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

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«На правах рукопису»
УДК 004.91

«До захисту допущено»
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___»_____2025 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-науковою програмою
«Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

зі спеціальності 121 Інженерія програмного забезпечення

на тему: «Метод та програмне забезпечення автоматичної класифікації природномовних текстових даних за наявністю агресії»

Виконав:
студент ІІ курсу, групи КП-31мн
Гришаєв Дмитро Ігорович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент,
Заболотня Тетяна Миколаївна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри СПіСКС, к.т.н., доцент,
Тарасенко-Клятченко Оксана Володимирівна _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2025 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – другий (магістерський)

Спеціальність (спеціалізація) – 121 «Інженерія програмного забезпечення»

Освітньо-наукова програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___»_____2023 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Гришаєву Дмитру Ігоровичу

1. Тема дисертації «Метод та програмне забезпечення автоматичної класифікації природномовних текстових даних за наявністю агресії», науковий керівник дисертації Заболотня Тетяна Миколаївна, к.т.н., доцент, затверджені наказом по університету №1219-С від «21» березня 2025 р.

2. Термін подання студентом дисертації «16» травня 2025 р.

3. Об'єкт дослідження: процес оброблення природномовних текстових даних для їх класифікації за наявністю агресії.

4. Предмет дослідження: методи, способи та алгоритми виявлення ознак агресії в природномовних текстових даних.

5. Перелік завдань, які потрібно розробити:
- дати визначення поняття агресії та розглянути її можливі ознаки в природномовних текстових даних;
- провести аналіз існуючих методів автоматичної класифікації природномовних текстових даних за наявністю агресії, виявити їх переваги та недоліки;
- проаналізувати метод автоматичної класифікації природномовних текстових даних за наявністю агресії на основі ALBERT;
- розробити метод автоматичної класифікації природномовних текстових даних за наявністю агресії;
- обрати та описати матрики для оцінювання ефективності запропонованого методу;
- виконати програмну реалізацію розробленого методу та оцінити її ефективність;
- провести порівняльний аналіз результатів роботи розробленого та базового методів.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:
- схема методу автоматичної класифікації природномовних текстових даних за наявністю агресії;

- архітектура програмного забезпечення автоматичної класифікації природномовних текстових даних за наявністю агресії;
- гістограма порівняння значень метрики Loss;
- гістограма порівняння значень метрики R^2;
- гістограма порівняння значень метрики MAE;
- гістограма порівняння значень метрики Accuracy.

7. Орієнтовний перелік публікацій:
- Тези доповіді “Програмна реалізація модифікованого методу автоматичної класифікації природномовних текстових даних за наявністю агресії”.

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент кафедри ПЗКС		

9. Дата видачі завдання «04» жовтня 2023 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	17.10.2023	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	04.12.2023	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	15.02.2024	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення	05.04.2024	
5.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	15.05.2024	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації	15.06.2024	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу; підготовка матеріалів доповіді на конференції ПМК-2024	05.11.2024	
8.	Оформлення текстової і графічної частини магістерської дисертації	31.01.2025	

Студент Дмитро ГРИШАЄВ

Науковий керівник Тетяна ЗАБОЛОТНЯ

РЕФЕРАТ

Актуальність теми. В останні роки прослідковується зростання кількості агресивних висловлювань в мережі Інтернет. Найчастіше з даною проблемою стикаються соціальні мережі, які пропонують користувачам певний рівень анонімності. Почуття безкарності зменшує страх перед наслідками, тому люди все більше вдаються до образливих, грубих та агресивних коментарів. В результаті чого певні шари суспільства часто стають об'єктом знущань, що вкрай негативно вливає на їх психологічних стан. Агресивні висловлювання часто використовуються у якості інструмента для політичних цілей, що може значно впливати на загальний інформаційний простір та настрій суспільства.

Власники соціальних мереж – одні з найбільш зацікавлених сторін у вирішенні проблеми розповсюдження агресивного контенту. Розвиток їх продуктів напряму пов'язаний із збільшенням обсягів природномовних текстових даних, які потрібно модерувати. Очевидно, що найбільш пріоритетним завданням у даному контексті є створення та оптимізація методів автоматичної класифікації природномовних текстових даних за наявністю агресії.

Варто зазначити, процес автоматичного сентимент аналізу, зокрема визначення ознак агресії, є багатогранним та комплексним. Визначення та обчислення великої кількості параметрів, що можуть вказувати на певні настрої, становлять основу для створення відповідних класифікаторів. При цьому, існує велика кількість різноманітних методів для пошуку агресії у природномовних текстових даних. Але кожен з них має певні обмеження та не демонструє абсолютної точності роботи.

Отже, дослідження, спрямоване на підвищення точності автоматичної класифікації природномовних текстових даних, є актуальним завданням, реалізація якого сприятиме вдосконаленню процесів аналізу та модерації текстової інформації в рамках онлайн-сервісів.

Об’єктом дослідження є процес оброблення природномовних текстових даних для їх класифікації за наявністю агресії.

Предметом дослідження є методи, способи та алгоритми виявлення ознак агресії в природномовних текстових даних.

Мета роботи: підвищення точності автоматичної класифікації природномовних текстових даних шляхом розроблення методу, який враховує додаткові структурні, лексичні та морфологічні ознаки агресії.

Методи дослідження. В роботі використовуються методи машинного навчання, статистичного, лексичного та морфологічного аналізу. Також у дослідженні застосовано методи класифікації тексту, методи векторизації тексту та методи оброблення емодзі.

Наукова новизна. Уперше розроблено метод автоматичної класифікації природномовних текстових даних за наявністю агресії, характерною рисою якого є використання розширеної кількості шарів у моделі ALBERT у поєднанні з додатковими структурними, лексичними, морфологічними ознаками, що підвищує точність визначення факту наявності агресії у тексті на 2-4% порівняно з базовим методом на основі ALBERT.

Практична цінність. На основі запропонованого методу реалізовано програмне забезпечення автоматичної класифікації природномовних текстових даних за наявністю агресії. Розроблена програмна реалізація запропонованого методу може бути використана як окремий модуль для систем оброблення природної мови або соціальних мереж для автоматизації фільтрації текстових даних.

Апробація роботи. Основні положення і результати роботи були представлені та обговорювались на науковій конференції магістрантів та аспірантів «Прикладна математика та комп’ютинг» ПМК 2024 (20-22 листопада, 2024 р., м. Київ).

Розширений опис запропонованого методу обговорюватиметься на міжнародній науковій конференції «The 8th International Conference on

Computer Science, Engineering and Education Applications (ICCSEEA2025)» (7-8 червня, 2025 р.).

Структура та обсяг роботи. Магістерська дисертація складається з вступу, чотирьох розділів, висновків та додатків.

У вступі надано загальну характеристику роботи, виконано оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі дослідження.

У першому розділі описано проблему класифікації природновомних текстових даних за наявністю агресії. Розглянуто основні ознаки агресії в тексті. Проаналізовано основні методи сентимент-аналізу тексту, наведено їх переваги та недоліки.

У другому розділі розглянуто базовий метод автоматичної класифікації природномовних текстових даних за наявності агресії на основі моделі ALBERT. Проаналізовано наявні проблеми базового методу до визначення агресії в тексті. Описано запропонований спосіб попереднього оброблення вхідних текстових даних. Розглянуто структурні, лексичні та морфологічні особливості текстових даних а також спосіб врахування впливу емодзі на емоційне забарвлення тексту. Представлено запропонований метод автоматичної класифікації природномовних текстових даних за наявності агресії. Обґрунтовано вибір метрик для оцінювання ефективності запропонованого методу.

У третьому розділі обґрунтовано вибір інструментів, бібліотек і технологій, що використовуються в програмній реалізації запропонованого методу. Описано архітектуру програмного забезпечення та особливості реалізації основних етапів оброблення текстів і їх класифікації за наявністю агресії. Наведено приклади результатів роботи програми.

У четвертому розділі проведено аналіз ефективності розробленої програмної реалізації та порівняння запропонованого методу з базовим та похідними від нього, що були розроблені в процесі дослідження. Наведено

обґрунтування отриманих результатів. Визначено напрямки подальшої роботи в рамках даного дослідження.

У висновках проаналізовано отримані результати роботи.

У додатках наведено графічні матеріали до пояснювальної записки, скріншоти презентації та лістинги коду розробленого програмного забезпечення.

Робота виконана на 98 аркушах, містить 3 додатки та посилання на список використаних літературних джерел з 59 найменувань. У роботі наведено 16 рисунків, 12 лістингів.

Ключові слова: інженерія програмного забезпечення, класифікація тексту, сентимент-аналіз, ALBERT, машинне навчання, ознаки агресії, емодзі.

ABSTRACT

Theme urgency. In recent years, there has been an increase in the number of aggressive statements on the Internet. This problem is most often encountered by social networks that offer users a certain level of anonymity. A sense of impunity reduces fear of consequences, so people increasingly resort to offensive, rude and aggressive comments. As a result, certain segments of society often become the object of bullying, which has an extremely negative impact on their psychological state. Aggressive statements are often used as a tool for political purposes, which can significantly affect the general information space and the mood of society.

Owners of social networks are one of the most interested parties in solving the problem of the spread of aggressive content. The development of their products is directly related to the increase in the volume of natural language text data that needs to be moderated. Obviously, the most priority task in this context is the creation and optimization of methods for automatic classification of natural language text data for the presence of aggression.

It is worth noting that the process of automatic sentiment analysis, in particular the identification of signs of aggression, is multifaceted and complex. The identification and calculation of a large number of parameters that can indicate certain moods form the basis for creating appropriate classifiers. At the same time, there are a large number of different methods for searching for aggression in natural language text data. But each of them has certain limitations and does not demonstrate absolute accuracy.

Therefore, research aimed at increasing the accuracy of automatic classification of natural language text data is a relevant task, the implementation of which will contribute to the improvement of the processes of analysis and moderation of text information within online services.

Object of research is the process of processing natural language text data for their classification by the presence of aggression.

Subject of research is methods, methods and algorithms for detecting signs of aggression in natural language text data.

Research objective: to increase the accuracy of automatic classification of natural language text data by developing a method that takes into account additional structural, lexical and morphological features of aggression.

Research methods. The work uses methods of machine learning, statistical, lexical and morphological analysis. The study also uses methods of text classification, text vectorization methods and emoji processing methods.

Scientific novelty. For the first time, a method for automatic classification of natural language text data for the presence of aggression has been proposed, based on the ALBERT model using an expanded number of layers in combination with additional structural, lexical, and morphological features, which increases the accuracy of determining the presence of aggression in the text by 2-4% compared to the basic method based on ALBERT.

Practical value. Based on the proposed method, software for automatic classification of natural language text data by the presence of aggression is implemented. The developed software implementation of the proposed method can be used as a separate module for natural language processing systems or social networks to automate text data filtering.

Approbation. The basic points and outcomes of the research have been presented and discussed at the 16th scientific conference for students and postgraduates «Applied mathematics and computing» 2024 (November 20-22, 2024, Kyiv).

An extended description of the proposed method will be discussed at the international scientific conference "The 8th International Conference on Computer Science, Engineering and Education Applications (ICCSEEA2025)" (June 7-8, 2025).

Structure and content of the thesis. The master's thesis consists of an introduction, four chapters, conclusions and appendices.

The introduction provides a general description of the work, an assessment of the current state of the problem is made, the relevance of the research direction is substantiated, the goal and objectives of the study are formulated.

The first chapter describes the problem of classifying natural language text data by the presence of aggression. The main signs of aggression in the text are considered. The main methods of sentiment analysis of the text are analyzed, their advantages and disadvantages are given.

The second chapter considers the basic method of automatic classification of natural language text data in the presence of aggression based on the ALBERT model. The existing problems of the basic method for determining aggression in the text are analyzed. The proposed method of pre-processing input text data is described. The structural, lexical and morphological features of text data are considered, as well as the method of taking into account the influence of emoji on the emotional coloring of the text. The proposed method of automatic classification of natural language text data in the presence of aggression is presented. The choice of metrics for assessing the effectiveness of the proposed method is justified.

The third section justifies the choice of tools, libraries and technologies used in the software implementation of the proposed method. The software architecture and the features of the implementation of the main stages of text processing and classification by the presence of aggression are described. Examples of the results of the program are given.

In the fourth section, an analysis was carried out of the effectiveness of the fragmented software implementation and the alignment of the established method with the basic and similar ones that were divided in the process of follow-up. The results have been grounded. Direct further work is indicated within the framework of this investigation.

The conclusions analyze the obtained results.

The appendices provide graphic materials for the explanatory note, screenshots of the presentation and listings of the code of the developed software.

The work is completed on 98 sheets, contains 3 appendices and links to a list of used literary sources from 59 titles. The work contains 16 figures, 12 listings.

Key words: software engineering, text classification, functional style, vectorization, machine learning, contextual analysis, statistical text parameters.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ 4
ВСТУП 7
1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ 9
1.1. Поняття і ознаки агресії в текстовій комунікації 9
1.2. Аналіз методів автоматизованої класифікації тексту за наявністю агресії 12
1.3. Огляд існуючих систем автоматизованої класифікації тексту за наявністю агресії 28
1.4. Висновки до першого розділу 31
2. РОЗРОБЛЕННЯ МЕТОДУ АВТОМАТИЧНОЇ КЛАСИФІКАЦІЇ ПРИРОДНОМОВНИХ ТЕКСТОВИХ ДАНИХ ЗА НАЯВНІСТЮ АГРЕСІЇ 33
2.1. Базовий метод автоматичної класифікації природномовних текстових даних за наявністю агресії 33
2.2. Проблеми автоматичної класифікації природномовних текстових даних за наявністю агресії та способи їх рішень 35
2.3. Опис запропонованого методу автоматичної класифікації природномовних текстових даних за наявністю агресії 43
2.4. Обґрунтування вибору метрик для оцінки ефективності запропонованого методу 50
2.5. Висновки до другого розділу 53
3. ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАПРОПОНОВАНОГО МЕТОДУ 55
3.1. Обґрунтування вибору засобів реалізації ПЗ 55
3.2. Архітектура розробленого ПЗ 65
3.3. Особливості реалізації ПЗ 68
3.4. Приклади результатів роботи ПЗ 72
3.5. Висновки до третього розділу 74
4. АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ЗАПРОНОВАНОГО МЕТОДУ 75
4.1. Підготовка до проведення оцінювання роботи методів 76
4.2. Оцінювання ефективності роботи методу 78
4.3. Результати аналізу метрик 85
4.4. Напрямки для подальшої роботи 86

4.5. Висновки до четвертого розділу 87
ВИСНОВКИ 89
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ 91
ДОДАТКИ 98

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Природномовні текстові дані – тексти, написані мовами, що використовуються людьми для комунікації, такі як українська, англійська чи китайська. Вони можуть містити граматичні структури, синоніми, омоніми та інші особливості природної мови.

Класифікація текстових даних – процес присвоєння текстам певних категорій або міток на основі їх змісту. Використовується в задачах аналізу тональності, виявлення спаму, визначення стилю тощо.

Емоційна полярність – властивість тексту, яка відображає його емоційне забарвлення – позитивне, негативне або нейтральне. Визначається за допомогою лексичних аналізаторів або машинного навчання.

Машинне навчання – галузь штучного інтелекту, яка створює алгоритми, здатні навчатися на основі даних та приймати рішення без явного програмування. Включає в себе методи регресії, класифікації, кластеризації тощо.

Ансамблевий підхід – підхід до розв’язання задачі за допомогою поєднання декількох різних технік, їх сильних сторін.

Вектор – математичний об'єкт, представлений списком чисел, що описують властивості певного елемента, наприклад, тексту. У NLP слова часто перетворюються у вектори для подальшого аналізу.

TF-IDF – метод оцінки важливості слова в тексті на основі його частоти появи у документі та всьому корпусі. Використовується для пошуку інформації, кластеризації та класифікації текстів.

Word2Vec – метод перетворення слів у вектори, що відображає їх семантичну близькість. Базується на нейронних мережах та вчить моделі зв’язків між словами у контексті.

Glove – алгоритм побудови векторних представлень слів, який враховує їх співвустрічальність у текстах. Відмінність від Word2Vec у тому, що він працює зі статистикою всього корпусу текстів.

API – інтерфейс для взаємодії програм між собою, що визначає спосіб виклику функцій та обміну даними. Дозволяє інтегрувати зовнішні сервіси у власні додатки.

N-грама – послідовність із *n* слів або символів, яка використовується для моделювання тексту та аналізу його структури. Часто застосовується у мовних моделях та предиктивному введенні.

Механізм уваги – техніка в нейромережах, яка дозволяє фокусуватися на найважливіших частинах вхідного тексту. Використовується в машинному перекладі, резюмуванні та генерації текстів.

Рекурентна нейронна мережа – тип нейромережі, що обробляє послідовні дані, зберігаючи інформацію про попередні стани. Використовується для оброблення текстів, мовлення та часових рядів.

Dropout – техніка регуляризації в нейромережах, яка випадково вимикає певні нейрони під час навчання. Допомагає запобігти перенавчанню та покращує узагальнюючу здатність моделі.

Гіперпараметри – налаштування алгоритму машинного навчання, які визначають його поведінку, наприклад, швидкість навчання або кількість нейронів. Вони налаштовуються перед тренуванням та впливають на якість моделі.

Ембедінг – векторне представлення об'єктів, таких як слова або зображення, у багатовимірному просторі. Використовується для збереження семантичних зв’язків між об’єктами.

Енкодер – компонент нейромережевої архітектури, що перетворює вхідні дані у векторне представлення. Використовується у трансформерах, машинному перекладі та автокодерах.

Декодер – частина нейромережевої архітектури, яка генерує вихідні дані на основі векторного представлення, створеного енкодером. Застосовується в задачах перекладу, генерації тексту та реконструкції даних.

Softmax – функція активації, що перетворює набір чисел у ймовірності, які сумуються до 1. Використовується в задачах класифікації для вибору найбільш ймовірного класу.

Стемінг – метод нормалізації слів, який скорочує їх до основної форми шляхом відсікання закінчень. Використовується в пошукових системах та NLP для зменшення розміру словника.

Емодзі – графічні символи, які передають емоції, об'єкти або дії в цифровій комунікації. Часто використовуються в соціальних мережах, месенджерах та текстових аналізах.

NLP – область штучного інтелекту, що досліджує обробку та аналіз природної мови комп’ютерами. Включає в себе машинний переклад, розпізнавання мови, аналіз тональності тощо.

ONNX – формат для обміну моделями машинного навчання між різними фреймворками. Дозволяє експортувати моделі з PyTorch, TensorFlow та інших бібліотек для оптимізованого виконання.

Ендпоінт – конкретна URL-адреса в API, яка відповідає за обробку певного запиту. Використовується для взаємодії між клієнтом і сервером у вебсервісах.

ВСТУП

Постійний розвиток інтернет-технологій разом із зростанням кількості їх користувачів активно призводить до збільшення об'єму генерації даних внаслідок життєдіяльності. Помітний вклад у дану тенденцію привносять соціальні мережі, що поступово стають невід'ємною частою життя людства. Так, вони надають користувачам змогу вільно висловлювати свої думки майже у будь-який доступний спосіб, іноді гарантуючи певний рівень анонімності. Але така свобода самовираження призвела до розповсюдження використання агресивних висловлювань, що в результаті може мати серйозні наслідки для певних шарів суспільства.

На сьогодні виявлення та модерація такого роду даних є однією з найважливіших завдань для захисту онлайн-простору та підтримки загальноприйнятого рівня культури. Умовно реалізацію подібних рішень можна розділити на дві категорії: мануальне, тобто використання людських ресурсів та автоматичне, тобто використання спеціалізованого програмного забезпечення. Очевидно, що для тих інтернет-ресурсів, якими щоденно користується велика кількість користувачів, актуальність другого підходу є набагато ефективнішою, адже персонал не здатен стабільно, якісно та швидко оброблювати великі об'єми даних. Тож постає необхідність у розробці такого програмного продукту, який міг би автоматично з високою точністю класифікувати текстові дані за наявністю ознак агресії у них.

Реалізація подібних продуктів ускладнюється тим фактом, що поняття агресивного висловлювання не є чітко визначеним. Воно може включати в себе образи, ворожнечу, кібербулінг, лайку, тощо. Саме тому кожен онлайн-сервіс самостійно для себе визначає даний термін. З цього випливає велика різноманітність підходів до створення програмних рішень [1].

З огляду на те, що генеровані людьми текстові дані є природномовними, тобто не завжди слідують певним правилам і шаблонам, можуть бути випадково або спеціально пошкодженими та зберігати у собі

додаткові ознаки, то складність розробки значно збільшується. Так, більшість поточних реалізацій, що використовуються для підтримки політик регулювання взаємодій між користувачами ігнорують багатогранні атрибути мови і агресивну поведінку, що призводить до неточних категоризацій та узагальнень.

Актуальність даного питання стало впливає на зріст кількості розроблювальних програмних рішень. Їх підходи базуються на різноманітних методах: від статистичного аналізу на основі словників до більш складного та універсального за допомогою штучних нейронних мереж. Усі вони застосовуються у різних випадках, потребують особливого налаштування та підтримки, мають як переваги у використанні, так і недоліки.

В рамках даної магістерської дисертації було проведено дослідження для створення програмного застосунку, що реалізує модифікований метод автоматичної класифікації природномовних текстових даних за наявністю ознак агресії, який базується на поєднанні контекстних залежностей слів з оцінкою тексту за інтенсивністю наявного емоційного забарвлення та додаткових значущих статистичних параметрів для підвищення точності роботи.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1. Поняття і ознаки агресії в текстовій комунікації

Агресія є широким психологічним поняттям, що означає форму ворожої або негативно-активної взаємодії, спрямованої на досягнення певної мети [2]. Агресія може виникати через стрес, конфлікт інтересів, неврівноваженість та проявлятися у фізичній, вербальній чи пасивній формах. Якщо особа виражає агресію за допомогою слів або образ, то така форма називається вербальною.

Даний феномен цікавий для дослідників з точки зору філософії, психології, медицини, біології, політології, соціології, філософії права, нейронауки [3]. Незважаючи на таку кількість галузей, в рамках яких фахівці намагаються характеризувати поняття «агресії» і навіть окреслюють для неї певні межі, вона не є повноцінно визначеною. Таким чином, кожен, хто використовує даний термін, повинен самостійно формулювати його опис, відповідно до поставлених задач.

Розвиток інформаційних технологій надає суспільству можливість збільшувати активність соціальної взаємодії [4]. Це спричинює зростання кількості чинників для прояву психоемоційних реакцій. Тому внаслідок інтенсивнішої соціальної взаємодії пропорційно утворюється більше умов для прояву агресії її суб’єктами.

На даний момент соціальні мережі є одним з основних засобів сталої комунікації між людьми. Не дивлячись на їх постійний розвиток з точки зору забезпечення комфорту, вони не здатні повноцінно контролювати безпеку спілкування. Такі особливості, як асинхронність, невидимість, соліпсизм, дисоціативна уява, є постійними супутниками людей в процесі електронної комунікації. Очевидно, що вони створюють повноцінне підґрунтя для прояву агресії в мережі Інтернет. Але основним фактором такої негативної поведінки є анонімність, що забезпечується соціальними мережами в тій чи іншій мірі. Вона сприяє прояву більш вільної поведінки

користувачів, що сподіваються на відсутність притягення їх до відповідальності.

Вплив агресії на людину може призвести до серйозних психологічних наслідків, як поява депресії, втрата віри у себе, тривожність. Саме тому сервіси, що забезпечують онлайн-комунікацію, зацікавлені у обмеженні негативних проявів поведінки своїх користувачів. Для цього застосовуються різноманітні інструменти, зокрема ті, що виявляють прояви агресії. З огляду на невизначеність даного поняття, обмеженість у ресурсах та варіативність політик, існує велика кількість різних підходів до вирішення проблеми. Але абсолютна більшість останніх використовує рішення для автоматизованої класифікації даних за наявністю агресії.

З іншого боку, користувачі вже звикли до певного рівня контролю за процесом їх спілкування. Тому більшість людей намагається приховувати справжній сенс своїх повідомлень за допомогою різноманітних практик. Найбільш розповсюдженим способом уникнення контролю є навмисне маскування, пошкодження створеного тексту [5]. Так, використовуються заміна літер на схожі символи, навмисні орфографічні помилки або повноцінне маскування певних слів.

Також слід відзначити, що мова спілкування носить абсолютно неформальний характер. Тобто вона не є обмеженою певними рамками. Тому текстова комунікація може включати у себе такі емоційні прояви, як, наприклад, емодзі.

Таким чином, створення повноцінного та точного інструмента для автоматизованої класифікації природномовних текстових даних за наявністю агресії є нетривіальним завданням та потребує врахування великої кількості параметрів, що можуть характеризувати ступінь її прояву.

Завдання класифікації природномовних текстових даних за наявністю агресії зводиться більшою мірою до проведення сентимент аналізу. Так, необхідно розглядати текст під різними кутами для об’єктивної оцінки ступеня інтенсивності певного настрою.

Умовно, ознаки, що певною мірою можуть характеризувати ступінь агресії, можна розділити на такі: структурні, морфологічні, лексичні та статистичні. Нижче проаналізуємо їх більш детально.

В рамках аналізу структурних ознак розглядається будова речень: їх довжина, використання розділових знаків, послідовності символів, емодзі та часткова або повна капіталізація тексту. Так, наприклад, короткі речення можуть бути індикатором емоційного висловлювання, а написання слів у верхньому регістрі, як правило, означає роздратування або агресію. В соціальних мережах користуються популярністю емодзі. Вони допомагають надати тексту більш очевидної та інтенсивної емоції. Тому виявлення та аналіз подібних структурних елементів може значною мірою вказувати на ступінь наявності агресії в текстових даних.

Морфологічні ознаки, як правило, виконують допоміжну роль в процесі аналізу природномовних текстових даних і вказують на розподіл частин мови. Наприклад, збільшення відносної кількості прикметників і дієслів може означати наявність емоційного забарвлення тексту [6].

Водночас, аналіз використаної лексики значно спрощує класифікацію тексту. Відомо, що лайливі, образливі або загалом негативно-забарвлені слова є основою для прояву агресії. З іншого боку, вживання емоційно-нейтральної лексики зазвичай навпаки свідчить про формальний стиль тексту та дотримання певних мовних норм.

Статистичний аналіз є одним із основних способів оброблення природномовних текстових даних. Так, врахування частоти вживання певних слів у тексті може допомогти виділити агресивні слова або фрази. На основі статистичних ознак формуються контекстні залежності слів та речень. Це дозволяє передбачувати, які слова або фрази можуть слідувати за даним або передувати йому. Врахування контексту є потужним інструментом, як загалом в задачах оброблення природної мови, так і при класифікації тексту за наявністю агресії, зокрема.

Залежно від обраного методу класифікації, певні ознаки можуть мати різну значущість в процесі аналізу даних. Більшість підходів до вирішення задач класифікації текстів за їх емоційним забарвленням вимагають ретельного та обґрунтованого вибору параметрів. Адже деякі з них мають вирішальне значення для класифікації, а інші, навпаки, створюють зайвий шум та знижають ефективність роботи програми. Тому вибір тих чи інших ознак текстових даних є вирішальним для точності роботи їх класифікатора за наявністю агресії.

1.2. Аналіз методів автоматизованої класифікації тексту за наявністю агресії

Існуючі методи для реалізації автоматичної класифікації тексту за наявністю агресії можна розділити на 2 категорії:

- на основі словників;
- на основі машинного навчання [7].

Дані підходи здатні забезпечити високу точність роботи класифікатора. Але при цьому вони застосовуються при різних умовах, адже обидва несуть в собі як переваги, так і недоліки. Намагаючись використати сильні сторони методів, дослідники поєднують їх, формуючи ще одну категорію – ансамблеву.

Спочатку розглянемо детально методи на основі словників. Він використовує ключові слова і є базовим в процесі класифікації природномовних текстових даних. Сенс даного методу полягає у формуванні певного за розміром словника, що вміщує потенційно агресивні, образливі та лайливі слова. Використовуючи такий список, обчислюються статистичні дані, наприклад, кількість подібних входжень у текст [8]. Очевидно, що великою перевагою даного методу є його швидкість. До того ж, програмні реалізації систем на основі даного методу є простими для розуміння та підтримки. У той самий час, їх точність напряму залежить від обсягу та релевантності використаного словника. З огляду на те, що терміни

з часом змінюються, підтримка його актуальності може становити серйозну задачу. Подібні системи не можуть враховувати факт того, що в природномовних текстових даних слова-маркери не завжди визначають загальну образливість.

Dennis Njagi та Damien Hanyurwimfura запропонували метод класифікації природномовних текстових даних за наявністю агресії на основі створення та використання спеціалізованого лексикону [9]. Їх робота полягає в аналізі суб'єктивної частини тексту, адже саме вона вміщає емоційно забарвлені вислови, що можуть означати агресію. Для цього використовується лексикон суб'єктивності SUBJCLUE, що вміщає більше 8000 слів із вказаними значеннями їх полярності і ступеня об'єктивності. Речення, що вміщають у собі більше двох слів із суб'єктивним забарвленням, ідентифікуються, як суб'єктивні. Необхідно зазначити, що суб'єктивність тексту оцінюється не тільки по окремим словам, а й з урахуванням їх семантичної ролі у реченні.

Наступний етап методу передбачає розробку словника «ненависті». Даний процес включає вилучення слів та виразів, що характеризуються агресивним настроєм. Лексикон вміщає такі компоненти, як слова з негативною полярністю, агресивні дієслова та тематичні граматичні патерни. Для збагачення лексикону використовуються такі ресурси, як WordNet [10].

В результаті сформований лексикон використовується для класифікації текстів. В рамках даного процесу застосовуються правила визначення частоти та інтенсивності появи подібних слів та патернів у реченні. Якщо зустрічається одне слово разом з граматичним патерном, то речення може бути класифіковане як агресивне. Таким чином враховуються лексичні та семантичні аспекти природномовних текстових даних.

Методи на основі машинного навчання є більш сучасними. Їх популярність позитивно вплинула на кількість досліджень, в результаті чого на даний момент існує багато різноманітних підходів до створення

автоматизованих класифікаторів природномовних текстових даних за наявністю агресії. Загальний підхід до реалізації таких методів полягає у обробленні набору даних таким чином, щоб кожен запис був представлений спеціальним набором ознак, тобто *n*-розмірним вектором, що використовується для навчання нейронної мережі. Причому такі набори повинні бути попередньо розмічені та вміщати достатню кількість записів. Обсяг набору даних залежить від потреб та обраної архітектури нейронної мережі.

Як було зазначено вище, методи на основі машинного навчання оперують даними у вигляді векторів, тобто наборів ознак. Для їх формування застосовують різноманітні підходи, що залежать від типу поставлених задач. Ступінь навчання нейронної мережі напряму залежить від якості поданого вектора, тобто від кількості релевантних нормалізованих параметрів у ньому. Тому в процесі його створення необхідно бути впевненим, що всі вагомі ознаки були включені, а ті, що створюють зайвий шум, відкинуті.

Найбільш розповсюдженим підходом до вилучення ознак є використання статистичної міри TF-IDF. За допомогою неї оцінюють ступінь вагомості слів у контексті документа, що є частиною корпусу [11].

Ще одним традиційним підходом є використання таких алгоритмів, як Word2Vec [12] або Glove [13], що дозволяють векторизувати текст з урахуванням семантичних зв’язків слів.

Традиційно для аналізу агресивної мови і емоцій використовуються методи навчання з вчителем, зокрема наївний баєсів класифікатор, дерева рішень, метод опорних векторів, лінійна регресія, логістична регресія.

Яскравим прикладом даного підходу є метод автора Ingmar Weber. Його дослідження було спрямоване на розроблення системи автоматичного виявлення мови ненависті та образливості на основі аналізу твітів [14]. Для побудови моделі було використано лексикон мови ненависті, що був сформований за допомогою платформи Hatebase.org. За його допомогою було сформовано корпус текстів шляхом пошуку по ключовим словам,

використовуючи Twitter API. В результаті чого було створено корпус із 85,4 мільйонів твітів. Для подального навчання нейронної мережі було випадковим чином обрано 25 тисяч унікальних записів, які були мануально проанотовані працівниками CrowdFlower.

