МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ

Програмний модуль визначення емоційного забарвлення коментарів у соціальних мережах

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма **«Аналітика даних»**

Освітній рівень: бакалавр

Виконала: студентка 4 курсу, групи АнД-41
Пономарьова Д.А. _____
(прізвище та ініціали)

Керівник Мінаєва Ю. І _____
(прізвище та ініціали)
к.т.н, доц. _____
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол №11 від 06.06.2022 р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ – 2022

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
інтелектуальних технологій
Іларіонов О.Є.

_____

"___"_____2022 р.

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Пономарьовій Даріні Андріївні

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)

Програмний модуль визначення емоційного забарвлення коментарів в соціальних мережах

затверджена протоколом засідання кафедри від «23» грудня 2021 р. №4

2. Термін здачі студентом закінченого проекту (роботи): 29 травня 2022 року

3. Вихідні дані до проекту (роботи): Статистика визначених в коментарях емоцій

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити):

Аналіз особливостей задачі сентиментального аналізу коментарів у соціальних мережах, проектування веб-застосунку, розробка програмного модуля.

5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій):

Мета дипломного проекту, актуальність та контекстна діаграма програмного модулю розробки (1 слайд), схема архітектури програмного модулю (1 слайд), опис задачі багатоміткової класифікації тексту(1 слайд), опис архітектури трансформерів та нейронної мережі (2 слайди), опис навчального набору даних (1 слайд), оцінка результатів та приклади визначення емоцій (2 слайди) , висновки(1 слайд).

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 15 лютого 2022 року

Керівник _____ / Мінаєва Ю.І. /
(підпис) (ПІБ)

Завдання прийняв до виконання _____ / Пономарьова Д.А. /
(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1.	Опрацювання літератури	15.02.2022 – 01.03.2022	
2.	Робота над розділом 1. Аналітичний огляд предметної області	02.03.2022 – 20.03.2022	
3.	Робота над розділом 2. Програмні рішення	21.03.2022 – 11.04.2022	
4.	Робота над розділом 3. Розробка програмного застосунку	12.04.2022 – 11.05.2022	
5.	Робота над оформленням пояснювальної записки та презентацією	22.05.2022– 29.05.2022	

Студент-дипломник _____ / Пономарьова Д.А. /
(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи _____ / Мінаєва Ю.І. /
(підпис) (ПІБ)

Анотація

Пономарьова Даріна Андріївна виконала випускну кваліфікаційну роботу на тему «Програмний модуль визначення емоційного забарвлення коментарів в соц. мережах» за спеціальністю 122 – «Комп’ютерні науки».

У випускній кваліфікаційній роботі досліджено методи обробки природної мови, розглянуто нейроні мережі, засновані на архітектурі трансформерів, розроблено програмний модуль, що реалізує багатомітковий сентиментальний аналіз для оцінки тональності тексту, і може набути широкого застосування як у визначенні статистики емоцій та ставлення групи людей до певної новини, події або звичайного авторського допису, так і впровадження в більш складні моніторингові системи. Розробка виконана на мові програмування Python.

Ключові слова: сентиментальний аналіз, класифікація тексту, багатоміткова, класифікація, глибоке навчання, нейронні мережі, трансформери, нейронна мережа BERT.

Annotation

A final Bachelor's Thesis: «Sentiment analysis software module of social media comments», 122 Computer Science specialty, was made by **Ponomarova Darina.**

The final qualification work examines natural language processing methods, considers neural networks based on transformer architecture, and developed a software module that implements multilabel sentimental analysis to assess the tone of the text and can be widely used in determining the statistics of emotions and attitudes on news, events, or regular author's posts, and implementation in more sophisticated monitoring systems. Development is performed in the Python programming language.

Keywords: sentiment analysis, text classification, multilabel classification, deep learning, neural networks, transformers, BERT neural network.

Зміст

Вступ

У сучасному світі просто неможливо уявити своє життя без інформаційних технологій, вплив яких на нашу буденність зростає з кожним днем. Все більше сервісів та послуг стають доступними онлайн і взаємодія з користувачами переноситься на простори мережі Інтернет.

Коментарі – є одним із головних видів сучасної онлайн-комунікації та реакції користувача на певний продукт або подію. Можливість аналізу думки клієнтів у соціальних мережах може мати вирішальне значення для бізнесу та дозволить брендам краще розуміти реальні потреби своїх користувачів, що є важливим аспектом на шляху до вдосконалення. Але для реалізації цієї можливості необхідно мати змогу автоматично аналізувати велику кількість неструктурованих текстових даних з різних джерел, обсяг яких з часом буде лише зростати.

І для реалізації даної задачі доречно звернутися до однієї зі сфер NLP (обробки природної мови) – сентиментального аналізу [1], використання якого дозволить, не залучаючи сотні експертів, визначити емоційне забарвлення коментаря.

Саме тому метою даного дипломного проекту є створення програмного модулю визначення емоційного забарвлення коментарів в соц. мережах, що реалізує багатомітковий сентиментальний аналіз для ефективної оцінки тональності тексту, і може набути широкого застосування у визначенні статистики певних дописів або новин, так і впровадження в складні моніторингові системи.

РОЗДІЛ 1. Аналітичний огляд предметної області

1.1 Огляд основних алгоритмів та методів

1.1.1 Задача класифікації

Задача класифікації – один з класичних методів машинного навчання[2]. Це задача розбиття множини об'єктів або спостережень на апріорно задані групи, називані класами, всередині кожної з яких, вони вважаються схожими між собою, та мають приблизно однакові властивості й ознаки. При цьому рішення здійснюється на основі аналізу значень атрибутів (ознак).

Основні методи класифікації поділяються на неконтрольовану (розраховану програмним забезпеченням, без учителя) і контрольовану (керовану людиною, з учителем) класифікацію.

При неконтрольованій класифікації результати об’єднань в групи за загальними характеристиками базуються на програмному аналізі об’єктів без надання користувачем зразків класів. Комп’ютером самостійно використовуються методи для визначення схожих об’єктів і їх групування в окремі класи. Користувач лише вказує алгоритм для застосування програмним забезпеченням та бажану кількість вихідних класів, але не допомагає і не керує процесом класифікації.

На практиці в більшості випадків для навчання моделі використовують контрольовану класифікацію. Під час навчання з учителем алгоритм «тренується» на навчальному наборі даних шляхом ітераційного прогнозування вихідних даних і його коригуванні для знаходження правильного розв’язку задачі класифікації. Моделі навчання з учителем, як правило, є більш точними, ніж моделі неконтрольованого навчання, але вони вимагають попереднього втручання людини для належного позначення даних.

У машинному навчанні класифікація відноситься до проблеми прогнозуючого моделювання, де для кожного прикладу вхідних даних передбачається мітка класу.

Модель використовує навчальний набір даних і розраховує, як найкраще зіставити приклади вхідних даних до певних міток класів. Таким чином, навчальний набір даних повинен бути достатньо репрезентативним для проблеми та містити багато прикладів кожного позначення класу.

1.1.2 Багатоміткова класифікація

Існує декілька основних типів класифікаційних задач і одним із них є багатоміткова класифікація.

Класифікація з кількома мітками виникла з дослідження проблеми категоризації тексту, де кожен документ може належати до кількох попередньо визначених тем одночасно.

Багатоміткова класифікація текстових даних є важливою проблемою. Приклади варіюються від статей новин до електронних листів. Наприклад, це можна використовувати, щоб визначити жанри, до яких належить фільм, на основі короткого змісту його сюжету.

У класифікації з кількома мітками навчальний набір складається з екземплярів, кожен з яких пов’язаний з набором міток, і завдання полягає в тому, щоб передбачити набори міток для нових, раніше невідомих вхідних екземплярів шляхом аналізу навчальних екземплярів з відомими наборами міток.

Різниця між багатокласовою (множинною) класифікацією [3] та класифікацією з кількома мітками полягає в тому, що в першій класи є взаємовиключними, тоді як для задач із кількома мітками всі з них відрізняються, але певним чином пов’язані між собою.

На відміну від звичайних завдань класифікації, де мітки класів є взаємовиключними, класифікація з кількома мітками вимагає спеціальних алгоритмів машинного навчання, які підтримують прогнозування кількох взаємно невиключних класів або міток.

1.1.3 Класифікація тексту

Класифікація тексту, також відома як категоризація тексту, – це процес класифікації тексту в організовані групи. Використовуючи обробку природної мови (NLP), класифікатори тексту можуть автоматично аналізувати текст, а потім, на основі його змісту, призначати певне значення з набору заздалегідь визначених міток або категорій.

Неструктурований текст є скрізь, наприклад, електронні листи, веб-сайти та соціальні медіа, але з таких даних важко отримати знання або інформацію, якщо вони не організовані певним чином. Раніше це було складним і високовартісним процесом, оскільки вимагало великих витрат часу та ресурсів на ручне сортування даних або створення ручних правил, які важко підтримувати автоматично. Класифікатори тексту за допомогою NLP виявились чудовою альтернативою швидкому, економічному та масштабованому структуруванню текстових даних.

Класифікація тексту стає все більш важливою частиною бізнесу, оскільки вона дозволяє легко отримувати статистичні дані та автоматизувати бізнес-процеси.

1.2 Сентиментальний аналіз

Одним з найпоширеніших прикладів та випадків використання автоматичної класифікації тексту є саме сентиментальний аналіз тексту.

Сентиментальний аналіз тексту – це техніка обробки природної мови, яка використовується для визначення емоційного забарвлення даних. Аналіз тексту часто проводиться з використанням текстових даних для того, щоб відстежувати настрої бренду та продукту у відгуках клієнтів та розуміти потреби споживачів. Це також ефективний інструмент моніторингу і оцінювання конкретних груп користувачів, з метою покращення якості продукції на основі вивчення запитів користувачів в Інтернеті.

Цей аналіз дозволяє віднайти в тексті емоційну оцінку, яка ще називається тональністю або сентиментом (від англ. Sentiment - почуття; думка, настрій) тексту. Людина оцінює світ відразу за багатьма шкалами (хороший-поганий, сильний-слабкий, великий-маленький, щасливий-нещасливий, радісний-сумний і т.п.), і шкали ці по-різному емоційно навантажені.

Ключовим аспектом сентиментального аналізу є саме класифікація полярності. Полярність стосується загального настрою, що передається конкретним текстом, фразою або навіть словом. Полярність може бути виражена у вигляді числової оцінки, відомої як «оцінка настроїв». Наприклад, ця оцінка може набувати значень від -100 до 100, де 0 представляє нейтральне забарвлення. Дана оцінка може бути розрахована для всього тексту або лише для окремої фрази.

Для ефективного аналізу оцінка настроїв може бути тонко налаштована для конкретного випадку використання. Категорії емоцій можуть виходити за межі лише «позитивних», «нейтральних» і «негативних». Наприклад, в даній дипломній роботі використано цілих 28 різних категорій емоційного забарвлення.

1.3 Постановка задачі

Метою дипломного проекту є створення програмного модулю, що реалізує багатоміткову класифікацію на основі сентиментального аналізу для оцінки тональності текстових коментарів користувачів під обраним дописом соціальної мережі Reddit.

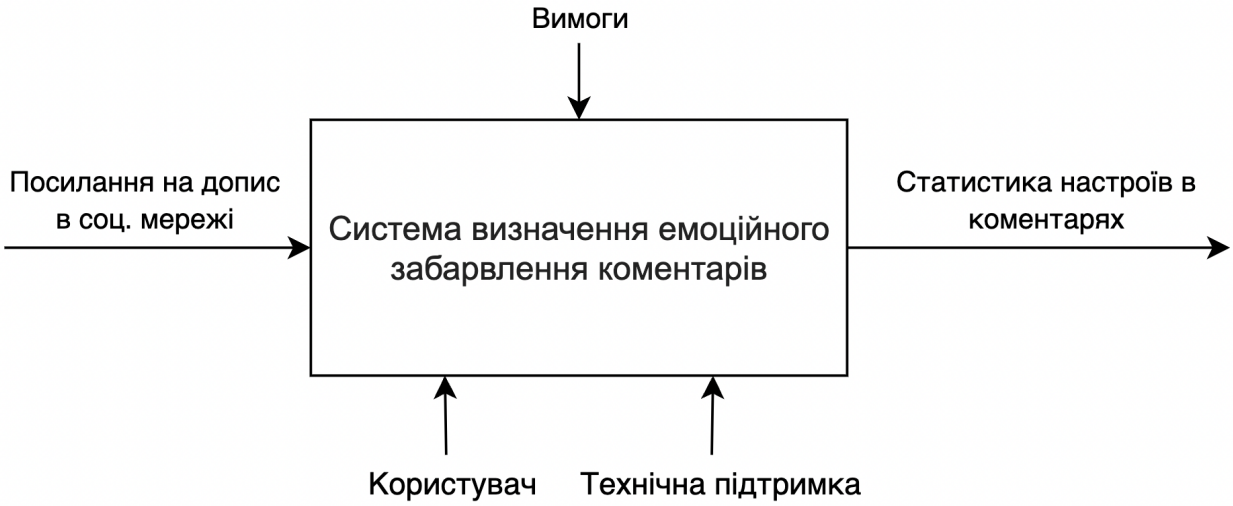

Рис. 1.1 Контекстна діаграма IDEF0 програмного модулю визначення емоційного забарвлення коментарів.

На вхід програмного модулю подається посилання на допис в соціальній мережі, коментарі якого ми хочемо проаналізувати. На виході маємо сформовану статистику настроїв в коментарях. Окрім вхідних та вихідних даних, на діаграмі ще зображено механізми, що взаємодіють із модулем: – «Користувач» – людина, що хоче дізнатися емоційну реакцію коментаторів на певний допис, а також елемент управління «Технічна підтримка», що необхідний для коректної роботи програмного модулю.

