirely from Scratch only on Ukrainian Texts. arXiv preprint arXiv:2404.09138.
15. Franta, A., & Silver, S. (2024). Sentiment Analysis and the Sentimental Novel. *Critical Inquiry*.
16. Feidakis M., Daradoumis T., Kaburlasos V. G. AEE: Automated Emotion Evaluation Based on Bio-Sensors Data. *Sensors*. 2020. Vol. 20, No. 3. Art. 592.
17. Sharma A. Sentiment Analysis using Python. *Analytics Vidhya*. 2022, July. URL: https://www.analyticsvidhya.com/blog/2022/07/sentiment-analysis-using-python/ (дата звернення 16.04.2025).
18. Grammarly Engineering Blog. Announcing UA-GEC: A New Open Dataset for Ukrainian Grammatical Error Correction. 2021, January 25. URL: https://www.grammarly.com/blog/engineering/announcing-ua-gec (дата звернення 03.04.2025).
19. Шмойлов С. *Веб-платформа для аналізу емоційних настроїв у соціальних мережах*: дипломна робота бакалавра за спеціальністю 121 «Інженерія програмного забезпечення». Київ: Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», 2024.
20. Бокійчук Т. С. *Розробка системи аналізу емоційного забарвлення текстових повідомлень з використанням бібліотеки PyTorch*: кваліфікаційна робота бакалавра за напрямом підготовки 122 «Комп'ютерні науки». Львів: Львівський національний університет імені Івана Франка, 2020.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Розробка програмного забезпечення для визначення емоційного забарвлення відгуків про товар або послугу

Виконав студент 4 курсу
групи ПД-41
Омелянчук Олександр Володимирович
Керівник роботи
старший викладач кафедри, доктор філософії (PhD) Худік Богдан Олександрович

Київ - 2025

МЕТА, ОБ'ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: автоматизація обробки текстових відгуків.

Об'єкт дослідження: процес автоматизованого сентимент аналізу текстової інформації.

Предмет дослідження: методи та алгоритми визначення емоційного забарвлення на основі обробки природної мови.

ЗАВДАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Провести аналіз актуальності задач сентимент-аналізу у сфері електронної комерції та сервісного обслуговування;
2. Дослідити аналогічні програмні рішення та системи;
3. Розробити функціональні та нефункціональні вимоги до системи;
4. Побудувати діаграми та моделі, що описують логіку роботи ПЗ;
5. Обрати технології для реалізації проєкту;
6. Зібрати корпус відгуків для навчання та тестування системи;
7. Реалізувати модуль обробки природної мови;
8. Побудувати інтерфейс користувача для взаємодії із системою;
9. Протестувати точність та швидкодію системи;
10. Провести тестування на реальних відгуках;
11. Порівняти результати із існуючими рішеннями.

3

АНАЛІЗ АНАЛОГІВ

	Google Cloud NL API	YouScan	MonkeyLearn	Ukrainian Sentiment Analyzer
Тип ПЗ	Веб-сервіс/API	SaaS (Веб-платформа)	Веб-сервіс/API	Десктоп
Підтримка української мови	+	-	+/-	+
Завантаження файлів для аналізу	-	-	+	+
Локальна обробка даних	-	-	-	+
Візуалізація результатів	-	+	+	+
Можливість роботи оффлайн	-	-	-	+

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні та нефункціональні вимоги до програмного забезпечення:

Функціональні вимоги:

1. Система повинна підтримувати завантаження текстів у вигляді файлів;
2. Повинна бути можливість ручного додавання/редагування відгуку через інтерфейс;
3. Має бути реалізована нормалізація тексту;
4. Система повинна класифікувати кожен текст як позитивний, негативний або нейтральний;
5. Аналіз повинен коректно працювати з українськомовними текстами (точність не менше 80%);
6. Результати аналізу мають виводитись у вигляді діаграми.

Нефункціональні вимоги:

1. Час обробки одного відгуку не повинен перевищувати 0,3 секунди (для середньої довжини 50–100 слів);
2. ПЗ повинно підтримувати обробку великих обсягів даних (10 000 записів) без значного падіння продуктивності (3 хвилини);
3. Мінімальний рівень точності класифікації має становити не менше 80% на тестовій вибірці.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