В рамках оброблення тексту слова твітів були переведені в нижній регістр та перетворені до початкової форми за допомогою алгоритму Портера [15]. Для формування вектору ознак було сформовано уніграми, біграми та триграми, що були зважені за допомогою TF-IDF. Додатково було визначено синтаксичні характеристики за допомогою тегування частин мови, використовуючи бібліотеку NLTK. Також використовувалися модифіковані індекси легкості читання Флеша-Кінкейда, показники сентименту та числові індикатори для хештегів, згадок, ретвітів, кількості символів, слів та складів [16].

В результаті векторизації було отримано великий за розмірами масив чисел. Тому автор використав механізм зменшення розмірності за допомогою логістичної регресії з регуляризацією L1. Надалі тренувалися різні моделі: баєсів класифікатор, дерева рішень, випадкові ліси, метод опорних векторів та логістична регресія. Моделі оцінювали за допомогою 5-кратної перехресної перевірки, що запобігала перенавчанню.

Логістична регресія з регуляризацією L2 була обрана через її здатність прогнозувати ймовірності приналежності до класів. Модель показала 90% точності класифікації тексту на наявність мови ненависті, але всього 40% для образливої мови. Модель добре спрацювала з поширеними типами мови ненависті, але допускала помилки при класифікації расистських чи гомофобних образ.

В рамках машинного навчання також застосовують ансамблеві підходи. За рахунок використання сильних сторін декількох нейронних мереж з різними архітектурами долаються недоліки окремих моделей. Такі комбінації дозволяють значно зменшити дисперсію та збільшити навчальну

здатність. Одним з найбільш популярних ансамблевих методів є випадковий ліс.

Girma Yohannis Bade та інші запропонували використовувати модель випадкового лісу для виявлення агресії в природномовних текстових даних [17]. Для проведення дослідження було використано публічний набір даних DravidianLangTech.

Попереднє оброблення даних включало заповнення пропусків у завантаженому csv файлі, щоб забезпечити цілісність інформації. Оброблення аномалій передбачало масштабування даних для зменшення впливу на навчання моделі. У якості метода векторизації було використано TF-IDF.

Для класифікації природномовних текстових даних за наявністю агресії автори обрали алгоритм випадкового лісу. Він поєднує кілька дерев рішень для підвищення точності прогнозів і зменшення перенавчання. Кожне дерево тренується на випадкових підмножинах даних і ознак, що зменшує кореляцію між деревами та підвищує стабільність результатів. В результаті навчання моделі випадкового лісу було досягнуто F1-міру на рівні 0,492.

Методи глибокого навчання, на відміну від попередніх, не потребують формування векторів ознак. В процесі навчання вони здатні самостійно виділяти інформацію, що призводить до збільшення точності їх роботи. Покращені результати класифікації обумовлені більш глибоким аналізом вхідних даних у прихованих шарах нейронної мережі.

Існує декілька типів моделей глибокого навчання:

- рекурентні нейронні мережі;
- згорткові нейронні мережі;
- на основі трансформерів.

Рекурентні нейронні мережі застосовуються для роботи з послідовностями текстових даних, зокрема для аналізу настрою і виявлення агресії. Найбільш популярними архітектурами рекурентних нейронних

мереж є RNN та його похідна LSTM. Обидві враховують порядок слів і їх контекст. LSTM застосовує механізм уваги, що дозволяє зосереджуватися на найбільш вагомих частинах тексту.

Хоча згорткові нейронні мережі були першочергово створені для оброблення зображень, згодом вони були адаптовані до вирішення NLP задач. Для виявлення ознак агресії CNN застосовує фільтри до послідовностей, фіксуючи локальні шаблони та особливості.

Puneet Mathur, Ramit Sawhney, Meghna Ayyar, Rajiv Ratn Shah в рамках конференції ALW2 презентували ансамблевий підхід на основі використання глибокого навчання для класифікації тексту на суміші англійської та індійської мови за наявністю агресії [18].

Для навчання використовувався набір даних Hinglish Offensive Tweet, що складався з твітів, зібраних через Twitter API. При пошуку даних було застосовано геолокаційні обмеження, щоб забезпечити, що вони походять з Індійського субконтиненту. Було застосовано фільтрацію, щоб відкинути тексти, що включають менше, ніж 3 слова на хінгліш. В результаті залишилось 3189 твітів, що були проанотовані як образливі, необразливі та ненависницькі.

Процес попереднього оброблення даних включав такі етапи, як видалення небажаних символів та структурних елементів, видалення стоп-слів, транслітерацію та переклад. Формування векторного представлення забезпечувалося такими інструментами, як Glove для врахування глобального контексту слів, Word2vec для використання локальних зв’язків слів та FastText. Набір даних був розмежований на тренувальну та валідаційну вибірки у співвідношенні 4:1. Додатково було обчислено оцінку сентиментальності речень за допомогою SentiWordNet та кількість образливих слів.

Модель, що була використана для навчання, складається з CNN та LSTM. На вхід їм подавався об’єднаний вектор ознак. CNN вміщав 3 послідовні згорткові шари із розмірами 20, 15, 10, шар dropout для

запобігання перенавчанню та підсумковий лінійний шар з трьома виходами. LSTM складався з одного шару розміром в 128 значень, що враховував довгострокові залежності у тексті, а також лінійний шар. Результати роботи обох моделей поєднувалися та подавалися у лінійний шар класифікатора.

В результаті навчання та оцінювання запропонована модель показала рівень F1-міри – 0,895, що більше, ніж при використанні машинного навчання на базі методу опорних векторів та випадкового лісу.

Трансформ-подібні моделі значно вплинули на розвиток оброблення природномовних текстових даних. Їх ефективність обумовлена здатністю враховувати довгострокові залежності між словами та паралельно оброблювати текст. Таке паралельне оброблення робить їх масштабованими та точними, перевершуючи RNN та CNN. Базова концепція моделей на основі архітектури трансформера полягає в механізмі уваги. Він дозволяє моделі зосереджуватися на відповідних частинах вхідного тексту під час прогнозування та враховувати контекстуальні зв’язки між кожним словом і всіма іншими в реченні. Використання таких моделей, як KET, GPT, BERT та його похідних забезпечили високу точність у задачах класифікації природномовних текстових даних за наявністю агресії.

З іншої сторони трансформ-подібні моделі мають низку недоліків. Для їх ефективного навчання необхідно мати великий обсяг навчальних даних. Дуже важливим фактором є налаштування гіперпараметрів. До того ж існують потреби у великому обсягу обчислювальних ресурсів, що ускладнює процес навчання.

Peixiang Zhong, Di Wang1 та Chunyan Miao в рамках вирішення задачі класифікації природномовних текстових даних за такими емоціями, як радість, сум, гнів, відраза, страх та здивування запропонували використання власної архітектури трансформеру – Knowledge enriched transformer [19]. Модифікація полягала у використанні зовнішніх баз знань. В процесі навчання моделі ці дані використовувалися для врахування ознак семантичного контексту. Джерелами для формування словників слугували

ConceptNet [20] та NRC_VAD [21], що надавали доступ до мультимовного семантичного графу із загальними поняттями, описаними природною мовою, та до англійського лексикону із вказаними значеннями емоційної інтенсивності.

Першим шаром даної моделі є утворення векторного представлення слів за допомогою поєднання ембедінгу та його позиції відносно інших слів (1):

$$t = Embed(t) + Pos(t). \quad (1)$$

Для врахування зовнішніх баз знань використовується механізм динамічної контекстно-залежної афективної графової уваги. В рамках даного процесу увага акцентується на поняттях залежно від їх значущості для поточного контексту. Для кожного токену t із тексту X_j^i KET шукає пов'язані поняття в базі знань ConceptNet, створюючи граф концепцій $g(t)$. Граф включає суміжні поняття для кожного слова. Щоб зменшити зайвий шум, концепції, які відносяться до множини стоп-слов чи відсутні у словнику моделі, видаляються. Після цього концептуальна репрезентація для токена t обчислюється як лінійна комбінація його концепцій (2):

$$c(t) = \sum_{k=0}^{|g(t)|} a_k c_k, \quad (2)$$

де c_k – ембедінг k концепції, a_k – вага уваги. Для обчислення ваги a_k застосовується функція softmax.

Релевантність концепції rel_k, тобто показник, наскільки концепція c_k пов'язана із загальним контекстом визначається за формулою (3):

$$rel_k = \min(s_k) - \max(s_k) abs(\cos(CR(X^i), c_k)), \quad (3)$$

де s_k – довірча оцінка концепції c_k, $CR(X^i)$ – середнє представлення контексту висловлювання.

Емоційна насиченість aff_k оцінює інтенсивність емоцій концепції c_k на основі її оцінок $V(c_k)$ та $A(c_k)$ (4).

$$aff_k = \min(V(c_k) - 0.5, A(c_k)/2) - \max(V(c_k) - 0.5, A(c_k)/2). \quad (4)$$

Комбінування релевантності та емоційної насиченості утворює значення ваги (5):

$$w_k = rel_k + aff_k. \quad (5)$$

Після цього кожне слово може бути представлене за допомогою концептуально збагаченого вектора (6):

$$t = W[t, c(t)], \quad (6)$$

де W – матриця параметрів, а $[t, c(t)]$ – конкатенація векторів слова та його концептуального представлення.

Наступним шаром є механізм ієрархічної самоуваги, що використовується для глибокого розуміння структури висловлювання. Його використання дозволяє враховувати як локальні зв’язки між словами в межах одного висловлювання, так і глобальні зв’язки між різними реченнями в діалозі.

На початку роботи моделі кожне висловлювання перетворюється у матрицю ембедінгів, яка проходить через шар самоуваги для визначення зв’язків між токенами всередині тексту. Це дозволяє моделі фокусуватися на різних властивостях тексту, комбінуючи інформацію з усіх токенів. Після оброблення кожного висловлювання формується контекстне їх представлення шляхом об'єднання попередніх висловлювань.

Для навчання моделі даної архітектури було використано набори даних DailyDialog [22], MELD [23], EmoryNLP [24] та IEMOCAP [25]. Для оцінки точності роботи моделі використовувалася міра F1. Результати дослідження показали, що KET визначала одну з 6 емоцій в природномовних текстових даних на 2-7% точніше, ніж аналогічні методи.

На даний момент найбільш популярною моделлю для вирішення задач NLP є BERT та його похідні. Bidirectional Encoder Representations from Transformers – результат дослідження науковців з Google AI Language. Його основною перевагою є здатність враховувати контекст з обох напрямків під

час оброблення тексту [26]. Це контрастує з попередніми спробами створити механізм уваги, що розглядали текстову послідовність або зліва направо, або комбіноване навчання зліва направо та справа наліво. Результати статті [27] демонструють, що дана модель може мати більш глибоке розуміння контексту, ніж однонапрямні аналоги.

BERT використовує механізм уваги трансформер, який вивчає контекстні зв'язки між словами в тексті. Базова форма трансформера включає 2 окремі механізми – енкодер, що читає текстовий вхід та декодер, що видає прогноз для задачі.

На відміну від направлених моделей, які враховують текстовий вхід послідовно, енкодер трансформера зчитує всю послідовність слів одразу. Така характеристика дозволяє BERT вивчати контекст слова на основі всього його оточення. Вхідні дані представляють собою послідовність токенів, що перетворюються у ембедінги, а далі оброблюються нейромережою. Вихідні дані представляють собою послідовність векторів розміром *N*, у яких кожен вектор відповідає вхідному токену з тим же індексом.

При навчанні мовних моделей виникає проблема визначення цілі прогнозування. Деякі передбачають наступне слово в послідовності. Такий підхід обмежує контекстне навчання. Щоб здолати цю проблему, BERT використовує 2 стратегії навчання: маскування та передбачення наступного речення.

Перед поданням послідовності слів у BERT 15% від загальної кількості у кожній послідовності замінюється токеном [MASK]. Далі модель намагається передбачити значення замаскованих слів на основі контексту, що надають інші. З технічної точки зору для цього необхідно:

- додати класифікаційний шар над вихідними даними енкодера;
- множення вихідних векторів на матрицю ембедінгів та перетворення їх вимірності відповідно до характеристики словника;
- обчислення вірогідності слова в словнику за допомогою softmax.

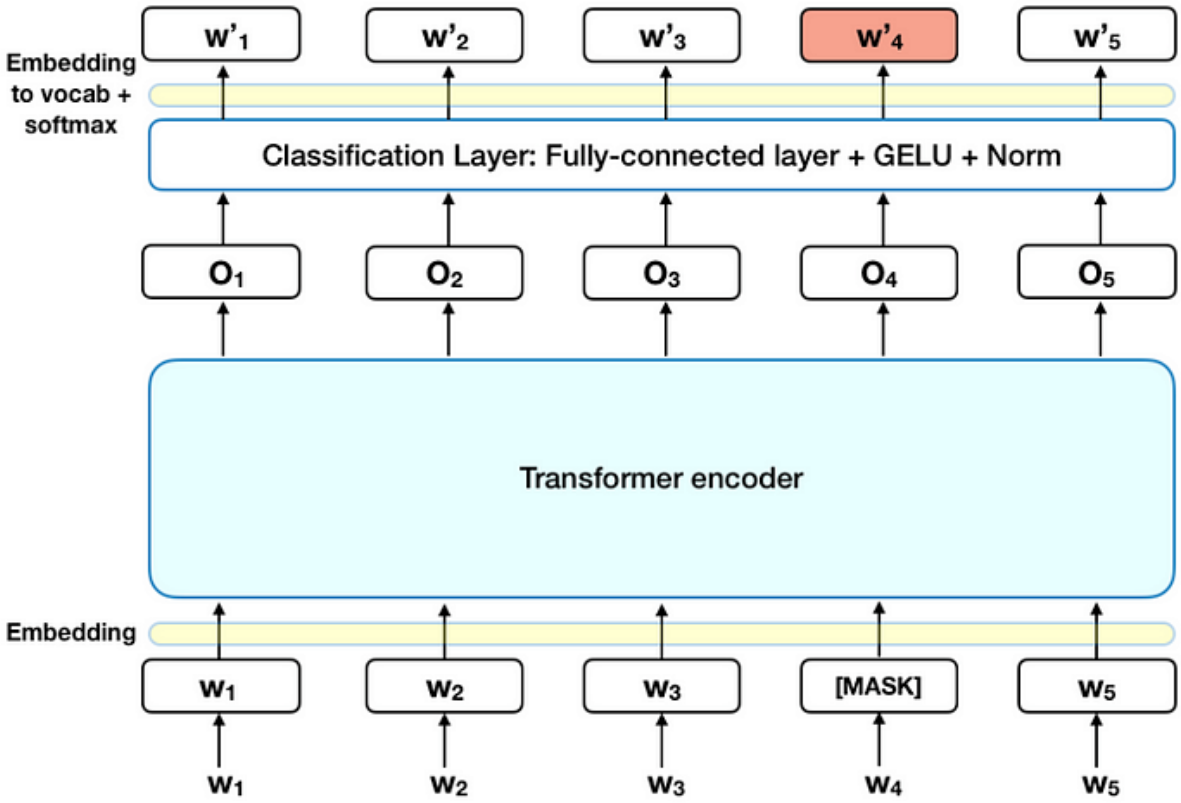


Рис. 1. Схема архітектури моделі BERT

Функція втрат бере до уваги лише передбачення замаскованих слів і не враховує прогнозування інших слів у послідовності. В результаті модель покращує свої результати повільніше, ніж напрямні моделі, що компенсується її підвищеною обізнаністю про контекст.

В процесі навчання BERT отримує пари речень в якості вхідних даних і навчається прогнозувати, чи є друге речення в парі з першим у вхідному документі. Одна половина вхідних даних представляють собою пару, у якій друге речення є наступним у відповідності до оригінального тексту. В той же час для інших 50% воно обирається випадковим чином. Щоб допомогти моделі відрізняти 2 речення при навчанні, перед входом в модель вхідні дані оброблюються наступним чином:

- [CLS] токен вставляється в початок першого речення, а [SEP] – в кінець речення;
- ембедінг речення, що вказує на перше чи друге, додається для кожного токену;

- позиційний ембедінг додається до кожного токену для вказання його позиції в послідовності.

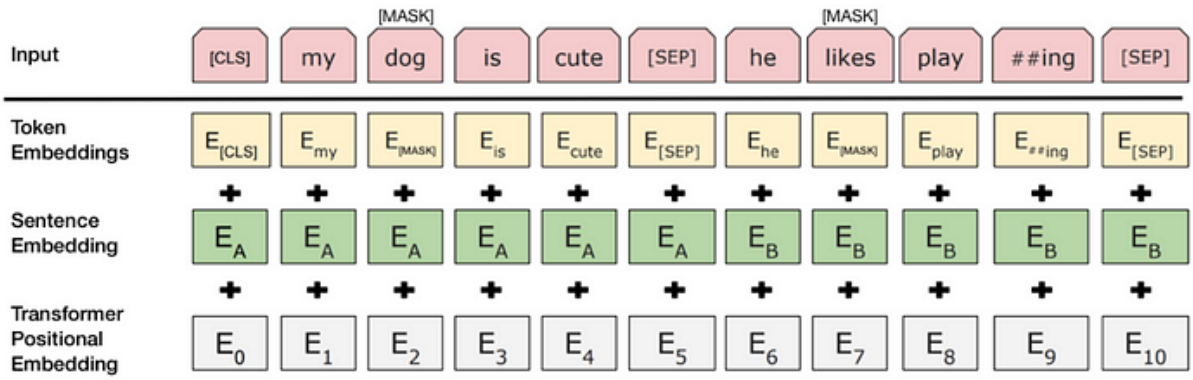


Рис. 2. Схема утворення представлення вхідного речення моделі BERT

Також BERT використовує підхід попереднього тренування, що дозволяє моделі вивчити загальні закономірності та структури конкретної мови з великого обсягу текстових даних перед тим, як підлаштовуватися під конкретне завдання. Це дозволяє досягати точних результатів навіть при невеликій кількості даних для донавчання.

Варто зазначити, що BERT часто використовується в якості метода векторизації. Так, попередньо натреновані моделі можуть оброблювати текстові послідовності та надавати ембедінги як для слів, так і для цілого тексту.

Моделі BERT можуть мати різну кількість шарів енкодерів. Кожен з них повертає вектори, насичені унікальною інформацією. Автори статті [28] вказують на можливість використання результатів роботи декількох таких шарів. Останні 4 є найбільш ефективними, тому їх варто використовувати для підвищення точності класифікації.

Існує декілька способів для врахування декількох шарів, зокрема сумування, середнє значення та конкатенація. Найбільш високе значення F1-міри демонструє підхід об’єднання векторів, що виходять з останніх 4 шарів енкодерів.

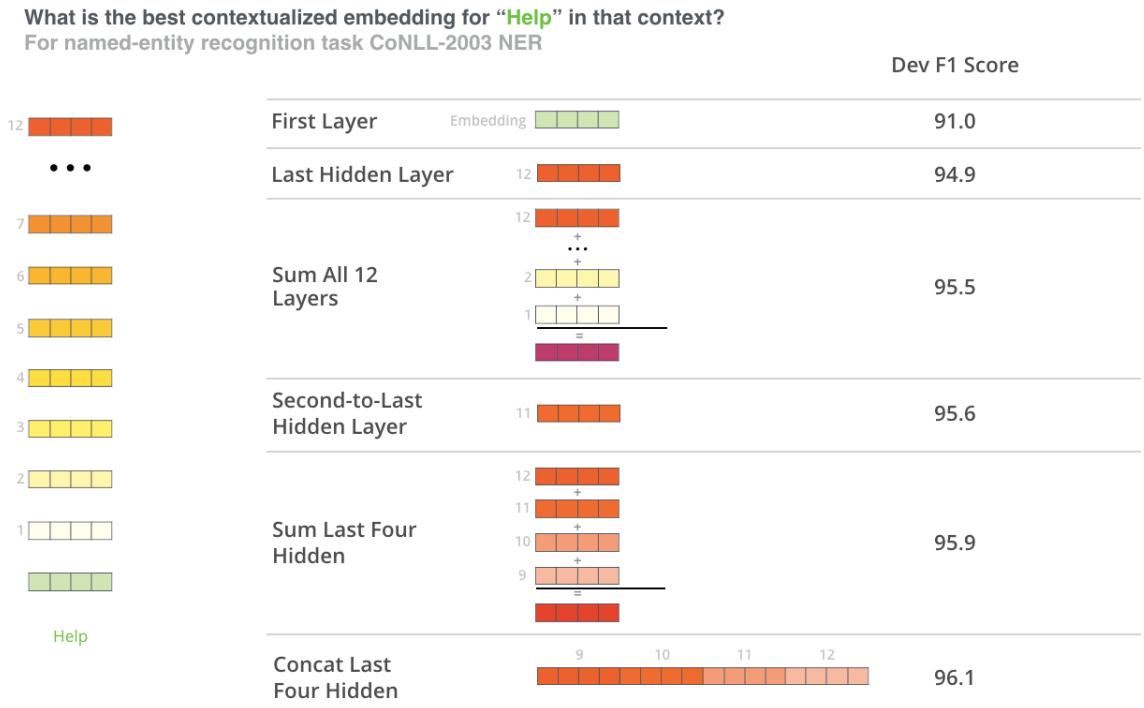


Рис. 3. Підходи до формування вихідного вектору BERT

В рамках автоматичної класифікації природномовних текстових даних BERT застосовується в більшості випадків. Дослідники фокусуються на точкових модифікаціях задля досягнення більшої точності класифікації.

Tommaso Caselli, Valerio Basile, Jelena Mitrovic, Michael Granitzer створили HateBERT [29], власну мовну модель на базі BERT для виявлення образливої поведінки в природномовних текстових даних. Метою дослідження була побудова попередньо навченої моделі, яку в перспективі можна було б використовувати для різних за специфікою задач.

У якості джерела даних було взято великий список онлайн-спільнот із соціальних мереж, які були заборонені за розміщення образливих, неправомірних та агресивних коментарів. Дослідники зберегли близько 1,5 мільйона повідомлень, що вміщали приблизно 43 мільйонів лексем.

Для створення дійсно ефективної моделі, було проведено 100 навчальних епох із розмірами батчей в 64 приклади, розміром по 512 лексем. Для боротьби з перенавчанням було використано Adam зі швидкістю в 5*e*-5.

В результаті було визначено, що HateBERT стабільно перевершує загальний BERT для різноманітних образливих мовних явищ.

Як було вказано вище, основною перепоною для навчання подібних моделей є необхідність у використанні великої кількості обчислювальних ресурсів. Наприклад, BERT-base [30], одна з двох класичних реалізацій, вміщає 110 мільйонів параметрів. В результаті чого дослідники NLP часто звертають увагу на оптимізації при зберіганні базової високої точності.

Існує велика кількість таких моделей, але найбільш популярною є ALBERT – A Lite BERT [31]. Вона з'явилась, як відповідь на проблеми, що виникли при масштабуванні BERT:

- збільшення кількості параметрів;
- висока обчислювальна складність;
- обмеження пам'яті сучасних апаратних засобів.

ALBERT вирішує дані задачі шляхом впровадження декількох рішень, серед яких факторизація параметрів ембедінгів, спільне використання параметрів між шарами та новий підхід до обчислення когерентності між реченнями.

Факторизація ембедінгів розділяє словникові ембедінги та приховані шари. У BERT ці компоненти мають однаковий розмір, що призводить до експоненційного збільшення кількості параметрів при зростанні розміру словника або прихованих шарів. ALBERT розв'язує цю проблему шляхом використання двох окремих матриць: одна відповідає за зменшення розмірностей ембедінгів, інша – за трансформацію в прихований простір. Це дозволяє суттєво скоротити обсяг пам'яті, необхідної для збереження параметрів, без втрати якості результатів.

Другим значним рішенням є спільне використання параметрів між шарами. У базовому BERT кожен шар має власні унікальні параметри, що призводить до збільшення їх загальної кількості зі зростанням глибини моделі. ALBERT використовує підхід, коли параметри уваги та зворотного зв'язку спільно використовуються всіма шарами. Це не тільки знижує їх

загальну кількість, але й забезпечує стабільність навчання. Спостереження показують, що цей підхід зменшує коливання вектора представлення на різних рівнях мережі, що сприяє кращому узагальненню.

Ще однією інновацією ALBERT є введення втрати прогнозування порядку речень – SOP. У BERT для навчання міжреченевих зв'язків використовувалась задача прогнозування наступного речення, але вона виявилась малоефективною, оскільки часто навчала модель визначати лише зміну теми тексту, а не когерентність між реченнями. SOP створює негативні приклади шляхом перестановки порядку двох сусідніх речень з одного документа, змушуючи модель навчатися розпізнавати тонкі дискурсні відмінності. Цей підхід дозволяє ALBERT демонструвати покращені результати на задачах з багатореченевим контекстом, таких як розуміння тексту та класифікація.

ALBERT має набагато менше параметрів у порівнянні з BERT. Наприклад, ALBERT-large [32] містить лише 17 мільйонів параметрів проти 336 мільйонів у BERT-large [33], але має майже ті самі або перевищує результати останньої на тестових наборах GLUE, SQuAD і RACE. Така ефективність досягається завдяки об'єднанню скорочення параметрів із використанням самоконтрольованих втрат. Варто зазначити, що при зменшенні кількості параметрів модель продемонструвала збільшену швидкість навчання.

Розповсюдженим підходом до аналізу природномовних текстових даних, зокрема класифікації за емоційним забарвленням є поєднання переваг різних за типом методів. Так, Bharat Gaind, Varun Syal та Sneha Padgalwar запропонували ансамблевий метод до визначення однієї з 6 емоцій дописів у соціальних мережах на основі використання словників та машинного навчання [34]. Автори фокусуються на дослідженні способів класифікації тексту за такими емоціями, як радість, сум, страх, здивування, агресія та огида.

Перший метод, що базується на застосуванні словників, враховує лексичні та граматичні особливості тексту, використовуючи маркери емоцій, анотації частин мови і граматичні залежності. Для цього створюється список слів разом із визначеною певною емоцією для кожного з них. Додатково їм у відповідність виставляється оцінка інтенсивності емоційного забарвлення. Для пошуку таких слів у тексті використовується їх базова форма. В результаті було сформовано словник із обсягом 1500 записів. Додатково використовується список із 50 слів-модифікаторів емоцій, що можуть впливати на оцінку інтенсивності.

Для лексичного аналізу тексту застосовується розбиття на слова, що були проанотовані по частинам мови, лемам та іменованим сутностям з використанням Stanford CoreNLP [35]. Токени порівнюються з лексемами із попередньо сформованого лексикону та додатково перевіряються, на кого направлена емоція, за допомогою аналіза граматичних суб'єктів та використання списку займенників. Якщо поряд з емоційним словом використовується заперечення, або попередньо визначений модифікатор, то оцінка його емоційної інтенсивності змінюється. Для тексту оцінка проводилася за формулою (7):

$$Score[EC_j] = \sum_{hits \in EC_j} (emotScore + perScore), \tag{7}$$

де $emotScore$ – інтенсивність емоції, а $perScore$ – вага, пов'язана з типом адресата. Для обчислення відносної оцінки певної емоції використовується формула (8):

$$\mathrm{Re}lScore[EC_j] = 100 \frac{Score[EC_j]}{\sum_{k=1}^{6} Score[EC_k]}. \tag{8}$$

Додатково до вищезгаданого підходу використовується машинний аналіз. У якості класифікатора застосовується SMO [36], похідну від SVM та дерево рішень. Для навчання використовується набір даних, що був зібраний

за допомогою Twitter API. Попереднє оброблення текстових даних включало приведення всіх символів до нижнього регістру та стемінг.

Для поєднання результатів двох підходів використовується наступна формулу оцінки емоційного забарвлення (9):

$$FinalScore[Lc] = Score[Lc] + 0.2Score[Mc], \quad (9)$$

де Lc – емоція, передбачена класифікатором, а Mc – емоція з найбільшою інтенсивністю, що було визначено за допомогою лексичного аналізу.

В результаті аналізу точності роботи даного методу було визначено, що на вибірці в 900 твітів для метрики Accuracy підхід разом із SMO мав значення 91,7%, а з деревом рішень – 85,4%.

1.3. Огляд існуючих систем автоматизованої класифікації тексту за наявністю агресії

З огляду на те, що проблема автоматичної класифікації природномовних текстових даних за наявністю агресії є актуальною та характеризується великою кількістю підходів до вирішення, існує багато готових онлайн-сервісів, що надають готові рішення для клієнтів.

Найбільш популярним та зручним інструментом є Perspective API [37]. Даний продукт був створений Jigsaw, підрозділом Google, та використовує алгоритми машинного навчання для аналізу природномовних текстів. Головна мета сервісу полягає у виявленні текстових даних, що можуть бути сприйняті як образливі, агресивні чи викликати конфлікти. Даний інструмент дозволяє легко модерувати контент у соціальних мережах та інших сервісах, що надають можливість онлайн-комунікації.

Perspective API аналізує вхідні дані за такими аспектами, як токсичність, агресивність, нахили до приниження, сексуальні домагання або інші форми образливої поведінки. Ключовим принципом його роботи є використання моделей глибокого навчання, які були натреновані на великій кількості анонімних текстів. Дані для тренування були сформовані за

допомогою використання реальних джерел: коментарів до статей, публікацій у соціальних мережах та відкритих обговореннях.

Сервіс дозволяє легко інтегрувати себе у будь-яку систему за допомогою REST API. За допомогою запиту можна передати текст та отримати оцінку за декількома параметрами. Виявлення агресії досягається за допомогою поля toxicity та threat.

Налаштування через API дає можливість адаптувати пороги чутливості залежно від контексту. Таким чином клієнт самостійно визначає, що для нього є агресія для максимального пристосування до поширених у даному випадку ситуацій.

Ще однією потужною системою є Text Moderation від Azure Cognitive Services [38]. Його задача є аналогічною до попереднього сервісу – виявлення та фільтрація небажаних або образливих висловлювань. Основною метою даного продукту є створення безпечного цифрового простору шляхом аналізу текстів за наявністю агресії, ненормативної лексики, погроз, мови ненависті та подібного контенту, що може задати шкоди користувачам під час спілкування.

Text Moderation використовує сучасні алгоритми машинного навчання, які були навчені на великій кількості текстових даних, що були відібрані з реальних випадків прояву агресивної поведінки в мережі Інтернет. Перевагою даного сервісу є підтримка оброблення декількох мов, що дозволяє використовувати його в міжнародних проєктах. Text Moderation враховує особливості граматики, сленгу, сарказму та культурних відмінностей різних мов, щоб забезпечити точніший аналіз тексту.

В рамках автоматичної класифікації природномовних текстових даних за наявністю агресії Text Moderation застосовує додаткові підходи до оброблення вхідного тексту, спрямовані на збільшення точності. Так, він виявляє непрямі форми агресії, саркастичні або пасивно-агресивні висловлювання. Також, як і його аналог, даний сервіс дозволяє задати порогові значення для адаптації до конкретних потреб клієнтів.

З гляду на те, що Text Moderation є частиною продуктів Azure, його легко інтегрувати з іншими сервісами компанії. Для зовнішніх користувачів він надає зручний REST API для модерації текстів в режимі реального часу. Існує можливість пакетного оброблення для аналізу великого об’єму даних.

MonkeyLearn – платформа для аналізу тексту, яка також використовує інструменти машинного навчання для оброблення природньої мови [39]. Легкість створення та налаштування моделі для класифікації тексту за конкретним значенням робить цей сервіс одним з найбільш популярних. Простий інтерфейс гарантує легкість роботи з ним навіть для непідготовлених користувачів.

MonkeyLearn дозволяє завантажувати власні дані та тренувати модель, підлаштовуючись під специфічні вимоги. Платформа підтримує мультикласову класифікацію, що додає гнучкості. Аналіз та виявлення агресії базується на ідентифікації мовних патернів, таких як вживання образливих слів, негативних емоційних маркерів або конструкцій, що мають на меті принижувати співрозмовника. Даний сервіс також надає REST API, що забезпечує легкість інтеграції в існуючі системи.