Даний програмний модуль може набути широкого застосування як у визначенні статистики інтернет-дописів, новин та подій, продуктів, так і впровадження в більш складні моніторингові системи.

Об’єкт дослідження – засоби автоматизованого визначення емоційного забарвлення тексту.

Предмет дослідження – методи визначення емоційного забарвлення текстових коментарів користувачів соціальної мережі Reddit, їх класифікація за допомогою застосування глибинного навчання.

Методи дослідження – в дипломному проекті використовуються методи обробки текстових документів, їх представлення у числовому форматі, для подання на вхід моделі глибинного навчання, а також методи багатоміткової класифікації тексту – для вирішення проблеми сентиментального аналізу.

В результаті дослідження була отримана висока точність оцінки тональності текстових коментарів.

Для виконання експериментальних досліджень було розроблено програму на мові програмування Python. Результати експериментальних досліджень оброблені і проаналізовані.

1.4 Огляд та застосування сентиментального аналізу

Сентиментальний аналіз може слугувати чудовим інструментом для отримання інформації, важливої для бізнесу. Завдяки його застосуванню, можна зрозуміти, що саме клієнти очікують від бренду, або шукають у продукті.

Моніторинг відгуків та управління репутацією є найпоширенішими цілями при застосуванні сентиментального аналізу. Власникам важливе та необхідне розуміння того, як споживачі сприймають їхній бренд, продукт або послуги. Це однаково корисно для технологічних компаній, маркетингових агентств, модних брендів, медіа-організацій та багатьох інших. Це дозволяє компаніям:

- відстежувати сприйняття бренду клієнтами;
- виокремити конкретні деталі, що варті уваги;
- знайти наявні закономірності та тенденції.

Все перелічене вище дозволяє власникам відповідним чином адаптуватися до поточної ситуації і внести до продукту необхідні зміни.

Загалом, аналіз настроїв можна використовувати для:

- автоматизації процесу моніторингу медіа та системи вчасного реагування та попередження;
- відстежування згадувань або відгуків про бренд на різних платформах (блоги, соціальні мережі, сайти, форуми тощо);
- класифікації цінності та терміновості згадувань відповідно до оцінки їхньої релевантності (тобто, яка платформа або тип користувача є життєво важливими для бренду)

На початковому етапі компанія реагує на результати, що надходять, і адаптується під них відповідним чином, але згодом сентиментальний аналіз може змінити навіть процес розвитку компанії в окремих напрямках.

Відмінним прикладом того, якого використання може набути сентиментальний аналіз для створення бренду та моніторингу є компанія KFC – американська мережа ресторанів швидкого харчування. У певний період свого існування на деякий час KFC використовували застарілі методи, поки їхня конкуренція рухалася вперед, акцентуючи увагу на здоровому харчуванні, приємному смаку та відчуттях.

Замість уподібнення до вже достатньо переповненої ніші, рішенням KFC стало використання сили самого бренду. Компанія почала використовувати меми (жартівливі картинки в Інтернеті) та іконографії поп-культури для просування цінності пропозицій бренду .

Такий підхід створив природню цікавість навколо бренду, доповнену посиланням на поп-культуру. В результаті користувачі взаємодіяли спочатку із брендом, а надалі були вимушені взаємодіяти і безпосередньо з самим продуктом.

Даний підхід поєднує в собі аналіз настроїв користувачів у моніторингу соціальних мереж та управління рекламою в результаті чого створюється цикл, який продовжує хід рекламної кампанії - користувачі активно залучаються до коментування або реакції на вміст реклами і це, у свою чергу, породжує подальші ідеї щодо розвитку даного напряму.

В результаті застосування методів сентиментального аналізу бренд KFC отримав постійну присутність у медіа-ландшафті, що гарантувало бренду неперервне зростання охоплення користувачів та, зрештою, частки ринку.

Окрім сприйняття бренду та вивчення думок клієнтів, дослідження ринку є, мабуть, найбільш впливовою сферою застосування сентиментального аналізу.

Так, компанія Apple є прекрасним прикладом застосування сентиментального аналізу на користь дослідження ринку та аналізу конкурентів. Комбінація цієї інформації з різних ресурсів якісно відображає

ринкову ситуацію та дозволяє розрахувати додаткову перспективу того, як диференціювати та посилити ціннісні пропозиції компанії.

Сентиментальний аналіз не є єдиним та основним інструментом дослідження ринку, однак даний інструмент може надати додатковий погляд на ринок і виявити корисні для бізнесу представлення того, як поточна ситуація може виглядати з початкового, базового рівня, тобто з боку споживачів. Крім того, подібний підхід може бути використаний для аналізу конкурентів та їх маркетингової діяльності.

При застосуванні сентиментального аналізу в даному розрізі стануть доступними такі можливості:

- збір інформації на різних платформах;
- розуміння реальних потреб користувачів, через залишені безпосередньо ними коментарі, відгуки, огляди тощо;
- уявлення про загальне сприйняття як всього продукту, так і конкретного його аспекту користувачем;
- надання результатів в режимі реального часу.

Сентиментальний аналіз дозволяє детально розглянути потреби та запити споживачів і на цій основі допомогти скоригувати ціннісну пропозицію таким чином, щоб вона набула очікуваних результатів.

1.5 Аналіз сучасних досліджень

Традиційні дослідження аналізу настроїв спрямовані на виявлення полярності в тексті, класифікуючи його як позитивний, негативний або нейтральний.

Для розв'язання задачі сентиментального аналізу можуть бути використані алгоритми машинного навчання. Серед класичних алгоритмів, які розв'язують задачі бінарної класифікації та класифікації тексту виділяють:

- Логістичну регресію (Logistic Regression) [4],
- Наївний байєсівський класифікатор (Naive Bayes) [5],
- Метод опорних векторів (Support Vector Machine, SVC) [6].

Так, у роботі «Twitter Sentiment Analysis using Machine Learning» [7] для вирішення задачі сентиментального аналізу для бінарної класифікації тексту – позитивного та негативного, було розглянуто та отримано такі точності роботи вищезазначених алгоритмів:

- Logistic Regression – 82.4%,
- Multinomial Naive Bayes – 80.6%,
- Linear Support Vector Machine – 83.7%.

У роботі присвяченій різним підходам до проблеми аналізу настроїв[8] було розглянуто гібридну версію рішення задачі:

- Support Vector Machine + Naive Bayes – 86%.

Для класифікації вже трьох і більше класів емоцій необхідно застосовувати інший, більш глибокий підхід – deep learning і нейронні мережі. У дослідженні «A Novel Machine Learning Approach for Sentiment Analysis on Twitter» [9] було застосовано та проаналізовано результати роботи алгоритмів глибинного навчання, а саме – нейронних мереж, на трьох мітках емоцій – позитивних, негативних та нейтральних. Також для порівняння з методами машинного навчання автор використовує в роботі вищезгадані алгоритми

Support Vector Machine та Multinomial Naïve Bayes.

Автором статті було отримано такі точності:

- Support Vector Machine – 78%,
- Multinomial Naïve Bayes – 80%,
- RNN (Recurrent Neural Network) [10] / LSTM (Long short-term memory) [11] – 83%,
- CNN (Convolutional Neural Network) [12] – 86.2%,
- RCNN (Region-Based Convolutional Neural Network) [13] – 88.7%,
- BERT large fine-tune UDA [14] – 95.8%

Дослідженнями доведено, що застосування нейромережевих технологій для небінарної класифікації тексту є актуальним та найбільш перспективним напрямком у вирішенні задачі. За отриманими результатами видно, що найкраще з класифікацією за трьома мітками емоцій впоралася нейронна мережа BERT та досягла точності у 96% на обраних даних. Тому для реалізації поставленого завдання даної дипломної роботи було обрано саме цю нейронну мережу.

1.6 Аналіз вимог

1.6.1 Функціональні вимоги

Функціональні вимоги – це вимоги до програмного забезпечення, які описують внутрішню роботу системи, її поведінку: обчислення даних, маніпулювання даними, обробка даних та інші специфічні функції, які має виконувати система.[15] На відміну від нефункціональних вимог, які визначають якою система повинна бути, функціональні вимоги визначають, що система повинна робити.

Даний програмний модуль має відповідати таким функціональним вимогам:

- Програма повинна перевіряти правильність введених користувачем даних і в разі помилки - повідомляти про це користувача.
- В результаті обробки користувацьких даних програмний модуль має видавати коректні результати.
- У разі некоректних даних програма має повідомляти про помилку, а не видавати некоректні результати.
- Програмний модуль має бути сумісною з різними браузерами та їхніми версіями.

1.6.2 Нефункціональні вимоги

Нефункціональні вимоги – вимоги, що не пов'язані з функціональним аспектом програмного забезпечення. Вони визначають очікувані характеристики програмного забезпечення.[16] Здебільшого ці вимоги походять від функціональних, заснованих на вкладі замовника та інших зацікавлених сторін. Припущення про нефункціональні вимоги можуть робити й самі користувачі.

Програмний модуль, представлений в дипломному проекті, має відповідати таким нефункціональним вимогам:

- Інтерфейс програмного модулю має бути зручним та інтуїтивно зрозумілим користувачу.
- Користувач повинен мати можливість зручно передати посилання на допис в соц. мережі або текстовий коментар на вхід програмного модулю.
- Програма повинна швидко обробляти інформацію.
- В результаті обробки користувацьких даних програмний модуль має видавати зрозуміло оформленні, надійні та доступні користувачеві результати.

1.7 Висновки до розділу

В першому розділі, що присвячений задачі класифікації, було описано загальні теоретичні відомості досліджуваних методів та алгоритмів, а саме: бінарної класифікації, класифікації тексту та сентиментального аналізу. Також було наведено причини актуальності проблеми класифікації текстових даних, яка з розвитком Інтернету буде лише зростати, та проаналізовано сучасні дослідження в даній області. Також в цьому розділі було представлено узагальнений вигляд програмного модулю, що розробляється та визначено його функціональні та функціональні вимоги.

РОЗДІЛ 2. Проектні рішення

2.1 Проектування архітектури програмного модулю

2.1.1 Дерево функцій

Модель дерева функцій належить до функціонального уявлення та призначена для опису ієрархічної структури функцій, включаючи статичні зв'язки між ними. Функція – це завдання, операція чи дія, які виконуються над певним об'єктом задля досягнення однієї чи кількох цілей. Між елементами різних рівнів моделі встановлені зв'язки типу “є ієрархічно вищим для” від функції верхнього рівня ієрархії до функцій нижнього рівня, отриманих в результаті декомпозиції комплексної функції.

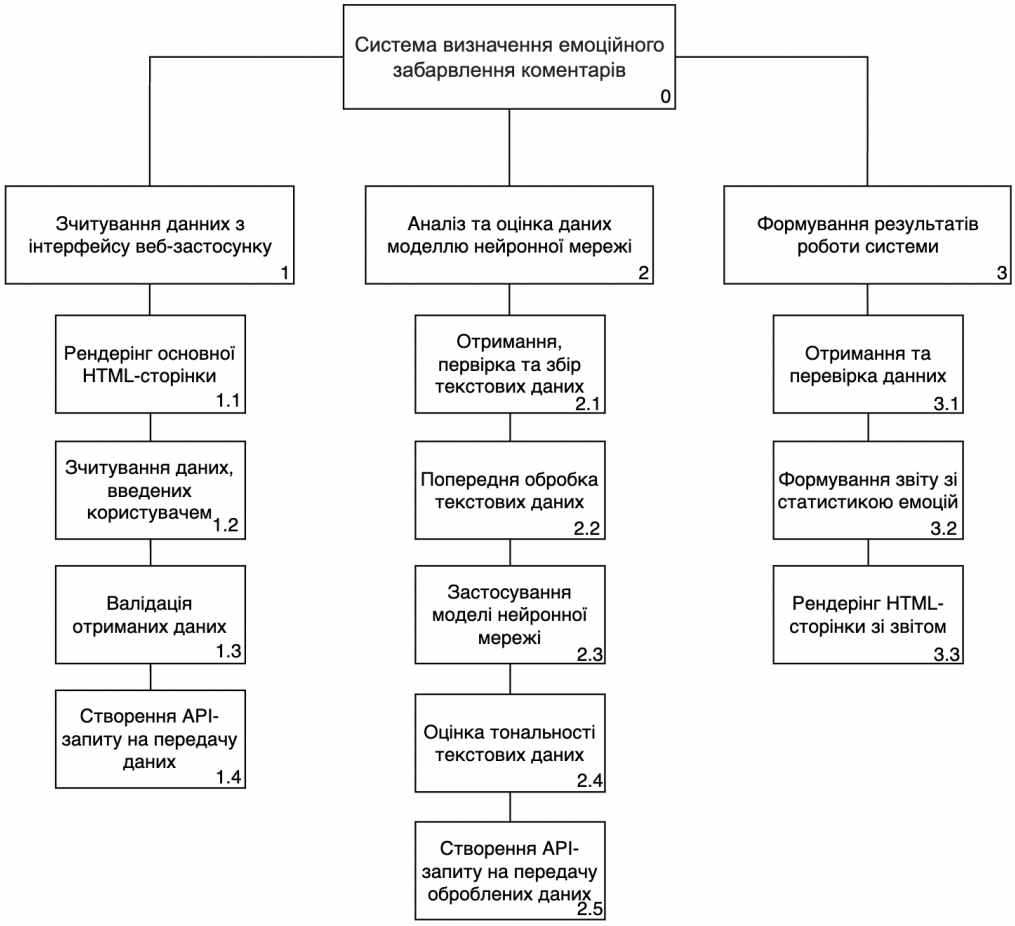

Рис. 2.1 Дерево функцій

В даному випадку ми маємо три рівня ієрархії, найвищий з них – нульовий, і є проектованим програмним модулем визначення емоційного

забарвлення коментарів в соціальних мережах. Наступний рівень складається з трьох основних блоків, кожен з яких має більш детальну декомпозицію, що також представлено на діаграмі:

1. Зчитування даних з інтерфейсу веб-застосунку,
    1.1. Рендерінг основної HTML-сторінки
    1.2. Зчитування даних, введених користувачем
    1.3. Валідація отриманих даних
    1.4. Створення API-запиту на передачу даних
2. Аналіз та оцінка даних моделлю нейронної мережі,
    2.1. Отримання, перевірка та збір текстових даних
    2.2. Попередня обробка текстових даних
    2.3. Застосування моделі нейронної мережі
    2.4. Оцінка тональності текстових даних
    2.5. Створення API-запиту на передачу оброблених даних
3. Формування результатів роботи програми.
    3.1. Отримання та перевірка даних
    3.2. Формування звіту зі статистикою емоцій
    3.3. Рендерінг HTML-сторінки зі звітом

2.1.2 Діаграма діяльності

Діаграма діяльності є розширеною версією блок-схеми, яка моделює перехід від однієї діяльності (операції системи) до іншої та використовується для опису динамічних аспектів системи, що розробляється. Даний тип діаграм часто використовуються не тільки для візуалізації динамічної природи системи, а й для побудови системи та моделювання бізнес-процесів.