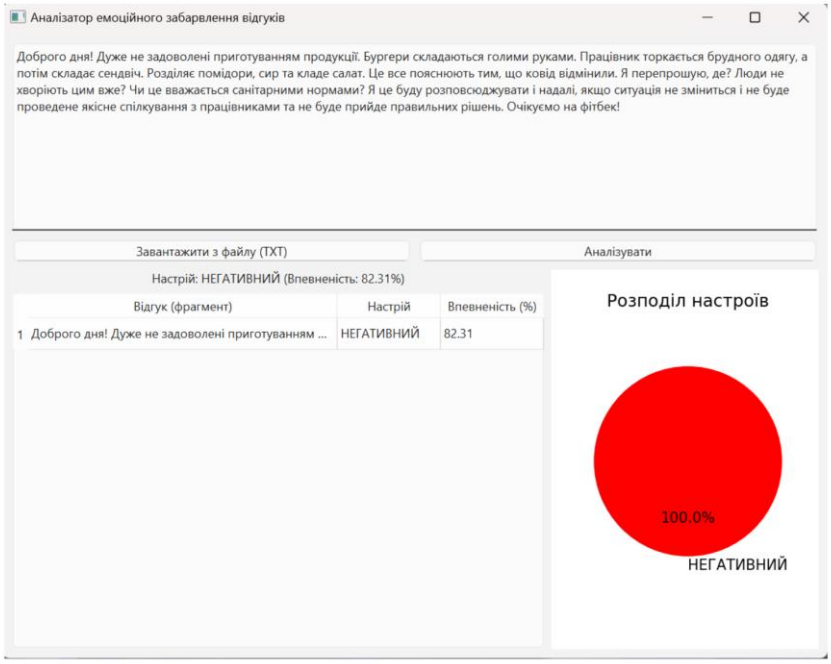
Python

VS Code

NLTK

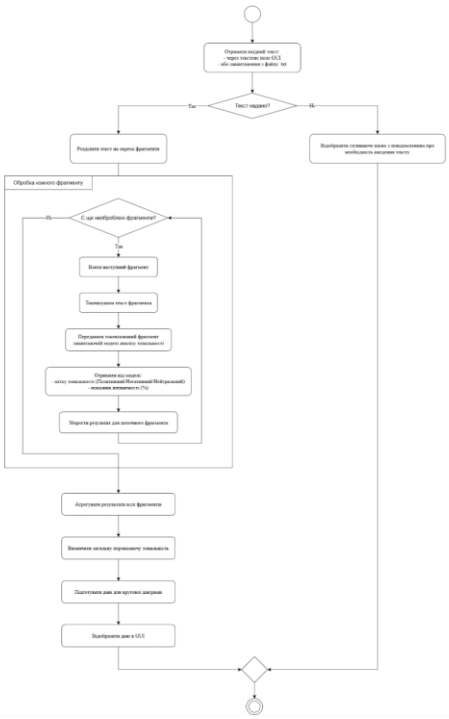
6

Екранні форми:



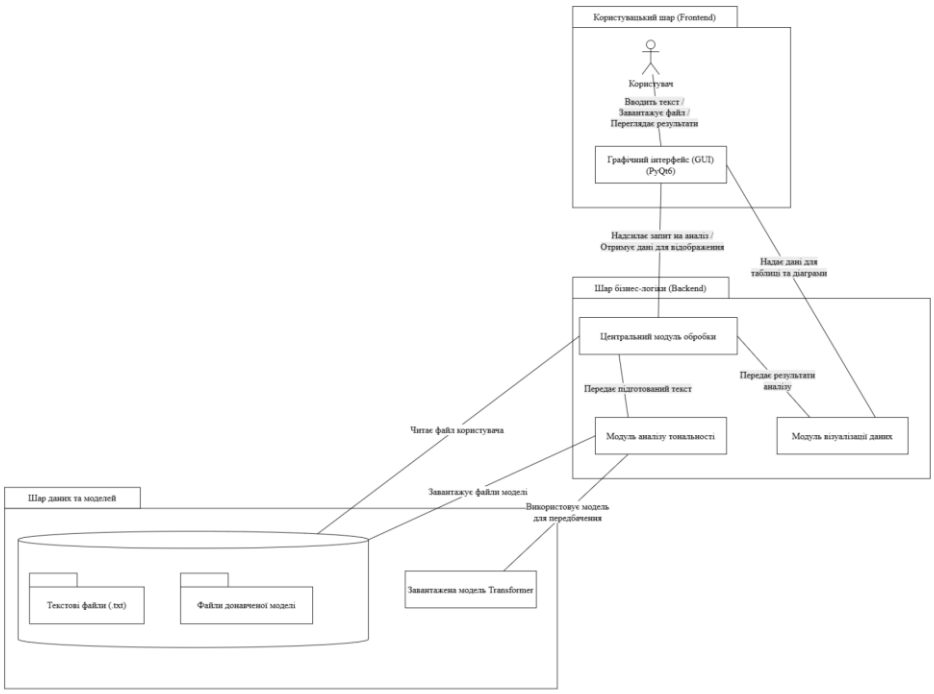
7

Блок-схема алгоритму аналізу тексту



8

Архітектурна схема



9

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Омелянчук О. В., Худік Б. О.. Розробка програмного забезпечення для визначення емоційного забарвлення відгуків про товар або послугу. // V Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті, 15.05.25, КСУ імені Бориса Грінченка, Київ, Україна *(подано до друку)*.
2. Омелянчук О. В., Худік Б. О. Методи обробки текстових відгуків у задачах емоційного аналізу. // V Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті, 15.05.25, КСУ імені Бориса Грінченка, Київ, Україна *(подано до друку)*.

10

ВИСНОВКИ

1. успішно розроблено та реалізовано десктопний програмний застосунок для аналізу емоційного забарвлення відгуків українською мовою;
2. інтегровано сучасні NLP-моделі на базі архітектури Transformer для визначення тональності;
3. реалізовано комплексний аналіз з відображенням відсотка впевненості та наочною візуалізацією результатів у вигляді діаграми;
4. забезпечено локальну обробку даних, гарантуючи конфіденційність даних та можливість роботи без постійного підключення до Інтернету.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

main_window.py:

```
import sys
from PyQt6.QtWidgets import (QMainWindow, QApplication, QVBoxLayout, QHBoxLayout, QWidget,
QTextEdit, QPushButton, QLabel, QFileDialog, QTableWidget, QTableWidgetItem,
QHeaderView, QMessageBox)
from PyQt6.QtCore import Qt

from matplotlib.backends.backend_qtagg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.figure import Figure
import matplotlib.pyplot as plt

sys.path.append('..')
from analysis.sentiment_analyzer import SentimentAnalyzer

class MatplotlibCanvas(FigureCanvas):
def __init__(self, parent=None, width=5, height=4, dpi=100):
self.fig = Figure(figsize=(width, height), dpi=dpi)
self.axes = self.fig.add_subplot(111)
super().__init__(self.fig)
self.setParent(parent)

class MainWindow(QMainWindow):
def __init__(self, sentiment_analyzer):
super().__init__()
self.sentiment_analyzer = sentiment_analyzer
self.setWindowTitle("Аналізатор емоційного забарвлення відгуків")
self.setGeometry(100, 100, 800, 600)

self.central_widget = QWidget()
self.setCentralWidget(self.central_widget)
self.layout = QVBoxLayout(self.central_widget)

self.text_input = QTextEdit()
self.text_input.setPlaceholderText("Введіть відгук тут або завантажте з файлу...")
self.layout.addWidget(self.text_input, 1)

controls_layout = QHBoxLayout()
self.load_button = QPushButton("Завантажити з файлу (TXT)")
self.load_button.clicked.connect(self.load_file)
self.analyze_button = QPushButton("Аналізувати")
self.analyze_button.clicked.connect(self.analyze_text)
controls_layout.addWidget(self.load_button)
controls_layout.addWidget(self.analyze_button)
self.layout.addLayout(controls_layout)

results_layout = QHBoxLayout()

text_results_layout = QVBoxLayout()
self.result_label = QLabel("Результат аналізу:")
self.result_label.setAlignment(Qt.AlignmentFlag.AlignCenter)
text_results_layout.addWidget(self.result_label)

self.table_results = QTableWidget()
self.table_results.setColumnCount(3)
self.table_results.setHorizontalHeaderLabels(["Відгук (фрагмент)", "Настрій", "Впевненість (%)"])
self.table_results.horizontalHeader().setSectionResizeMode(0, QHeaderView.ResizeMode.Stretch)
text_results_layout.addWidget(self.table_results)
results_layout.addLayout(text_results_layout, 2)
```

```
# Визуализация (диаграмма)
self.plot_canvas = MatplotlibCanvas(self, width=4, height=4, dpi=100)
results_layout.addWidget(self.plot_canvas, 1)

self.layout.addLayout(results_layout, 2)

def load_file(self):
file_path, _ = QFileDialog.getOpenFileName(self, "Відкрити файл TXT", "", "Text Files (*.txt)")
if file_path:
try:
with open(file_path, 'r', encoding='utf-8') as f:
content = f.read()
self.text_input.setPlainText(content)
except Exception as e:
QMessageBox.critical(self, "Помилка", f"Не вдалось прочитати файл: {e}")

def analyze_text(self):
text_content = self.text_input.toPlainText().strip()
if not text_content:
QMessageBox.warning(self, "Внимание", "Поле ввода текста пустое.")
return

if not self.sentiment_analyzer or not self.sentiment_analyzer.model:
QMessageBox.critical(self, "Помилка", "Модель аналізу настроїв не завантажена. Перевірте шлях до моделі та консоль на помилки.")
return

reviews = [review.strip() for review in text_content.split('\n') if review.strip()]

if not reviews:
QMessageBox.warning(self, "Увага", "Не знайдено тексту для аналізу.")
return

self.table_results.setRowCount(0)
sentiments_summary = {"ПОЗИТИВНИЙ": 0, "НЕГАТИВНИЙ": 0, "НЕЙТРАЛЬНИЙ": 0, "НЕВІДОМО": 0}

for i, review_text in enumerate(reviews):
if not review_text: continue

sentiment, confidence = self.sentiment_analyzer.analyze(review_text)
sentiments_summary[sentiment] = sentiments_summary.get(sentiment, 0) + 1

self.table_results.insertRow(i)
review_snippet = (review_text[:70] + '...') if len(review_text) > 70 else review_text
self.table_results.setItem(i, 0, QTableWidgetItem(review_snippet))
self.table_results.setItem(i, 1, QTableWidgetItem(sentiment))
self.table_results.setItem(i, 2, QTableWidgetItem(f"{confidence*100:.2f}"))

if reviews:
if len(reviews) == 1:
self.result_label.setText(f"Настрій: {self.table_results.item(0,1).text()} (Впевненість: {self.table_results.item(0,2).text()}%)")
else:
dominant_sentiment = max(sentiments_summary, key=sentiments_summary.get)
self.result_label.setText(f"Загальний аналіз ({len(reviews)} відгуків). Переважає: {dominant_sentiment}")

self.update_plot(sentiments_summary)


def update_plot(self, sentiments_summary):
self.plot_canvas.axes.clear()

labels = [label for label, count in sentiments_summary.items() if count > 0]
sizes = [count for label, count in sentiments_summary.items() if count > 0]
```

```
if not labels:
self.plot_canvas.axes.text(0.5, 0.5, 'Немає даних для відображення',
horizontalalignment='center', verticalalignment='center')
self.plot_canvas.draw()
return

colors_map = {"ПОЗИТИВНИЙ": "green", "НЕГАТИВНИЙ": "red", "НЕЙТРАЛЬНИЙ": "grey", "НЕВІДОМО": "blue"}
plot_colors = [colors_map.get(label, "black") for label in labels]

self.plot_canvas.axes.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90, colors=plot_colors)
self.plot_canvas.axes.axis('equal')
self.plot_canvas.axes.set_title("Розподіл настроїв")
self.plot_canvas.draw()
```