Аналіз існуючих систем для автоматизованої класифікації текстових даних за наявністю агресії, таких як Perspective API, Text Moderation від Azure та MonkeyLearn, показує, що ці інструменти забезпечують значну функціональність для виявлення агресивних висловлювань. Однак, незважаючи на їх сильні сторони, вони не завжди задовольняють всі потреби користувачів у точності та гнучкості.

Хоча ці сервіси дозволяють налаштовувати пороги чутливості для виявлення агресії, вони не здатні повноцінно справлятися з більш складними формами агресії, такими як культурні особливості мови. Крім того, наявність орфографічних помилок у вхідних текстових даних може негативно вплинути на результат класифікації, адже ці системи не завжди враховують всі нюанси мовних конструкцій.

1.4. Висновки до першого розділу

У даному розділі розглянуто проблему автоматичної класифікації природномовних текстових даних за наявністю агресії. Було проаналізовано та визначено агресію, як явище, її вплив на процес спілкування та на його учасників.

Дане поняття є багатогранним та визначається великою кількістю параметрів. Точність класифікації тексту напряму залежить від якості оброблення вхідних текстових даних та врахування усіх важливих їх ознак. Задача визначення агресії є окремим випадком від загального сентимент-аналізу. Саме тому для її рішення важливими є такі текстові характеристики, як структурні, морфологічні, лексичні та статистичні.

Проведено аналіз методів автоматичної класифікації тексту за наявністю агресії. Зазначено, що існує 2 загальні підходи – використання словників та методів машинного навчання. Перший підхід реалізується за допомогою лексичного та морфологічного аналізу. Він є відносно простим для створення та підтримки, швидким у роботі, але в той самий час прямо залежить від якості використовуваних словників та їх обсягів. Другий підхід використовує методи машинного навчання. Він також поділяється на класичний та глибокий. Різницею між ними є якість проведення аналізу вхідних даних, вимога до обсягу та якості навчальних корпусів текстів та кінцева точність роботи.

Розглянуто методи класифікації тексту за наявністю агресії на основі вищезазначених підходів. Моделі глибокого навчання, а саме, трансформ-подібні продемонстрували найбільшу точність. Так, BERT є найбільш популярною в рамках NLP, але при цьому потребує значної кількості обчислювальних ресурсів. Для оптимізації процесу навчання та підтримки високої точності класифікації створено модель ALBERT.

Наведено опис найбільш популярних онлайн-сервісів, що надають можливість класифікації текстових даних за наявністю агресії онлайн. Так, було описано Perspective API, Text Moderation та MonkeyLearn. Усі вони

надають інтерфейс взаємодії в рамках архітектурного стилю REST для простої інтеграції з існуючими сервісами.

Розглянуті системи мають обмеження у точності та гнучкості, зокрема при обробці складних форм агресії та культурних відмінностей. Також їх здатність справлятися з орфографічними помилками у текстах часто призводить до неточних результатів класифікації.

2. РОЗРОБЛЕННЯ МЕТОДУ АВТОМАТИЧНОЇ КЛАСИФІКАЦІЇ ПРИРОДНОМОВНИХ ТЕКСТОВИХ ДАНИХ ЗА НАЯВНІСТЮ АГРЕСІЇ

2.1. Базовий метод автоматичної класифікації природномовних текстових даних за наявністю агресії

В якості базового методу автоматичної класифікації природномовних текстових даних за наявністю агресії вирішено використати модель ALBERT саме через її здатність проводити глибокий структурний та контекстуальний аналіз текстів та, у порівнянні з класичним BERT, використовувати значно меншу кількість обчислювальних ресурсів в процесі її навчання та використання.

В класичному вигляді даний метод включає у себе такі етапи:

1. Валідація тексту.
2. Очищення вхідних даних від текстового шуму.
3. Застосування моделі ALBERT для створення векторного представлення тексту.
4. Класифікація тексту за наявністю агресії.
5. Представлення результатів.

Етап 1. Нейронні мережі, створені на основі архітектури «трансформер», мають певні обмеження. Так, в залежності від обраної моделі, визначається максимальний розмір тексту, що може бути оброблений в процесі векторизації. Хоча перевищення встановленого ліміту не призведе до помилок, але модель, зокрема ALBERT, відкине зайву частину тексту. Така поведінка є небажаною, тому має бути створений інструмент для запобігання даній ситуації. В рамках даного дослідження пропонується класифікувати текст англійською мовою. Такий вибір пояснюється існуванням великої вибірки навчальних корпусів текстових даних. Очевидно, що така умова вимагає, щоб вхідний текст був саме англійською

мовою. Проведення аналізу текстів занадто малих розмірів також не є бажаним, адже шанс їх хибної класифікації буде збільшуватися.

Тому в рамках першого етапу базового методу здійснюється перевірка розміру вхідного тексту та його мови.

Етап 2. Очищення вхідних даних від зайвого шуму першочергово передбачає визначення, які особливості тексту є зайвими в процесі класифікації. З огляду на те, що обраним методом векторизації є модель ALBERT, то саме вміст її вбудованого словника є визначальним фактором для видалення зайвого шуму. Наприклад, деякі трансформ-побідні моделі не здатні з'ясувати роль емодзі у тексті, мають проблеми з обробленням хеш-тегів, невірно трактують символьні послідовності. Тому поширеним серед дослідників способом очищення тексту є видалення усіх супутніх елементів, окрім слів та розділових знаків.

Етап 3. Застосування моделі ALBERT в рамках векторизації також має різні варіанти. Як і її аналог BERT, дана модель на початку своєї роботи токенізує вхідний текст. Вектори створюються для кожного токена. При цьому, існує можливість отримати векторне представлення і для всієї послідовності. Найчастіше дослідники використовують результати роботи останнього шару моделі, адже вважається, що його вихідні вектори перевершують за якістю свого наповнення ті, що отримуються із попередніх шарів. Тому класичним підходом для класифікації тексту за наявністю агресії є використання вектору для [CLS] токену, тобто представлення всієї текстової послідовності.

Етап 4. Для визначення, чи є агресивний вміст у певному тексті, застосовується бінарна класифікація. Здатність ALBERT створювати для тексту вектори з великою кількістю їх вагомих структурних та контекстуальних ознак надає можливість використовувати в якості класифікатора найпростішу модель нейронної мережі – лінійний шар. Дослідження показали, що даний підхід є достатнім та вичерпним для рішення великої кількості задач класифікації.

Етап 5. Представлення результатів є фінальним етапом методу. В його рамках користувачу надається інформація про статус вмісту агресії в наданому ним тексті.

2.2. Проблеми автоматичної класифікації природномовних текстових даних за наявністю агресії та способи їх рішень

Факт того, що потенційні вхідні тексти не є формалізованими, створює необхідність розробникам автоматичних класифікаторів тексту за наявністю агресії звертати увагу на людський фактор та його вплив. Зазвичай, користувачі соціальних мереж, де з'являється найбільша кількість негативних висловлювань, звикли до механізмів виявлення та протидії прояву агресії у повідомленнях. Саме тому вони намагаються заважати роботі таких інструментів, навмисно пошкоджуючи власні дописи. Пропускаючи літери у словах, замінюючи їх на символи, використовуючи іншу мову, вони намагаються обмежити здатність наявних систем до проведення якісного аналізу. Саме тому існує необхідність у додатковому механізмі, що дозволив би виправити текст та передати його у систему в оригінальному вигляді.

Варто зазначити, що передача емоцій, включаючи негативний та агресивний підтекст, не обмежується у способах лише словами. Дуже розповсюдженим явищем є використання емодзі. Їх роль – підсилення емоційного забарвлення висловлювання. Вони можуть повторюватися та позначати діаметрально-протилежні емоції. Ступінь аналізу сенсу емодзі та їх впливу на текст є вагомим чинником в процесі його класифікації за наявністю агресії. Але при цьому треба враховувати також і можливу їх відсутність.

Ще одним способом підсилення висловлення емоцій є використання повторюваних розділових знаків. Так, під час написання лайливого повідомлення люди часто в кінці речення ставлять декілька знаків оклику чи

питання. Даний параметр є вагомим та має сильно впливати на рішення класифікатора.

Очевидно, що висока частотність агресивних, негативно забарвлених лайливих, образливих слів у тексті може бути також маркером подібного вмісту. В той самий час, вони можуть бути використані і для звичайного підсилення позитивної емоції. Цей факт часто стає перепоною для методів, подібних запропонованому, на основі лексичного аналізу.

З огляду на те, що поняття агресії не є чітко окресленим та може бути по-різному визначеним, однозначно стверджувати, що певне висловлювання є агресивним, не можна. Найбільший вплив на рішення класифікатора створюється навчальним набором даних. Так, при мануальному анотуванні корпусів текстів кожна оцінка є суб'єктивною і не може бути абсолютною для кожного користувача системи. При цьому, така проблема є незмінною та не може бути повноцінно вирішеною з точки зору формування набору даних.

Можна зробити висновок, що ігнорування подібних ознак негативних емоцій та проблем може значно зменшити точність визначення наявності агресії в природномовних текстових даних. Саме тому однією з цілей даного дослідження є врахування згаданих вище вагомих параметрів в процесі створення та навчання класифікатора.

2.2.1. *Попереднє оброблення вхідних текстових даних*

Як було згадано вище, пошкоджений текст не може бути повноцінно проаналізований та точно класифікований за наявністю агресії. Для цього перед поданням текстових даних в основну частину методу необхідно їх виправити.

В рамках даного дослідження було вирішено вважати за основні типи пошкоджень тексту, що потенційно можуть вплинути на точність класифікації наступні:

- орфографічні помилки;
- спотворення слів;

- зайві або некоректні символи;
- скорочення та жаргонізми.

Існує велика кількість інструментів для виявлення та виправлення подібних проблем. Більшість з них базуються на використанні нейронних мереж та словників. З огляду на різний підхід до їх створення, в даній роботі проведено дослідження ступеню якості приведення тексту до нормальної форми за допомогою таких інструментів, як TextBlob [40], symspellpy [41] та language_tool_python [42]. В рамках експериментів на спотворених дописах із соціальної мережі Twitter було виявлено, що language_tool_python частіше коректно нормалізує природномовні текстові дані, ніж його аналоги.

Також в рамках попередньої обробки запропоновано видаляти повторювані пробіли, URL-адреси та хеш-теги для зменшення кількості текстового шуму при подальшому обробленні.

2.2.2. *Структурні особливості текстових даних*

Серед структурних особливостей текстових даних, що можуть бути ознакою прояву агресії нами було визначено наступні:

- використання повторювальних знаків оклику;
- використання повторювальних знаків питання;
- використання слів у верхньому регістрі.

Для врахування перших 2 параметрів достатньо використати регулярний вираз за допомогою створеної функції (лістинг 1). Результуюче булеве значення представляється у вигляді 0 або 1.

Лістинг 1. Функція для визначення наявності повторювальної пунктуації в тексті

```
def isWithRepeatedPunctuation(text, symbol):
  escapedSymbol = re.escape(symbol)
  pattern = rf"({escapedSymbol}{{2,}})"
  match = re.search(pattern, text)
  return bool(match)
```

Тут врахування третього параметру передбачає обчислення кількості слів, що були вжиті у верхньому регістрі. Для цього необхідно вилучити з тексту список всіх слів в оригінальному вигляді та порахувати кількість повністю капіталізованих. При цьому треба зважати на існування перших слів у реченні, що складаються з 1 букви.

Оскільки довжина текстової послідовності, що надходить у систему, різна, не можна зберігати такий кількісний параметр без нормалізації. Для цього застосовано нормалізацію кількості вживань слів у верхньому регістрі відносно загальної кількості слів (лістинг 2).

Лістинг 2. Функція визначення відносної кількості слів у верхньому регістрі

```
def getWords(text):
  tokens = word_tokenize(text)
  words = [word for word in tokens if word.isalnum()]
  return words


MIN_WORD_LENGTH = 2

def countUppercaseWords(text):
  words = getWords(text)
  uppercaseWords = [word for word in words if word.isupper() and len(word)
>= MIN_WORD_LENGTH]
  return len(uppercaseWords) / len(words)
```

Таким чином, формується вектор із трьох параметрів, що приймають значення від 0 до 1. В подальшому даний набір буде поданий на вхід у класифікатор.

2.2.3. *Лексичні та морфологічні особливості текстових даних*

Лексичний аналіз – однин із основних підходів, що може бути застосований для визначення наявності агресії в природномовних текстових даних. Він базується на використанні словників та напряму залежить від їх якості та обсягу.

Лайливі, образливі та агресивні слова часто є ознакою використання відповідної емоції у тексті. Для їх пошуку та обчислення кількості вживань

із відкритих джерел завантажено корпус слів hurtlex_EN [43], що вміщає у собі понад 8 тисяч слів різноманітного негативного характеру. Також сформовано словник із найбільш популярних нецензурних слів англійською обсягом в 200 прикладів.

Після обчислення кількості входжень слів із словників у тексті було отримано 2 відповідні числові значення, що були нормалізовані відносно загальної кількості слів у тексті. Таким чином, було створено набір із двох параметрів для подальшого використання при класифікації тексту.

Що стосується морфологічних особливостей, то нами визначено, що ознакою агресивного забарвлення текстових даних може бути надмірне використання дієслів та прикметників. Для пошуку даних частин мови пропонується використання бібліотеки, що здатна провести морфологічний аналіз тексту.

Так, обчислюється частота вживання дієслів і прикметників та здійснюється відносна нормалізація шляхом ділення значень на загальну кількість слів у тексті (лістинг 3).

Лістинг 3. Функція обчислення кількості дієслів та прикметників

```
def countVerbs(analyzedText):
  return sum(1 for token in analyzedText if token.pos_ == 'VERB')

def countAdjectives(analyzedText):
  return sum(1 for token in analyzedText if token.pos_ == 'ADJ')
```

В результаті лексичного та морфологічного аналізу тексту формується масив із 4 значень для подальшого використання в процесі класифікації.

2.2.4. *Врахування впливу емодзі на емоційне забарвлення тексту*

Визначення впливу емодзі на загальне емоційне забарвлення тексту може бути здійснене кількома способами:

- за допомогою словників із значеннями їх емоцій;

- за допомогою паттернів їх використання;
- за допомогою глибокого аналізу методами машинного навчання.

Дослідники Deng Yang, Liu Kejian та Yang Cheng в рамках статті про метод сентимент аналізу коментарів у мікроблогах запропонували використання Emoji2vec для глибокого аналізу сенсу емодзі та їх впливу на загальну емоцію [44].

Emoji2Vec – це модель нейронної мережі, спеціально розроблена для представлення емодзі у вигляді векторів, які можна інтегрувати у різні задачі оброблення природної мови. Завдяки цьому підходу, емодзі перестають бути лише графічними символами, які складно аналізувати традиційними методами, і стають повноцінними об’єктами при семантичному аналізі тексту. Модель забезпечує можливість враховувати змістові та контекстуальні зв’язки між емодзі та текстовими елементами, що відкриває нові перспективи для аналізу комунікацій у соціальних мережах, чатах та інших платформах, де емодзі активно використовуються.

На відміну від простого підходу, де емодзі трактуються лише як окремі символи, Emoji2Vec дозволяє перетворювати їх у векторне представлення. Це векторне представлення не тільки зберігає графічну природу емодзі, але й враховує їхній змістовний зв’язок із текстом, у якому вони використовуються. Такий підхід є особливо корисним для аналізу емоційного забарвлення повідомлень, виявлення сарказму чи агресії, а також для покращення роботи алгоритмів класифікації тексту.

Подібно до моделі Word2Vec, яка створює векторні представлення для слів на основі їхнього контексту у тексті, Emoji2Vec працює за аналогічним принципом. Кожне емодзі розглядається як окреме «слово», для якого визначається унікальний вектор, що відображає його значення. Цей вектор формується на основі аналізу контексту, у якому емодзі використовується. Наприклад, емодзі серця може асоціюватися з концептами любові, прихильності або підтримки, тоді як емодзі сміху матиме зв’язок із гумором. Таким чином, Emoji2Vec створює багатовимірні представлення, які

дозволяють враховувати складні семантичні та емоційні аспекти використання емодзі.

Завдяки залученню Emoji2Vec значно розширюються можливості NLP-моделей, що працюють з текстами, які містять емодзі. Це особливо актуально у завданнях класифікації емоцій, аналізу настроїв, побудови чат-ботів та перекладачів, а також для створення моделей, орієнтованих на соціальну взаємодію. Впровадження такої технології дозволяє краще зрозуміти наміри користувачів і точніше інтерпретувати їхні повідомлення, що є важливим для сучасних застосунків у сфері штучного інтелекту.

Оскільки фінальний класифікатор являє собою нейронну мережу, то на вхід до нього має подаватися вектор сталого розміру. З огляду на те, що кількість емодзі в тексті може бути необмеженою, вирішено створити механізм векторизації та врахування невизначеної кількості емодзі. Аналіз тренувального набору даних показав, що середня кількість емодзі на 1 твіт сягає 4. При цьому максимальна кількість – 12.

Для формування загального векторного представлення емодзі всього тексту нами було визначено, що їх максимальна кількість має бути до 10 включно. Таке число було обране через необхідність підтримки балансу між мінімізацією розміру результуючого вектору та максимізацією наповнення його вагомими ознаками. Якщо емодзі недостатньо, то відповідні вектори заповнюються нулями. При більшій їх кількості застосовується формування десятого вектора на основі усіх надлишкових. Для цього береться усереднений за значеннями вектор, щоб врахувати їх загальний вплив на емоційне забарвлення.

В результаті аналізу емодзі для вхідного тексту створюється вектор розміром 3000 параметрів.

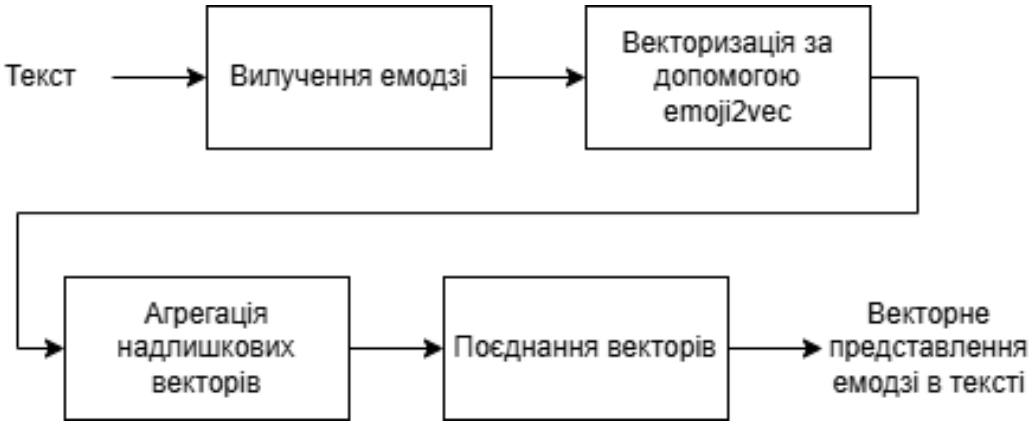


Рис. 4. Схема формування векторного представлення емодзі в тексті

2.2.5. *Спосіб вирішення проблеми нечітко визначеного поняття агресії*

Як було згадано раніше, термін «агресивне висловлювання» є досить абстрактним та трактується по-різному в залежності від суб’єктивної точки зору та потреб.

Використання традиційної бінарної класифікації, яка розділяє тексти на ті, що містять агресію, і ті, що її не містять, є надмірно спрощеним підходом. У такій моделі ігноруються проміжні стани, різноманітні точки зору користувачів або випадки, коли текст може мати ознаки агресії, але не виражати її явно. Відтак, існує необхідність у переході до більш гнучкого підходу.

Для вирішення цієї проблеми було впроваджено підхід, що передбачає використання регресійної моделі для визначення рівня агресії у тексті. Замість того, щоб однозначно класифікувати текст як агресивний чи неагресивний, система оцінює рівень агресії за шкалою від 0 до 1. Значення «0» вказує на повну відсутність агресії, тоді як значення «1» означає високу інтенсивність агресії. Такий підхід дозволяє краще враховувати нюанси змісту текстів і пропонує більш об'єктивний спосіб оцінювання. Важливим аспектом цього підходу є можливість встановлення користувачем граничної оцінки, яка визначає, від якого значення текст слід вважати агресивним. Наприклад, у деяких випадках агресія може вважатися присутньою, якщо

рівень перевищує 0,7, тоді як в інших контекстах це значення може бути нижчим чи вищим. Такі пороги визначаються за допомогою мануального калібрування на необхідних для класифікації даних.

Запропонований підхід вирішує кілька важливих завдань. По-перше, він дозволяє враховувати різні інтерпретації агресії у тексті, які можуть залежати від індивідуального або соціокультурного контексту. Це особливо важливо в багатомовних і багатокультурних середовищах, де поняття агресії може суттєво відрізнятися. По-друге, використання регресійної моделі сприяє більш точному аналізу тональності тексту та дозволяє виявляти приховану агресію, яка могла б бути пропущена бінарними класифікаторами. По-третє, система стає більш адаптивною до змін у сприйнятті агресії, оскільки граничне значення може бути змінене залежно від потреб користувача або специфіки завдання.

Для створення класифікатора, що вирішував би задачу регресії, необхідно проводити тренування на спеціалізованому проанотованому наборі даних із вказаними оцінками рівня агресії. При навчанні також треба використовувати відповідну функцію втрат для задач регресії, як середньоквадратична помилка.

Таким чином, перехід від бінарної класифікації до регресійної моделі є значним кроком уперед у врахуванні суб'єктивності поняття агресії. Цей підхід забезпечує більшу гнучкість, точність та адаптивність. Використання моделі, яка оцінює агресію у вигляді неперервного значення, дозволяє уникнути спрощення та сприяє створенню системи, здатної враховувати різноманітність сприйняття людської комунікації.

2.3. Опис запропонованого методу автоматичної класифікації природномовних текстових даних за наявністю агресії

Для забезпечення гарантії високої точності класифікації тексту за наявністю агресії необхідно чітко визначити характеристики вхідних текстових даних, щоб система змогла повноцінно їх проаналізувати. Як було

вказано раніше, запропонований метод може працювати лише з текстовими даними англійською мовою. До того ж, через системні обмеження ALBERT, розмір вхідного тексту повинен бути в межах від 3 до 160 слів. Таким чином, перший процес, що виконується при роботі запропонованого метода – валідація вхідних даних. За результатами його роботи метод може отримати текст для його класифікації або користувач повідомляється про помилку та її причини.

Ще одним, не менш важливим етапом попереднього оброблення тексту є виправлення потенційних помилок. Для цього визначається, чи знаходяться помилки у вхідних даних. Якщо визначено, що у послідовності наявні слова, що були пошкоджені, то ініціалізується процес їх виправлення та приведення до нормальної форми. Метою даного підходу є досягнення таких цілей:

- мінімізація надлишкових операцій – відмова від процесу виправлення тексту, якщо у ньому не було зроблено помилок;
- застереження від потенційно некоректного виправлення слів – мінімізація впливу хибної роботи імпортованих інструментів.

В результаті до основної частини методу надходитиме коректний за системними вимогами текст із зменшеною до мінімуму кількістю помилок та навмисних пошкоджень, що гарантує повноцінний аналіз та класифікацію вхідних текстових даних за наявністю агресії.

Оскільки в якості класифікатора використовується модель нейронної мережі – лінійний шар, точність її роботи напряму залежить від якості ознак природномовних текстових даних. Ці параметри мають бути вагомими та, при цьому, нормалізованими. Таким чином, наступний етап роботи методу, аналіз вхідних даних, має надавати класифікатору одномірний вектор параметрів фіксованої довжини із значеннями від 0 до 1. При цьому, необхідно слідкувати за обсягом вектору. Розмір, що не буде співпадати з тим, на якому навчався класифікатор, при базовій реалізації методу може негативно впливати на точність його роботи.

Проблеми оригінального методу в основному включають недостатньо глибокий процес аналізу додаткових ознак: лексичних, морфологічних, структурних. Базуючись на контекстуальному аналізі BERT, втрачаються дійсно вагомі параметри, за якими часто можна визначити характер переданої емоції, навіть не занурюючись у сенс. Саме тому основною ідеєю запропонованого методу є обчислення та врахування усіх вищезазначених ознак та поєднання їх разом з результуючим ембедінгом тексту від ALBERT.

З огляду на те, що моделі на основі глибокого навчання, у даному випадку ALBERT, не потребують додаткових оброблень вхідних даних, як токенізація або видалення стоп-слів, основну частину методу можна умовно розділити на 2 частини:

- аналіз додаткових текстових ознак власними засобами;
- контекстуальний аналіз на основі моделі ALBERT.

Розглянемо детальніше процес формування вектору додаткових текстових ознак.

Для обчислення лексичних ознак, як було вказано раніше, використовуються 2 словники із словами-маркерами агресії, ненависті та загалом негативного настрою. Для цього необхідно провести токенізацію вхідного тексту та відфільтрувати результат так, щоб залишилися лише слова. З огляду на те, що словники вміщають у собі слова в різних формах, попередня лематизація не має сенсу. Тому для обчислення кількості їх входжень у запропонований текст необхідно 1 раз обійти вхідний масив слів із загальною складністю алгоритму – $O(n)$. При цьому система визначає розмір вхідних даних – кількість слів у тексті, що потрібно для проведення нормалізації. В результаті формується вектор із двох значень від 0 до 1.

Морфологічні ознаки в даному випадку характеризують текст з точки зору кількості вживань дієслів та прикметників. Для цього використовуються інструменти морфологічного аналізу тексту для визначення їх приналежності до частин мови. В результаті із алгоритмічною складністю $O(n)$ обчислюється окремо кількість слів із частиною мови

дієслово та прикметник відповідно. На цьому етапі також застосовується нормалізація відносно загальної кількості слів у вхідному тексті.

Структурні ознаки включають визначення, чи було використано повторювальні знаки оклику, питання та скільки вжито слів у верхньому регістрі. Для перших двох параметрів достатньо перевірити текст за допомогою регулярного виразу (лістинг 2). Останній параметр також передбачає використання списку всіх слів із тексту для обчислення загальної кількості тих, що повністю складені із великих літер і при цьому складаються із більш, ніж однієї. Також в результаті застосовується подібна до попередніх етапів відносна нормалізація. Булеві значення перетворюються у 0 або 1. В результаті цього формується масив із 3 значень.

Для аналізу вливу емодзі на текст застосовується модель нейронної мережі Emoji2vec. При цьому, першочергово виконується формування масиву емодзі окремо від тексту. Для цього необхідно використовувати додатковий інструмент, що здатен якісно визначати, чи є поточний символ емодзі. Модель Emoji2vec залежить від вбудованого словника. Цей факт ставить обмеження на здатність нейромережі повноцінно аналізувати ознаки емодзі. Саме тому вводиться додаткова перевірка, яка виключає із загального списку ті емодзі, які не можуть бути розглянуті запропонованою моделлю.

Наступним етапом є формування векторів ознак, розміром 300 значень кожен. Як було згадано вище, для формування постійного розміру вихідного вектору, було визначено, що максимальна кількість емодзі в тексті дорівнює 10. Саме тому застосовується механізм усереднення значень надлишкових векторів. В результаті аналізу емодзі система отримує одномірний вектор розміром 3000 значень.

Основним етапом роботи методу є створення ембедінгів тексту за допомогою ALBERT. Обрана модель, albert-base-v2, вміщає 12 шарів та дозволяє отримати ембедінги для кожного з них. У порівнянні з базовим методом, пропонується використовувати результати роботи останніх 4, замість лише останнього. При цьому існує декілька варіантів, як можна

врахувати одразу декілька векторів. Зокрема, конкатенація, сума та середнє значення. З огляду на те, що поєднання векторів не позбавляє систему потенційно вагомих значень та не несе втрат під час округлень, саме такий підхід був обраний для формування результуючого набору структурних та контекстуальних ознак.

По завершенню усіх вищезгаданих етапів метод зберігає 5 одномірних векторів різного розміру. Класифікатор на вхід очікує лише 1. Тому перед початком фінальної класифікації необхідно виконати їх конкатенацію.

Очевидно, що в результаті сформується вектор дуже великої довжини. Лінійний шар в такому випадку втрачає свою високу здатність до класифікації. Для виправлення даної проблеми застосовується механізм ущільнення вектору за допомогою використання ще одного лінійного шару. Таким чином, відбувається зменшення розміру вектору до 600, ідентичного для оригінального методу. Результат подається у фінальний класифікатор.

Результат класифікації – оцінка тексту з точки зору наявності агресії у ньому. В залежності від користувацьких налаштувань, система повинна або подати оцінку у оригінальному вигляді, або, якщо було задано порогове значення, інтерпретувати результат у вигляді бінарного значення, тобто повідомити, чи наявна агресія у вхідному тексті. Варто зазначити, що система має змогу опрацьовувати одразу декілька текстів. Саме тому в процесі інтерпретації до кожного вхідного тексту у відповідність ставиться результат його класифікації.

В результаті роботи методу формується повідомлення, чи включає певний текст у собі елементи агресії у відповідності до базових налаштувань, що гарантує підлаштування системи до особливих потреб користувача.

Нижче наведено фрагмет коду надбудови над моделлю ALBERT для врахування додаткових структурних, лексичних та морфологічних ознак природномовних текстових даних (лістинг 4).

Лістинг 4. Надбудова над моделлю ALBERT з урахуванням додаткових ознак агресії

```
class AlbertForSequenceClassification(AlbertPreTrainedModel):
    def __init__(self, config):
        super().__init__(config)
        self.num_labels = config.num_labels

        self.albert = AlbertModel(config)
        self.dropout = nn.Dropout(config.classifier_dropout_prob)
        self.hidden_size = config.hidden_size
        self.intermediate_layer = nn.Linear(config.hidden_size + 3000 + 6,
config.hidden_size)
        self.relu = nn.ReLU()
                    self.classifier    =    nn.Linear(config.hidden_size,
self.config.num_labels)
        self.init_weights()
def forward(
        self,
        input_ids=None,
        attention_mask=None,
        token_type_ids=None,
        position_ids=None,
        head_mask=None,
        inputs_embeds=None,
        labels=None,
        emoji_embedings=None,
        output_attentions=None,
        output_hidden_states=None,
        additional_features=None,
        return_dict=None,
    ):
        return_dict  =  return_dict  if  return_dict  is  not  None  else
self.config.use_return_dict
        outputs = self.albert(
            input_ids=input_ids,
            attention_mask=attention_mask,
token_type_ids=token_type_ids,
            position_ids=position_ids,
            head_mask=head_mask,
            inputs_embeds=inputs_embeds,
            output_attentions=output_attentions,
            output_hidden_states=output_hidden_states,
            return_dict=return_dict,
        )
        pooled_output=outputs[1]
           combined_output  =  torch.cat((pooled_output,  emoji_embedings,
additional_features), dim=1)
                                              intermediate_output          =
self.relu(self.intermediate_layer(combined_output))
        intermediate_output = self.dropout(intermediate_output)
        logits = self.classifier(intermediate_output)

        loss = None
        if labels is not None:
            if self.num_labels == 1:
                #  We are doing regression
                loss_fct = MSELoss()
                loss = loss_fct(logits.view(-1), labels.view(-1))
            else:
                loss_fct = CrossEntropyLoss()
```

Продовження лістингу 4

```
                        loss = loss_fct(logits.view(-1, self.num_labels),
labels.view(-1))

        if not return_dict:
            output = (logits,) + outputs[2:]
            return ((loss,) + output) if loss is not None else output

        return SequenceClassifierOutput(
            loss=loss,
            logits=logits,
            hidden_states=outputs.hidden_states,
            attentions=outputs.attentions,
        )
```

Для загального опису запропонованого методу створено його схематичне представлення, що включає опис послідовності процесів для підвищення точності класифікації природномовних текстових даних за наявністю агресії (рис. 5).