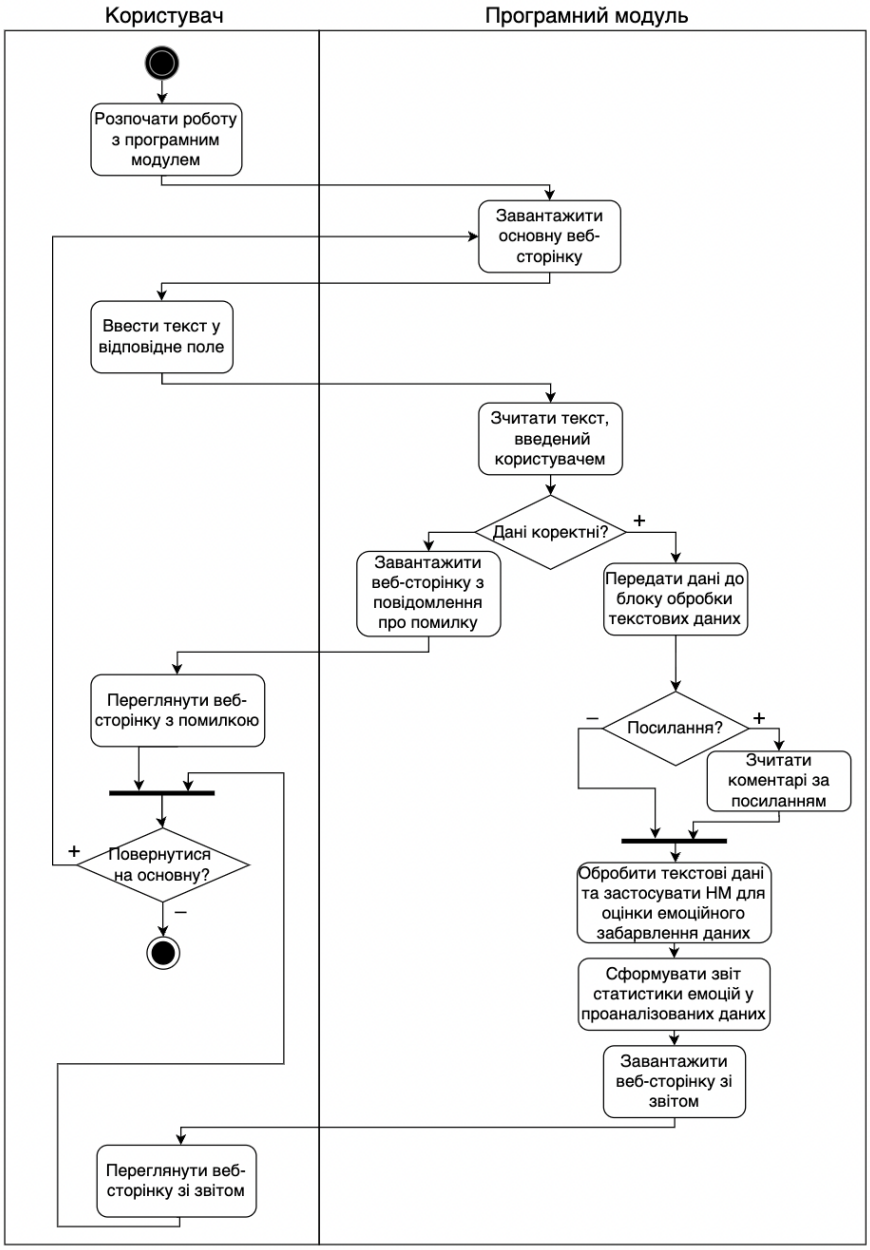

Рис. 2.2 Діаграма діяльності

Побудована діаграма діяльності окрім динамічних процесів також зображує взаємодію користувача та розроблюваного програмного модулю. Так, користувач може виконувати такі операції:

- Розпочати роботу з програмним модулем,
- Ввести текст або посилання, аналіз якого хоче отримати на виході, у відповідне поле,
- Переглянути помилку при введені даних, якщо вона присутня,
- Переглянути сформований програмним модулем звіт зі статистикою емоцій у проаналізованих текстових даних,
- Повернутися на основну веб-сторінку програмного модуля,
- Закінчити роботу з програмним модулем.

У той час як потік операцій програмного модуля налічує такі основні пункти:

- Формування та завантаження веб-сторінок в залежності від попередніх дій користувача або програми,
- Валідація введених користувачем даних та перевірка їхньої відповідності умовам роботи програмного модулю,
- Обробка даних для подання на вхід моделі нейронної мережі та її безпосереднє застосування для визначення емоційного забарвлення текстових даних.
- Формування звіту статистики емоцій у проаналізованих текстових даних.

2.1.3 Опис архітектури

В основі програмного модулю лежить концепція мікросервісної архітектури – поширений підхід до розробки програмного забезпечення, коли програма розбивається на невеликі автономні компоненти (мікросервіси) з чітко визначеними інтерфейсами. Саме ця архітектура характерна для хмарних додатків, які зараз популярні завдяки перевагам, що відкривають для бізнесу хмарні середовища.

Кожен із сервісів відповідає за конкретне бізнес-завдання, має власне сховище даних та взаємодіє з іншими сервісами через прості API-інтерфейси для вирішення складніших завдань. Так, у даному проекті можна виділити два основних мікросервіси – веб-застосунок та програмний інтерфейс моделі нейронної мережі.

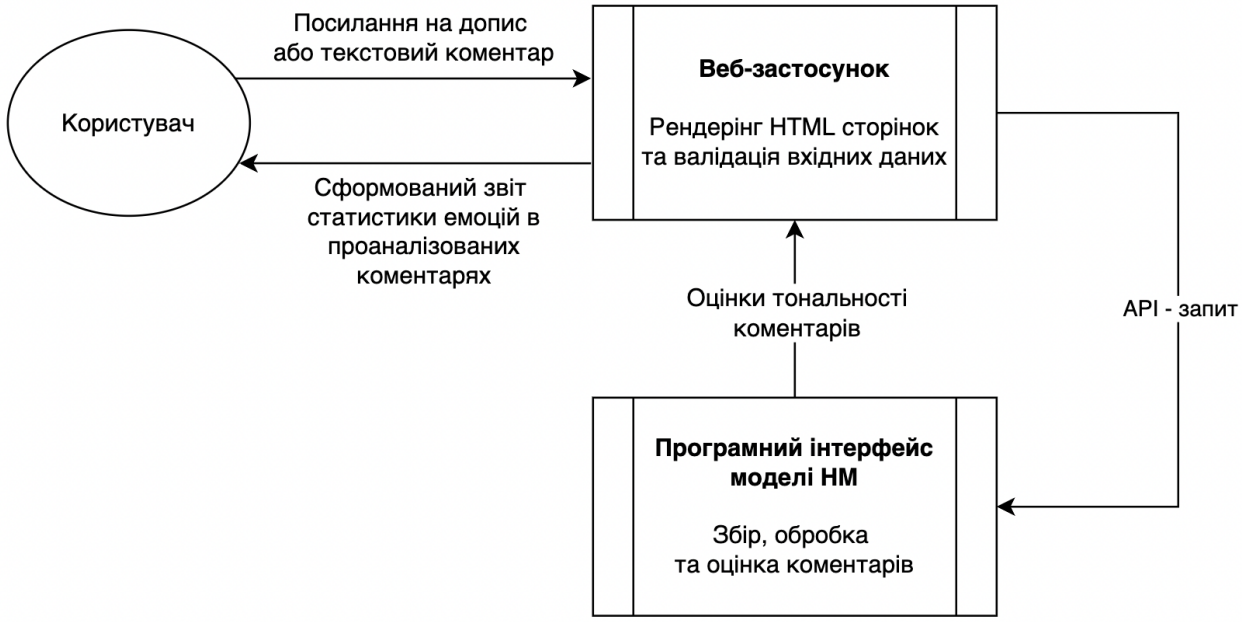

Рис. 2.3 Загальний вигляд архітектури системи

Веб-застосунок отримує на вхід посилання або текстовий коментар, введені користувачем у відповідне поле HTML-сторінки, яка формується одразу після запуску серверу. Далі дані валідуються та через API-запит надсилаються на вхід до частини програмного інтерфейсу моделі нейронної мережі. В залежності від отриманих даних та пройденої ними перевірки, в

даному блоці виконується певний перелік операцій, такі як збір коментарів (якщо на вхід було подане посилання на певний допис, а не текст коментаря), їхня обробка та оцінка тональності моделлю нейронної мережі. Отримані оцінки коментарів, також за допомогою API-запиту, повертаються до блоку веб-застосунку, де на їх основі формується звіт та відображається для користувача на відповідній HTML-сторінці.

Дана мікросервісна архітектура має такі переваги:

- Простота розгортання (можливість розгортання тільки мікросервісів, що змінюються, незалежно від решти програмного модулю)
- Оптимальність масштабування (можливість розширення тільки того функціоналу, який цього потребує, залишаючи працювати інші частини програмного модулю)
- Стійкість до збоїв (відмова одного сервісу не призводить до зупинення програмного модулю загалом – коли помилка виправлена, потрібну зміну можна розгорнути лише для відповідного сервісу замість повторного розгортання всієї програми)
- Можливість вибору технологій (різні набори технологій, оптимальні для вирішення завдань, які стоять перед окремими сервісами)

2.2 Алгоритми глибинного навчання

Глибинне навчання [17] – це підмножина машинного навчання, де штучні нейронні мережі є алгоритмами, що змодельовані працювати по аналогії з мозком людини й навчатись на великих обсягах даних.

Глибинне навчання використовує багатошарові нейронні мережі, що класифікують інформацію завдяки взаємопов’язаним шарам вузлів - нейронів. При переході через один із шарів, кожен нейрон цього шару виконує прості операції з даними і вибірково передає результати на наступні вузли. Кожен наступний шар фокусується на об’єкті вищого рівня, ніж попередній, доки дані, що обробляються не досягнуть кінцевого шару.

Між вхідним та вихідним шарами є приховані шари. Саме тут і криється основна різниця між нейронними мережами та глибинним навчанням: базова нейронна мережа може мати один або два прихованих шари, а мережа глибинного навчання може мати десятки або навіть сотні шарів. Збільшення кількості різних шарів і вузлів може підвищити точність мережі, однак це також може означати, що модель потребуватиме налаштувань більшої кількості параметрів та обчислювальних ресурсів.

Однією з головних переваг глибинного навчання є те, що нейронні мережі використовуються для виявлення прихованих уявлень і зв’язків в даних. Алгоритми глибинного навчання можна навчити переглядати текстові дані, аналізувати публікації в соціальних мережах, новини та опитування, щоб надати цінну інформацію для бізнесу та клієнтів.

Використовуючи глибинне навчання, модель сентиментального аналізу можна навчити розуміти не лише прості визначення, а читати текст з урахуванням контексту та розуміти фактичний настрій та почуття автора цього тексту.

2.2.1 Архітектура трансформерів

Трансформер – це архітектура, спрямована на вирішення проблеми паралелізації обробки текстових послідовностей, яка працює на механізмі “self-attention” [18].

Загалом архітектура трансформеру складається з двох основних блоків – кодувальнику та декодеру, на вхід яким подаються послідовності слів. Але перш ніж потрапити на вхід до блоків, у шарі вбудовування ці слова перетворюються у числові вектори в залежності від місця в словнику, де слова згруповані за мірою близькості значень. Наступний етап – позиційні кодувальники, що задають контекст слова за його позицією в реченні.

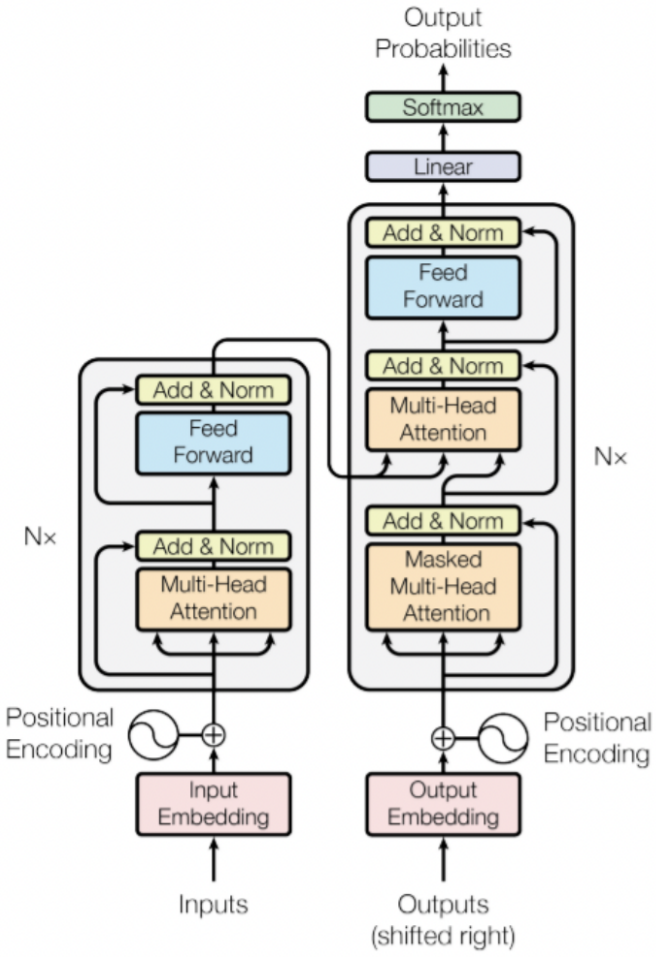

Рис. 2.4 Приклад архітектури трансформерів

І тільки після цього переходимо на вхід блоку кодувальника, де оцінюється те, наскільки конкретне слово є релевантним для інших слів у

цьому реченні. Для кожного слова створюється вектор уваги, який фіксує контекстний зв’язок між словами в цьому реченні, з яких далі обчислюється середнє зважене значення для визначення остаточного вектору уваги слів.