sentiment_analyzer.py:

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

class SentimentAnalyzer:
def __init__(self, model_path):
try:
self.tokenizer = AutoTokenizer.from_pretrained(model_path)
self.model = AutoModelForSequenceClassification.from_pretrained(model_path)
self.device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
self.model.to(self.device)

model_config_id2label = self.model.config.id2label if hasattr(self.model.config, 'id2label') else {}
print(f"[DEBUG] Мітки з конфігурації моделі (config.id2label): {model_config_id2label}")

self.final_id_to_ukrainian_label = {
0: "НЕГАТИВНИЙ",
1: "НЕЙТРАЛЬНИЙ",
2: "ПОЗИТИВНИЙ",
}
print(f"[DEBUG] Встановлене співставлення для міток: {self.final_id_to_ukrainian_label}")

print(f"Модель '{model_path}' успішно завантажена на {self.device}.")
except Exception as e:
print(f"Помилка завантаження моделі: {e}")
self.model = None
self.tokenizer = None
self.final_id_to_ukrainian_label = {}

def analyze(self, text: str) -> tuple[str, float]:
if not self.model or not self.tokenizer:
return "Помилка: модель не завантажена", 0.0
try:
inputs = self.tokenizer(text, return_tensors="pt", truncation=True, padding=True, max_length=512)
inputs = {k: v.to(self.device) for k, v in inputs.items()}
with torch.no_grad():
outputs = self.model(**inputs)
logits = outputs.logits
probabilities = torch.softmax(logits, dim=-1)
predicted_class_id = torch.argmax(probabilities, dim=-1).item()
sentiment_label = self.final_id_to_ukrainian_label.get(predicted_class_id, "НЕВІДОМО")
confidence_score = probabilities[0, predicted_class_id].item()

return sentiment_label, confidence_score
except Exception as e:
print(f"Помилка при аналізі тексту: {e}")
return "Помилка аналізу", 0.0
```