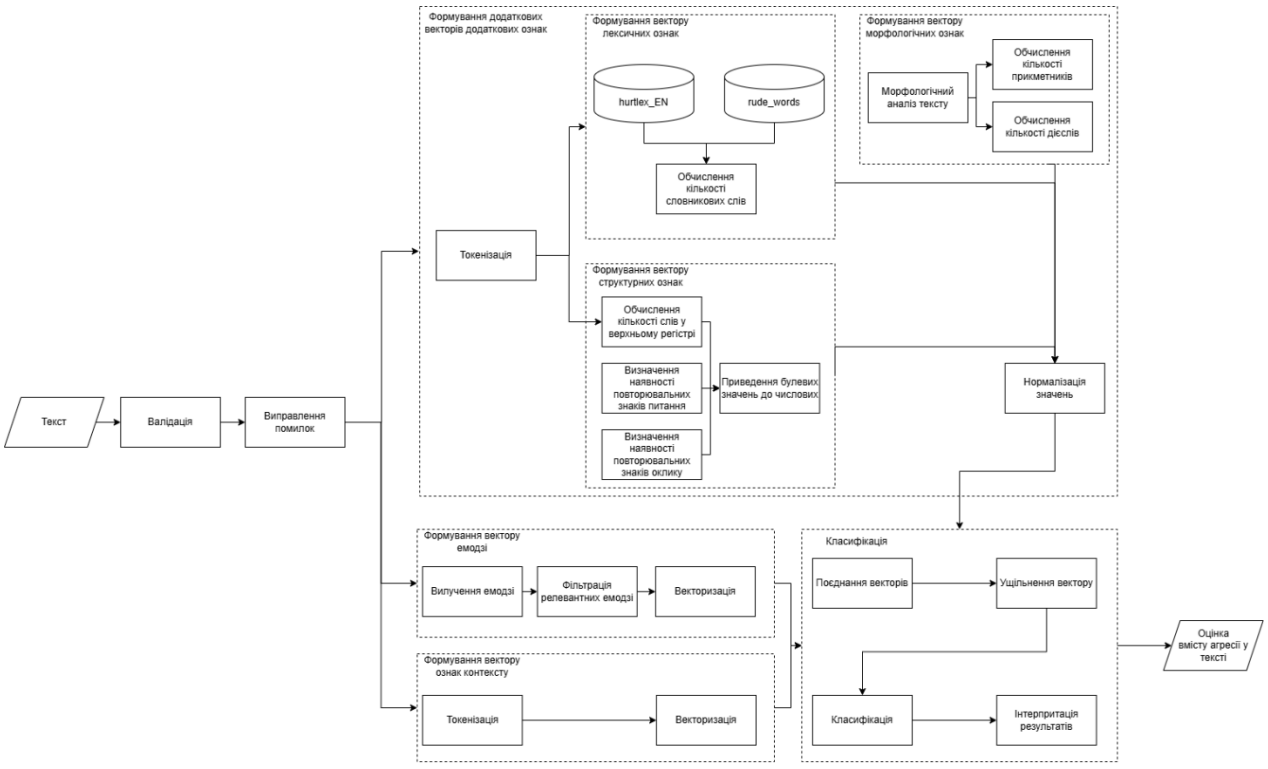


Рис. 5. Схема запропонованого методу автоматичної класифікації природномовних текстових даних за наявністю агресії

2.4. Обґрунтування вибору метрик для оцінки ефективності запропонованого методу

Метою даного дослідження є підвищення точності класифікації природномовних текстових даних за наявністю агресії. Щоб довести ефективність запропонованого методу, необхідно порівнювати його характеристики з аналогами. Найбільш коректним в даному випадку є порівняння за допомогою кількісних метрик. При цьому необхідно враховувати те, що класифікатором, роботу якого необхідно оцінити, виступає нейронна мережа. Цей факт окреслює множину метрик, що можуть бути використані.

Для глибокого аналізу точності роботи необхідно обчислювати декілька метрик, оскільки кожна з них характеризує систему з певної точки зору.

Перехід до вирішення задачі регресії замість оригінальної бінарної чи мультикласової класифікації також впливає на релевантність метрик. Так, використання F1-міри, AUC, Recall, Precision вже не має сенсу. При цьому Accuracy є актуальною метрикою, хоча і має вираховуватися інакше.

Таким чином для оцінювання точності класифікації природномовних текстових даних за наявністю агресії обрано наступні: Loss, MSE, MAE, R^2, Accuracy. Нижче наведено детальний опис даних метрик разом із формулами для їх обчислення:

1. Loss – це кількісна міра, яка вимірює похибку між прогнозованими результатами та фактичними цілями [45]. Вона використовується під час навчання моделі нейронної мережі, визначаючи, наскільки добре працює модель і надсилає сигнал зворотнього зв’язку до оптимізатора для налаштування параметрів таким чином, щоб зменшити втрати при наступній ітерації. Існують різні типи функцій втрат і обираються відповідно до характеру розв’язуваної задачі.

Для класифікатора, що визначає неперервну величину, функція втрат має бути середньоквадратичною помилкою.

2. MSE – метрика, що вимірює середньоквадратичну помилку, тобто значення квадратів відхилень між передбаченими значеннями моделі та фактичними [46]. Чим менше значення MSE, тим ближчими є передбачення моделі до реальних даних. MSE дає кількісну оцінку середньої похибки, але її значення є залежним від одиниць вимірювання цільової змінної. Завдяки піднесенню до квадрату, MSE надає більше значення більшим відхиленням, що робить її корисною для виявлення значних похибок. При цьому великі помилки мають непропорційно великий вплив на значення MSE, що може призвести до викривлення оцінки в разі наявності викидів у даних. Для обчислення MSE використовується формула (10):

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - j_i)^2, \qquad (10)$$

де n – загальна кількість зразків у наборі даних, y_i – реальне значення для i зразка, j_i – передбачене значення для і зразка.

3. MAE – це метрика, яка використовується для оцінки якості моделі, що працює з безперервними величинами [47]. MAE обчислює середню абсолютну різницю між передбаченими значеннями моделі та фактичними значеннями. Вона показує, наскільки в середньому передбачення відхиляються від істинних значень у тих самих одиницях, що й цільова змінна. MAE вимірює середню абсолютну похибку для всіх передбачень моделі. Вона не підносить відхилення до квадрату, як у MSE, а бере лише їх модуль, що робить метрику менш чутливою до великих викидів у даних. На відміну від MSE, MAE менш чутлива до впливу великих викидів, оскільки не підносить відхилення до квадрату. MAE надає однакову вагу всім

похибкам незалежно від їх величини, що може бути недоліком у задачах, де великі похибки є критичними. Для обчислення MAE використовується формула (11):

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} | y_i - j_i |, \tag{11}$$

де n – загальна кількість зразків у наборі даних, y_i – реальне значення для i зразка, j_i – передбачене значення для і зразка.

4. R^2 є статистичною метрикою, яка використовується для оцінки якості регресійної моделі [48]. Вона показує, яку частку варіації залежної змінної у можна пояснити незалежними змінними в моделі. R^2 варіюється від 0 до 1, де значення, ближчі до 1, свідчать про те, що модель добре описує дані. R^2 базується на порівнянні двох величин: TSS (загальна варіація залежної змінної) та ESS (залишкова варіація). Для обчислення R^2 використовується формула (12):

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - j_i)^2}{\sum_{i=1}^{n} (y_i - f_i)^2}, \tag{12}$$

де n – загальна кількість зразків у наборі даних, y_i – реальне значення для i зразка, j_i – передбачене значення для і зразка. f_i – середнє значення істиних значень.

5. Accuracy – метрика для оцінки якості моделей у задачах класифікації та регресії [49]. Значення Accuracy легко інтерпретується але при цьому при невдалому визначенні порогового значення може бути менш інформативним. У даному випадку метрика використовується для оцінки точності регресійної моделі. В якості порогового значення було емпірично обрано 0,0025. Таке число дозволяє отримувати точні значення метрики, завдяки

чому збільшується об’єктивність порівняння з результатами роботи інших методів. Для обчислення Accuracy використовується формула (13):

$$\text{Accuracy} = \frac{\sum_{i=1}^{n} e_i < 0.0025}{n}, \tag{13}$$

де n – загальна кількість зразків у наборі даних, e_i – квадрат різниці між передбаченним та істинним значенням. $e_i < 0{,}002$ – індикаторна функція, яка дорівнює 1, якщо умова виконується і 0, якщо ні.

2.5. Висновки до другого розділу

У даному розділі розглянуто підхід на основі використання моделі глибокого навчання ALBERT, що був використаний у якості базису для формування запропонованого методу.

В рамках оригінального методу визначено потенційні проблеми, що можуть негативно впливати на точність класифікації природномовних текстових даних за наявністю агресії. Зокрема, було перелічено:

- наявність потенційних помилок та навмисних пошкоджень тексту;
- відсутність врахування типової для прояву агресії лексики;
- відсутність врахування структурних особливостей тексту, що можуть означати агресію;
- відсутність врахування морфологічних ознак тексту;
- відсутнє або обмежене врахування емодзі;
- неможливість надання об’єктивної оцінки ступеню агресії тексту.

Для підвищення точності роботи методу було запропоновано та описано такі етапи:

- валідація вхідних даних;
- виправлення помилок;
- врахування структурних особливостей текстових даних;

- врахування морфологічних особливостей текстових даних;
- врахування контекстуальних особливостей текстових даних;
- врахування значення емодзі;
- перехід до вирішення задачі регресії;
- ущільнення розміру результуючого вектору.

Запропонований метод автоматичної класифікації природномовних текстових даних було представлено у вигляді схеми.

Для всебічної оцінки роботи методу та порівняння точності його класифікації обрано та обґрунтовано набір метрик, до складу яких входять: MSE, MAE, R^2, Accuracy та інші.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАПРОПОНОВАНОГО МЕТОДУ

3.1. Обґрунтування вибору засобів реалізації ПЗ

Для створення програмної реалізації запропонованого методу автоматичної класифікації природномовних текстових даних за наявністю агресії розроблено консольний застосунок, що надає можливість налаштовувати особливості власної роботи за допомогою користувацького інтерфейсу. В процесі розроблення використовувалися наступні технології:

- мова програмування Python;
- бібліотека Transformers для оброблення природномовних текстових даних за допомогою трансформ-подібних нейронних мереж;
- фреймворк PyTorch для створення моделей нейронних мереж глибокого навчання;
- бібліотека Scikit-learn, що забезпечує допоміжними інструментами процес навчання та оцінювання нейронних мереж;
- бібліотека Pandas для здійснення контролю над набором даних;
- бібліотека NLTK, що надає набір інструментів для оброблення природномовних текстових даних;
- бібліотека emoji для виявлення та оброблення емодзі в текстових даних;
- бібліотека Spacy, що дозволяє здійснювати морфологічний аналіз тексту.

3.1.1. Мова програмування Python

Python – інтерпретована об’єктно-орієнтована мова програмування високого рівня із динамічною семантикою [50]. Її популярність обумовлена простим, легким для вивчення синтаксисом, що, у свою чергу, полегшує сприйняття чужого коду. Python підтримує механізм використання модулів та бібліотек для можливості повторного використання коду. Зазначимо, що

етапу компіляції в процесії її роботи немає, що полегшує процес налагодження коду. При цьому інтерпретатор викликає виняток при виявленні помилки в програмі.

Python зазвичай застосовується для розроблення таких продуктів:

- веб-застосунки;
- десктопне програмне забезпечення;
- застосунки для аналізу та оброблення даних;
- машинне навчання [51].

Python має відкритий вихідний код, що позитивно впливає на розвиток її екосистеми бібліотек і фреймворків [52]. Вони дозволяють значно пришвидшити процес створення застосунків будь-якого типу та надають можливість розробникам приділяти більше уваги саме вирішенню проблем, а не написанню існуючих інструментів. Необхідно зазначити, що суспільство користувачів даної мови є активним, що забезпечує інших великою кількістю навчальних матеріалів та надає підтримку на різноманітних форумах.

Зазначимо, що Python постійно розвивається та вводить значні поліпшення продуктивності та нові функції при цьому не втрачаючи існуючі переваги.

Розглянемо переваги Python, що стали вирішальними в процесі вибору мови програмування для створення програмної реалізації методу для автоматичної класифікації природномовних текстових даних за наявністю агресії:

- велика кількість бібліотек та фреймворків, що надають багато інструментів для аналізу природньої мови, що значно прискорює процес створення застосунку;
- краща підтримка Python від Google Colab. З огляду на необхідність проведення навчання нейронної мережі із архітектурою «трансформер», потреба у наявності великої кількості

обчислювальних ресурсів спровокувала використання даного сервісу;

- значна кількість навчальних ресурсів, як по машинному навчанню, так і по обробці природномовних текстових даних беруть за основу саме Python. Можливість використовувати готові підходи та методи для реалізації запропонованого методу в рамках даної мови програмування дозволяють економити час розроблення застосунку;
- можливість розгалуження потоків виконання коду. З огляду на те, що компоненти запропонованого методу вибагливі до потужності виконуючої машини та їх робота може займати доволі тривалий час, то існує об'єктивна причина для розмежування процесу виконання програми на декілька потоків. Такий механізм дозволяє робити паралельні обчислення та зменшувати загальний час роботи автоматичної класифікації текстових даних;
- наявність великої кількості інструментів для створення консольного користувацького інтерфейсу. Зокрема, за допомогою Python можна відображати стан оброблення вхідних даних. Також наявні зручні функції-утиліти для забезпечення валідації користувацького вводу.

Таким чином, Python обрано у якості мови програмування для створення консольного застосунку, що реалізує процес автоматичної класифікації природномовних текстових даних за наявністю агресії.

3.1.2. Бібліотека Transformers

Бібліотека Transformers від Hugging Face – це відкритий інструмент, що забезпечує доступ до сучасних моделей оброблення природної мови на основі архітектури Transformer [53].

Ця архітектура є основою для побудови моделей з високою пропускною здатністю, які здатні ефективно працювати з великими обсягами даних і демонструвати високі результати у багатьох завданнях NLP.

Для даної бібліотеки можна виділити такі особливості:

- підтримує різноманітні моделі, серед яких BERT, GPT, RoBERTa, T5, DistilBERT та інші. Кожна модель налаштовується для різних завдань завдяки гнучкому API;
- реалізовано централізований Model Hub, що містить тисячі моделей, готових до використання. Користувачі можуть завантажувати, адаптувати й розгортати моделі за кілька рядків коду;
- підтримує PyTorch і TensorFlow, забезпечуючи можливість перенесення моделей між цими середовищами;
- оснащена оптимізованими токенізаторами, написаними на Rust, що значно пришвидшує підготовку даних. Також підтримується експорт моделей у формати, оптимізовані для розгортання, такі як ONNX.

Архітектура бібліотеки складається з трьох основних компонентів: токенізатор, який перетворює текст у числові індекси, власне модель на основі Transformer, що обробляє ці індекси для отримання контекстуальних векторів і голова, яка відповідає за прогноз для конкретного завдання.

Бібліотека широко використовується в дослідницькій спільноті, індустрії та середовищі розробників для вирішення задач аналізу тексту, побудови чат-ботів, створення систем автоматичного перекладу тощо.

Таким чином, Transformers є потужним інструментом, що дозволяє в рамках даного дослідження використати необхідні моделі архітектури Трансформер для створення автоматичного класифікатора природномовних текстових даних за наявністю агресії.

3.1.3. Фреймворк PyTorch

PyTorch – це фреймворк, що застосовується для побудови моделей глибокого навчання [54]. Він реалізований на мові Python та має легкий для розуміння та використання інтерфейс. Сильною стороною PyTorch є

можливість підтримки графічних процесорів і використання автоматичної диференціації у зворотньому, що дозволяє автоматично змінювати графіки обчислень. Саме тому він є одним з найпопулярніших інструментів для проведення швидких експериментів та створення прототипів.

Фреймворк поєднує в собі ефективні та гнучкі бібліотеки Torch із прискоренням GPU та інтуїтивно зрозумілий інтерфейс Python, який зосереджений на швидкому прототипуванні, читабельному коді та підтримці широкого спектру моделей глибокого навчання. Pytorch дозволяє розробникам використовувати знайомий підхід імперативного програмування.

До ключових переваг даного фреймворку можна віднести:

- у спільноті PyTorch.org є велика та активна спільнота з чудовою документацією та посібниками. Форуми активні та підтримуються;
- написаний на Python і інтегрований із такими популярними бібліотеками Python, як NumPy для наукових обчислень, SciPy і Cython для компіляції Python у C для кращої продуктивності. Оскільки його синтаксис і використання подібні до Python, розробникам Python відносно легко освоїти PyTorch;
- добре підтримується основними хмарними платформами;
- мова сценаріїв, яка називається TorchScript, проста у використанні та гнучка в режимі очікування. Це режим швидкого запуску, у якому операції виконуються негайно, коли вони викликаються з Python, але який також можна перейти до графової моделі для швидкості та оптимізації в середовищах виконання C++;
- підтримує CPU, GPU та паралельну обробку, а також розподілене навчання. Це означає, що обчислювальну роботу можна розподілити між кількома ядрами CPU та GPU, а навчання можна проводити на кількох GPU на кількох машинах;

- підтримує динамічні обчислювальні графіки, що дозволяє змінювати поведінку мережі під час виконання. Це забезпечує значну перевагу гнучкості перед більшістю фреймворків машинного навчання, які вимагають визначення нейронних мереж як статичних об’єктів перед виконанням;
- нові користувальницькі компоненти можна створювати як підкласи стандартного класу Python, параметрами можна легко поділитися із зовнішніми наборами інструментів, як-от TensorBoard, а бібліотеки можна легко імпортувати та використовувати всередині;
- має добре оцінений набір API, які можна використовувати для розширення основних функцій;
- підтримує як «режим бажання» для експериментів, так і «режим графіка» для високопродуктивного виконання;
- у ньому є велика колекція інструментів і бібліотек у різних сферах: від комп’ютерного зору до навчання з підкріпленням;
- підтримує чистий зовнішній інтерфейс C++, знайомий програмістам Python, і може використовуватися для створення високопродуктивних програм C++.

3.1.4. Бібліотека Scikit-learn

Scikit-learn – це одна з найпопулярніших бібліотек на Python, що експортують допоміжні інструменти для створення нейронних мереж [55]. Вона надає вибір ефективних інструментів для машинного навчання та статистичного моделювання, включаючи класифікацію, регресію, кластеризацію та зменшення розмірності через узгоджений інтерфейс у Python. В основу даної бібліотеки ввійшли такі, як NumPy, SciPy і Matplotlib.

Замість того, щоб зосереджуватися на завантаженні, маніпулюванні та узагальненні даних, бібліотека Scikit-learn зосереджена на моделюванні даних. Деякі з найпопулярніших груп моделей, наданих Sklearn, такі:

- алгоритми керованого навчання. Майже всі популярні алгоритми керованого навчання, як-от лінійна регресія, метод опорних векторів, дерево рішень тощо, є частиною scikit-learn;
- алгоритми неконтрольованого навчання. З іншого боку, він також має всі популярні алгоритми неконтрольованого навчання від кластеризації, факторного аналізу, аналізу основних компонентів до неконтрольованих нейронних мереж;
- кластеризація. Ця модель використовується для групування немаркованих даних;
- перехресна перевірка. Використовується для перевірки точності контрольованих моделей на невидимих даних;
- зменшення розмірності. Використовується для зменшення кількості атрибутів у даних, які в подальшому можна використовувати для підсумовування, візуалізації та вибору ознак;
- методи ансамблю. Використовується для об’єднання прогнозів кількох контрольованих моделей;
- вилучення функцій. Використовується для вилучення ознак із даних для визначення атрибутів у даних зображення та тексту;
- вибір функцій. Використовується для визначення корисних атрибутів для створення контрольованих моделей;
- ця бібліотека з відкритим вихідним кодом, яка також може використовуватися в комерційних цілях за ліцензією BSD.

3.1.5. Бібліотека Pandas

Pandas – найпопулярніша бібліотека програмного забезпечення для оброблення та аналізу даних для мови програмування Python [56]. Як бібліотека програмного забезпечення з відкритим вихідним кодом, створена на основі Python спеціально для оброблення та аналізу даних, Pandas пропонує структуру даних і операції для потужного, гнучкого та легкого у

використанні аналізу й оброблення даних. Pandas посилює Python, надаючи цій популярній мові програмування можливість працювати з даними, подібними до електронних таблиць, уможливлюючи швидке завантаження, вирівнювання, маніпулювання та об’єднання на додаток до інших ключових функцій. Pandas цінується за високу продуктивність, коли внутрішній вихідний код написаний на C або Python.

За словами організаторів Python Package Index – сховища програмного забезпечення для мови програмування Python – Pandas добре підходить для роботи з кількома типами даних, зокрема:

- табличні дані з неоднорідно типізованими стовпцями, як у таблиці SQL або електронній таблиці Excel;
- упорядковані та невпорядковані дані часових рядів;
- довільні матричні дані з мітками рядків і стовпців;
- будь-яка інша форма наборів спостережень або статистичних даних.

Pandas надає кілька ключових переваг як науковцям, так і розробникам даних, зокрема:

- легке оброблення відсутніх даних як у даних із плаваючою комою, так і без неї;
- стовпці можна вставляти та видаляти з DataFrames і об’єктів більшої розмірності;
- об’єкти можуть бути явно вирівняні за набором міток, або користувач може просто ігнорувати мітки та дозволити серіям, DataFrame тощо автоматично вирівнювати дані під час обчислень;
- потужна, гнучка функція групування для виконання операцій розділення, застосування та об’єднання над наборами даних для агрегування та перетворення даних;
- полегшення конвертації обірваних, по-різному індексованих даних в інших структурах даних Python і Numpy в об’єкти DataFrame;

- інтелектуальне нарізання на основі міток, фантастичне індексування та піднабір великих наборів даних;
- гнучка зміна форми та поворот наборів даних;
- ієрархічне маркування осей;
- надійні інструменти вводу/виводу для завантаження даних із плоских файлів, файлів Excel, баз даних і збереження/завантаження даних із надшвидкого формату HDF5;
- спеціальна функціональність часових рядів: генерація діапазону дат і перетворення частоти, статистика рухомого вікна, зміщення дати та відставання.

3.1.6. Бібліотека NLTK

Набір інструментів природної мови NLTK – це середовище програмування Python для створення програм для статистичного оброблення природної мови [57].

Він включає в себе бібліотеки оброблення мови для токенізації, синтаксичного аналізу, класифікації, визначення кореня, маркування та семантичного обґрунтування. Одна з найпотужніших бібліотек NLP, містить інструменти, які дозволяють комп'ютерам розуміти природну мову та відповідним чином реагувати на її використання.

NLTK підтримує широкий діапазон мов, а не лише англійську. Він надає інструменти токенізації, визначення коренів і морфологічного аналізу. На додаток до стандартних завдань NLP, таких як токенізація та парсинг, NLTK містить інструменти для аналізу настроїв. Це дає змогу набору інструментів визначати настрій певного фрагмента тексту, що може бути корисним для таких програм, як моніторинг соціальних медіа чи аналіз огляду продукту.

NLTK має велику й активну спільноту користувачів і співавторів, що означає велику кількість ресурсів, доступних для навчання та усунення

недоліків. На додаток до книги NLTK і навчальної програми, згаданих у статті, доступні онлайн-форуми, навчальні посібники та приклади кодів.

3.1.7. Бібліотека emoji

Бібліотека emoji для Python забезпечує зручну роботу з емодзі у текстах [58]. Вона дозволяє легко інтегрувати, знаходити, замінювати чи видаляти емодзі в рядках. Основні функції бібліотеки включають:

- використання коротких кодів, таких як «:smile:», для вставки емодзі;
- перевірку, чи містить рядок емодзі, або підрахунок їхньої кількості;
- заміну емодзі на відповідні коди;
- можливість видалення всіх емодзі з тексту;
- підтримку останніх версій емодзі відповідно до Unicode.

3.1.8. Бібліотека Spacy

Spacy – це безкоштовна бібліотека з відкритим вихідним кодом, написана мовою Python для розширеного оброблення природної мови [59]. Її розроблено спеціально для створення програм, які обробляють і розуміють великі обсяги текстових даних. Однією з важливих сильних сторін Spacy є її адаптивність для створення та використання конкретних моделей для завдань NLP, таких як ідентифікація іменованих сутностей або тегування частин мови. Розробники можуть точно налаштувати свої програми, використовуючи відповідні дані, щоб задовольнити потреби в конкретних випадках використання.

Spacy має складні функції розпізнавання об’єктів, токенізації та синтаксичного аналізу. Він також підтримує широкий спектр широко використовуваних мов. Spacy є підходящим варіантом для розробки NLP-додатків виробничого рівня, оскільки він швидкий і ефективний під час виконання.

В рамках консольного застосунку для автоматичної класифікації природномовних текстових даних за наявністю агресії Spacy використовується для ефективного визначення частин мови слів, що вміщаються у вхідних даних.

3.2. Архітектура розробленого ПЗ

Основний підхід, що використовувався при створенні архітектури програмного забезпечення автоматичної класифікації природномовних текстових даних за наявністю агресії – модульність. Кожна логічна частина застосунку відокремлена від інших та працює лише в рамках власної області відповідальності.

Необхідно зазначити, що запропонований метод був реалізований у вигляді консольного застосунку з користувацьким інтерфейсом. При цьому, головний модуль виконує роль провайдера власного інтерфейсу, що дозволяє швидко розгорнути даний застосунок на будь-якому веб-сервері, замінивши керування через командний рядок контроллером для створення власної обгортки.

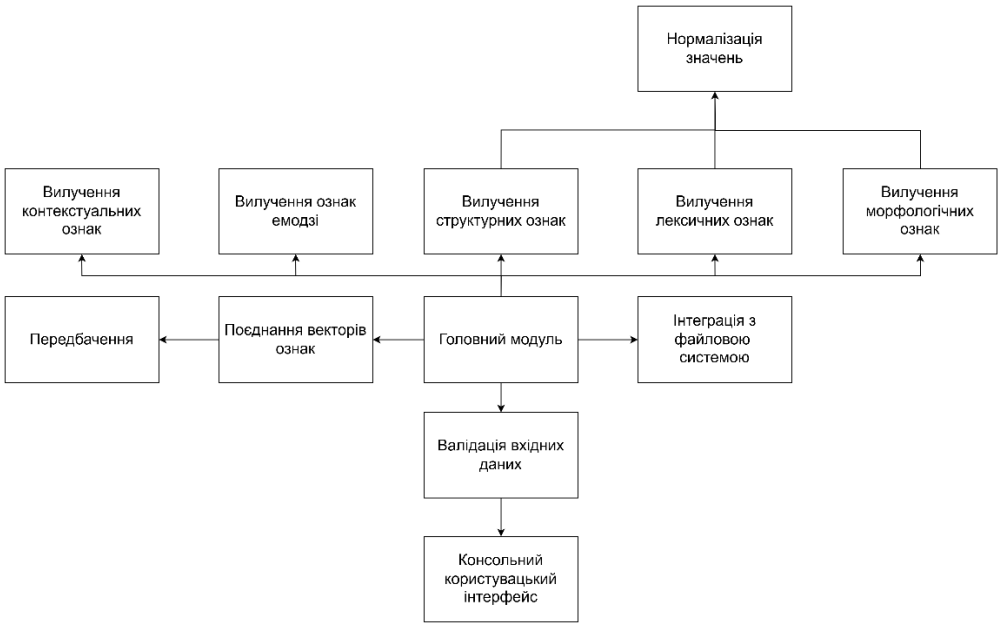


Рис. 6. Архітектура програмного забезпечення автоматичної класифікації природномовних текстових даних за наявністю агресії

Розглянемо модулі, що реалізують логіку створеного програмного забезпечення:

1. Console user interface – модуль, що надає користувачу можливість керувати застосунком за допомогою командного рядка. Його логіка передбачає декілька сценаріїв використання з можливістю почати спочатку без неочікуваного завершення процесу у випадку успішної класифікації чи виникнення неочікуваних помилок.
2. Input data validation – модуль, що забезпечує стабільну роботу застосунку з точки зору введених користувачем даних. З огляду на те, що користувач може за будь-яких причин передати системі майже будь-яку інформацію, то важливо їх валідувати, щоб основний процес оброки тексту не здійснювався над даними неочікуваного формату. Наявність даного модулю гарантує, що результат передбачення буде ґрунтуватися на конкретних особливостях тексту, а не буде випадковим.
3. Main – кореневий модуль, що відповідає за логіку викликів усіх інших модулів та передачу даних між ними. Саме він формує програмний API, що використовується Console user interface. Тому для користування даною програмною реалізацією запропонованого методу достатньо використовувати лише ті функції, що експортуються з Main.
4. Contextual features extraction – модуль, що відповідає за використання механізму векторизації моделі ALBERT. Він приймає перевірений текст та повертає вектор, що в подальшому процесі буде поєднаний іншими. Робота даного модуля займає найбільшу кількість часу, що обумовлено процесом глибокого аналізу вхідних текстових даних.
5. Emoji features extraction – модуль, завдання якого полягає у пошуку емодзі з тексту та їх векторизації за допомогою Emoji2vec моделі. Необхідно зазначити, що на вхід моделі Emoji2vec подаються лише

емодзі. Результатом роботи модулю є вектор, що описує ступінь впливу знайдених емодзі на загальний сенс оброблюваного тексту.

6. Structural features extraction – модуль, що відповідає за визначення таких структурних ознак тексту, як слова у верхньому регістрі, повторювальні знаки оклику та питання. Він повертає вектор із трьох значень, одне з яких проходить етап нормалізації.
7. Lexical features extraction – модуль, що обчислює кількість вживаних слів із попередньо зібраних словників із словами-маркерами агресивної поведінки. Необхідно зазначити, що даний модуль потребує нормалізації значень, щоб вони були в межах від 0 до 1.
8. Morphological features extraction – схожий на попередній модуль. При цьому його задача – визначити морфологічні особливості тексту, а саме: кількість вживаних дієслів та прикметників. Також потребує нормалізації значень.
9. Values normalization – модуль, що експортує лише одну функцію-утиліту. Її завданням є проведення відносної нормалізації. Приймає на вхід 2 числових значення та повертає їх відношення.
10. Filesystem interaction – модуль, який відповідає за взаємодію із файловою системою. Так як розроблений застосунок має оперувати файлом з вагами натренованого класифікатора а також словниками для визначення лексичних особливостей тексту, то їх необхідно зберігати та забезпечувати доступ для їх читання.
11. Features combining – модуль, що виконує підготовчу перед класифікацією роль. Його задачею є поєднати усі сформовані вектори в один та провести його мінімізацію, тобто зменшення розміру для подальшого подання у класифікатор.
12. Prediction – модуль, що забезпечує роботу класифікатора. Він викликає попередньо натреновану нейронну мережу для здійснення передбачень для вхідних текстів. По завершенню

процесу класифікації результат проходить етап перетворення у зручний користувача вигляд та повертає його у Main для подальшого виводу на екран.

3.3. Особливості реалізації ПЗ

Оскільки основною метою даного дослідження є максимізація точності автоматичної класифікації природномовних текстових даних за наявністю агресії, існує необхідність у використанні способів, технік та інструментів, які дозволяють якісно і повноцінно зібрати, обробити та проаналізувати вагомі властивості вхідних текстів для передбачення результату. Нижче детально розглянуто особливості реалізації програмного забезпечення для різних етапів роботи методу.

3.3.1. Особливості валідації вхідних даних

Основним завданням валідації вхідних даних є забезпечення того, що у класифікатор будуть надходити лише відповідні певним правилам тексти. Ці обмеження визначені на основі параметрів, що були передані методу токенізації ALBERT (лістинг 5).