Другою частиною кодувальник є нейронна мережа прямого поширення, яка застосовується до кожного вектору уваги, і її головна мета – перетворити ці вектори уваги у форму, прийнятну для наступного шару кодувальника або декодера. Що важливо на даному етапі – наші вектори незалежні один від одного, тому тут можна застосувати паралелізацію і отримати набір закодованих векторів для кожного слова одночасно.

Далі переходимо до блоку, де відбувається навчання нашого трансформеру – до блоку декодера, на вхід якого подається послідовність міток коментарів. Аналогічно до блоку кодувальника, спочатку необхідне проходження крізь шар вбудовування, кодувальника позицій та блок самоуваги, де вектори уваги генеруються для кожної мітки для аналізу їхнього зв’язку між собою.

Але в даному випадку ми маємо замаскований блок уваги, бо спочатку визначається мітка, відповідно до попередніх результатів, а потім перевіряється її відповідність до фактичної. Після порівняння значення матриці оновлюються. Даний блок називається замаскованим, бо кожного разу для ефективного навчання ми приховуємо справжню фактичну мітку для того, щоб навчання відбувалося за зверненням до попередньо отриманих результатів.

Далі результуючі вектори уваги з попереднього шару та вектори з блоку кодувальника передаються в інший блок уваги кодувальника-декодеру. Цей блок з’ясовує зв’язки між нашим коментарем та міткою, відбувається основне зіставлення; на виході отримується вектор зв’язку.

Потім вектори передаються в блок прямого поширення, який перетворює їх до вигляду для подання на інший шар декодера або лінійний шар - останній шар прямого поширення, який повертає вектор значень рівний

розміру вектору міток. Далі він пропускається через Softmax в розподіл ймовірностей, який вже інтерпретується людиною. І на виході передбачена мітка має найбільше значення ймовірності.

2.2.2 Нейронна мережа BERT

BERT - Bidirectional Encoder Representations from Transformers – це попередньо навчена нейронна мережа, що працює на основі трансформерів, але не включає в себе блок декодеру. Архітектура нейронної мережі складається з блоку кодера та класифікатора, що діє як своєрідний декодер, який намагається передбачити необхідний вихід, але на вхід якому нічого не подається, так як передбачення визначається за попередньо отриманими результатами. BERT використовує інші гіперпараметри для кращої продуктивності, наприклад, 12 і 16 міні блоків уваги, а не 8, як трансформер, а також має не тільки вбудовування слів і позиційне кодування, а ще й вбудовування сегментів.

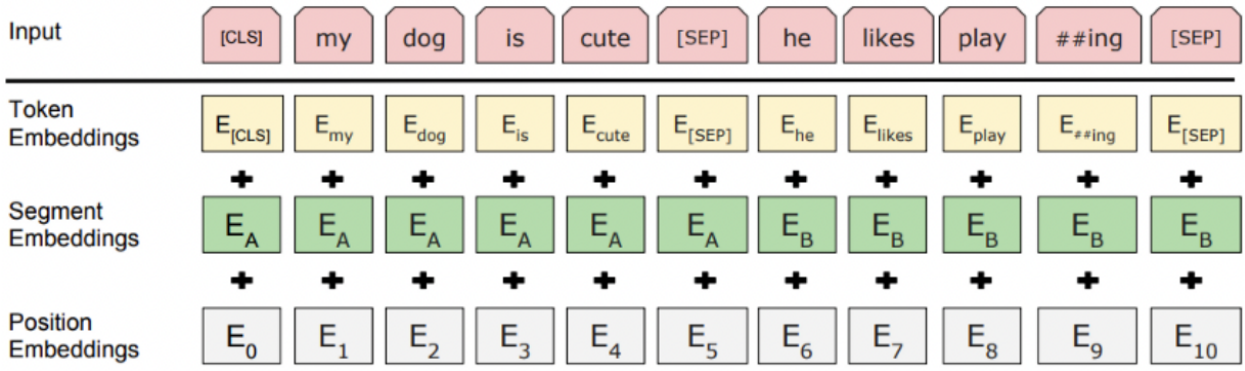

Рис. 2.5 Вигляд даних, що подаються на вхід моделі.

- Token Embeddings – вбудовування слів – список унікальних ідентифікаторів для токенізованого тексту;
- Segment Embeddings – вбудовування сегментів – встановлюється значення 1 для реальних токенів, 0 для токенів заповнення;
- Position Embeddings – позиційне кодування – в нашому випадку – список одиниць.

2.3 Висновки до розділу

В другому розділі, що присвячений проектуванню програмного модулю було розглянуто особливості його архітектури, побудовано та описано дерево функцій, діаграму діяльності та загальний вигляд основних модулів програми, що розробляється. Також у даному розділі було розглянуто нейронну мережу BERT, що застосовується для вирішення поставленої задачі, та описано архітектуру трансформерів, на якій дана мережа базується.

РОЗДІЛ 3. Розробка програмного застосунку

3.1 Вибір програмного середовища

Інструментом для вирішення поставленої задачі було обрано мову програмування Python, через низку її переваг, однією з яких, є існування обширної системи Python-бібліотек – великого набору існуючих кодових баз та готових прикладів/макетів для вирішення задач. Ці бібліотеки налічують як алгоритми для базових математичних операцій, так і для більш складних завдань.

В даній роботі було використано такі модулі:

- Numpy – для наукових розрахунків і обчислень.
- Pandas – для зручної роботи з великими наборами даних.
- PyTorch, Tez, Transformers – для створення, налаштування та навчання нейронної мережі.
- Flask – для створення та роботи з веб-застосунком

Для зручної роботи з кодом та поєднання різних блоків програмного модулю було використано програмне середовище PyCharm – гібридну платформу, розроблена JetBrains як IDE для мови програмування Python. Дане програмне середовище має такі особливості:

1. Інтелектуальний редактор коду:

- дозволяє створювати високоякісний програмний код,
- складається з різних кольорових тем для виділення ключових слів, класів і функцій, що допомагає підвищити читабельність і розуміння коду,
- інтерактивно, в режимі реального допомагає виявити помилки,
- надає функцію автозаповнення та інструкції для завершення коду.

2. Зручна навігація по коду:

- дозволяє редагувати та покращувати код з меншими зусиллями та часом,
- дозволяє легко перейти до конкретної функції, класу або файлу,
- дозволяє швидко знайти елемент, символ або змінну у вихідному коді.

3. Підтримка великого різноманіття популярних веб-фреймворків Python, одним з яких є Flask, що був використаний в даному дипломному проекті.

Для ефективної роботи нейронна мережа була натренована у онлайн програмному середовищі Google Colab, яке було розроблено компанією Google, для надання безкоштовного доступу до графічних процесорів та TPU для створення моделі машинного або глибинного навчання.

3.2 Опис навчального набору даних

Емоції є ключовим аспектом соціальних взаємодій, які впливають на поведінку людей і формують їхні стосунки. Особливо це стосується мови — лише кількома словами ми можемо висловити найрізноманітніші тонкі та складні емоції. Таким чином, важливо дати можливість машинам розуміти контекст та емоції, що, у свою чергу, дозволить використовувати різноманітні програми, включаючи чат-боти, моделі для виявлення шкідливої поведінки в Інтернеті та покращену підтримку клієнтів. взаємодії.

За останнє десятиліття дослідницька спільнота НЛП зробила доступними кілька наборів даних для класифікації емоцій на основі мови. Більшість із них створені вручну та охоплюють цільові області (заголовки новин, субтитри до фільмів і навіть казки), але, як правило, є відносно невеликими або зосереджені лише на шести основних емоціях (злість, здивування, огида, радість, страх і смуток), які були запропоновані в 1992 році. Хоча ці набори даних емоцій дозволили досліджувати класифікацію емоцій, вони також підкреслили потребу у великомасштабному наборі даних з більш розширеним набором емоцій, який міг би сприяти більш широкому спектру майбутніх потенційних застосувань.

«GoEmotions: A Dataset of Fine-Grained Emotions»[19] – розмічений людьми набір даних із 58 тисячами коментарів соц. мережі Reddit, здобутих із популярних англомовних субредітів і позначених 27 категоріями емоцій. Як найбільший на сьогоднішній день повністю анотований набір даних про емоції англійською мовою, таксономія GoEmotions враховує психологічні особливості та застосовність даних.

Positive		Negative		Ambiguous
admiration 👏	joy 😃	anger 😡	grief 😢	confusion 😕
amusement 😂	love ❤️	annoyance 😒	nervousness 😬	curiosity 🤔
approval 👍	optimism 🤞	disappointment	remorse 😔	realization 💡
caring 🤗	pride 😌	disapproval 👎	sadness 😞	surprise 😲
desire 😍	relief 😅	disgust 🤮		
excitement 🤩		embarrassment 😳		
gratitude 🙏		fear 😨		

Рис. 3.1 Таксономія GoEmotions

На відміну від шести основних емоцій, які включають лише одну позитивну емоцію (радість), дана таксономія включає 12 позитивних, 11 негативних, 4 неоднозначні категорії емоцій і 1 «нейтральну», що робить датасет широко придатним для завдань, які вимагають тонкої диференціації між проявами емоцій.

3.3 Представлення даних

Початковий тренувальний набір даних GoEmotions від Google містить 58,000 рядків та 3 колонки з даними, де колонки:

- *text* – містить текстові коментарі користувачів
- *label* – мітку емоційного забарвлення тексту
- *id* – унікальний ідентифікатор кожного коментаря.

Первинний вигляд даних:

	text	labels	id
6	Yes I heard abt the f bombs! That has to be why. Thanks for your reply:) until then hubby and I will anxiously wait 😜	[15]	ee3b6wu
7	We need more boards and to create a bit more space for [NAME]. Then we'll be good.	[8, 20]	ef4qmod
8	Damn youtube and outrage drama is super lucrative for reddit	[0]	ed8wbdn
9	It might be linked to the trust factor of your friend.	[27]	eczgv1o
10	Demographics? I don't know anybody under 35 who has cable tv.	[6]	eel6g5h
11	Aww... she'll probably come around eventually, I'm sure she was just jealous of [NAME]... I mean, what woman wouldn't be! lol	[1, 4]	edex4ki
12	Hello everyone. Im from Toronto as well. Can call and visit in personal if needed.	[27]	ef83m1s

Рис. 3.2 Приклад вхідних даних

, де позначення міток:

```
{0: 'admiration',
 1: 'amusement',
 2: 'anger',
 3: 'annoyance',
 4: 'approval',
 5: 'caring',
 6: 'confusion',
 7: 'curiosity',
 8: 'desire',
 9: 'disappointment',
 10: 'disapproval',
 11: 'disgust',
 12: 'embarrassment',
 13: 'excitement',
 14: 'fear',
 15: 'gratitude',
 16: 'grief',
 17: 'joy',
 18: 'love',
 19: 'nervousness',
 20: 'optimism',
 21: 'pride',
 22: 'realization',
 23: 'relief',
 24: 'remorse',
 25: 'sadness',
 26: 'surprise',
 27: 'neutral'}
```

Рис. 3.3 Позначення міток емоцій

Кодування міток відбувається з використанням One-Hot Encoding:

```
n_labels = len(mapping)

def one_hot_labels(df):
    dict_labels = []
    for i in tqdm(range(len(df)), leave=False):
        d = dict(zip(range(n_labels), [0]*n_labels))
        labels = df.loc[i]["labels"]
        for label in labels:
            d[label] = 1
        dict_labels.append(d)
    df_labels = pd.DataFrame(dict_labels)
    return df_labels
```

Рис. 3.4 Функція кодування міток емоцій

Вигляд даних після застосування кодування:

	text	labels	id	0	1	2	3	4	5	6	...	18	19	20	21	22	23	24	25	26	27
6	Yes I heard abt the f bombs! That has to be why. Thanks for your reply:) until then hubby and I will anxiously wait 😜	[15]	ee3b6wu	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
7	We need more boards and to create a bit more space for [NAME]. Then we'll be good.	[8, 20]	ef4qmod	0	0	0	0	0	0	0	...	0	0	1	0	0	0	0	0	0	0
8	Damn youtube and outrage drama is super lucrative for reddit	[0]	ed8wbdn	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9	It might be linked to the trust factor of your friend.	[27]	eczgv1o	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
10	Demographics? I don't know anybody under 35 who has cable tv.	[6]	eel6g5h	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
11	Aww... she'll probably come around eventually, I'm sure she was just jealous of [NAME]... I mean, what woman wouldn't be! lol	[1, 4]	edex4ki	0	1	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
12	Hello everyone. Im from Toronto as well. Can call and visit in personal if needed.	[27]	ef83m1s	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1

7 rows × 31 columns

Рис. 3.5 Вигляд набору даних після кодування

Візуалізація розподілу даних міток:

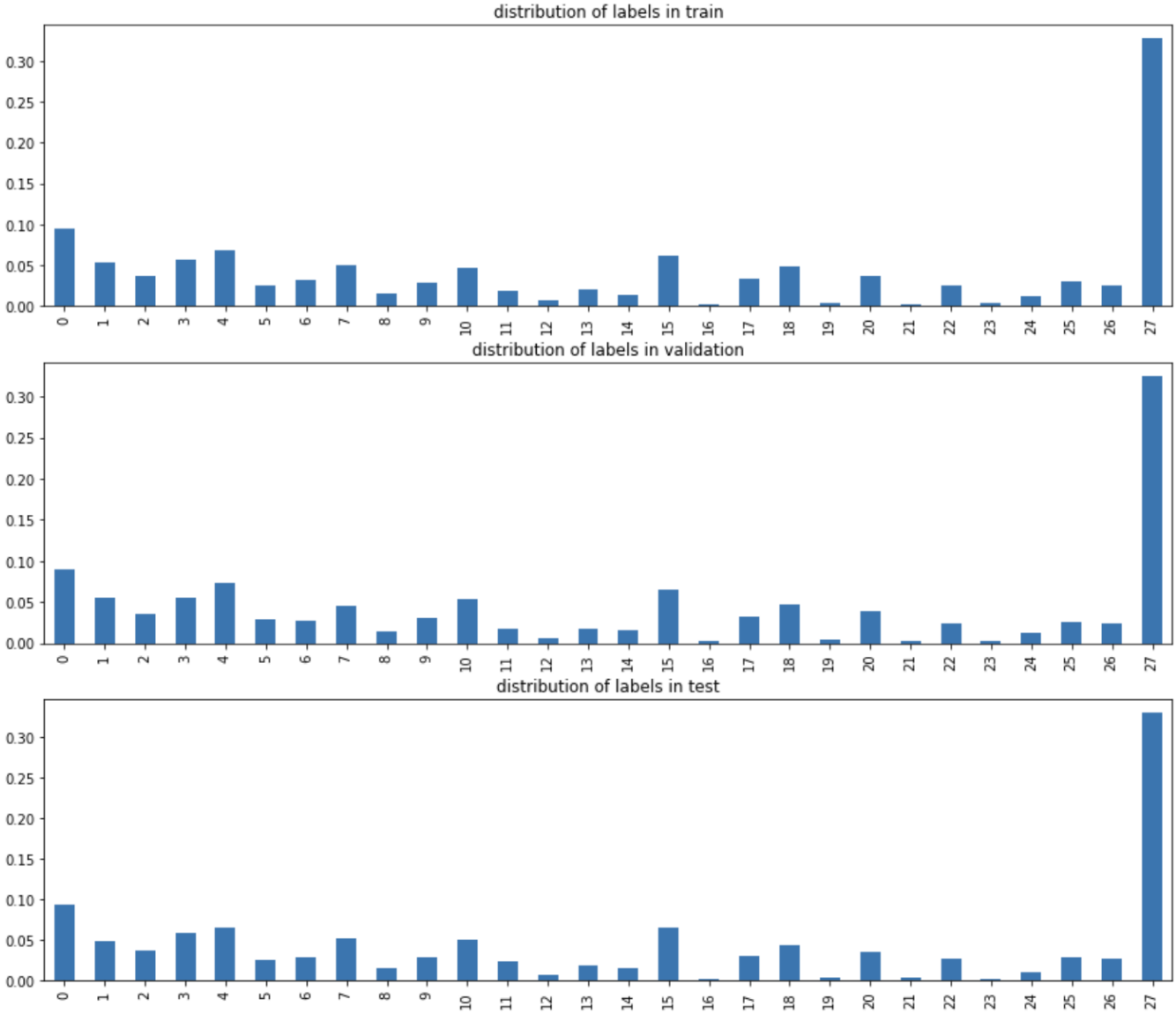

Рис. 3.6 Розподіл міток емоцій

Функція виведення коментарів за певними мітками та приклади коментарів під мітками «захоплення» та «злість»:

```
def inspect_data(label, n=5):
    samples = train[train[label] == 1].sample(5)
    sentiment = mapping[label]

    print(f"examples from {sentiment}")
    print()
    for text in samples["text"]:
        print(text)
        print("---")
```

```
inspect_data(0)
```

```
examples from admiration

Thanks to the finicky nature of the reddit app, i accidentially gilded this post. Congrats :p
---
Your sacrifice will be remembered.
---
That. Is flipping Cute. He's definately a dog person.
---
*Bud Light presents...real men of genius. 🎶 Mr Take Your Cat Up Tiger Mountain On Your Shoooouldererrrrr* 🎶
---
You're awesome! :)
---
```

```
inspect_data(2)
```

```
examples from anger

I hate from the bottom of my heart, when animals behave like this..
---
Try reading up. Not going to play these stupid games with a butt hurt Meltzer fan boy.
---
I accidentally swiped. I don't have time right now I swiped for ego and got it so I don't actually want to talk to anyone.
---
"I doubt I'm done writing" Kill me
---
Irrational anger especially certain family members
---
```

Рис. 3.7 Функція виведення коментарів за певними мітками та приклади виводу коментарів

3.4 Опис та застосування моделей нейронних мереж

Для представлення даних у необхідному форматі для датасету було описано клас. На вхід він отримує тексти та цільові значення, а на виході отримуємо тензорне представлення індексів, масок та цільових значень:

```
class GoEmotionDataset():
    def __init__(self, texts, targets):
        self.texts = texts
        self.targets = targets
        self.tokenizer = transformers.SqueezeBertTokenizer.from_pretrained(
            "squeezebert/squeezebert-uncased", do_lower_case=True
        )
        self.max_len = 35

    def __len__(self):
        return len(self.texts)


    def __getitem__(self, index):
        target = self.targets[index]
        text = self.texts[index]

        inputs = self.tokenizer.encode_plus(text,
                                            None,
                                            add_special_tokens=True,
                                            max_length=self.max_len,
                                            padding="max_length",
                                            truncation=True)

        ids = inputs["input_ids"]
        mask = inputs["attention_mask"]

        return {
            "ids": torch.tensor(ids, dtype=torch.long),
            "mask": torch.tensor(mask, dtype=torch.long),
            "targets": torch.tensor(self.targets[index], dtype=torch.long),
        }
```

Рис. 3.8 Клас для представлення набору даних у необхідному форматі

Клас моделі містить нашу результуючу модель. Вона складається з попередньо натренованої нейронної мережі, методу регуляризації Dropout, повнозв'язного шару із кількістю вихідних нейронів рівною кількості класів:

```
class EmotionClassifier(tez.Model):
    def __init__(self, num_train_steps, num_classes):
        super().__init__()
        self.bert = transformers.SqueezeBertModel \
        .from_pretrained("squeezebert/squeezebert-uncased")
        self.bert_drop = nn.Dropout(0.3)
        self.out = nn.Linear(768, num_classes)
        self.num_train_steps = num_train_steps
        self.step_scheduler_after = "batch"

    def fetch_optimizer(self):
        param_optimizer = list(self.named_parameters())
        no_decay = ["bias", "LayerNorm.bias"]
        optimizer_parameters = [
            {
                "params": [
                    p for n, p in param_optimizer \
                    if not any(nd in n for nd in no_decay)
                ],
                "weight_decay": 0.001,
            },
            {
                "params": [
                    p for n, p in param_optimizer \
                    if any(nd in n for nd in no_decay)
                ],
                "weight_decay": 0.0,
            },
        ]
        opt = AdamW(optimizer_parameters, lr=3e-5)
        return opt
```

Рис. 3.9 Клас моделі нейронної мережі

У якості алгоритму оптимізації моделі (метода, який використовується для зміни таких атрибутів нейронної мережі, як: вагові коефіцієнти та коефіцієнт швидкості навчання), було обрано алгоритм оптимізації Адама. Метод обчислює індивідуальні коефіцієнти швидкості навчання для різних параметрів на основі оцінок першого та другого моментів градієнтів.

Параметри конфігурації Адама:

1. learning_rate=0.001 – швидкість навчання або розмір кроку. Пропорція, в якій ваги оновлюються.

2. beta_1=0.9 – експоненційна оцінка швидкості розпаду для першого моменту .

3. beta_2=0.999 – експоненційна оцінка швидкості розпаду для другого моменту.

4. epsilon=1e-07 – дуже мале число, близьке до нуля, щоб запобігти ділення на нуль у реалізації.

```
def fetch_scheduler(self):
    sch = get_linear_schedule_with_warmup(
        self.optimizer, num_warmup_steps=0,
        num_training_steps=self.num_train_steps
    )
    return sch

def loss(self, outputs, targets):
    if targets is None:
        return None
    return nn.BCEWithLogitsLoss()(outputs, targets.float())


def monitor_metrics(self, outputs, targets):
    if targets is None:
        return {}

    outputs = torch.sigmoid(outputs)
    outputs = outputs.cpu().detach().numpy()
    targets = targets.cpu().detach().numpy()

    fpr_micro, tpr_micro, _ = metrics.roc_curve(targets.ravel(),
                                                outputs.ravel())
    auc_micro = metrics.auc(fpr_micro, tpr_micro)
    return {"auc": auc_micro}


def forward(self, ids, mask, targets=None):
    o_2 = self.bert(ids, attention_mask=mask)["pooler_output"]
    b_o = self.bert_drop(o_2)
    output = self.out(b_o)
    loss = self.loss(output, targets)
    acc = self.monitor_metrics(output, targets)
    return output, loss, acc
```

Рис. 3.10 Функції оптимізації моделі

3.5 Оцінка результатів

Натренована на 8 епохах модель досягнула валідаційної точності - 95%:

```
100%|██████████| 679/679 [02:42<00:00,  4.17it/s, auc=0.74, loss=0.189, stage=train]
  0%|          | 0/340 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py
  cpuset_checked))
100%|██████████| 340/340 [00:12<00:00, 27.41it/s, auc=0.859, loss=0.126, stage=valid]
Validation score improved (inf --> 0.12639302707770292). Saving model!
  0%|          | 0/679 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py
  cpuset_checked))
100%|██████████| 679/679 [02:48<00:00,  4.03it/s, auc=0.89, loss=0.115, stage=train]
  0%|          | 0/340 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py
  cpuset_checked))
100%|██████████| 340/340 [00:12<00:00, 26.87it/s, auc=0.922, loss=0.105, stage=valid]
Validation score improved (0.12639302707770292 --> 0.10477457371923854). Saving model!
  0%|          | 0/679 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py
  cpuset_checked))
100%|██████████| 679/679 [02:50<00:00,  3.98it/s, auc=0.928, loss=0.0993, stage=train]
  0%|          | 0/340 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py
  cpuset_checked))
100%|██████████| 340/340 [00:13<00:00, 24.69it/s, auc=0.938, loss=0.0949, stage=valid]
Validation score improved (0.10477457371923854 --> 0.09489963703295763). Saving model!
  0%|          | 0/679 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py
  cpuset_checked))
100%|██████████| 679/679 [02:47<00:00,  4.05it/s, auc=0.943, loss=0.0905, stage=train]
  0%|          | 0/340 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py
  cpuset_checked))
100%|██████████| 340/340 [00:12<00:00, 27.37it/s, auc=0.944, loss=0.0911, stage=valid]
Validation score improved (0.09489963703295763 --> 0.09105225608629339). Saving model!
  0%|          | 0/679 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py
  cpuset_checked))
100%|██████████| 679/679 [02:47<00:00,  4.06it/s, auc=0.952, loss=0.0846, stage=train]
  0%|          | 0/340 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py
  cpuset_checked))
100%|██████████| 340/340 [00:12<00:00, 27.29it/s, auc=0.945, loss=0.0891, stage=valid]
Validation score improved (0.09105225608629339 --> 0.08914107015246854). Saving model!
  0%|          | 0/679 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py
  cpuset_checked))
```

Рис. 3.11 Навчання моделі нейронної мережі

Розподіл якості прогнозування кожної мітки:

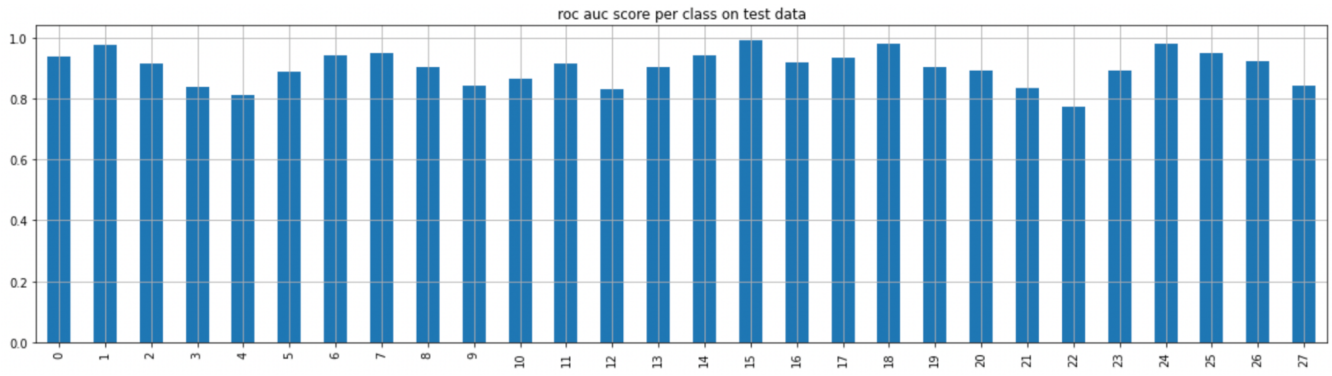

Рис. 3.12 Розподіл якості прогнозування міток емоцій

На даному графіку зображено якість розпізнавання кожної з міток емоцій, що наявні в коментарях тестового набору даних. Загалом для всіх емоцій точність їхнього виявлення є доволі високою, що свідчить про добре

натреновану модель.