Лістинг 5. Функція ALBERT токенізації

```
def extractContextFeatures(text):
    return   tokenizer(text,   truncation=True,   padding="max_length",
max_length=256)
```

Так, результатом роботи даної функції буде токен розміром в 256 елементів. Необхідно зазначити, що у зв’язку з тим, що тренування нейронної мережі було здійснено на прикладах англійською мовою, то для того, щоб гарантувати, що вхідний текст не буде обрізаний в процесі токенізації, було визначено, що максимальним розміром вхідного тексту є 160 слів. При цьому, варто згадати, що спроба передбачити оцінку ступеню агресії тексту, що складається з 1, 2 чи 3 слів не є доцільною через малу

кількість даних для аналізу. Саме тому в рамках валідації вхідних даних було задано такі допустимі рамки їх розміру.

Як було згадано раніше, створений класифікатор не може якісно визначати ступінь агресії для усіх мов, окрім англійської. Саме тому було додано перевірку мови вхідного тексту (лістинг 6).

Лістинг 6. Перевірка мови вхідного тексту

```
from langdetect import detect
language = detect(text)
```

При цьому, застосунок має бути зрозумілим будь-якого користувача. Через це було реалізовано можливість пояснювати причину, за якою певний текст не було допущено до подальшого оброблення. В результаті роботи валідації вхідних даних формується 2 об’єкти:

- масив перевірених текстів;
- масив кортежів, де перший елемент є текстом, що не пройшов валідацію, а другий – текстовий опис причини.

3.3.2. Особливості консольного користувацького інтерфейсу

Для забезпечення кращого користувацького досвіду було запроваджено 2 варіанти користування застосунком:

- введення текстів вручну. Так, можна вводити нескінченну кількість текстів для їх подальшої оцінки. Для цього при запуску програми користувач має ввести опції -t або --text. Після цього інтерфейс надасть інструкції щодо подальших кроків;
- зчитування текстів з файлу. Для використання даної можливості необхідно ввести опцію -f або --file та поруч шлях до цільового файлу. Після цього застосунок перевіряє, чи існує він у файловій системі та чи можна його прочитати. При виконанні обох вимог починається процес класифікації, інакше – завершення програми із вийнятком.

Також вкрай важливим аргументом при запуску програми є порогова оцінка ступеня агресії. Її роль полягає у розмежуванні випадків, коли агресія наявна у тексті і коли її немає. Дана оцінка приймає значення від 0 до 1 і є суб'єктивною для кожного окремого користувача. При цьому, передбачена перевірка, що було введено вірне з точки зору обмежень число. Застосунок завершить свою роботу з вийнятком, якщо умови не були задовільнені.

Лістинг 7. Функція ініціалізації аргументів командного рядка

```
import argparse

def validate_number(value):
  try:
    number = float(value)
    if 0 <= number <= 1:
      return number
    else:
      raise argparse.ArgumentTypeError(INVALID_LIMIT_SCORE)
  except ValueError:
    raise argparse.ArgumentTypeError(NAN_TYPE)

parser = argparse.ArgumentParser()
parser.add_argument('-t', '--text', action='store_true')
parser.add_argument('-f', '--file', type=str)
parser.add_argument('-n', '--number', type=validate_number)
```

3.3.3. Особливості поєднання ознак текстових даних

Процес класифікації за допомогою власної моделі нейронної мережі, заснованої на класі AlbertForSequenceClassification, може займати відносно великий період часу. При цьому, існує можливість як тренування моделі, так і її використання із вхідними даними у вигляді обмеженого набору, замість розгляду елементів почергово. Такий підхід дозволяє значного економити час та може позитивно впливати на точність роботи. Саме тому в процесі поєднання усіх ознак тестових даних створюється об'єкт, де значення кожного поля приймає значення типу torch.stack (лістинг 8).

Лістинг 8. Код для поєднання ознак текстових даних

```
textsFeatures = []
  for i, text in enumerate(validatedTexts, 1):
    bareText = remove_emojis(text)
    emojis = extract_emojis(text)
```

Продовження лістингу 8

```
    features = extractContextFeatures(bareText)
    structuralFeatures = extractStructuralFeatures(bareText)
    emojiFeatures = extractEmojiFeatures(emojis)
    features["emoji_embedings"] = emojiFeatures
    features["additional_features"] = additionalFeatures
    textsFeatures.append(features)
batch = {
    'input_ids': torch.stack([torch.tensor(item['input_ids']) for item in
features]),
     'attention_mask':  torch.stack([torch.tensor(item['attention_mask'])
for item in features]),
     'token_type_ids':  torch.stack([torch.tensor(item['token_type_ids'])
for item in features]),
    'emoji_embedings': torch.stack([torch.tensor(item['emoji_embedings'])
for item in features]),
                                                    'additional_features':
torch.stack([torch.tensor(item['additional_features'])    for    item    in
features])
  }
  return batch
```

Таким чином на вхід класифікатору подається єдиний об’єкт, що в процесі передбачення розділяється на партії до 16 елементів.

3.3.4. Особливості обмеження генерації випадкових значень

Через факт того, що на всіх етапах роботи класифікатор опирається на випадкові значення, то, відповідно, існує велика вірогідність того, що при різних запусках на одних і тих самих вхідних даних, класифікатор буде передбачувати завжди різні значення.

Для того, щоб уникнути цієї проблеми, застосована фіксація вихідного значення генератора випадкових чисел для всіх джерел випадковості Python API, Numpy, PyTorch.

Лістинг 9. Приклад фіксації вихідного значення генератора випадкових чисел

```
SEED = 42
random.seed(SEED)
np.random.seed(SEED)
torch.manual_seed(SEED)
torch.cuda.manual_seed_all(SEED)
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = False
```

3.4. Приклади результатів роботи ПЗ

В даному підрозділі наведено приклади результатів роботи розробленого застосунку з елементами виведення відповідних помилок при подачі некоректних вхідних даних.

При запуску консольного застосунку необхідно задати порогову оцінку визначення агресії, джерело до файлу із цільовими прикладами або самі тексти.

При використанні опції з файлом користувач зможе автоматично отримати результати передбачення. У консоль виводиться порядковий номер тексту, його перші 20 символів, передбачена оцінка, що приймає значення від 0 до 1 та результат бінарної класифікації (рис. 7).

```
PS C:\Users\Dmytro\Desktop\diplomaDocumentation\app\method> python .\main.py -f .\texts.txt -n 0.29
1. Question: These 4 br... 0.6374067068099976 вміщає агресію
```

Рис. 7. Результат передбачення для приклада із файлу

Як було згадано раніше, кількість прикладів програмно не обмежується. Тому результати класифікації текстів за наявністю агресії можна виконати одночасно для декількох прикладів (рис. 8).

```
PS C:\Users\Dmytro\Desktop\diplomaDocumentation\app\method> python .\main.py -f .\texts.txt -n 0.29
1. Question: These 4 br... 0.6374067068099976 вміщає агресію
2. Many of the families... 0.32954108715057373 вміщає агресію
3. Hero Rohit Sharma lo... 0.261223167181015 не вміщає агресію
4. i love you my bunny.... 0.2823463976383209 не вміщає агресію
```

Рис. 8. Результат передбачення для декількох прикладів із файлу

Зміна порогової оцінки, що вказується після опції -n впливає на результат бінарної класифікації, що відображається з правого краю. На рисунку нижче зображено результат впливу підвищеної оцінки (рис 9).

```
PS C:\Users\Dmytro\Desktop\diplomaDocumentation\app\method> python .\main.py -f .\texts.txt -n 0.33
1. Question: These 4 br... 0.6374067068099976 вміщає агресію
2. Many of the families... 0.32954108715057373 не вміщає агресію
3. Hero Rohit Sharma lo... 0.261223167181015 не вміщає агресію
4. i love you my bunny.... 0.2823463976383209 не вміщає агресію
```

Рис. 9. Результат передбачення для декількох прикладів із зміненою пороговою оцінкою

Як було вказано раніше, якщо текст не підпадає під встановлені обмеження, то після класифікації усіх інших текстів, з'явиться причина, чому перший не був оцінений.

Для цього було підготовано текст, що був написаний на українській мові а також такий, що складається лише з двох слів (рис. 10).

```
PS C:\Users\Dmytro\Desktop\diplomaDocumentation\app\method> python .\main.py -f .\texts.txt -n 0.33
1. Question: These 4 br... 0.6374067068099976 вміщає агресію
2. Many of the families... 0.32954108715057373 не вміщає агресію
3. Hero Rohit Sharma lo... 0.261223167181015 не вміщає агресію
4. i love you my bunny.... 0.2823463976383209 не вміщає агресію
5. Це нормальний текст ... Неправильна мова. Очікується англійська, знайдено: uk.
6. Small text... Неправильна довжина. Текст повинен містити від 5 до 100 слів.
```

Рис. 10. Результат передбачення з виводом помилок для текстів, що не пройшли валідацію

Якщо користувач вирішив вводити текст напряму у консоль, то він може використати відповідну опцію. Після запуску застосунок відобразить повідомлення про дії, що очікуються від користувача. Так, він може вводити тексти без обмеження по їх кількості. А для їх оцінки натиснути клавішу Enter 2 рази (рис. 11).

```
PS C:\Users\Dmytro\Desktop\diplomaDocumentation\app\method> python .\main.py -t -n 0.33
Введіть текст (від 3 до 160 слів). Для завершення введення тексту натисніть Enter двічі:
Question: These 4 broads who criticize America, what country did they flee to get here? And now they want to make OUR America like
OUR country.
Many of the families who are being separated at the border and who are being kept in poor conditions did try to cross legally, thr
Hero Rohit Sharma love From Pakistan

1. Question: These 4 br... 0.6374067068099976 вміщає агресію
2. Many of the families... 0.32954108715057373 не вміщає агресію
3. Hero Rohit Sharma lo... 0.2612231373786926 не вміщає агресію
```

Рис. 11. Результат передбачення для декількох прикладів, що були введені у консоль

Якщо користувач не введе порогову оцінку, або вона не прийматиме значення від 0 до 1, то програма завершиться із відповідним вийнятком (рис. 12).

```
PS C:\Users\Dmytro\Desktop\diplomaDocumentation\app\method> python .\main.py -f .\texts.txt -n
usage: main.py [-h] [-t] [-f FILE] [-n NUMBER]
main.py: error: argument -n/--number: expected one argument
```

Рис. 12. Помилка при виклику застосунку без вказаної порогової оцінки

3.5. Висновки до третього розділу

У даному розділі обґрунтовано та детально розглянуто засоби розроблення, що були основними та вагомими для створення програмної реалізації запропонованого методу автоматичної класифікації природномовних текстових даних за наявністю агресії. До них відносяться: мова програмування Python, бібліотека Transformers, фреймворк PyTorch, бібліотека Scikit-learn, бібліотека Pandas, бібліотека NLTK, бібліотека Spacy.

Описано архітектуру розробленого програмного забезпечення, наведено її схему та детально розглянуто кожен модуль системи.

Виділено особливості та техніки, що були використані під час процесу реалізації застосунку. Серед них можна виділити наступні:

- сувора валідація вхідних даних для гарантування точності результатів класифікації;
- форматування векторів із вагомими особливостями текстових даних у зручний для класифікатора об’єкт для його розбиття на батчі, що позитивно впливає на витрати часу для передбачення;
- суворо обмежена генерація випадкових величин для забезпечення сталих результатів класифікації для одних і тих самих текстів.

Описано та продемонстровано варіанти використання консольного програмного застосунку в умовах мануального введення текстів та подання шляху до файлу з ними.

4. АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ЗАПРОНОВАНОГО МЕТОДУ

Розроблений метод автоматичної класифікації природномовних текстових даних за наявністю агресії є ансамблем відомих ефективних підходів до вирішення подібних до даної задач. В процесі розроблення схеми роботи методу, здійснення аналізу можливостей для поліпшення його точності та надійності було проведено декілька практичних дослідів. Кожен з них здійснювався на основі різних архітектурних рішень. Метою такого підходу було:

- виявлення дійсно ефективного для даного випадку варіанта методу;
- відслідковування динаміки зміни значень метрик, щоб розуміти, як впливає на результат нова компонента.

Необхідно зазначити, що запропонований метод має за мету максимізувати точність класифікації. Саме тому даний розділ присвячений аналізу показників метрик, що були обрані у другому, для кожного з варіантів реалізації.

Розглянемо типи розроблених в рамках даного дослідження методів:

- ALBERT. Базовий метод, що базується на використанні існуючого класу AlbertForSequenceClassification з бібліотеки Transformers. Враховує контекстні залежності тексту відповідно до характеристик відповідної моделі глибокого навчання.
- ALBERT + Emoji. Розширення базового методу шляхом визначення контекстних ознак емодзі за допомогою моделі Emoji2vec.
- ALBERT + Emoji + BILSTM. Використання механізму уваги BILSTM для визначення ступеня взаємодії між словами та емодзі. Механізм уваги реалізовувався за допомогою матриці зв’язків кожного слова та кожного емодзі та LSTM моделі.

- ALBERT + Emoji + extra features. Поєднання результатів роботи векторизації ALBERT та Emoji2vec було розширено за допомогою додаткових ознак текстових даних, що були обчислені самостійно.
- ALBERT + Emoji + extra features + linear layer. Модель, з основою попереднього варіанта, що передбачав зменшення розміру об'єднаного вектору за допомогою додаткового лінійного шару для подальшого подання у класифікатор.
- Extended ALBERT + Emoji + extra features + linear layer. Фінальний варіант, в рамках якого було використано здатність ALBERT до вилучення векторів із усіх 13 шарів та конкатенації 4 останніх. Необхідно зазначити, що використання такого інженерного рішення вимагає значного збільшення обсягів обчислювальних ресурсів. Автоматичне обчислення обраних метрик в процесі навчання неможливе на базі Google Colab, адже він не здатен забезпечити достатнім об'ємом відео пам'яті. При цьому, він надає можливість натренувати таку модель та експортувати фінальний класифікатор. Для оцінки його роботи було використано мануальне тестування шляхом надсилання на вхід твітів з подальшим порівнянням результатів передбачення та фактичних оцінок.

4.1. Підготовка до проведення оцінювання роботи методів

Очевидно, що для проведення ефективного аналізу точності роботи вищезгаданих методів необхідно натренувати відповідні моделі класифікаторів. Для цього було завантажено набір даних measuring-hate-speech, що вміщає близько 40 тисяч унікальних твітів із проставленими оцінками ступеню агресії. З точки зору об'єктивності даних оцінок, то вони були визначені групою з 7912 науковців.

Для проведення тренування класифікаторів набір даних було розмежовано на тренувальну та валідаційну вибірки у співвідношенні 7:3 (лістинг 10).

Лістинг 10. Розділення набору даних на тренувальну на валідаційну вибірки

```
train_df, evaluate_df = train_test_split(
      df, test_size=0.3, random_state=42
      )
```

Загальний обсяг валідаційної вибірки сягав 12000 твітів.

Тренування проводилося за стратегією епох з наступними параметрами:

- темп навчання – 2*e*-5;
- розмір батчу – 32;
- кількість епох – 5.

Лістинг 11. Код для створення об’єкту для тренування

```
from transformers import TrainingArguments


BASE_MODEL = 'albert-base-v2'
LEARNING_RATE = 2e-5
MAX_LENGTH = 256
BATCH_SIZE = 32
EPOCHS = 5


training_args = TrainingArguments(
    output_dir="./results",
    learning_rate=LEARNING_RATE,
    per_device_train_batch_size=BATCH_SIZE,
    per_device_eval_batch_size=BATCH_SIZE,
    num_train_epochs=EPOCHS,
    evaluation_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
    weight_decay=0.01,
)
```

У кінці кожної з епох автоматично викликалася функція обчислення значень метрик (лістинг 12).

Лістинг 12. Функція обчислення значень метрик

```
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

Продовження лістингу 12

```
def compute_metrics_for_regression(eval_pred):
    logits, labels = eval_pred
    labels = labels.reshape(-1, 1)

    mse = mean_squared_error(labels, logits)
    mae = mean_absolute_error(labels, logits)
    r2 = r2_score(labels, logits)
    single_squared_errors = ((logits - labels).flatten()**2).tolist()
     accuracy = sum([1 for e in single_squared_errors if e < 0.0025]) /
len(single_squared_errors)