Як можна побачити з графіку – найкраще розпізнаються емоції вдячності, жалю та кохання. Найгірше – горя, гордості та усвідомлення.

Приклади прогнозування:

Текстовий коментар про кохання:

```
score_sentence("Love is an unexplainable feeling. \
It’s so huge that doesn’t fit in any words, \
especially these three simple words “I love you”.  \
I want our love to be special and unique. I want it to be eternal.")
```

```
love 93.11 %
admiration 2.41 %
neutral 1.53 %
approval 1.26 %
joy 0.96 %
```

Рис. 3.13 Приклад визначених емоцій у текстовому коментарі про кохання

Коментар, в якому негативно оцінюється зовнішність людини:

```
score_sentence("Ewww, look at you, ugly face! \
You can't even fit in your moms jeans...")
```

```
disgust 64.0 %
annoyance 14.75 %
neutral 7.56 %
disapproval 7.55 %
disappointment 3.78 %
```

Рис. 3.14 Приклад визначених емоцій у негативному текстовому коментарі

Коментар, в якому міститься негативна оцінка фільму:

```
score_sentence("It is the most stupid film ever. \
I regret wasting my precious time")
```

```
annoyance 56.65 %
anger 52.05 %
disgust 5.4 %
neutral 4.95 %
disapproval 3.15 %
```

Рис. 3.15 Приклад визначених емоцій у негативному текстовому коментарі

Наведені приклади чудово демонструють роботу нейронної мережі і доводять високу точність розпізнавання емоцій, що приховані в коментарях.

3.6 Розроблений інтерфейс програмного модулю

Інтерфейс користувача (англ. User Interface, UI, дружній інтерфейс) – засіб зручної взаємодії користувача з інформаційною системою[20]. Інтерфейс даного програмного модулю розроблявся згідно нефункціональних вимог, описаних в першому розділі даного дипломного проекту та відповідє таким властивостям:

- Ясність. Елементи інтерфейсу не мають двозначності та є зрозумілими навіть при першому використанні.
- Виразність. Інтерфейс є мінімалістичним та всі його елементи уточнені.
- Відповідність. Інтерфейс забезпечує користувача гарним зворотнім зв'язком стосовно того, що відбувається, та чи успішно обробляються введені дані.

На основній сторінці веб-застосунку знаходиться заголовок, поле для вводу тексту користувачем та кнопка, що надсилає дані до програми.

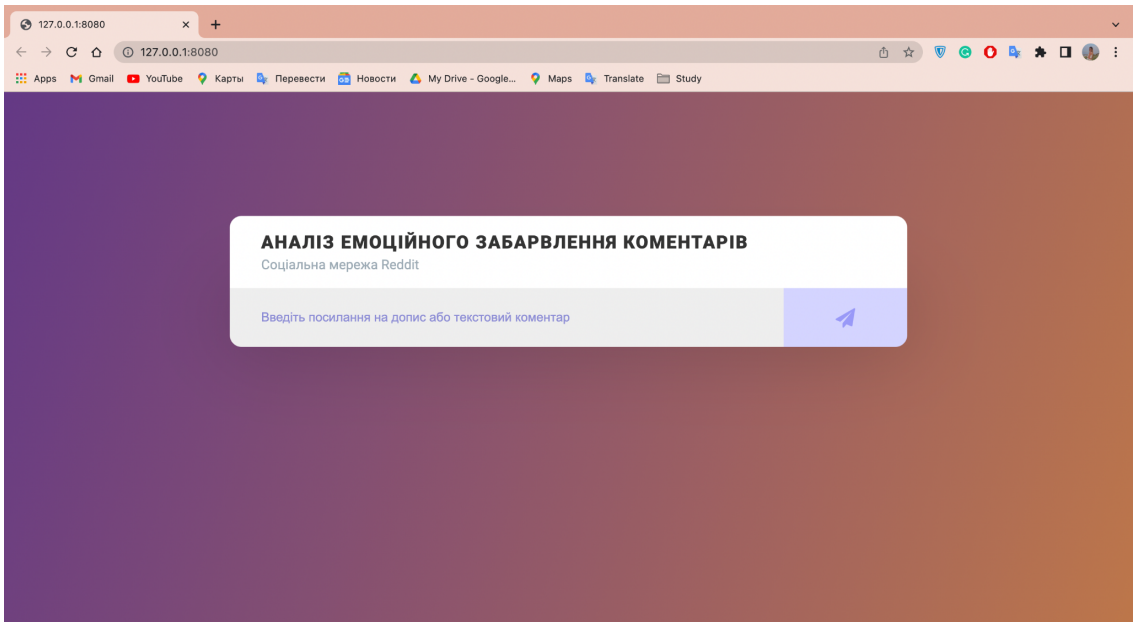

Рис. 3.16 Основна сторінка веб-застосунку

При вводі некоректних даних, користувач побачить повідомлення про помилку, яке буде містити уточнення проблеми:

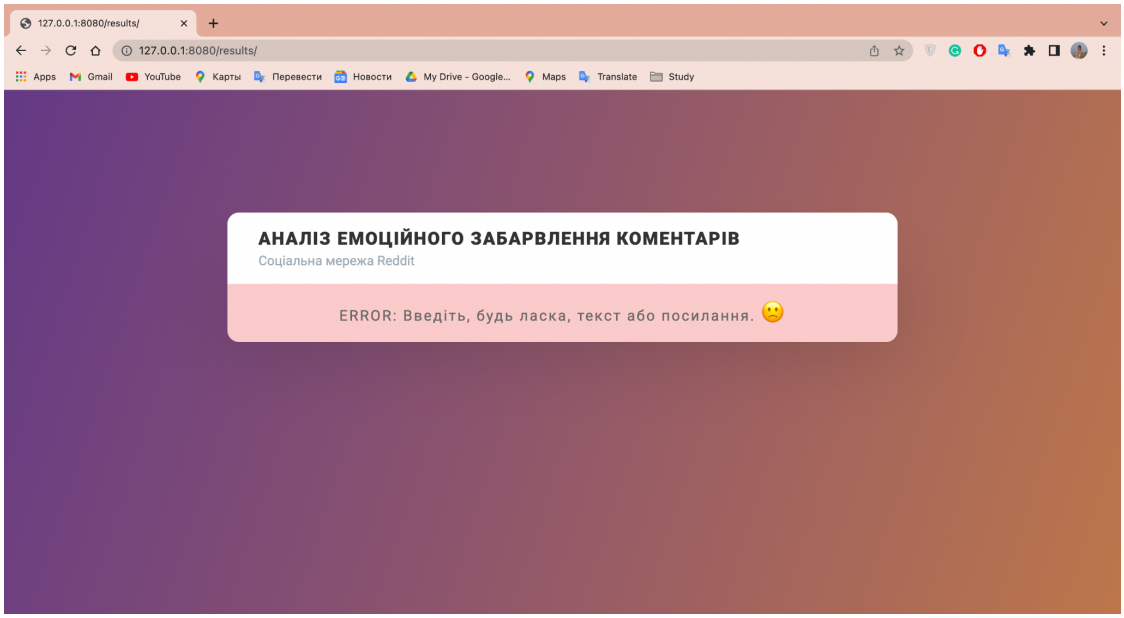

Рис. 3.17 Приклад повідомлення про помилку

При введені коректних даних, після проведеного програмним модулем аналізу, користувач потрапить на сторінку зі звітом по оцінених емоціях:

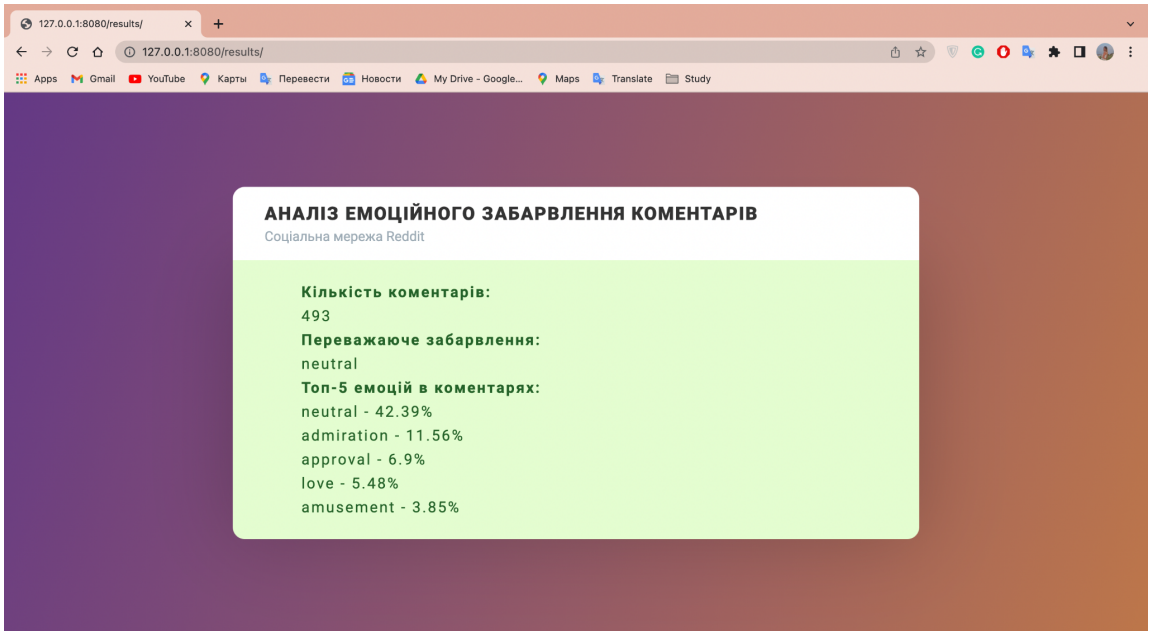

Рис. 3.18 Приклад веб-сторінки зі звітом

Але обробка та аналіз введених користувачем даних може зайняти певний час, якщо на вхід подане посилання. І під час даного процесу

користувач буде бачити анімовану картинку завантаження:

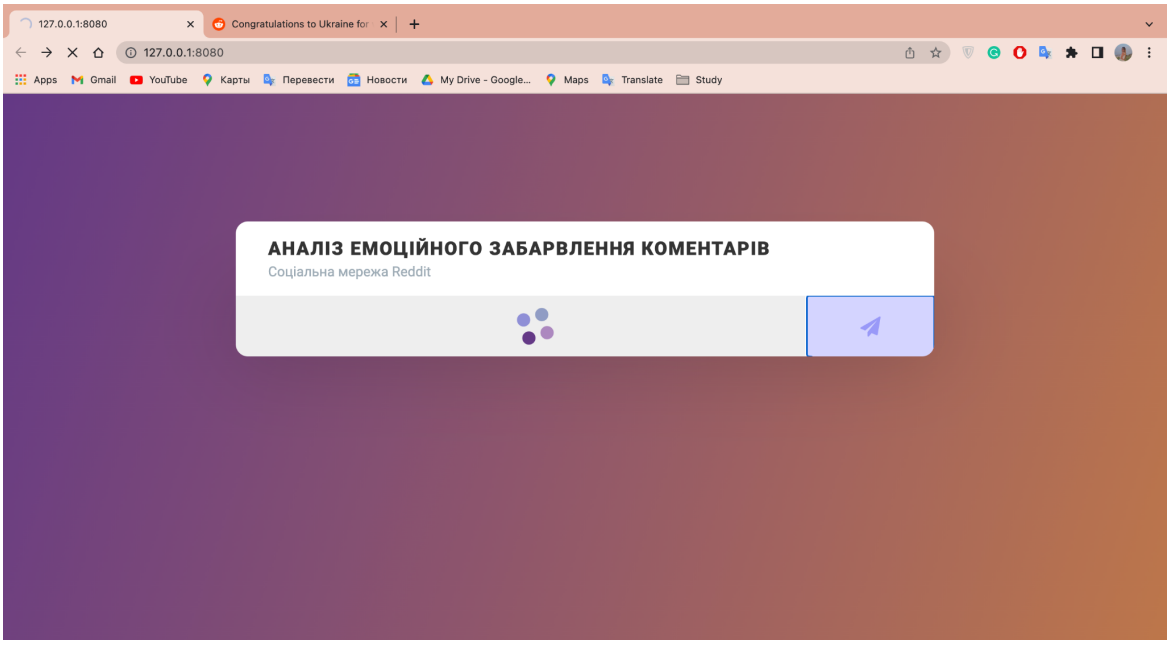

Рис. 3.19 Вигляд завантаження веб-сторінки зі звітом

Створений інтерфейс програмного модулю відповідає сформованим функціональним та нефункціональним вимогам, є зручним та інтуїтивно зрозумілим користувачеві.

3.7 Висновки до розділу

В третьому розділі дипломного проекту, що присвячений розробці програмного модулю, обґрунтовано вибір програмного середовища, описано тренувальний набір даних, продемонстровано вигляд створеного інтерфейсу модуля та блоки коду, що відповідають за основні функції роботи програми: обробки та кодування текстових даних, класу моделі нейронної мережі та алгоритму її оптимізації. Також в даному розділі було проведено оцінку результатів та продемонстровано приклади роботи програмного модулю.

ВИСНОВКИ

В даному дипломному проекті мною було розроблено програмний модуль визначення емоційного забарвлення коментарів в соціальних мережах, який реалізує багатомітковий сентиментальний аналіз для ефективної оцінки тональності тексту.

В першому розділі, що присвячений аналітичному огляду предметної області, було описано загальні теоретичні відомості досліджуваних методів та алгоритмів, а саме: бінарної класифікації, класифікації тексту та сентиментального аналізу. Також було наведено причини актуальності проблеми класифікації текстових даних, яка з розвитком Інтернету буде лише зростати. Також в цьому розділі було представлено узагальнений вигляд програмного модулю, що розробляється та визначено його функціональні та функціональні вимоги.

В другому розділі, що присвячений проектуванню програмного модулю було розглянуто особливості його архітектури, побудовано та описано дерево функцій, діаграму діяльності та загальний вигляд основних модулів програми, що розробляється. Також у даному розділі було розглянуто нейронну мережу BERT, що застосовується для вирішення поставленої задачі, та описано архітектуру трансформерів, на якій дана мережа базується.

В третьому розділі дипломного проекту, що присвячений розробці програмного модулю, обґрунтовано вибір програмного середовища, описано тренувальний набір даних, продемонстровано вигляд створеного інтерфейсу модуля та блоки коду, що відповідають за основні функції роботи програми: обробки та кодування текстових даних, класу моделі нейронної мережі та алгоритму її оптимізації. Також в даному розділі було проведено оцінку результатів та продемонстровано приклади роботи програмного модулю.

Натренована нейронна мережа BERT, що вбудована в реалізований програмний модуль, вирішує задачу багатоміткової класифікації тексту з точністю 95% на тестових даних і показує гарні результати на реальних даних.

Сам програмний модуль надає користувачам можливість проаналізувати введений ними один текстовий коментар або всі коментарі під дописом, що будуть автоматично зібрані програмою за введеним посиланням. Даний модуль наразі розроблений тільки для аналізу коментарів соціальної мережі Reddit, але в подальшому може набути підтримки й інших соціальних мереж, форумів або блогів. Тим не менш вже на даному етапі, описаний програмний модуль може набути широкого застосування як у визначенні статистики емоцій та ставлення групи людей до певної новини, події або звичайного авторського допису, так і впровадження в більш складні моніторингові системи.

Список використаних джерел

1. Рудзевич, А.-М. П. Методи машинного навчання в сентимент аналізі текстової інформації: магістерська дис: 122 Комп'ютерні науки / Рудзевич Анна-Марія Павлівна. – Київ, 2020. – 88 с.

2. Ashish Vaswani, Noam Shazeer, …: Attention is All you Need, Computer Science- Computation and Language, 2017, 15 с.

3. Хайкін С. Нейронні мережі: повний курс, 2-ге вид., Випр.: Пер. з англ. - М.: ТОВ «І.Д.Вільямс», 2006. - 1104 с.

4. Jurafsky and Martin: Speech and Language Processing, 2nd Edition, 2009, 1032 с.

5. Bird, Klein, and Loper: Natural Language Processing with Python, O'Reilly Media, Inc., 2009, 502 с.

6. Handbook of Natural Language Processing, Chapman & Hall/CRC Machine Learning & Pattern Recognition, 2nd Edition, 2010, 702 с.

7. Foster Provost, Tom Fawcett, Data Science for Business: What you need to know about data mining and data-analytic thinking, Paperback, 2013, 413 с.

8. Sakshi Ranjan, Subhankar Mishra: Comparative Sentiment Analysis of App Reviews, IEEE, 2020, 10 с.

9. Francisco Herrera, Francisco Charte, Antonio J. Rivera, María J. del Jesus: Multilabel Classification. Problem Analysis, Metrics and Techniques, Springer International Publishing Switzerland 2016, 72 с.

10. Denis Rothman: Transformers for Natural Language Processing, Packt Publishing, 2021, 384 с.

11. Sudharsan Ravichandiran: Getting Started with Google BERT, Packt Publishing, 2021, 352 с.

12. https://www.researchgate.net/publication/341264906_Twitter_Sentiment_Analysis_using_Supervised_Machine_Learning

13. https://www.mygreatlearning.com/blog/multiclass-classification-explained/
14. https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML#:~:text=Machine%20learning%20(ML)%20is%20a,to%20predict%20new%20output%20values.
15. https://arxiv.org/pdf/1512.01043.pdf
16. https://ua.sawakinome.com/articles/technology/difference-between-functional-and-non-functional-requirements-2.html
17. https://www.researchgate.net/publication/341264906_Twitter_Sentiment_Analysis_using_Supervised_Machine_Learning
18. https://machinelearningmastery.com/the-transformer-attention-mechanism/
19. https://paperswithcode.com/dataset/goemotions
20. https://docs.python.org/3/

Додатки

Додаток А: програмний код веб-застосунку:

```
from flask import Flask

from core.main.greetings import GreetingsRender
from core.processing.result import ResultRender

application = Flask(__name__)

GreetingsRender.register(application, route_base='/')
ResultRender.register(application, route_base='/results')

if __name__ == '__main__':
    application.run(host='0.0.0.0')
```

```
import requests

from flask import render_template
from flask import request

from flask_classful import FlaskView

from pydantic import ValidationError

from core.processing.models import InputForm
from core.processing.report import generate_report

DEFAULT_MODEL_ADDRESS = 'http://127.0.0.1:5000/predict'

class ResultRender(FlaskView):
    def get(self):
        return render_template('index.html')

    def post(self):
        form_data = request.form.to_dict(flat=True)

        if not form_data:
            return render_template('error_page.html', message={'text':
'Nothing to process'})
        try:
            val_data = InputForm(**form_data).dict()
        except ValidationError as error:
            print(error.errors())
            return render_template('error_page.html', message={'text':
error.errors()[-1]['msg']})
        else:
            response = requests.get(DEFAULT_MODEL_ADDRESS, json={'api_key':
1234, 'input_string': val_data['text']}).json()
            # print(response)

        print(request.args.keys())
        n, df = generate_report(response)
        return render_template('report.html', message={'num': n, 'top5': df})
```

```
import pandas as pd


def generate_report(response: dict):
    df = pd.DataFrame(response['result'])
```

```
    if df['input'].nunique() == 1:
        a = [[row['class'], row['score']] for row in df['scores'].head()]
        return 1, a
    else:
        df_exploded = df.explode(['scores'])
        df_scores_parsed = pd.json_normalize(df_exploded['scores'])
        df_scores_parsed['idx'] = df_exploded.index
        df_scores_parsed =
df_scores_parsed.set_index(df_scores_parsed['idx'])
        df_res = pd.concat([df_exploded, df_scores_parsed], axis=1)
        total_df = pd.pivot_table(data=df_res, values=['score'],
index=['input'], columns=['class']).reset_index()
        a = round(total_df['score'].idxmax(axis=1).value_counts(1) * 100,
2).head()
        res_df = pd.DataFrame(a).reset_index().rename(columns={'index':
'emotion', 0: 'ratio'}).to_dict('records')
        return total_df.shape[0], pd.DataFrame(res_df).values.tolist()
```

```
from pydantic import BaseModel, validator


class InputForm(BaseModel):
    text: str

    @validator('text')
    def check_data(cls, v):
        if not v or v.isdigit():
            raise ValueError('Введіть, будь ласка, текст або посилання.')
        if len(v) < 10:
            raise ValueError('Введений текст дуже короткий.')
        if 'http' in v and 'reddit.com' not in v:
            raise ValueError('Інші соц. мережі, крім Reddit, поки не
підтримуються, але Ви можете ввести скопійований текст.')
        return v.title()
```

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1">

   <link rel="stylesheet" href="/static/styles/styles.css">
   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.5.0/css/font-awesome.min.css">
   <link href='https://fonts.googleapis.com/css?family=Roboto:400,700,900'
rel='stylesheet' type='text/css'>

   <title></title>
</head>
<body>

   <div class="back"></div>

   <div class="registration-form">
      <header>
           <h1>Аналіз емоційного забарвлення коментарів</h1>
           <p>Соціальна мережа Reddit</p>
       </header>

       <form action="/results" class="form" method="POST" autocomplete="off">
          <div class="input-section link-section">
```

```
            <input class="link" type="text" name="text" id="text"
                   placeholder="Введіть посилання на допис або текстовий
коментар" autocomplete="off" />
             <button class="animated-button" type="submit"
onclick="myLoad()">
                <span class="icon-paper-plane">
                    <i class="fa fa-paper-plane"></i></i>
                 </span>
                  <div id="loader"></div>
             </button>
          </div>
       </form>
   </div>

    <script>
        function myLoad() {
            document.getElementById("text").style.color = "#eee";
            document.getElementById("loader").style.display = "block";
        }
    </script>
</body>
</html>
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="stylesheet" href="/static/styles/styles.css">
   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.5.0/css/font-awesome.min.css">
   <link href='https://fonts.googleapis.com/css?family=Roboto:400,700,900'
rel='stylesheet' type='text/css'>
   <title>

   </title>
</head>
<body>
   <div class="back"></div>
   <div class="report">

       <header>
           <h1>Аналіз емоційного забарвлення коментарів</h1>
           <p>Соціальна мережа Reddit</p>
       </header>

       <div class="report-section">
          <p>
             <b>Кількість коментарів:</b>
             <br>{{ message['num'] }}
                <br><b>Переважаюче забарвлення:</b>
                <br>{{ message['top5'][0][0] }}
             <br><b>Топ-5 емоцій в коментарях:</b>
                <br>{{ message['top5'][0][0] }}  -  {{ message['top5'][0][1]
}}%
                <br>{{ message['top5'][1][0] }}  -  {{ message['top5'][1][1]
}}%
                <br>{{ message['top5'][2][0] }}  -  {{ message['top5'][2][1]
}}%
                <br>{{ message['top5'][3][0] }}  -  {{ message['top5'][3][1]
}}%
                <br>{{ message['top5'][4][0] }}  -  {{ message['top5'][4][1]
}}%
          </p>
```

```
        </div>
    </div>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="/static/styles/styles.css">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.5.0/css/font-awesome.min.css">
    <link href='https://fonts.googleapis.com/css?family=Roboto:400,700,900'
rel='stylesheet' type='text/css'>
    <title>

    </title>
</head>
<body>
    <div class="back"></div>
    <div class="error">

        <header>
            <h1>Аналіз емоційного забарвлення коментарів</h1>
            <p>Соціальна мережа Reddit</p>
        </header>

        <div class="error-section">
           <p> ERROR: {{ message['text'] }} <span style='font-
size:30px;'>&#128577;</span></p>
        </div>
    </div>
</body>
</html>
```

Додаток Б: програмний код інтерфейсу нейронної мережі:

```
PROD_PORT=8080

API_KEY=1234

SEED=42

ENABLE_TRANSFER_LEARNING=1

PRETRAINED_BERT_MODEL=squeezebert/squeezebert-uncased
APP_TRAINED_MODEL=files/ml/bert-m.pth
OUTPUT_MULTI_LABELS=28

USER_AGENT='sentiment_analysis'
CLIENT_ID='BsxobdWkVEGrhXrZFbl4rg'
CLIENT_SECRET='ICtjJZJGb-awVEQOdpsyvZMoLM75yQ'

import multiprocessing as mp

from datetime import datetime

from flask import request
from flask_restful import Resource

from services.predict.validation import Inputs
```

```
from services.predict.model import Estimation

from services.predict.reddit import get_comments

from pydantic import ValidationError
from loguru import logger

from utils.config import is_url


class InputClassification(Resource):
    """
        Pass
    """

    def __init__(self, model, tokenizer, transfer_learning):
        self.model = model
        self.tokenizer = tokenizer
        self.transfer_learning = transfer_learning

    def get(self) -> dict:
        """
            Pass
        """
        output = {'time': str(datetime.now())}
        try:
            args = request.get_json()
            if args is None:
                args = {}

            val_args = Inputs(**args).dict()
            val_args = is_url(val_args['input_string'])
            _input = val_args['input_string'].lower()

            if val_args['useParser']:
                _input = get_comments(_input)

            model_interface = Estimation(model=self.model,
                                         tokenizer=self.tokenizer,

transfer_learning=self.transfer_learning
                                         )

            if isinstance(_input, list):
                with mp.Pool(processes=mp.cpu_count()) as pool:
                    result = pool.map(model_interface.score_sentence, _input)
                # result = [model_interface.score_sentence(i) for i in
_input]

            else:
                result = model_interface.score_sentence(_input)

            output.update({'result': {"scores": result, "input": _input}})

        except ValidationError as val_error:
            logger.error(val_error.json())
            output.update({
                'status': 'ValidationError',
                'message': val_error.errors()
            })
        return output
```

```
import torch.nn as nn

from sklearn import metrics
from transformers import AdamW, get_linear_schedule_with_warmup

from utils.config import mapping


class EmotionClassifier(tez.Model):
    """
        Model class
    """

    def __init__(self, num_train_steps, num_classes, bert):
        super().__init__()
        self.bert = bert
        self.bert_drop = nn.Dropout(0.3)
        self.out = nn.Linear(768, num_classes)
        self.num_train_steps = num_train_steps
        self.step_scheduler_after = "batch"

    def fetch_optimizer(self):
        param_optimizer = list(self.named_parameters())
        no_decay = ["bias", "LayerNorm.bias"]
        optimizer_parameters = [
            {
                "params": [
                    p for n, p in param_optimizer if not any(nd in n for nd
in no_decay)
                ],
                "weight_decay": 0.001,
            },
            {
                "params": [
                    p for n, p in param_optimizer if any(nd in n for nd in
no_decay)
                ],
                "weight_decay": 0.0,
            },
        ]
        opt = AdamW(optimizer_parameters, lr=3e-5)
        return opt

    def fetch_scheduler(self):
        """
            Scheduler
        """
        sch = get_linear_schedule_with_warmup(
            self.optimizer, num_warmup_steps=0,
num_training_steps=self.num_train_steps
        )
        return sch

    @staticmethod
    def loss(outputs, targets):
        if targets is None:
            return None
        return nn.BCEWithLogitsLoss()(outputs, targets.float())

    @staticmethod
    def monitor_metrics(outputs, targets) -> dict:
        out_dict = {}
        if not targets:
            return out_dict
```