    return {"mse": mse, "mae": mae, "r2": r2, "accuracy": accuracy}
```

З огляду на те, що механізм глибокого навчання потребує значних обчислювальних ресурсів навіть при використанні ALBERT, навчання та валідація моделей були проведені на платформі Google Colab за допомогою серверного прискорювача Python 3 на базі GPU.

Розглянемо апаратні характеристики, що були використані під час навчання:

- апаратний прискорювач NVIDIA V100 з об'ємом пам'яті у 16 ГБ;
- об’єм оперативної пам’яті 51 ГБ.

Необхідно зазначити, що для задачі регресії, що була взята за основу для кожного з методів поняття метрики Accuracy не є чітко визначеним. В рамках даної роботи було визначено, що передбачення класифікатора вважається достовірним, якщо квадрат похибки менший за 0.0025. Таке значення було визначено експериментально під час цього дослідження.

4.2. Оцінювання ефективності роботи методу

Для проведення глибокого аналізу значень усіх метрик, розглянемо кожну з них окремо.

4.2.1. Порівняння значень метрики Loss

В процесі оцінювання згаданих вище первісних запропонованого методу у кінці кожної з 5 епох обчислювалася метрика Loss. Фактично, вона є показником ступеню роботи моделі для надання зворотнього зв’язку

оптимізатору і визначає похибку між прогнозованим і фактичним результатом для кожного конкретного прикладу.

Поступове зменшення значення Loss протягом декількох епох свідчить про те, що модель успішно мінімізує помилку.

Динаміка зміни значень даної метрики дозволяє визначити, наскільки оптимальним з точки зору узагальнення даних є вибір тієї чи іншої варіації методу.

Варто зазначити, що у даному випадку значення Loss вираховувалося за допомогою функції MSE. Фактично, обидві метрики вимірюють середнє значення квадратів різниць між прогнозованими значеннями моделі та фактичними значеннями. Тому вони дозволяють оцінити, наскільки точно модель прогнозує значення.

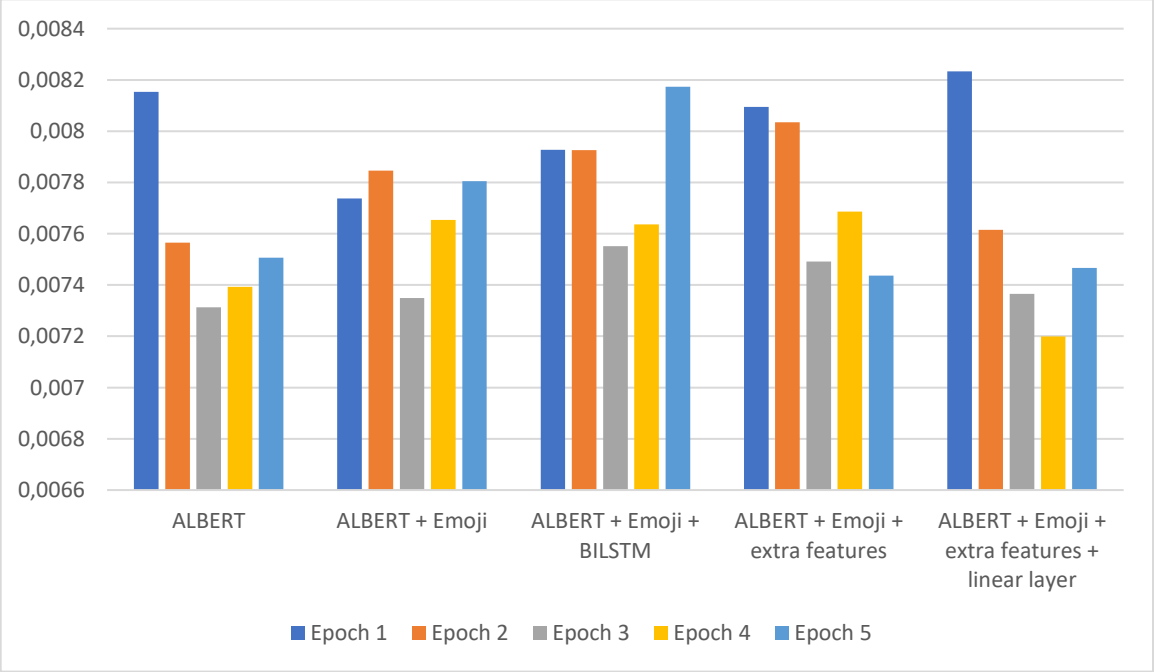


Рис. 13. Гістограма порівняння значень метрики Loss

Результати вимірювань свідчать про те, що моделі ALBERT, ALBERT + Emoji + extra features, ALBERT + Emoji + extra features + linear layer мають кращі здатності для узагальнення даних, ніж інші. При цьому,

для перших двох прослідковується тенденція підвищення значення loss після третьої епохи.

Варіація ALBERT + Emoji + BILSTM після повного циклу навчання демонструє найвищий показник loss. Так, у порівнянні зі звичайним ALBERT відбувся приріст приблизно на 8,87%, що є значним регресом, адже очікувалось зниження даного значення.

Що стосується моделей ALBERT + Emoji, ALBERT + Emoji + extra features, то вони в середньому отримали гірші оцінки loss, ніж базова. При цьому, врахування додаткових ознак текстових даних позитивно вплинуло на динаміку якості навчання. Так, після 2 епохи спостерігається значне зменшення оцінки.

В даному контексті остання модель, а саме, ALBERT + Emoji + extra features + linear layer демонструє дещо кращі результати та при цьому менш здатна до перенавчання. Так як вона є похідною від 4 за порядком моделі, то можна зробити висновок, що при навчанні на 6 та більше епохах вона здатна значно покращити значення метрики Loss та MSE. Необхідно зазначити, що їх значення в кінці 1 епохи було найбільше у порівнянні з іншими моделями.

4.2.2. Порівняння значень метрики MAE

MAE метрика є однією з найбільш значущих для задач регресії. Вона вимірює середнє значення абсолютних різниць між прогнозованими значеннями моделі та фактичними значеннями. Очевидно, що в рамках даного дослідження метою є мінімізація таких метрик, як MAE.

Розглянемо її фактичні значення для кожної моделі протягом 5 навчальних епох (рис. 14).

Очевидно, що тренди MAE, MSE та Loss майже збігаються. При цьому моделі ALBERT + Emoji та ALBERT + Emoji + extra features демонструють погіршення результатів одразу в кінці 2 епохи. Але вже після наступної ітерації поєднання інформації про емодзі та контексту демонструють значу позитивну динаміку, що перевершує оцінки методу-нащадка.

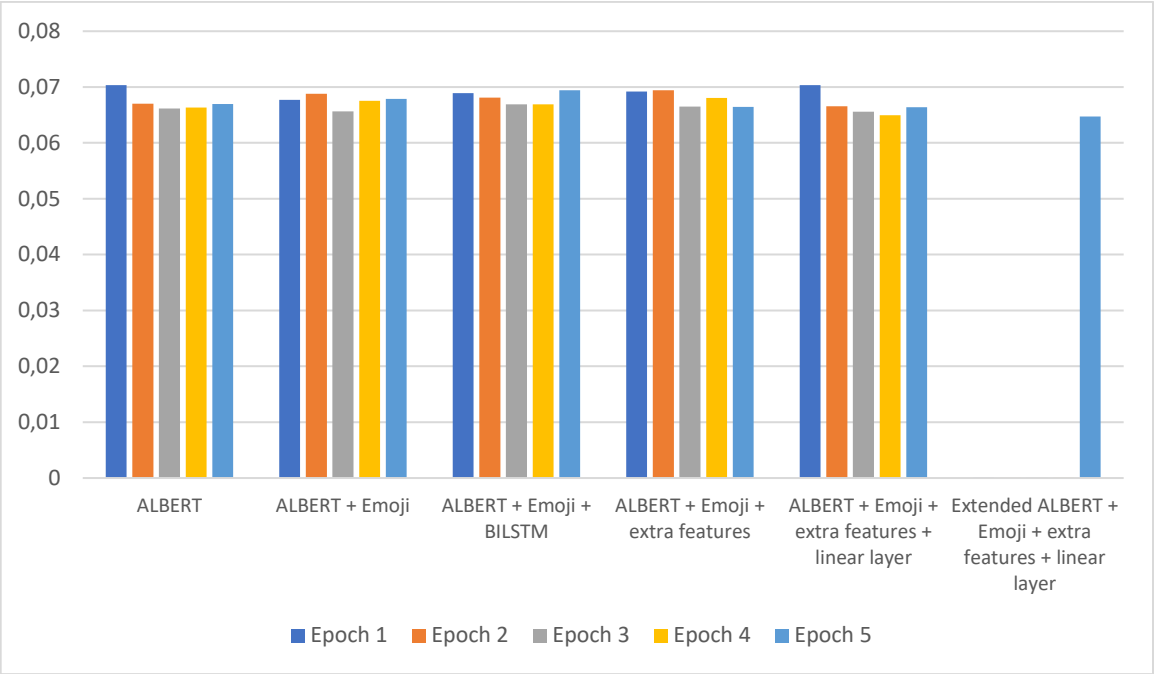


Рис. 14. Гістограма порівняння значень метрики MAE

Базовий метод ALBERT та його п’ята похідна отримали в процесі валідації значно ліпші результати з перевагою на користь останньої.

При порівнянні показників метрик MAE моделі ALBERT + Emoji + extra features + linear layer та базової на останньому циклі можна побачити приріст на 0,84%.

При цьому, в кінці 4 епохи, що є найкращою з точки зору результату для останньої моделі приріст сягає 2,11%.

Можна зробити висновок: поєднання контекстної інформації природномовних текстових даних, числової характеристики емодзі та додаткових структурних, морфологічних та лексичних особливостей тексту дійсно позитивно вливає як на процес навчання, так і на фінальний результат.

Значення MAE, яке вдалось обчислити мануально для запропонованого методу, що використовує розширені можливості ALBERT, після 5 епох навчання приймає найнижчі значення серед усіх. У порівнянні з базовим ALBERT приріст значення даної метрики сягає 3,46%.

4.2.3. Порівняння значень метрики R^2

Наступною вагомою метрикою є R^2, коефіцієнт детермінації. Він показує ступінь пояснення моделлю варіативності залежної змінної. Дана оцінка використовується для визначення якості регресійних моделей.

Чим ближче значення R^2 до 1, тим краще. При цьому, якщо відбувається екстремально велике наближення до даної граничної оцінки, то це можна вважати індикатором перенавчання.

Розглянемо обчислені значення R^2 протягом 5 епох.

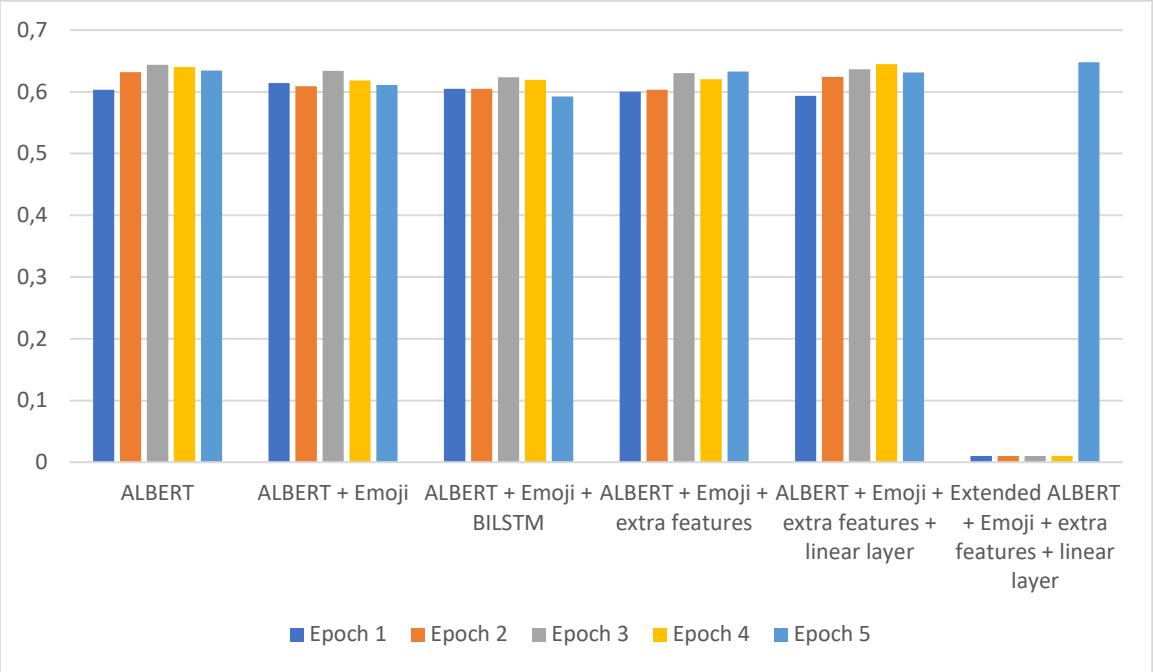


Рис. 15. Гістограма порівняння значень метрики R^2

Можна зробити висновок, що відображені на гістограмі (рис. 3) дані за трендом відрізняються від попередніх метрик. Так, можна побачити, що найнижче значення в кінці 1 епохи має модель моделі ALBERT + Emoji + extra features + linear layer. При цьому, вона демонструє стабільне приближення до граничного значення 1 і має втрати лише в самому кінці, що може свідчите про початок перенавчання.

ALBERT, в свою чергу, набуває найкращого результату вже в кінці 3 епохи і надалі значення погіршуються.

Варто зазначити, шо модель ALBERT + Emoji + extra features демонструє найкращу динаміку змін. Так, після 3 епохи відбулося незначне погіршення значення R^2, але при цьому вже на 5 епосі вона набуває найкращих результатів.

Моделі ALBERT + Emoji а також ALBERT + Emoji + BILSTM демонструють найгірші результати. Судячи з результату 5 епохи та загальної тенденції змін, вони не є підходящими для використання в рамках методу автоматичної класифікації природномовних текстових даних за наявністю агресії.

Що стосується найбільших значень, то протягом усіх 5 епох саме остання модель набуває найкращого R^2 серед усіх наявних варіантів. При порівнянні її з базовою моделлю було виявлено приріст приблизно у 2,07%.

4.2.4. Порівняння значень метрики Accuracy

У випадку розв'язання задачі регресії метрика Accuracy не є основною та однозначно визначеною. Вона застосовується для оцінки якості роботи моделей зазвичай для мультикласової класифікації. Як було вказано у другому розділі, її роль у даному дослідженні – визначити точність регресійної моделі. В якості порогового значення було обрано 0,0025, оскільки параметр, що має бути визначений, ступінь вмісту агресії, обмежений від 0 до 1.

Отже, результат передбачення моделі вважається достовірним, якщо різниця між ним та фактичним значенням оцінки ступеню агресії не перевищує 0,05.

Параметр для обчислення значення даної метрики був визначений в процесі дослідження. Саме тому Accuracy є найбільш вагомою оцінкою в рамках проведення аналізу отриманих результатів.

Розглянемо результати тестування розроблених моделей на 5 епохах.

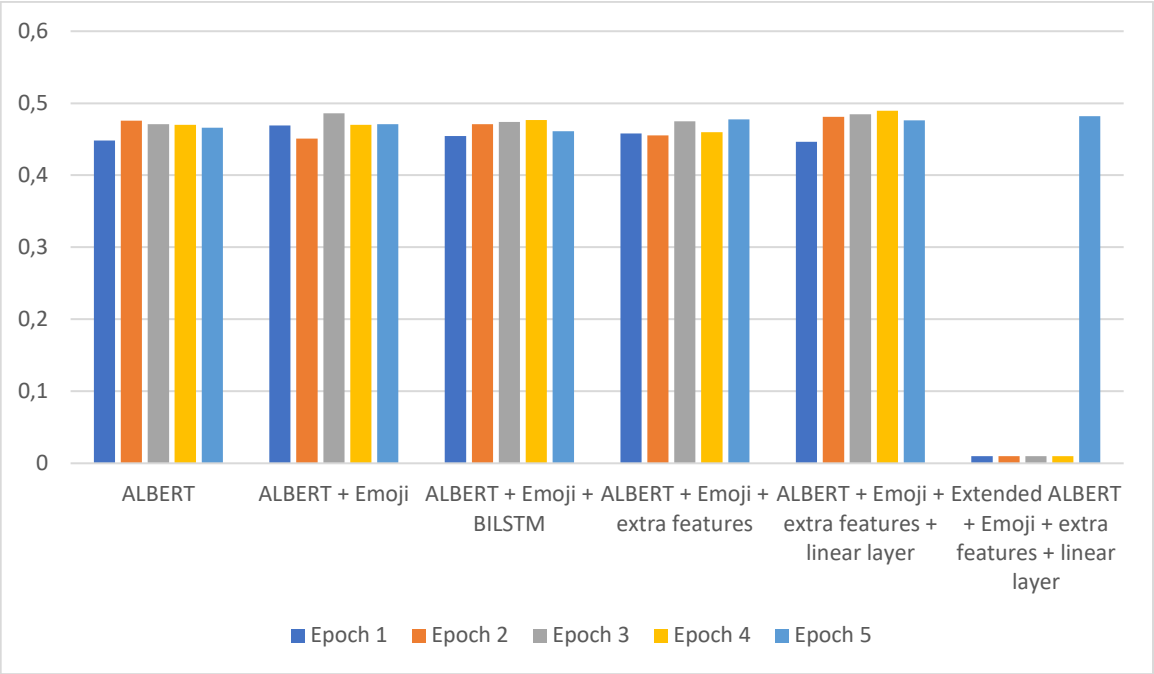


Рис. 16. Гістограма порівняння значень метрики Accuracy

Очевидно, що моделі на основі ALBERT + Emoji + extra features показують значно кращі результати, ніж базовий ALBERT. При цьому, варіант із додаванням додаткового лінійного шару перед використанням класифікатора для зменшення вхідного розміру демонструє найвищі значення в середньому і при цьому найбільшу стабільність та динаміку.

Відмітимо, Accuracy для моделі ALBERT + Emoji в кінці 2 епохи приймає майже найбільше значення серед усіх. Але надалі спостерігається погіршення.

Можна зазначити, що модель на основі ALBERT поступово втрачає можливість до гнучкого налаштування та стає більш грубою. При цьому її негативна динаміка є сталою але не стрімкою.

Порівнюючи первісну модель ALBERT з її найкращою похідною extended ALBERT + Emoji + extra features + linear layer, можна визначити, що після 5 епох навчання відбувся приріст приблизно на 3,9%.

4.3. Результати аналізу метрик

В результаті проведеного аналізу обраних метрик було виявлено досить чіткий паттерн поведінки різних моделей під час навчання протягом 5 епох.

Видно, що використання механізму уваги BILSTM не дало позитивного результату. Навпаки, майже по всім розглянутим метрикам цей варіант моделі є найгіршим. Під час даного дослідження було проведено аналіз причини такого незадовільного результату. Основною причиною невдачі було визначено саме навчальний набір даних, його особливості. Так, механізм уваги BILSTM, що реалізовувався у вигляді матриці взаємодії кожного слова та кожного емодзі був дуже чутливим до відсутності емодзі. Визначено, що набір даних вміщав всього до 10% твітів, що включали у себе емодзі. При цьому, процес розробленої векторизації передбачав заповнення векторів нулями замість невистачаючої інформації.

Таким чином, процес передбачення даною моделлю для 9 з 10 прикладів був ускладнений саме використанням нульової матриці.

Що стосується загального результату дослідження, то чітко прослідковується динаміка розвитку методу під час його розширення додатковими параметрами. При цьому, було застосовано механізм вилучення результатів роботи з усіх 12 шарів ALBERT та поєднано 4 останніх для подальшого використання при класифікації.

Через обмеження кількості доступних обчислювальних ресурсів на платформі Google Colab обчислювати значення метрик для останньої версії моделі в кінці кожної епохи було не можливо. При цьому, по закінченню процесу навчання її було власноруч протестовано та оцінено якість роботи.

Таким чином, остання варіація, що передбачала врахування контекстної інформації, характеристики емодзі, структурних, лексичних, морфологічних параметрів, зменшення загального розміру вхідного у класифікатор вектору та поєднання результуючих векторів з 4 останніх

шарів моделі ALBERT, показала приріст по більшості метрикам відносно базового методу на 2-4%.

4.4. Напрямки для подальшої роботи

Процес створення та покращення методу автоматичної класифікації природномовних текстових даних за наявністю агресії є комплексним та багатогранним. Одним із ключових та очевидних напрямків є оптимізація оброблення тексту за допомогою ALBERT. Оскільки використання інформації з усіх шарів моделі значно ускладнює класифікацію, то важливо правильно комбінувати їх. Один можливих із підходів – використання механізму уваги, що дозволило б моделі робити акцент на найважливіших аспектах тексту.

При цьому існують можливості для вдосконалення процесу оброблення емодзі. Замість базового використання моделі Emoji2vec можна створити власну на актуальному наборі даних та зменшити розмір вихідного вектору для того, щоб класифікатор звертав більше уваги на текстові особливості. Комбінування векторизації Emoji2vec та алгоритмів глибокого навчання, як CNN може допомогти краще аналізувати семантику емодзі.

Також є необхідність у врахуванні виявлених недоліків у механізмі уваги за допомогою BILSTM та внесенні відповідних виправлень. При використанні подібної адаптованої системи класифікатор може значно підвищити точність передбачень.

Очевидно, що основним недоліком розробленого методу є його вибагливість до кількості обчислювальних ресурсів. Тому у якості подальшої роботи доцільно звернути увагу на пошук більш оптимізованих за часом обрахунків рішень. Доцільно провести експерименти із такими моделями, як TinyBERT, MobileBERT, тощо.

Факт того, що запропонований метод може працювати лише з текстовими даними англійською мовою, ставить жорсткі обмеження на

вхідні дані. В рамках подальшої роботи планується натренувати класифікатор на прикладах інших розповсюджених мов.

Розроблений застосунок є консольним та доступний до використання лише після його завантаження. Такий підхід вимагає правильного встановлення великої кількості залежностей та не є зручним. Саме тому в рамках подальшої роботи планується створити web api, що буде надавати користувачам доступ до інтуїтивно-зрозумілих ендпоінтів.

З іншого боку, для користувачів, які бажають одноразово використати даний функціонал, необхідно розробити більш зручний для роботи графічний інтерфейс.

4.5. Висновки до четвертого розділу

У даному розділі описано загальний підхід до проведення аналізу отриманих результатів. Для цього було розроблено, натреновано протягом 5 епох та протестовано такі моделі:

- ALBERT;
- ALBERT + Emoji;
- ALBERT + Emoji + BILSTM;
- ALBERT + Emoji + extra features;
- ALBERT + Emoji + extra features + linear layer;
- Extended ALBERT + Emoji + extra features + linear layer.

Після проведення кожної ітерації навчання кожна модель, окрім останньої автоматично оцінювалася за метриками Loss, MSE, MAE, R^2, Accuracy.

Через вибагливість до надзвичайно великого об’єму обчислювальних ресурсів, які не надає Google Colab, модель Extended ALBERT + Emoji + extra features + linear layer було протестовано самостійно із її станом на кінець 5 епохи.

Проведено детальний аналіз результатів кожної із згаданих метрик, під час якого було виявлено схожі тенденції для одних і тих самих моделей. Варіація на основі використання механізму уваги BILSTM виявилася надлишковою та недоцільною в рамках даного дослідження. При цьому, чітко прослідковувалося поліпшення точності для моделей на основі ALBERT + Emoji + extra features. Остання, з яких продемонструвала приріст значень розглянутих метрик відносно базової моделі ALBERT на 2-4%.

Запропоновано подальші кроки до покращення та розвитку розробленого застосунку. Вони включали підвищення точності роботи, оптимізацію обчислень текстових параметрів, покращення користувацького досвіду.

ВИСНОВКИ

Дана дисертація присвячено покращенню оригінального метода автоматичної класифікації природномовних текстових даних за наявністю агресії на основі моделі глибокого навчання ALBERT шляхом використання розширеної кількості шарів у поєднанні з додатковими структурними, лексичними та морфологічними ознаками.

Під час проведення дослідження розглянуто актуальність проблеми впливу агресивних висловлювань на суспільство та їх автоматичного виявлення. Визначено, що поняття агресії не є чітко окресленим та може бути описано суб'єктивно. Для розроблення власного відповідного методу розглянуто ознаки природномовних текстових даних, що можуть бути вагомими під час їх класифікації за наявністю агресії. Наведено та проаналізовано існуючі методи і підходи для проведення сентимент-аналізу тексту та, зокрема, виявлення агресивних настроїв. Знайдено та описано популярні сервіси що надають можливість класифікації текстових даних за наявністю агресії онлайн.

Обґрунтовано обраний підхід на основі використання моделі глибокого навчання ALBERT, на базисі якого побудовано запропонований метод. Визначено потенційні проблеми, що не вирішує оригінальний метод. Запропоновано етапи роботи розроблюваного застосунку та наведено відповідне графічне представлення. Для проведення глибокого та різнобічного аналізу результатів роботи методу проаналізовано та обрано метрики, що дозволяють оцінити якість роботи вирішення задачі регресії.

Наведено основні засоби розроблення програмного застосунку та обґрунтовано їх ефективність в процесі розв'язання задачі автоматичної класифікації тексту за наявністю агресії. Описано архітектуру розробленого програмного забезпечення та її модулів. Виділено та надано перелік особливостей реалізації запропонованого методу. Продемонстровано

варіанти використання даного консольного застосунку на різних вхідних даних.

Проведено детальний аналіз точності роботи розроблених під час даного дослідження варіантів методу. Визначено, що підхід, який базується на використанні додаткових шарів ALBERT, інформації про емодзі, структурних, лексичних та морфологічних особливостей природномовних текстових даних, демонструє приріст точності роботи у порівнянні з базовим ALBERT на проміжку від 2% до 4% для різних метрик. Наведено напрямки для подальшої роботи над методом для його покращення з точки зору точності, швидкості та користувацького інтерфейсу.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Pete Burnap and Matthew L. Williams. 2015. Cyber hate speech on Twitter: An application of machine classification and statistical modeling for policy and decision making. Policy Internet 7, 2 (2015), 223–242.
2. Поняття агресії [Електронний ресурс]. — Режим доступу: https://www.rozmova.me/blog/agresia#:~:text=%D0%90%D0%B3%D1%80%D0%B5%D1%81%D1%96%D1%8F%20%D1%94%20%D1%88%D0%B8%D1%80%D1%88%D0%B8%D0%BC%20%D1%82%D0%B0%20%D0%B1%D1%96%D0%BB%D1%8C%D1%88,%D1%84%D1%96%D0%B7%D0%B8%D1%87%D0%BD%D1%96%D0%B9%2C%20%D0%B2%D0%B5%D1%80%D0%B1%D0%B0%D0%BB%D1%8C%D0%BD%D1%96%D0%B9%20%D1%87%D0%B8%20%D0%BF%D0%B0%D1%81%D0%B8%D0%B2%D0%BD%D1%96%D0%B9%20%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%85 — (01.05.2024).
3. Особливості прояву агресії в соціальних мережах [Електронний ресурс]. — Режим доступу: https://almanac.npu.kiev.ua/index.php/almanac/article/view/292/273 — (24.04.2024).
4. Вплив інформаційних технологій [Електронний ресурс]. — Режим доступу: https://essuir.sumdu.edu.ua/bitstream-download/123456789/52494/9/Boyko_computer_culture.pdf;jsessionid=CA602DD6AAA588B80938B6E4401CC52E — (15.07.2024).
5. Сучасна комунікація [Електронний ресурс]. — Режим доступу: https://dspace.nadpsu.edu.ua/bitstream/123456789/2822/1/%D0%A1%D1%83%D1%87%D0%B0%D1%81%D0%BD%D0%B0_%D0%BA%D0%BE%D0%BC%D1%83%D0%BD%D1%96%D0%BA_%D0%BD.pdf — (01.03.2024).
6. An experimental study on feature engineering and learning approaches for aggression detection in social media [Електронний ресурс]. — Режим

доступу: https://www.academia.edu/69783715/An_experimental_study_on_feature_engineering_and_learning_approaches_for_aggression_detection_in_social_media — (05.07.2024).
7. Hate speech detection: Challenges and solutions [Електронний ресурс]. — Режим доступу: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0221152 — (20.08.2024).
8. Hate speech detection [Електронний ресурс]. — Режим доступу: https://pmc.ncbi.nlm.nih.gov/articles/PMC6701757/ — (14.05.2024).
9. A Lexicon-based Approach for Hate Speech Detection [Електронний ресурс]. — Режим доступу: https://www.researchgate.net/publication/283125668_A_Lexicon-based_Approach_for_Hate_Speech_Detection — (01.04.2024).
10. WordNet [Електронний ресурс]. — Режим доступу: https://wordnet.princeton.edu/ — (03.04.2025).
11. TF-IDF [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/TF-IDF — (30.04.2024).
12. Word2Vec [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/Word2vec — (05.08.2024).
13. GloVe [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/GloVe — (05.09.2024).
14. Automated Hate Speech Detection and the Problem of Offensive Language [Електронний ресурс]. — Режим доступу: https://www.researchgate.net/publication/314942659_Automated_Hate_Speech_Detection_and_the_Problem_of_Offensive_Language — (20.04.2024).
15. The Porter stemming algorithm: Then and now [Електронний ресурс]. — Режим доступу:

https://www.researchgate.net/publication/33038304_The_Porter_stemming_algorithm_Then_and_now — (01.06.2024).
16. Flesch Reading Ease Score [Електронний ресурс]. — Режим доступу: https://rockcontent.com/blog/flesch-reading-ease-score/ — (13.06.2023).
17. Social Media Hate and Offensive Speech Detection Using Machine Learning Method [Електронний ресурс]. — Режим доступу: https://aclanthology.org/2024.dravidianlangtech-1.40.pdf — (22.03.2024).
18. Did you offend me? Classification of Offensive Tweets in Hinglish Language [Електронний ресурс]. — Режим доступу: https://www.researchgate.net/publication/334117654_Did_you_offend_me_Classification_of_Offensive_Tweets_in_Hinglish_Language — (05.04.2024).
19. Knowledge-Enriched Transformer for Emotion Detection in Textual Conversations [Електронний ресурс]. — Режим доступу: https://www.researchgate.net/publication/336018210_Knowledge-Enriched_Transformer_for_Emotion_Detection_in_Textual_Conversations — (01.09.2024).
20. ConceptNet [Електронний ресурс]. — Режим доступу: https://conceptnet.io/ — (05.04.2024).
21. NRC VAD Lexicon v2: Norms for Valence, Arousal, and Dominance for over 55k English Terms [Електронний ресурс]. — Режим доступу: https://doi.org/10.48550/arXiv.2503.23547 — (05.04.2024).
22. DailyDialog [Електронний ресурс]. — Режим доступу: https://paperswithcode.com/dataset/dailydialog — (05.04.2024).
23. MELD [Електронний ресурс]. — Режим доступу: https://paperswithcode.com/dataset/meld — (05.04.2024).
24. EmoryNLP [Електронний ресурс]. — Режим доступу: https://paperswithcode.com/dataset/emorynlp — (05.04.2024).
25. IEMOCAP [Електронний ресурс]. — Режим доступу: https://paperswithcode.com/dataset/iemocap — (05.04.2024).

26. BERT Explained: State of the art language model for NLP [Електронний ресурс]. — Режим доступу: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270 — (10.11.2024).
27. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [Електронний ресурс]. — Режим доступу: https://arxiv.org/pdf/1810.04805 — (24.05.2024).
28. The Illustrated BERT [Електронний ресурс]. — Режим доступу: https://jalammar.github.io/illustrated-bert/ — (01.10.2024).
29. HateBERT: Retraining BERT for Abusive Language Detection in English [Електронний ресурс]. — Режим доступу: https://arxiv.org/abs/2010.12472 — (05.04.2024).
30. BERT-base-uncased [Електронний ресурс]. — Режим доступу: https://huggingface.co/google-bert/bert-base-uncased — (05.04.2024).
31. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations [Електронний ресурс]. — Режим доступу: https://huggingface.co/docs/transformers/model_doc/albert — (05.04.2024).
32. ALBERT-large-v2 [Електронний ресурс]. — Режим доступу: https://huggingface.co/albert/albert-large-v2 — (05.04.2024).
33. BERT-large-uncased [Електронний ресурс]. — Режим доступу: https://huggingface.co/google-bert/bert-large-uncased — (05.04.2024).
34. Emotion Detection and Analysis on Social Media [Електронний ресурс]. — Режим доступу: https://arxiv.org/pdf/1901.08458 — (12.06.2024).
35. Stanford CoreNLP [Електронний ресурс]. — Режим доступу: https://stanfordnlp.github.io/CoreNLP/ — (05.04.2024).
36. Sequential Minimal Optimization for Support Vector Machines [Електронний ресурс]. — Режим доступу: https://bitmask93.github.io/ml-blog/Sequential-Minimal-Optimization-for-Support-Vector-Machines/ — (05.04.2024).

37. Using machine learning to reduce toxicity online [Електронний ресурс]. — Режим доступу: https://perspectiveapi.com/ — (01.01.2024).
38. Content Moderator: Text Moderation API (Microsoft) [Електронний ресурс]. — Режим доступу: https://learn.microsoft.com/ru-ru/rest/api/cognitiveservices/contentmoderator/text-moderation?view=rest-cognitiveservices-contentmoderator-v1.0 — (05.04.2024).
39. MonkeyLearn Help Center [Електронний ресурс]. — Режим доступу: https://help.monkeylearn.com/en/ — (05.04.2024).
40. TextBlob Documentation [Електронний ресурс]. — Режим доступу: https://textblob.readthedocs.io/en/dev/ — (05.04.2024).
41. SymSpellPy Documentation [Електронний ресурс]. — Режим доступу: https://symspellpy.readthedocs.io/en/latest — (05.04.2024).
42. Language-tool-python Documentation [Електронний ресурс]. — Режим доступу: https://pypi.org/project/language-tool-python — (05.04.2024).
43. Hurtlex: A Lexicon of Offensive Language [Електронний ресурс]. — Режим доступу: https://vocabularyserver.com/hurtlex/en/ — (05.04.2024).
44. Emoji-based Fine-grained Attention Network for Sentiment Analysis in the Microblog Comments [Електронний ресурс]. — Режим доступу: https://arxiv.org/abs/2206.12262 — (05.04.2024).
45. Loss function [Електронний ресурс]. — Режим доступу: https://uk.eitca.org/%D1%88%D1%82%D1%83%D1%87%D0%BD%D0%B8%D0%B9-%D1%96%D0%BD%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D1%82/eitc-ai-tff-tensorflow-%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B8/%D0%B2%D1%81%D1%82%D1%83%D0%BF-%D0%B4%D0%BE-%D1%82%D0%B5%D0%BD%D0%B7%D0%BE%D1%80%D0%BD%D0%BE%D0%B3%D0%BE-

%D0%BF%D0%BE%D1%82%D0%BE%D0%BA%D1%83/%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D0%B8%D0%B9-%D0%BA%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B8%D0%B9-%D0%B7%D1%96%D1%80-%D0%B7-%D0%BC%D0%BB/%D0%B5%D0%BA%D0%B7%D0%B0%D0%BC%D0%B5%D0%BD%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B8%D0%B9-%D0%BE%D0%B3%D0%BB%D1%8F%D0%B4-%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B8-%D0%BA%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%BE%D0%B3%D0%BE-%D0%B7%D0%BE%D1%80%D1%83-%D0%B7-%D0%BC%D0%BB/%D1%8F%D0%BA%D0%B0-%D1%80%D0%BE%D0%BB%D1%8C-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%97-%D0%BE%D0%BF%D1%82%D0%B8%D0%BC%D1%96%D0%B7%D0%B0%D1%82%D0%BE%D1%80%D0%B0-%D1%82%D0%B0-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%97-%D0%B2%D1%82%D1%80%D0%B0%D1%82-%D1%83-%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%BC%D1%83-%D0%BD%D0%B0%D0%B2%D1%87%D0%B0%D0%BD%D0%BD%D1%96/ — (01.05.2024).

46. Mean squared error [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Mean_squared_error — (21.07.2024).
47. Mean absolute error [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Mean_absolute_error — (21.07.2024).
48. Coefficient of determination [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Coefficient_of_determination — (21.07.2024).

49. Accuracy and precision [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Accuracy_and_precision — (21.07.2024).
50. What is Python? Executive Summary [Електронний ресурс]. — Режим доступу: https://www.python.org/doc/essays/blurb/ — (01.05.2024).
51. What is Python [Електронний ресурс]. — Режим доступу: https://aws.amazon.com/what-is/python/ — (05.01.2024).
52. What is Python? Everything You Need to Know to Get Started [Електронний ресурс]. — Режим доступу: https://www.datacamp.com/blog/all-about-python-the-most-versatile-programming-language#rdl — (24.06.2024).
53. HuggingFace's Transformers: State-of-the-art Natural Language Processing [Електронний ресурс]. — Режим доступу: https://arxiv.org/abs/1910.03771 — (09.10.2024).
54. PyTorch [Електронний ресурс]. — Режим доступу: https://www.nvidia.com/en-us/glossary/pytorch/— (02.08.2024).
55. Scikit Learn - Introduction [Електронний ресурс]. — Режим доступу: https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.htm — (01.07.2024).
56. Pandas [Електронний ресурс]. — Режим доступу: https://www.nvidia.com/en-us/glossary/pandas-python/ — (01.02.2024).
57. Natural Language Toolkit [Електронний ресурс]. — Режим доступу: https://www.nltk.org/ — (01.02.2024).
58. Emoji for Python [Електронний ресурс]. — Режим доступу: https://pypi.org/project/emoji/ — (16.01.2024).
59. Natural Language Processing With SpaCy (A Python Library) [Електронний ресурс]. — Режим доступу: https://www.comet.com/site/blog/natural-language-processing-with-spacy-a-python-library/ — (02.11.2024).

ДОДАТКИ

Додаток 1

Копії графічних матеріалів

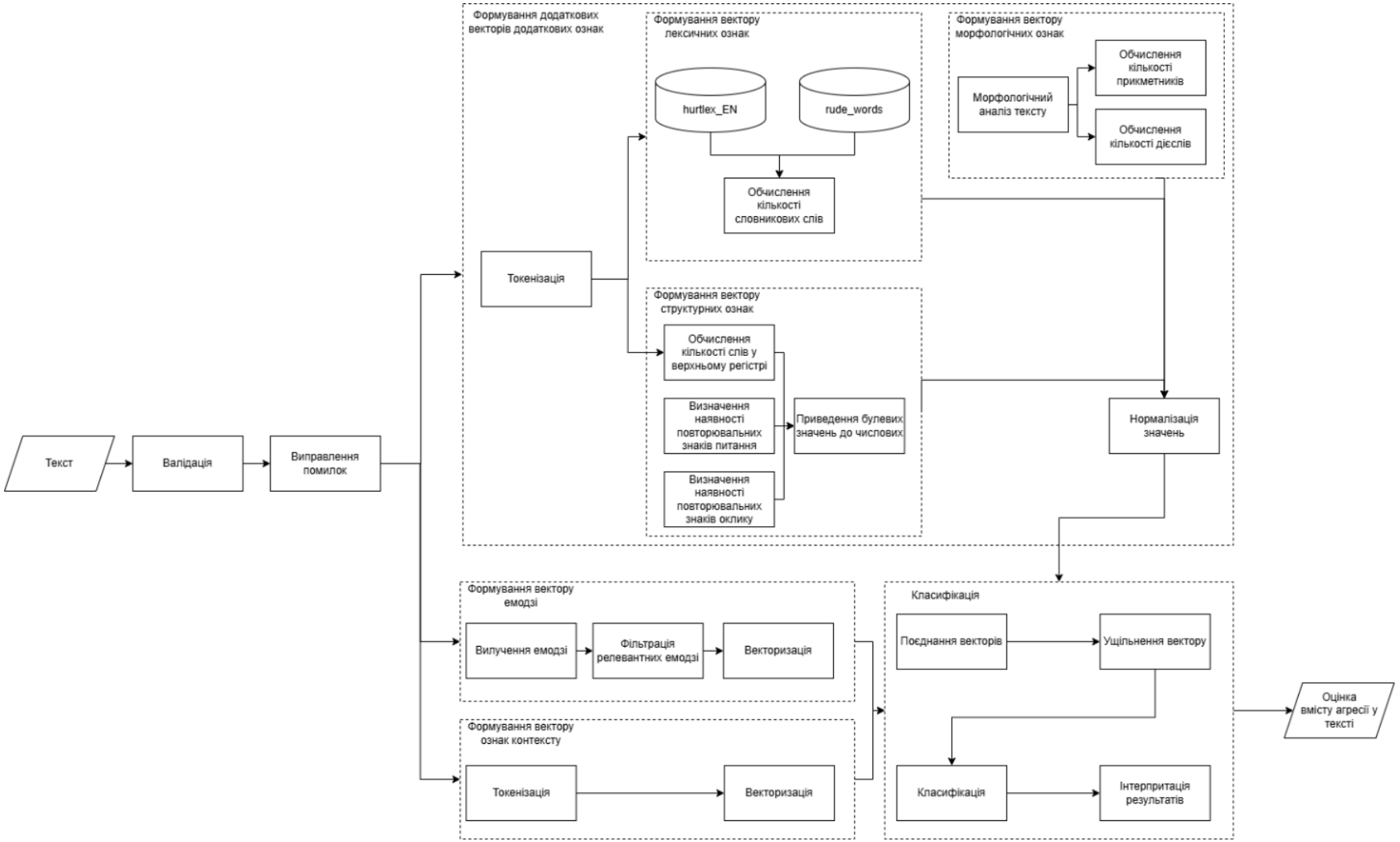
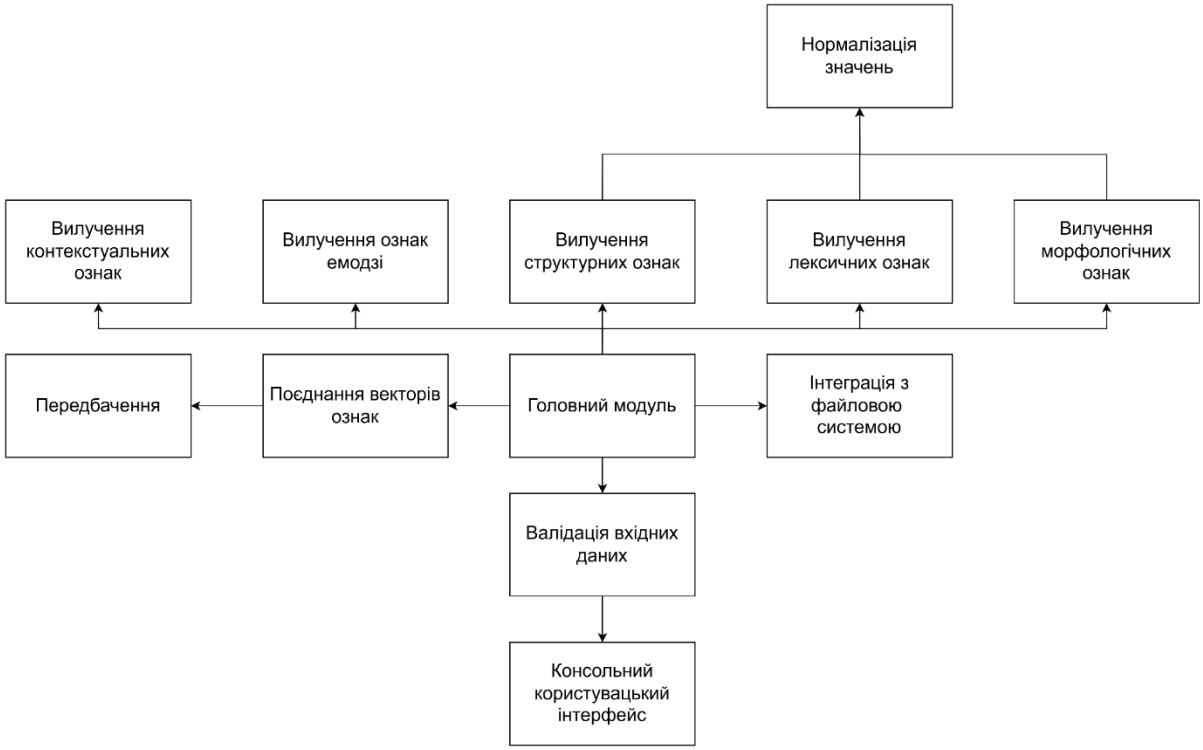
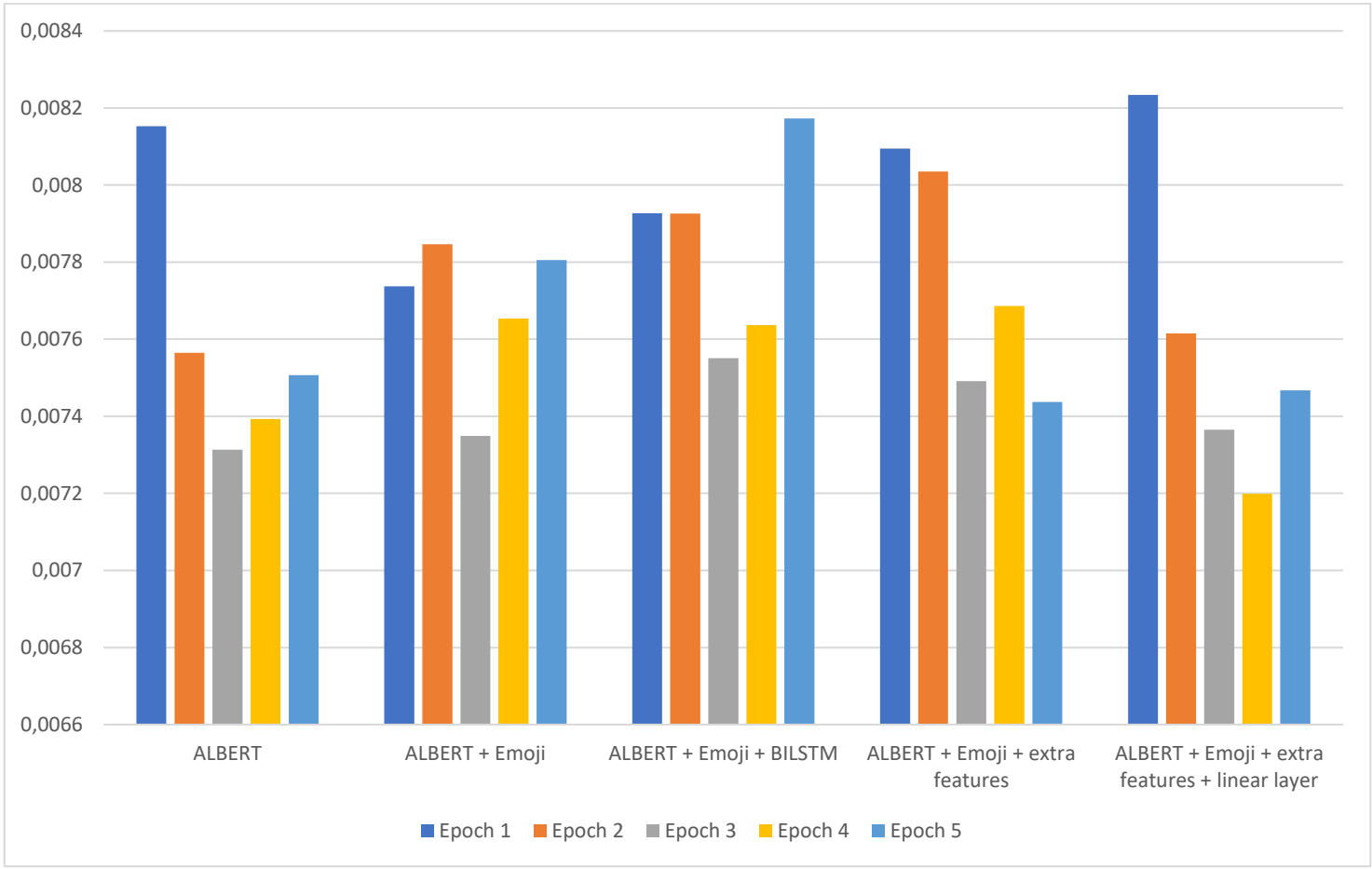


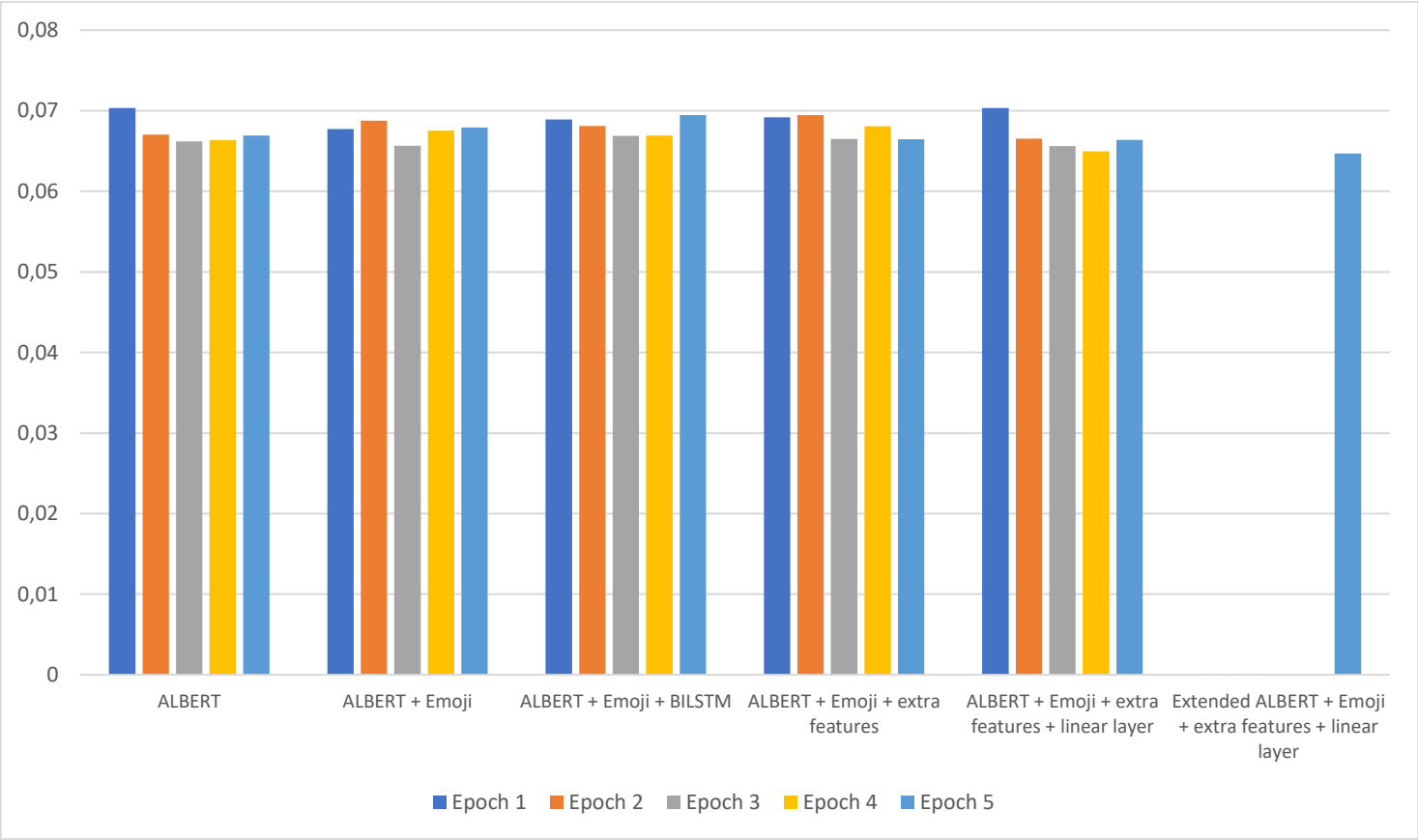
Схема запропонованого методу автоматичної класифікації природномовних текстових даних за наявністю агресії
Гришаєв Д.І., група КП-31мн



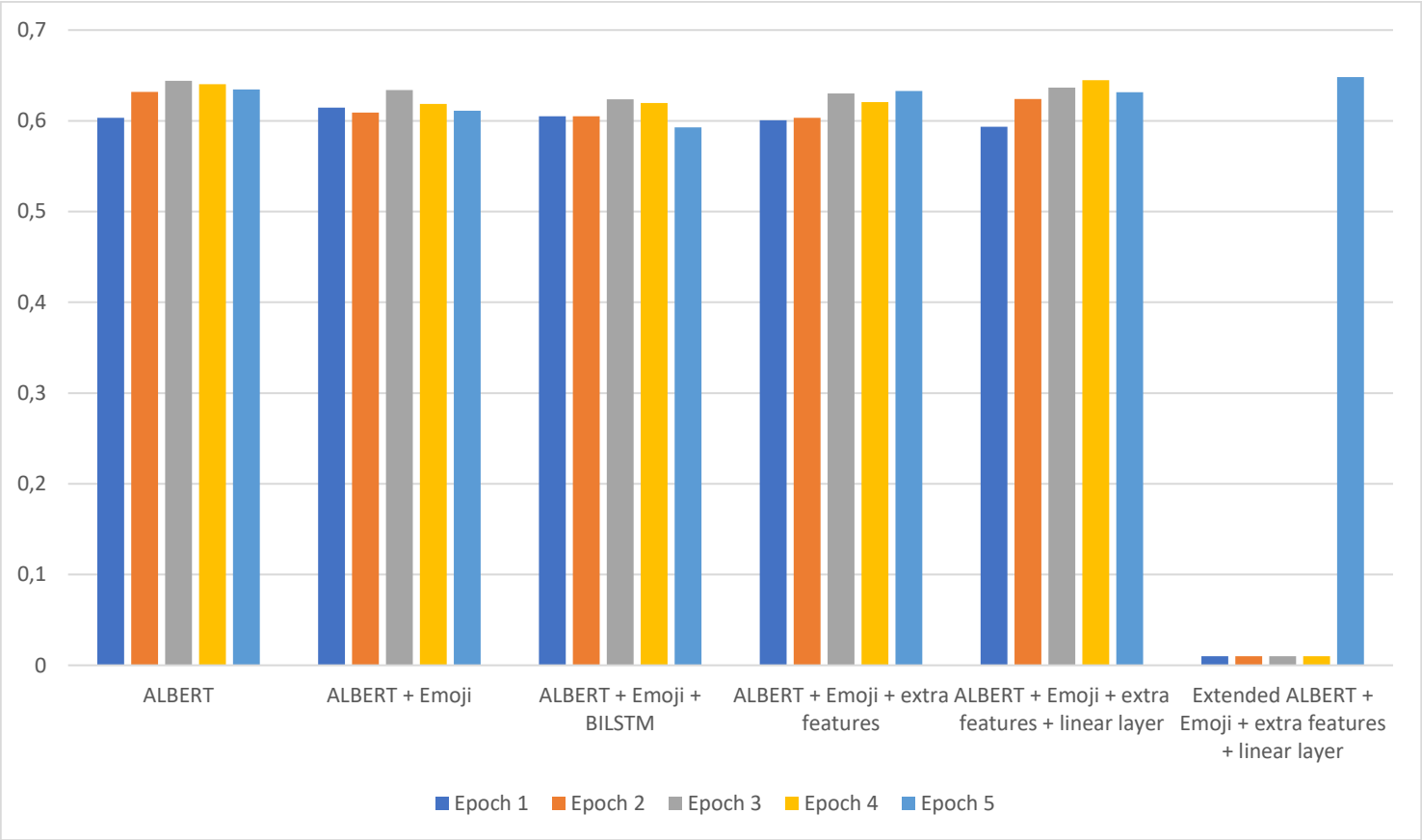
Архітектура програмного забезпечення автоматичної класифікації природномовних текстових даних за наявністю агресії
Гришаєв Д.І., група КП-31мн



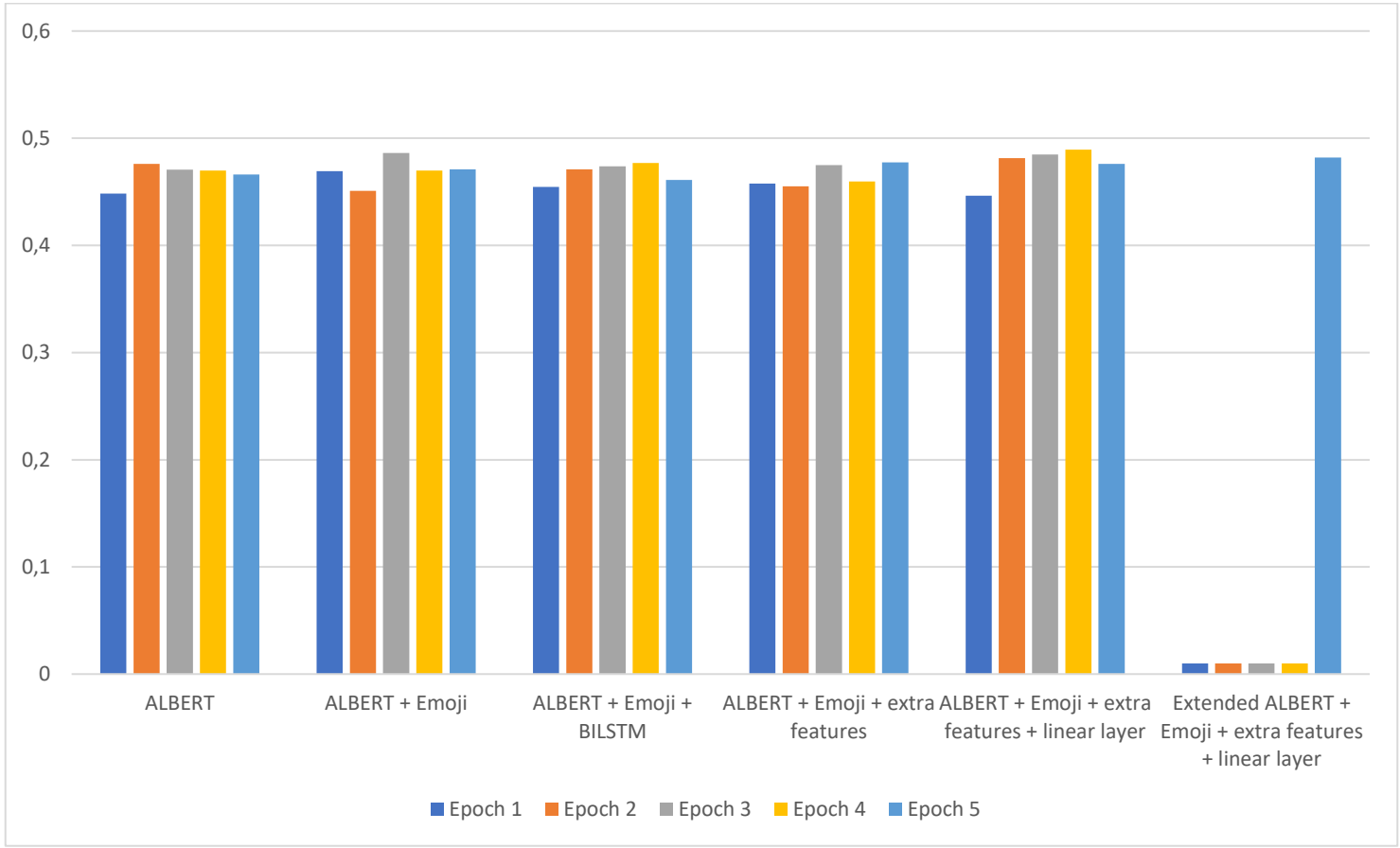
Гістограма порівняння значень метрики Loss
Гришаєв Д.І., група КП-31мн



Гістограма порівняння значень метрики MAE
Гришаєв Д.І., група КП-31мн



Гістограма порівняння значень метрики R^2
Гришаєв Д.І., група КП-31мн



Гістограма порівняння значень метрики Accuracy
Гришаєв Д.І., група КП-31мн

Додаток 2

Лістинг програми