```
        outputs = torch.sigmoid(outputs)
        outputs = outputs.cpu().detach().numpy()
        targets = targets.cpu().detach().numpy()

        fpr_micro, tpr_micro, _ = metrics.roc_curve(targets.ravel(),
outputs.ravel())
        auc_micro = metrics.auc(fpr_micro, tpr_micro)
        out_dict.update({"auc": auc_micro})

        return out_dict

    def forward(self, ids, mask, targets=None):
        o_2 = self.bert(ids, attention_mask=mask)["pooler_output"]
        b_o = self.bert_drop(o_2)
        output = self.out(b_o)
        loss = self.loss(output, targets)
        acc = self.monitor_metrics(output, targets)
        return output, loss, acc


class Estimation:
    """
        Pass
    """

    def __init__(self, model, tokenizer, transfer_learning):
        self.model = model
        self.tokenizer = tokenizer
        self.transfer_learning = transfer_learning

    def tokenize(self, text) -> [torch.LongTensor, torch.LongTensor]:
        """
            Tokenize text,
        """
        MAX_LEN = None

        if self.transfer_learning:
            MAX_LEN = 35

        inputs = self.tokenizer.encode_plus(text,
                                            None,
                                            add_special_tokens=True,
                                            max_length=MAX_LEN,
                                            padding="max_length",
                                            truncation=True)

        ids = inputs["input_ids"]
        ids = torch.LongTensor(ids).cpu().unsqueeze(0)

        attention_mask = inputs["attention_mask"]
        attention_mask = torch.LongTensor(attention_mask).cpu().unsqueeze(0)

        return ids, attention_mask

    def score_sentence(self, text, top_labels=28):
        """
            Pass
        """

        with torch.no_grad():
            ids, attention_mask = self.tokenize(text)

            if self.transfer_learning:
```

```
                output = self.model.forward(ids, attention_mask)[0]
                output = torch.sigmoid(output)

            if not self.transfer_learning:
                output = self.model(ids, attention_mask)

                # noinspection PyUnresolvedReferences
                output = nn.functional.softmax(output.logits, dim=-1)

            probas, indices = torch.sort(output)

        probas = probas.cpu().numpy()[0][::-1]
        indices = indices.cpu().numpy()[0][::-1]

        if self.transfer_learning:
            mapper = mapping

        else:
            mapper = self.model.config.id2label

        result = [
            {
                'class': mapper[i], 'score': round(p * 100, 2)
            }
            for i, p in zip(indices[:top_labels], probas[:top_labels])
        ]

        return result

    def __del__(self):
        gc.collect()
```

```
import os
import praw

from dotenv import load_dotenv

load_dotenv()


def get_comments(input_url) -> list:
    reddit = praw.Reddit(
        user_agent=os.environ['USER_AGENT'],
        client_id=os.environ['CLIENT_ID'],
        client_secret=os.environ['CLIENT_SECRET']
    )

    submission = reddit.submission(url=input_url)
    submission.comments.replace_more(limit=0)

    return [comment.body for comment in submission.comments.list()]
```

```
import os
import torch
import random

import transformers

import numpy as np

from pathlib import Path
```

```
from flask import Flask
from flask_restful import Api
from loguru import logger

from services.predict.model import EmotionClassifier

from services.health.status import ApiHealth
from services.predict.classification import InputClassification

from dotenv import load_dotenv

load_dotenv()

seed = int(os.environ['SEED'])
os.environ['PYTHONHASHSEED'] = str(seed)

# Torch RNG
torch.manual_seed(seed)
torch.cuda.manual_seed(seed)
torch.cuda.manual_seed_all(seed)

# Python RNG
np.random.seed(seed)
random.seed(seed)

device = torch.device('cpu')
global model, tokenizer


TRANSFER_LEARNING_FLAG = int(os.environ['ENABLE_TRANSFER_LEARNING'])
PRE_TRAINED_MODEL_NAME = os.environ['PRETRAINED_BERT_MODEL']

if TRANSFER_LEARNING_FLAG:
    logger.debug("Init self trained model")
    # Application model
    BERT_PRETRAINED =
transformers.SqueezeBertModel.from_pretrained(PRE_TRAINED_MODEL_NAME)

    APP_MODEL = Path(os.environ['APP_TRAINED_MODEL']).absolute()
    model = EmotionClassifier(1, int(os.environ['OUTPUT_MULTI_LABELS']),
bert=BERT_PRETRAINED)
    model.load_state_dict(torch.load(APP_MODEL, map_location=device))

    tokenizer =
transformers.SqueezeBertTokenizer.from_pretrained(PRE_TRAINED_MODEL_NAME,
do_lower_case=True)

if not TRANSFER_LEARNING_FLAG:
    logger.debug("Init remote trained model")
    model =
transformers.AutoModelForSequenceClassification.from_pretrained(PRE_TRAINED_M
ODEL_NAME)
    tokenizer =
transformers.AutoTokenizer.from_pretrained(PRE_TRAINED_MODEL_NAME)


application = Flask(__name__)
api = Api(application)
api.add_resource(ApiHealth, '/')
api.add_resource(InputClassification, '/predict',
                 resource_class_kwargs={'model': model,
                                        'tokenizer': tokenizer,
                                        'transfer_learning':
```

```
TRANSFER_LEARNING_FLAG})

if __name__ == '__main__':
    load_dotenv()
    application.run()
```

```
import os

from pydantic import BaseModel, validator

GENERAL_API_KEYS = {
    int(os.environ['API_KEY'])
}


class Inputs(BaseModel):
    """
        Val model service input
    """
    api_key: int
    input_string: str

    @validator('api_key')
    def check_api_key(cls, v):
        """
            Pass
        """
        if v not in GENERAL_API_KEYS:
            raise ValueError('Invalid API-key.')
        return v
```

```
from datetime import datetime

from flask_restful import Resource


class ApiHealth(Resource):
    """
        Just to check if server works well
    """
    def get(self) -> dict:
        """
            Get method
        """
        return {'status': 'ok', 'time': str(datetime.now())}
```

```
labels = ['admiration', 'amusement', 'anger', 'annoyance', 'approval',
          'caring', 'confusion', 'curiosity', 'desire', 'disappointment',
          'disapproval', 'disgust', 'embarrassment', 'excitement', 'fear',
'gratitude',
          'grief', 'joy', 'love', 'nervousness', 'optimism', 'pride',
'realization', 'relief',
          'remorse', 'sadness', 'surprise', 'neutral']

mapping = {i: v for i, v in zip(range(len(labels)), labels)}


def is_url(text: str) -> dict:
    output = {
        "input_string": text,
        "useParser": False
```

```
    }

    if 'http' in text.lower():
        output['useParser'] = True

    return output
```

Додаток В: стилі веб-застосунку

```
body {
   font-family: "Roboto";
}

.back {
  background: linear-gradient(120grad, #643986, #bf7849);
  position: absolute;
  width: 100%;
  height: 100%;
  top: 0;
  left: 0;
  right: 0;
}

.registration-form {
  width: 60%;
  position: absolute;
  left: 50%;
  transform: translate(-50%, 0%);
  top: 20%;
  background: transparent;
}

.registration-form header {
  position: relative;
  z-index: 4;
  background: white;
  padding: 20px 40px;
  border-radius: 15px 15px 0 0;
}

.registration-form header h1 {
  font-weight: 900;
  letter-spacing: 1.5px;
  color: #333;
  font-size: 23px;
  text-transform: uppercase;
  margin: 0;
}

.registration-form header p {
  word-spacing: 0px;
  color: #9facb6;
  font-size: 17px;
  margin: 0;
  margin-top: 5px;
}

.registration-form form {
  position: relative;
}
```

```
.registration-form .input-section {
  width: 100%;
  position: absolute;
  display: flex;
  left: 50%;
  transform: translate(-50%, 0);
  height: 75px;
  border-radius: 0 0 15px 15px;
  overflow: hidden;
  z-index: 2;
  box-shadow: 0px 0px 100px rgba(0, 0, 0, 0.2);
  transition: all 0.2s ease-in;
}

.registration-form form input {
  background: #eee;
  color: #8f8fd6;
  width: 80%;
  border: 0;
  padding: 20px 40px;
  margin: 0;
  font-size: 16px;
}

.registration-form form input:focus {
  outline: none;
}

.registration-form form input::placeholder {
  color: #8f8fd6;
  font-weight: 100;
}

button {
  border: none;
  cursor: pointer;
}

.animated-button {
  width: 20%;
  background-color: #d4d4ff;
}

.animated-button span {
  display: flex;
  flex-direction: row;
  justify-content: space-around;
  align-items: center;
  line-height: 75px;
  text-align: center;
  height: 75px;
  transition: all 0.2s ease-in;
}

.animated-button span i {
  font-size: 25px;
  color: #9999f8;
}

.animated-button .next-button {
  background: transparent;
  color: #9999f8;
  font-weight: 100;
```

```
  width: 100%;
  border: 0;
}

.next {
  margin-top: -75px;
}

#loader  {
  animation: rotate 1s infinite;
  position: absolute;
  display:none;
  left: 40%;
  top: 20%;
  z-index: 1;
  width: 40px;
  height: 40px;
}

#loader:before,
#loader:after {
  border-radius: 50%;
  content: '';
  display: block;
  height: 20px;
  width: 20px;
}
#loader:before {
  animation: ball1 1s infinite;
  background-color: #8f8fd6;
  box-shadow: 30px 0 0 #ac88bf;
  margin-bottom: 10px;
}
#loader:after {
  animation: ball2 1s infinite;
  background-color: #643986;
  box-shadow: 30px 0 0 #909cc4;
}

@keyframes rotate {
  0% {
    -webkit-transform: rotate(0deg) scale(0.8);
    -moz-transform: rotate(0deg) scale(0.8);
  }
  50% {
    -webkit-transform: rotate(360deg) scale(1.2);
    -moz-transform: rotate(360deg) scale(1.2);
  }
  100% {
    -webkit-transform: rotate(720deg) scale(0.8);
    -moz-transform: rotate(720deg) scale(0.8);
  }
}

@keyframes ball1 {
  0% {
    box-shadow: 30px 0 0 #909cc4;
  }
  50% {
    box-shadow: 0 0 0 #909cc4;
    margin-bottom: 0;
    -webkit-transform: translate(15px,15px);
    -moz-transform: translate(15px, 15px);
```

```
  }
  100% {
    box-shadow: 30px 0 0 #909cc4;
    margin-bottom: 10px;
  }
}

@keyframes ball2 {
  0% {
    box-shadow: 30px 0 0 #ac88bf;
  }
  50% {
    box-shadow: 0 0 0 #ac88bf;
    margin-top: -20px;
    -webkit-transform: translate(15px,15px);
    -moz-transform: translate(15px, 15px);
  }
  100% {
    box-shadow: 30px 0 0 #ac88bf;
    margin-top: 0;
  }
}

.success {
  width: 100%;
  position: absolute;
  display: flex;
  align-items: center;
  left: 50%;
  transform: translate(-50%, 0);
  height: 75px;
  border-radius: 0 0 15px 15px;
  overflow: hidden;
  z-index: 2;
  box-shadow: 0px 0px 100px rgba(0, 0, 0, 0.2);
  transition: all 0.2s ease-in;
  background: limegreen;
  margin-top: -75px;
}

.success p {
  color: white;
  font-weight: 900;
  letter-spacing: 2px;
  font-size: 18px;
  width: 100%;
  text-align: center;
}

.report{
  width: 60%;
  position: absolute;
  left: 50%;
  transform: translate(-50%, 0%);
  top: 15%;
  background: transparent;
}

.report header {
  position: relative;
  z-index: 4;
  background: white;
  padding: 20px 40px;
```

```
  border-radius: 15px 15px 0 0;
}

.report header h1 {
  font-weight: 900;
  letter-spacing: 1.5px;
  color: #333;
  font-size: 23px;
  text-transform: uppercase;
  margin: 0;
}

.report header p {
  word-spacing: 0px;
  color: #9facb6;
  font-size: 17px;
  margin: 0;
  margin-top: 5px;
}

.report form {
  position: relative;
}

.report .report-section {
  width: 100%;
  position: absolute;
  display: flex;
  align-items: center;
  left: 50%;
  transform: translate(-50%, 0);
  height: 350px;
  border-radius: 0 0 15px 15px;
  overflow: hidden;
  z-index: 2;
  box-shadow: 0px 0px 100px rgba(0, 0, 0, 0.2);
  transition: all 0.2s ease-in;
  background: #e4fdd0;
}

.report .report-section p{
  color: #256329;
  font-weight: 500;
  letter-spacing: 2px;
  font-size: 19px;
  width: 100%;
  text-align: left;
  margin-left: 10%;
  line-height: 30px;
}

.error{
  width: 60%;
  position: absolute;
  left: 50%;
  transform: translate(-50%, 0%);
  top: 20%;
  background: transparent;
}

.error header {
  position: relative;
  z-index: 4;
```

```
  background: white;
  padding: 20px 40px;
  border-radius: 15px 15px 0 0;
}

.error header h1 {
  font-weight: 900;
  letter-spacing: 1.5px;
  color: #333;
  font-size: 23px;
  text-transform: uppercase;
  margin: 0;
}

.error header p {
  word-spacing: 0px;
  color: #9facb6;
  font-size: 17px;
  margin: 0;
  margin-top: 5px;
}

.error form {
  position: relative;
}

.error .error-section {
  width: 100%;
  position: absolute;
  display: flex;
  align-items: center;
  left: 50%;
  transform: translate(-50%, 0);
  height: 75px;
  border-radius: 0 0 15px 15px;
  overflow: hidden;
  z-index: 2;
  box-shadow: 0px 0px 100px rgba(0, 0, 0, 0.2);
  transition: all 0.2s ease-in;
  background: #facbcb;
}

.error .error-section p{
  color: dimgrey;
  font-weight: 500;
  letter-spacing: 2px;
  font-size: 19px;
  width: 100%;
  text-align: center;
}
```