```
from ui import ui, outputResults
from features.structural import extractStructuralFeatures
from features.emoji import extractEmojiFeatures, remove_emojis,
extract_emojis
from features.context import extractContextFeatures
from classifier import makePrediction
from explanation import explainResults
from featuresCombiner import combineFeatures

def main():
  validatedTexts, score = ui()
  textsFeatures = []
  for i, text in enumerate(validatedTexts, 1):
    bareText = remove_emojis(text)
    emojis = extract_emojis(text)
    features = extractContextFeatures(bareText)
    structuralFeatures = extractStructuralFeatures(bareText)
    emojiFeatures = extractEmojiFeatures(emojis)
    features["emoji_embedings"] = emojiFeatures
    features["additional_features"] = structuralFeatures

    textsFeatures.append(features)
  batch = combineFeatures(textsFeatures)
  scores = makePrediction(batch)
  textsWithResults = explainResults(scores, score)
  outputResults(validatedTexts, scores, textsWithResults)

main()

import torch
import warnings

warnings.filterwarnings("ignore", category=UserWarning)
torch.set_warn_always(False)

def combineFeatures(features):
  batch = {
     'input_ids': torch.stack([torch.tensor(item['input_ids']) for item in
features]),
    'attention_mask': torch.stack([torch.tensor(item['attention_mask']) for
item in features]),
    'token_type_ids': torch.stack([torch.tensor(item['token_type_ids']) for
item in features]),
     'emoji_embedings': torch.stack([torch.tensor(item['emoji_embedings'])
for item in features]),
                                                    'additional_features':
torch.stack([torch.tensor(item['additional_features']) for item in
features])
  }
  return batch

def explainResults(scores, limitScore):
  if isinstance(scores, (list, tuple)):
    return [score > limitScore for score in scores]
  else:
    return [scores > limitScore]

from transformers import AlbertPreTrainedModel, AlbertModel, AlbertConfig
import torch.nn as nn
import torch
from torch.nn import CrossEntropyLoss, MSELoss
```

```
from transformers.modeling_outputs import SequenceClassifierOutput
from safetensors.torch import load_file

import numpy as np
import random

SEED = 42
random.seed(SEED)
np.random.seed(SEED)
torch.manual_seed(SEED)
torch.cuda.manual_seed_all(SEED)
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = False

BASE_MODEL = 'albert-base-v2'

class AlbertForSequenceClassification(AlbertPreTrainedModel):
    def __init__(self, config):
        super().__init__(config)
        self.num_labels = config.num_labels

        self.albert = AlbertModel(config)
        self.dropout = nn.Dropout(config.classifier_dropout_prob)
        self.hidden_size = config.hidden_size
         self.intermediate_layer = nn.Linear(config.hidden_size + 300 + 3,
config.hidden_size)
        self.relu = nn.ReLU()
                      self.classifier    =    nn.Linear(config.hidden_size,
self.config.num_labels)
        self.init_weights()

    def forward(
        self,
        input_ids=None,
        attention_mask=None,
        token_type_ids=None,
        position_ids=None,
        head_mask=None,
        inputs_embeds=None,
        labels=None,
        emoji_embedings=None,
        output_attentions=None,
        output_hidden_states=None,
        additional_features=None,
        return_dict=None,
    ):
            return_dict  =  return_dict  if  return_dict  is  not  None  else
self.config.use_return_dict
        outputs = self.albert(
            input_ids=input_ids,
            attention_mask=attention_mask,
            token_type_ids=token_type_ids,
            position_ids=position_ids,
            head_mask=head_mask,
            inputs_embeds=inputs_embeds,
            output_attentions=output_attentions,
            output_hidden_states=output_hidden_states,
            return_dict=return_dict,
        )
        pooled_output=outputs[1]
             combined_output  =  torch.cat((pooled_output,  emoji_embedings,
additional_features), dim=1)
```

```
                                                intermediate_output          =
self.relu(self.intermediate_layer(combined_output))
        intermediate_output = self.dropout(intermediate_output)
        logits = self.classifier(intermediate_output)

        loss = None
        if labels is not None:
            if self.num_labels == 1:
                #  We are doing regression
                loss_fct = MSELoss()
                loss = loss_fct(logits.view(-1), labels.view(-1))
            else:
                loss_fct = CrossEntropyLoss()
                       loss  =  loss_fct(logits.view(-1,  self.num_labels),
labels.view(-1))

        if not return_dict:
            output = (logits,) + outputs[2:]
            return ((loss,) + output) if loss is not None else output

        return SequenceClassifierOutput(
            loss=loss,
            logits=logits,
            hidden_states=outputs.hidden_states,
            attentions=outputs.attentions,
        )

config = AlbertConfig.from_pretrained(BASE_MODEL, num_labels=1)

model = AlbertForSequenceClassification(config)

weights_path = "./features/model.safetensors"
state_dict = load_file(weights_path)
model.load_state_dict(state_dict)

def makePrediction(batch):
  with torch.no_grad():
    outputs = model(**batch)
    return outputs.logits.squeeze().tolist()

from nltk.tokenize import word_tokenize
import re
import torch

EXCLAMATION_MARK = "!"
QUESTION_MARK = "?"
MIN_WORD_LENGTH = 2

def getWords(text):
  tokens = word_tokenize(text)
  words = [word for word in tokens if word.isalnum()]
  return words

def isWithRepeatedPunctuation(text, symbol):
  escapedSymbol = re.escape(symbol)
  pattern = rf"({escapedSymbol}{{2,}})"
  match = re.search(pattern, text)
  return bool(match)

def countUppercaseWords(text):
  words = getWords(text)
  uppercaseWords = [word for word in words if word.isupper() and len(word)
>= MIN_WORD_LENGTH]
```

```
  return len(uppercaseWords) / len(words)

def extractStructuralFeatures(text):
     features    =    isWithRepeatedPunctuation(text,    EXCLAMATION_MARK),
isWithRepeatedPunctuation(text, QUESTION_MARK), countUppercaseWords(text)
  return torch.tensor(features, dtype=torch.float32)

import emoji
from gensim.models import KeyedVectors
import torch

e2v     =     KeyedVectors.load_word2vec_format('./features/emoji2vec.bin',
binary=True)

def extract_emojis(text):
  return [char for char in text if char in emoji.EMOJI_DATA]

def remove_emojis(text):
  return emoji.replace_emoji(text, replace='')

def extractEmojiFeatures(emojis, embedding_dim=300):
  emoji_embeddings = []
  for emoji in emojis:
      if emoji in e2v:
          emoji_embeddings.append(e2v[emoji])

  if emoji_embeddings:
           emoji_embeddings_tensor   =   torch.stack([torch.tensor(embedding,
dtype=torch.float32) for embedding in emoji_embeddings], dim=0)
      averaged_embedding = emoji_embeddings_tensor.mean(dim=0)
  else:
    averaged_embedding = torch.zeros(embedding_dim, dtype=torch.float32)
  return averaged_embedding

from transformers import AlbertTokenizer

BASE_MODEL = 'albert-base-v2'

tokenizer = AlbertTokenizer.from_pretrained(BASE_MODEL)

def extractContextFeatures(text):
     return    tokenizer(text,    truncation=True,    padding="max_length",
max_length=256)

import argparse
from validation import validate_number, validate_texts

INPUT_TEXT = "Введіть текст (від 3 до 160 слів). Для завершення введення
тексту натисніть Enter двічі:"
NO_FILE_ERROR = "Файл не знайдено."
READING_FILE_ERROR = "Виникла помилка при читанні файлу"
INPUT_NUMBER = "Введіть число від 0 до 1."
RESULT_TEXT = "Текст номер"
WITH_AGRESSION = "вміщає агресію"
WITHOUT_AGRESSION = "не вміщає агресію"

notValidTexts = []

def read_multiline_text():
    print(INPUT_TEXT)
    texts = []
    while True:
        text = input()
```

```
        if text.strip() == "":
            if len(texts) > 0:
                break
        texts.append(text)
    return texts

def ui():
    global notValidTexts
    parser = argparse.ArgumentParser()
    parser.add_argument('-t', '--text', action='store_true')
    parser.add_argument('-f', '--file', type=str)
    parser.add_argument('-n', '--number', type=validate_number)


    args = parser.parse_args()

    texts = []

    if args.text:
        texts = read_multiline_text()

    if args.file:
        try:
            with open(args.file, 'r', encoding='utf-8') as file:
                content = file.read()
                texts = content.splitlines()
        except FileNotFoundError:
            print(NO_FILE_ERROR)
        except Exception as e:
            print(f"{READING_FILE_ERROR}: {e}")

    validationResults = validate_texts(texts)
    notValidTexts = validationResults[1]
    return validationResults[0], args.number

def outputResults(texts, scores, results):
  scoresFixed = scores
  if not isinstance(scores, (list, tuple)):
    scoresFixed = [scoresFixed]
  for i, text in enumerate(texts, 0):
    result = WITH_AGRESSION if results[i] else WITHOUT_AGRESSION
    print(f"{i + 1}. {text[:20]}... {scoresFixed[i]} {result}")
  for i, text in enumerate(notValidTexts, 0):
    print(f"{i + 1 + len(texts)}. {text[0][:20]}... {text[1]}")


import argparse
from langdetect import detect, LangDetectException

INVALID_LIMIT_SCORE = "Число повинно бути в діапазоні від 0 до 1."
NAN_TYPE = "Введене значення не є числом."
WRONG_TEXT_LENGTH = "Неправильна довжина. Текст повинен містити від 5 до 100
слів."
WRONG_TEXT_LANGUAGE = "Неправильна мова. Очікується англійська, знайдено"
UNDEFINED_TEXT_LANGUAGE = "Неправильна мова. Не вдалося визначити мову."

MIN_WORDS_COUNT = 3
MAX_WORDS_COUNT = 160

def validate_number(value):
  try:
    number = float(value)
    if 0 <= number <= 1:
```

```
      return number
    else:
      raise argparse.ArgumentTypeError(INVALID_LIMIT_SCORE)
  except ValueError:
    raise argparse.ArgumentTypeError(NAN_TYPE)


def validate_texts(texts):
  valid_texts = []
  invalid_texts = []

  for text in texts:
    words = text.split()
    if len(words) < MIN_WORDS_COUNT or len(words) > MAX_WORDS_COUNT:
      invalid_texts.append((text, WRONG_TEXT_LENGTH))
      continue

    try:
      language = detect(text)
      if language != 'en':
         invalid_texts.append((text, f"{WRONG_TEXT_LANGUAGE}: {language}."))
          continue
    except LangDetectException:
        invalid_texts.append((text, UNDEFINED_TEXT_LANGUAGE))
        continue

    valid_texts.append(text)

  return valid_texts, invalid_texts

import re
import spacy
from nltk.tokenize import word_tokenize
nlp = spacy.load('en_core_web_md')

EXCLAMATION_MARK = "!"
QUESTION_MARK = "?"
MIN_WORD_LENGTH = 2

def isWithRepeatedPunctuation(text, symbol):
  escapedSymbol = re.escape(symbol)
  pattern = rf"({escapedSymbol}{{2,}})"
  match = re.search(pattern, text)
  return bool(match)

def getWords(text):
  tokens = word_tokenize(text)
  words = [word for word in tokens if word.isalnum()]
  return words

def countUppercaseWords(text):
  words = getWords(text)
  uppercaseWords = [word for word in words if word.isupper() and len(word)
>= MIN_WORD_LENGTH]
  return len(uppercaseWords) / len(words)

def countVerbs(analyzedText):
  return sum(1 for token in analyzedText if token.pos_ == 'VERB')

def countAdjectives(analyzedText):
  return sum(1 for token in analyzedText if token.pos_ == 'ADJ')

def extractFeatures(text):
```

```
  analyzedText = nlp(text)
  wordsCount = sum(1 for token in analyzedText if not token.is_punct)
    isWithRepeatedExclamationMark   =   int(isWithRepeatedPunctuation(text,
EXCLAMATION_MARK))
     isWithRepeatedQuestionMark    =    int(isWithRepeatedPunctuation(text,
QUESTION_MARK))
  uppercaseWordsFrequency = countUppercaseWords(text)
  verbsFrequency = countVerbs(analyzedText) / wordsCount
  adjectivesFrequency = countAdjectives(analyzedText) / wordsCount
      return     [uppercaseWordsFrequency,     isWithRepeatedQuestionMark,
isWithRepeatedExclamationMark, verbsFrequency, adjectivesFrequency]

import torch
from  transformers  import  AlbertTokenizer,  Trainer,  TrainingArguments,
AlbertPreTrainedModel, AlbertModel
import torch.nn as nn
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
import emoji
from datasets import Dataset
from gensim.models import KeyedVectors

df = pd.read_csv('./filtered.csv')

def extract_emojis(text):
  return [char for char in text if char in emoji.EMOJI_DATA]

def remove_emojis(text):
  return emoji.replace_emoji(text, replace='')

df['emojis'] = df['text'].apply(extract_emojis)
df['bare_text'] = df['text'].apply(remove_emojis)

min_score = df['hate_speech_score'].min()
max_score = df['hate_speech_score'].max()
df['score'] = (df['hate_speech_score'] - min_score) / (max_score - min_score)

train_df, evaluate_df = train_test_split(
    df, test_size=0.3, random_state=42
)

evaluate_df, test_df = train_test_split(
    evaluate_df, test_size=0.4, random_state=42
)

BASE_MODEL = 'albert-base-v2'
LEARNING_RATE = 2e-5
MAX_LENGTH = 256
BATCH_SIZE = 32
EPOCHS = 5

tokenizer = AlbertTokenizer.from_pretrained(BASE_MODEL)

e2v = KeyedVectors.load_word2vec_format('./emoji2vec.bin', binary=True)

max_emoji_count = 10

def get_emoji_embeddings(emojis, embedding_dim=300):
  emoji_embeddings = []
  for emoji in emojis:
      if emoji in e2v:
          emoji_embeddings.append(e2v[emoji])
```

```
  if emoji_embeddings:
        emoji_embeddings_tensor    =    torch.stack([torch.tensor(embedding,
dtype=torch.float64) for embedding in emoji_embeddings], dim=0)
    if emoji_embeddings_tensor.size(0) < max_emoji_count:
                    padding      =      torch.zeros((max_emoji_count      -
emoji_embeddings_tensor.size(0), embedding_dim), dtype=torch.float64)
     emoji_embeddings_tensor = torch.cat([emoji_embeddings_tensor, padding],
dim=0)
      return emoji_embeddings_tensor
  else:
    return torch.zeros((10, embedding_dim), dtype=torch.float64)

def preprocess_function(examples):
  labels = examples["score"]
  emojisList = examples["emojis"]
     examples     =     tokenizer(examples["bare_text"],     truncation=True,
padding="max_length", max_length=256)
  examples["label"] = [float(label) for label in labels]
  examples["emoji_embedings"] = [get_emoji_embeddings(emojis) for emojis in
emojisList]
  return examples

#  train_dataset  =  Dataset.from_pandas(train_df).map(preprocess_function,
batched=True)
#                              evaluate_dataset                                =
Dataset.from_pandas(evaluate_df).map(preprocess_function, batched=True)
test_dataset     =     Dataset.from_pandas(test_df).map(preprocess_function,
batched=True)
test_dataset.set_format(type='torch', columns=['emoji_embedings'])

emoji_embeddings_tensor = test_dataset[0]["emoji_embedings"]

_, M = emoji_embeddings_tensor.size()

_, N = emoji_embeddings_tensor.size()

print(N, M)

torch.matmul(emoji_embeddings_tensor, emoji_embeddings_tensor.T)

print(torch.matmul(emoji_embeddings_tensor, emoji_embeddings_tensor.T))

import torch

training_args = TrainingArguments(
    output_dir="./results",
    learning_rate=LEARNING_RATE,
    per_device_train_batch_size=BATCH_SIZE,
    per_device_eval_batch_size=BATCH_SIZE,
    num_train_epochs=EPOCHS,
    evaluation_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
    weight_decay=0.01,
)


class RegressionTrainer(Trainer):
    def compute_loss(self, model, inputs, return_outputs=False):
        labels = inputs.pop("labels")
        outputs = model(**inputs)
        logits = outputs[0][:, 0]
```

```
        loss = torch.nn.functional.mse_loss(logits, labels)
        return (loss, outputs) if return_outputs else loss

trainer = RegressionTrainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=evaluate_dataset,
    compute_metrics=compute_metrics_for_regression
)
```

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

МЕТОД ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ АВТОМАТИЧНОЇ КЛАСИФІКАЦІЇ ПРИРОДНОМОВНИХ ТЕКСТОВИХ ДАНИХ ЗА НАЯВНІСТЮ АГРЕСІЇ

Виконав: Гришаєв Дмитро Ігорович

Керівник: доц. кафедри ПЗКС, к.т.н., доц. Заболотня Тетяна Миколаївна

Київ – 2025

АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ

- Збільшення кількості агресивних висловлювань у мережі Інтернет, як результат зростання популярності соціальних мереж та їх політики анонімності.
- Відсутність врахування мовних особливостей, прихованих емоцій та нестандартних форм вираження агресії у наявних системах автоматичної модерації текстів.
- Потреба в розробленні інструментів для виявлення агресії в природномовних текстах з урахуванням контексту, лексики та емоційного забарвлення.
- Підвищення культури спілкування та безпеки користувачів.
- Нові можливості для захисту та розвитку цифрового середовища.

ЗАГАЛЬНА ІНФОРМАЦІЯ ПРО ДОСЛІДЖЕННЯ

Об’єкт дослідження: процес оброблення природномовних текстових даних для їх класифікації за наявністю агресії.

Предмет дослідження: методи, способи та алгоритми виявлення ознак агресії в природномовних текстових даних.

Мета дослідження: підвищення точності автоматичної класифікації природномовних текстових даних шляхом розроблення методу, який враховує додаткові структурні, лексичні та морфологічні ознаки агресії.

ПОСТАНОВКА ЗАДАЧІ

1. Пошук та аналіз методів для автоматичної класифікації природномовних текстових даних.
2. Опис обраного методу для подальшої модифікації.
3. Розроблення модифікованого методу для автоматичної класифікації природномовних текстових даних за наявністю агресії.
4. Створення програмної реалізації запропонованого методу.
5. Аналіз результатів роботи модифікованого методу у порівнянні з оригінальним методом.
6. Визначення напрямків для подальшої роботи над запропонованим методом

ІСНУЮЧІ МЕТОДИ

ПРОБЛЕМИ АВТОМАТИЧНОЇ КЛАСИФІКАЦІЇ ТЕКСТУ ЗА НАЯВНІСТЮ АГРЕСІЇ

- Недостатня точність класифікації текстових даних за наявністю агресії, як результат ігнорування певних вагомих її ознак.
- Навмисне пошкодження власних дописів, як спосіб протидії існуючим механізмам модерації текстів.
- Обмеження гнучкості при використанні бінарної класифікації.
- Жорсткі системні вимоги для можливості тренування та запуску автоматичних класифікаторів на основі глибокого навчання.

БАЗОВИЙ МЕТОД АВТОМАТИЧНОЇ КЛАСИФІКАЦІЇ ТЕКСТУ ЗА НАЯВНІСТЮ АГРЕСІЇ

Обрано модель ALBERT, як менш вимогливу похідну від BERT.

Етапи базового методу:
- валідація тексту
- очищення вхідних даних від текстового шуму
- застосування моделі ALBERT для створення векторного представлення тексту
- класифікація тексту за наявністю агресії
- представлення результатів

АНАЛІЗ КОНТЕКСТНИХ ОСОБЛИВОСТЕЙ ТЕКСТОВИХ ДАНИХ

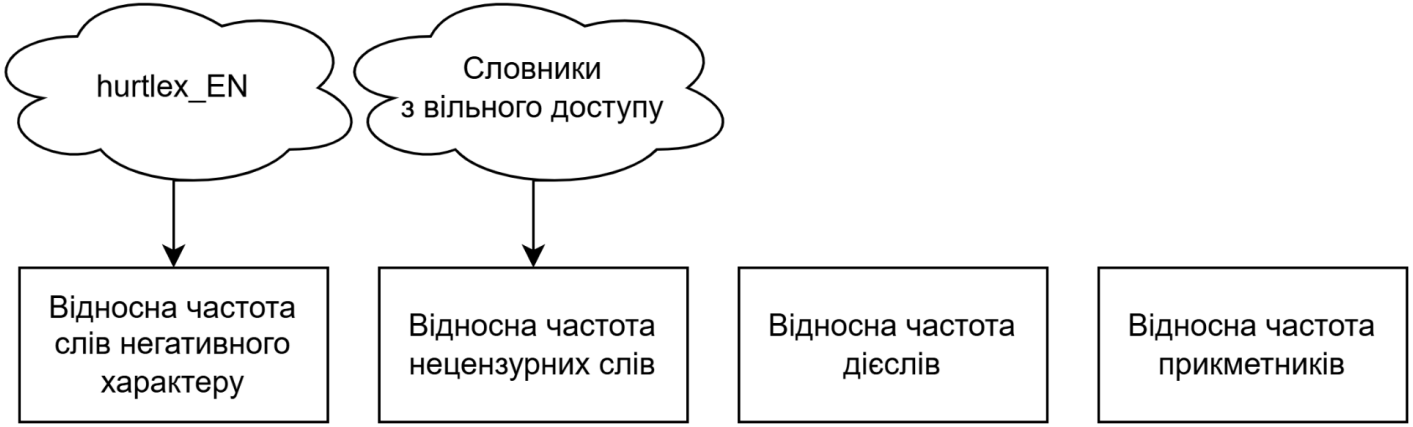
- Допомога у виявленні прихованої агресії.
- Використання ALBERT для створення векторного представлення тексту.
- Використовується конкатенація останніх чотирьох шарів ALBERT.
- Albert-base-v2 – використана в рамках дослідження модель.

СТРУКТУРНІ ОСОБЛИВОСТІ ТЕКСТОВИХ ДАНИХ

Визначено наступні особливості тексту, що можуть бути ознакою прояву агресії:

- використання повторювальних **знаків оклику**;
- використання повторювальних **знаків питання**;
- використання **слів у верхньому регістрі**;

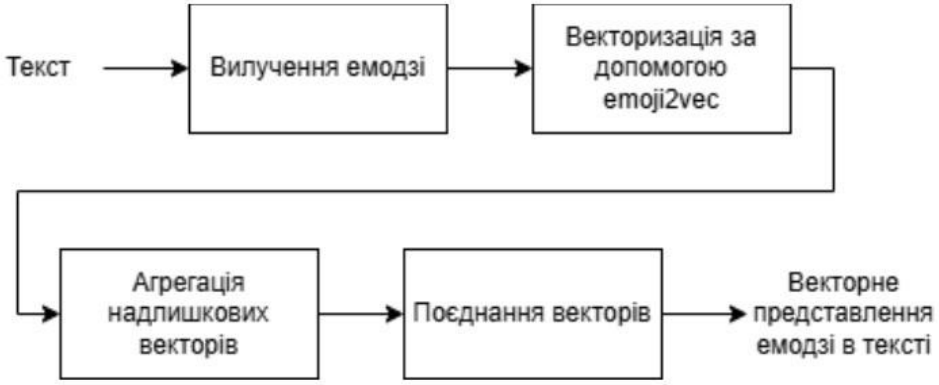
ЛЕКСИЧНІ ТА МОРФОЛОГІЧНІ ОСОБЛИВОСТІ ТЕКСТОВИХ ДАНИХ



ВПЛИВ ЕМОДЗІ НА ЕМОЦІЙНЕ ЗАБАРВЛЕННЯ ТЕКСТУ

Приклади використання емодзі для підсилення емоційного забарвлення тексту:

- "Love how my boss ignores every idea I bring up 💔
- "Sure, take credit for my work again 🙃 Go ahead.
- "Absolutely LOVE being left on read for 3 days 🥰"



РІШЕННЯ ПРОБЛЕМИ НЕЧІТКО ВИЗНАЧЕНОГО ПОНЯТТЯ АГРЕСІЇ

Проблема невизначеності поняття «агресивне висловлювання».

Використання регресійної моделі, як спосіб збільшення гнучкості у використанні.

Необхідність у спеціалізованому проанотованому тренувальному датасеті із вказаними оцінками рівня агресії.

Функція втрат – середньоквадратична помилка.

РІШЕННЯ ПРОБЛЕМИ НАВМИСНОГО ПОШКОДЖЕННЯ ВХІДНИХ ДАНИХ

Основні типи пошкодження тексту:

- орфографічні помилки;
- спотворені слова;
- зайві або некоректні символи;
- скорочення або жаргонізми;

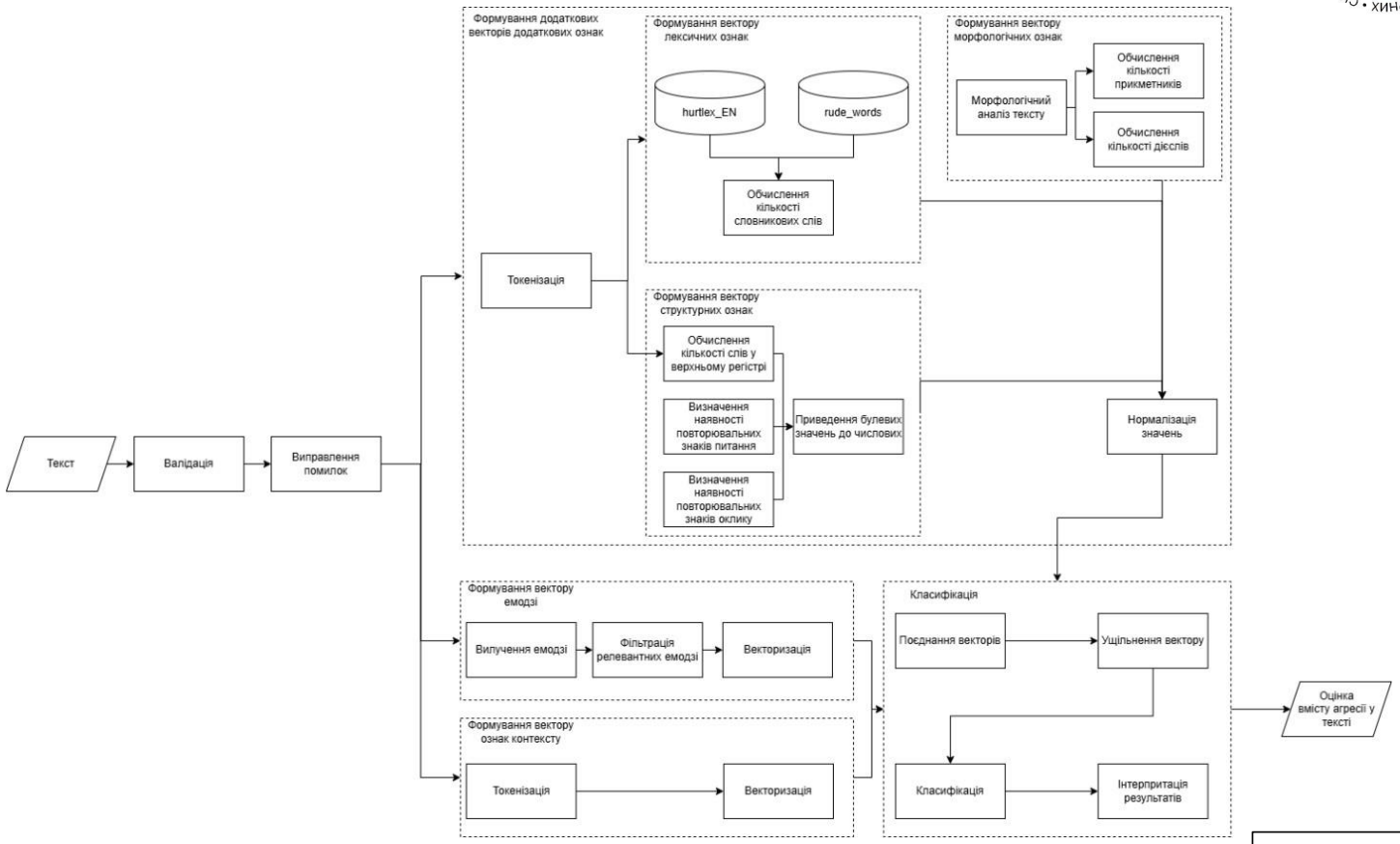
Виправлення орфографічних помилок за допомогою language_tool_python, як частина попереднього оброблення вхідних даних.

Додаткове видалення повторювальних пробілів, URL-адрес та хеш-тегів.

ЗАПРОПОНОВАНА МОДИФІКАЦІЯ МЕТОДУ

- Валідація тексту.
- **Виправлення потенційних орфографічних помилок.**
- Застосування моделі ALBERT для створення векторного представлення тексту **із врахуванням результатів роботи останніх 4 шарів**.
- **Обчислення структурних ознак агресії.**
- **Обчислення лексичних ознак агресії.**
- **Обчислення морфологічних ознак агресії.**
- **Векторизація емодзі.**
- **Конкатенація векторів та зменшення його розміру за допомогою лінійного шару.**
- Класифікація тексту за наявністю агресії.
- Представлення результатів.

УЗАГАЛЬНЕНА СХЕМА ЗАПРОПОНОВАНОГО МЕТОДУ



ТЕХНОЛОГІЇ РОЗРОБЛЕННЯ

Python

Transformers

PyTorch

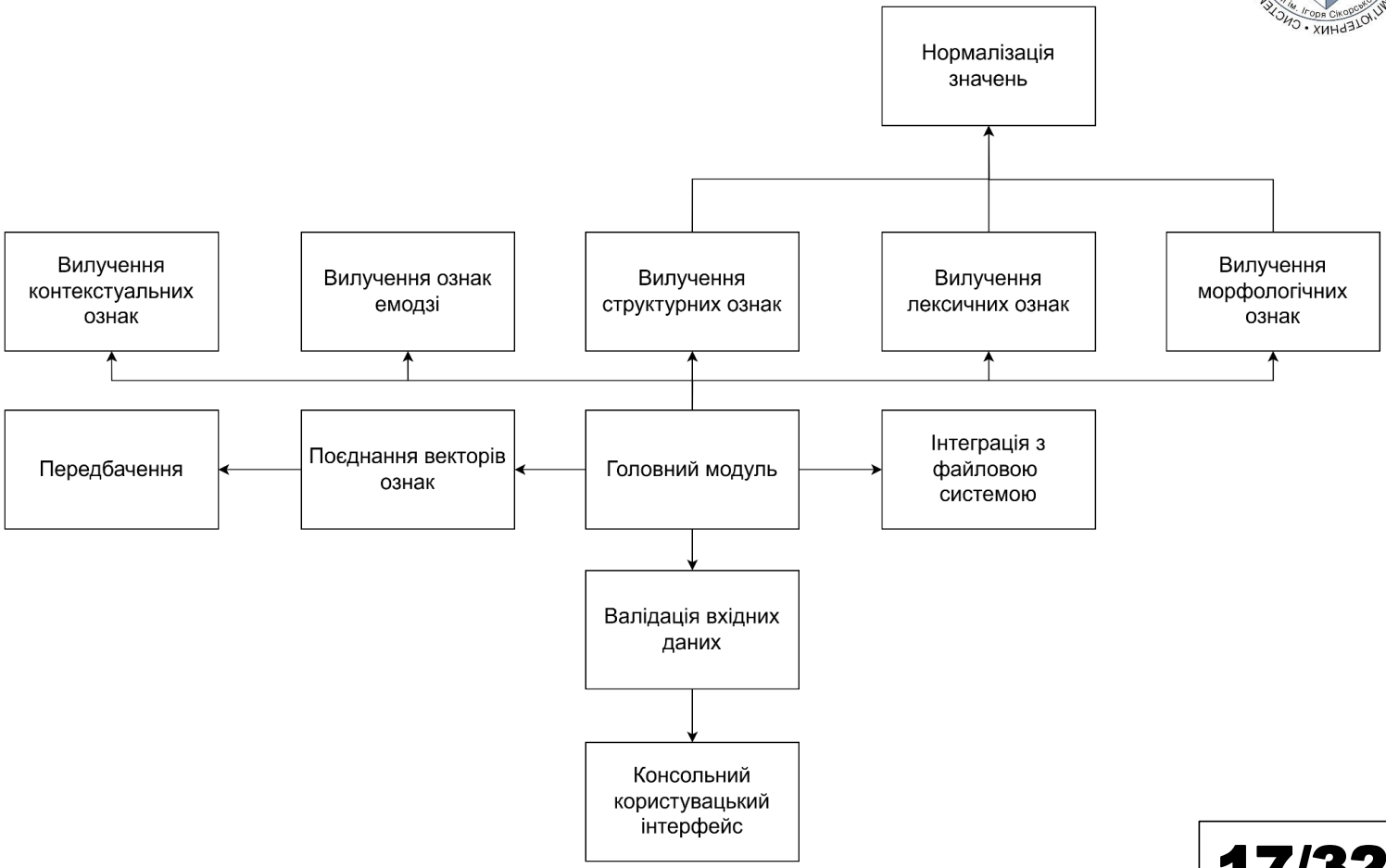
scikit-learn

Pandas

NLTK

SpaCy

АРХІТЕКТУРА РЕАЛІЗОВАНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



РЕЗУЛЬТАТИ КЛАСИФІКАЦІЇ ТЕКСТІВ ЗА НАЯВНІСТЮ АГРЕСІЇ

Результат передбачення для прикладу із файлу

```
PS C:\Users\Dmytro\Desktop\diplomaDocumentation\app\method> python .\main.py -f .\texts.txt -n 0.29
1. Question: These 4 br... 0.6374067068099976 вміщає агресію
```

Результат передбачення для декількох прикладів із

```
PS C:\Users\Dmytro\Desktop\diplomaDocumentation\app\method> python .\main.py -f .\texts.txt -n 0.29
1. Question: These 4 br... 0.6374067068099976 вміщає агресію
2. Many of the families... 0.32954108715057373 вміщає агресію
3. Hero Rohit Sharma lo... 0.261223167181015 не вміщає агресію
4. i love you my bunny.... 0.2823463976383209 не вміщає агресію
```

Результат передбачення для декількох прикладів із зміненою пороговою оцінкою

```
PS C:\Users\Dmytro\Desktop\diplomaDocumentation\app\method> python .\main.py -f .\texts.txt -n 0.33
1. Question: These 4 br... 0.6374067068099976 вміщає агресію
2. Many of the families... 0.32954108715057373 не вміщає агресію
3. Hero Rohit Sharma lo... 0.261223167181015 не вміщає агресію
4. i love you my bunny.... 0.2823463976383209 не вміщає агресію
```

Результат передбачення для декількох прикладів, що були введені у консоль

```
PS C:\Users\Dmytro\Desktop\diplomaDocumentation\app\method> python .\main.py -t -n 0.33
Введіть текст (від 3 до 160 слів). Для завершення введення тексту натисніть Enter двічі:
Question: These 4 broads who criticize America, what country did they flee to get here? And now they want to make OUR America like
OUR country.
Many of the families who are being separated at the border and who are being kept in poor conditions did try to cross legally, thr
Hero Rohit Sharma love From Pakistan♦♦♦♦

1. Question: These 4 br... 0.6374067068099976 вміщає агресію
2. Many of the families... 0.3295410871505737 не вміщає агресію
3. Hero Rohit Sharma lo... 0.2612231373786926 не вміщає агресію
```

ПРИКЛАДИ ПОМИЛОК ВАЛІДАЦІЇ ВВЕДЕНОГО ТЕКСТУ

Помилка при виклику застосунку без вказаної порогової оцінки

```
PS C:\Users\Dmytro\Desktop\diplomaDocumentation\app\method> python .\main.py -t -n 0.33
Введіть текст (від 3 до 160 слів). Для завершення введення тексту натисніть Enter двічі:
Question: These 4 broads who criticize America, what country did they flee to get here? And now they want to make OUR America like
OUR country.
Many of the families who are being separated at the border and who are being kept in poor conditions did try to cross legally, thr
Hero Rohit Sharma love From Pakistan♦♦♦♦

1. Question: These 4 br... 0.6374067068099976 вміщає агресію
2. Many of the families... 0.32954108715057373 не вміщає агресію
3. Hero Rohit Sharma lo... 0.2612231373786926 не вміщає агресію
```

Результат передбачення з виводом помилок для текстів, що не пройшли валідацію

```
PS C:\Users\Dmytro\Desktop\diplomaDocumentation\app\method> python .\main.py -f .\texts.txt -n 0.33
1. Question: These 4 br... 0.6374067068099976 вміщає агресію
2. Many of the families... 0.32954108715057373 не вміщає агресію
3. Hero Rohit Sharma lo... 0.261223167181015 не вміщає агресію
4. i love you my bunny.... 0.2823463976383209 не вміщає агресію
5. Це нормальний текст ... Неправильна мова. Очікується англійська, знайдено: uk.
6. Small text... Неправильна довжина. Текст повинен містити від 5 до 100 слів.
```

ОБРАНІ МЕТРИКИ ДЛЯ ОЦІНКИ ЗАПРОПОНОВАНОГО МЕТОДУ

$$Loss = \frac{1}{n} \sum_{i=1}^{n} (y_i - j_i)^2$$

$$MAE = \frac{1}{n} \sum_{i=1}^{n} | y_i - j_i |$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - j_i)^2}{\sum_{i=1}^{n} (y_i - f_i)^2}$$

$$Accuracy = \frac{\sum_{i=1}^{n} e_i < 0.0025}{n}$$

УМОВИ ПРОВЕДЕННЯ ОЦІНЮВАННЯ РОБОТИ ЗАПРОПОНОВАНОГО МЕТОДУ

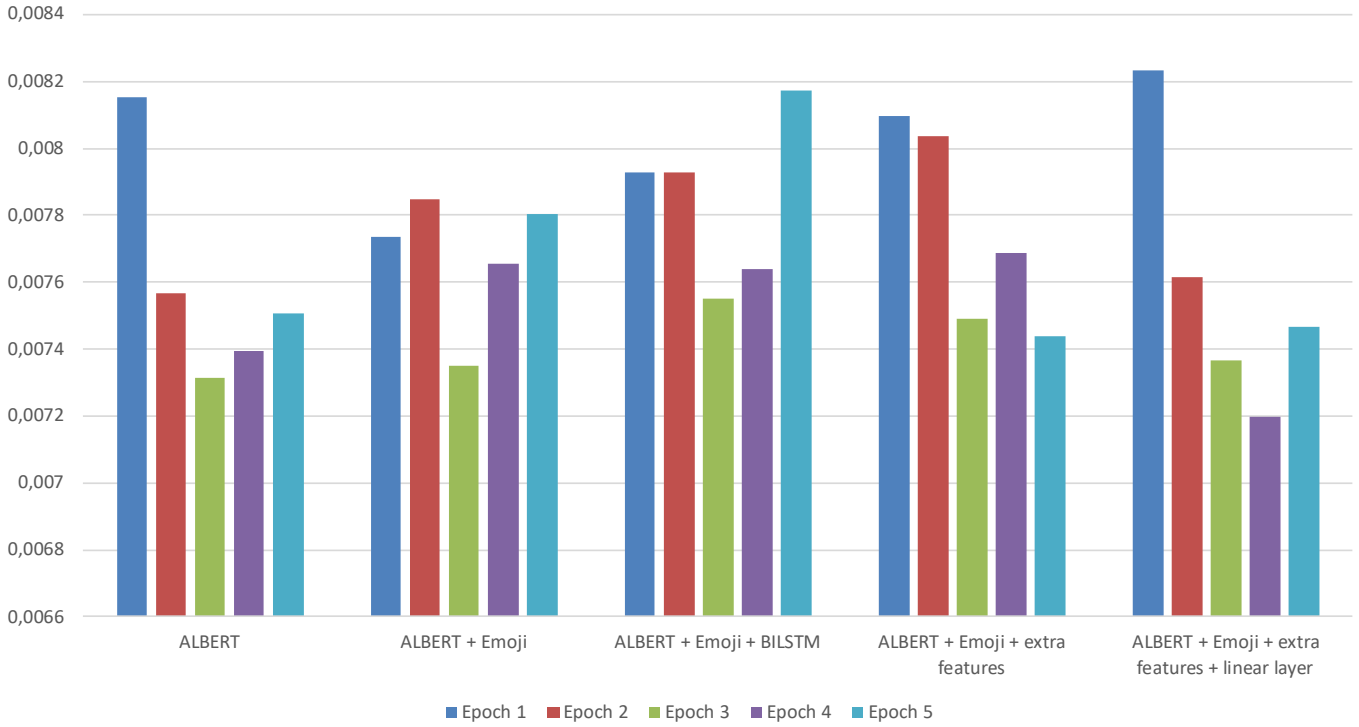
Тренувальний датасет - measuring-hate-speech.

Тренування проводилося за стратегією епох з наступними параметрами:

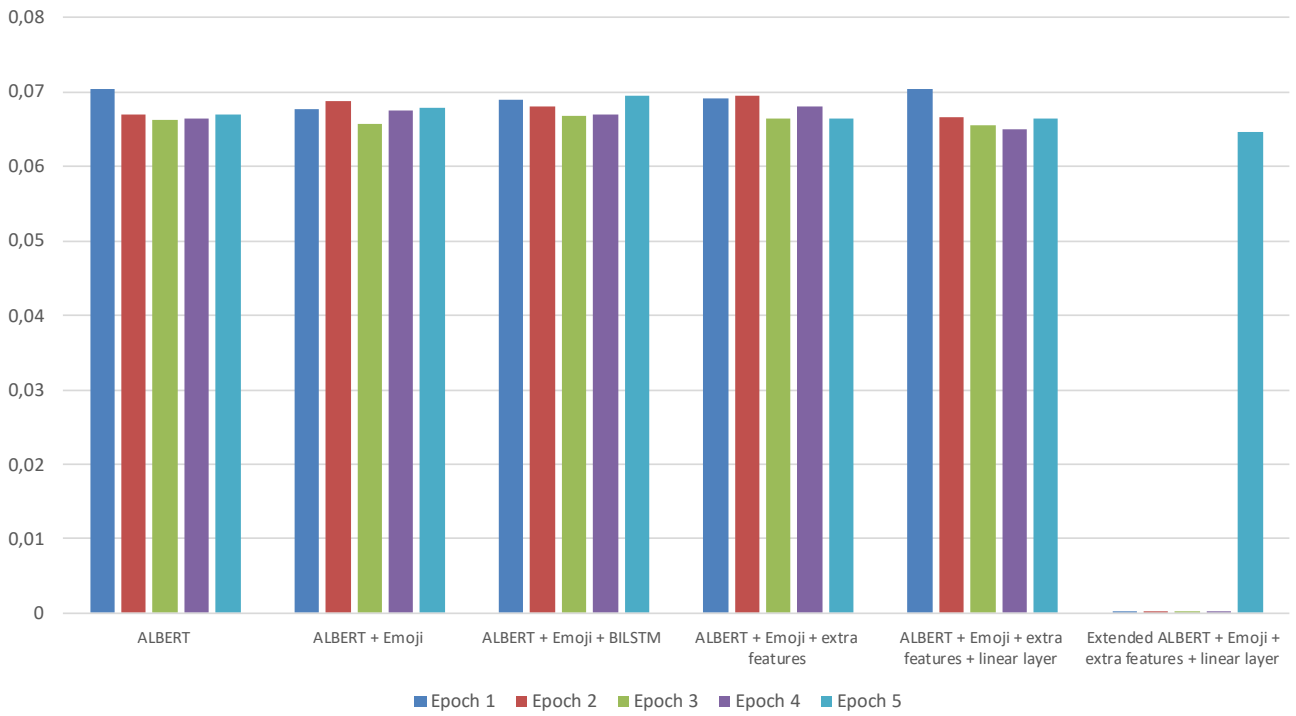
- темп навчання – 2*e*-5;
- розмір батчу – 32;
- кількість епох – 5.

Платформа для тренування та запуску – Google Colab.

РЕЗУЛЬТАТИ ВИМІРЮВАНЬ ЗНАЧЕНЬ МЕТРИКИ LOSS

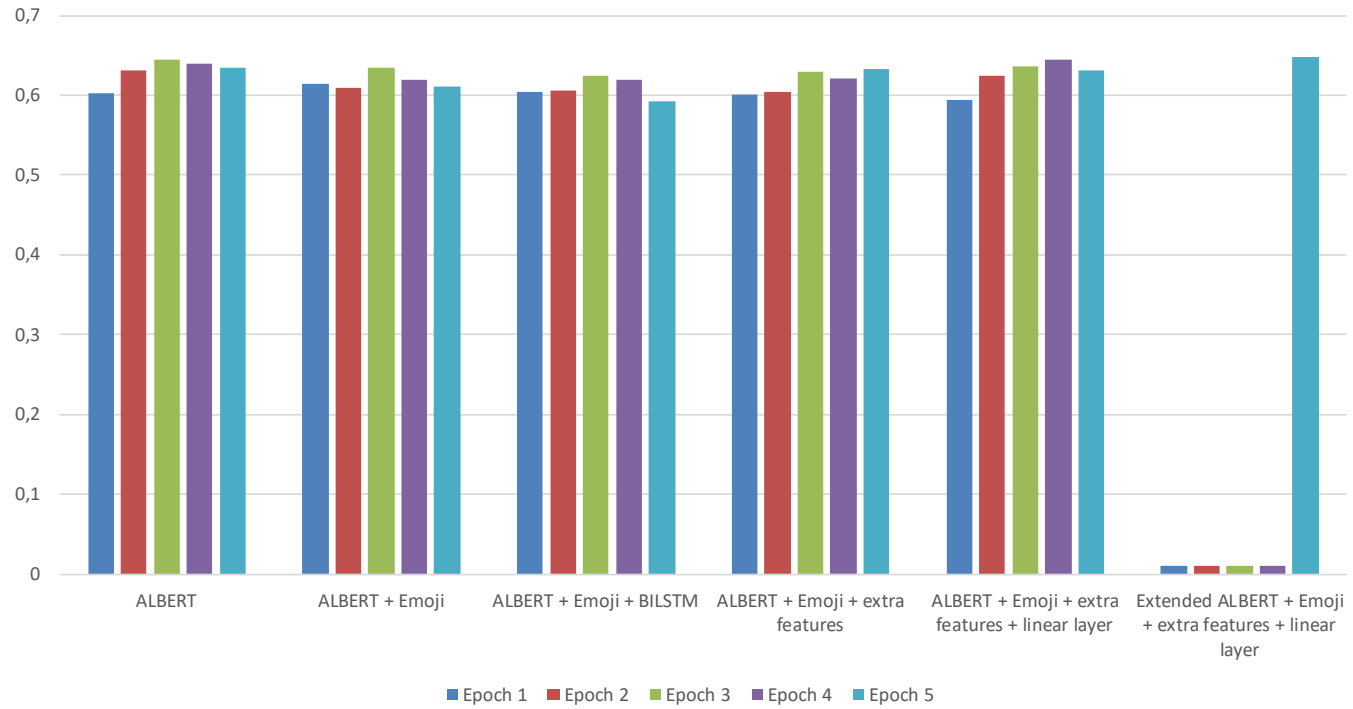


РЕЗУЛЬТАТИ ВИМІРЮВАНЬ ЗНАЧЕНЬ МЕТРИКИ MAE



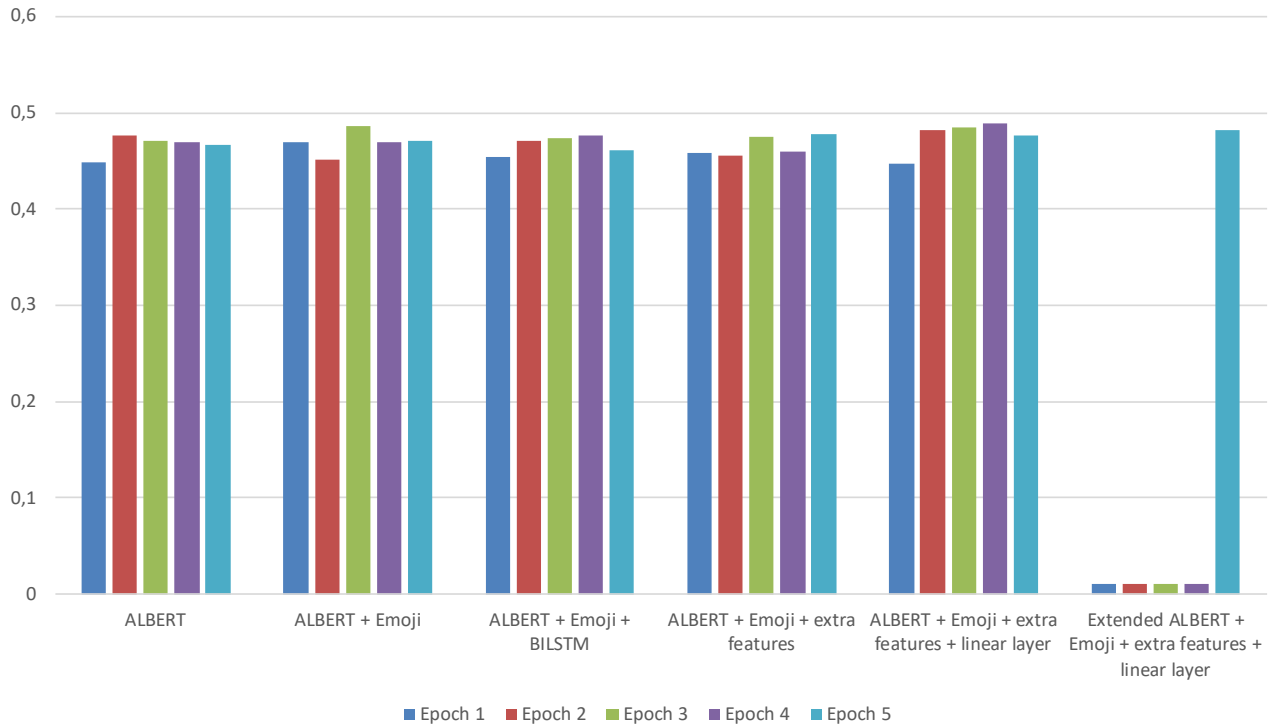
Відносне покращення точності запропонованого методу порівняно з базовим - **3,46%.**

РЕЗУЛЬТАТИ ВИМІРЮВАНЬ ЗНАЧЕНЬ МЕТРИКИ R^2



Відносне покращення точності запропонованого методу порівняно з базовим - **2,07%.**

РЕЗУЛЬТАТИ ВИМІРЮВАНЬ ЗНАЧЕНЬ МЕТРИКИ ACCURACY



Відносне покращення точності запропонованого методу порівняно з базовим - **3,9%.**

НАУКОВО-ІННОВАЦІЙНА НОВИЗНА

Уперше запропоновано метод автоматичної класифікації природномовних текстових даних за наявністю агресії, що базується на моделі ALBERT з використанням розширеної кількості шарів у поєднанні з додатковими структурними, лексичними, морфологічними ознаками, що підвищує точність визначення факту наявності агресії у тексті на 2-4% порівняно з базовим методом на основі ALBERT.

ПРАКТИЧНА ЦІННІСТЬ РОБОТИ

На основі запропонованого методу реалізовано програмне забезпечення автоматичної класифікації природномовних текстових даних за наявністю агресії. Розроблена програмна реалізація запропонованого методу може бути використана як окремий модуль для систем оброблення природної мови або соціальні мережі для автоматизації фільтрації текстових даних.

АПРОБАЦІЯ

Основні положення та результати роботи представлені та обговорювалися на науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК 2024 (20-22 листопада, 2024 р., м. Київ).

Розширений опис запропонованого методу обговорюватиметься на міжнародній науковій конференції «The 8th International Conference on Computer Science, Engineering and Education Applications (ICCSEEA2025)» (7-8 червня, 2025 р.).

УНІКАЛЬНІСТЬ РОБОТИ

За результатами перевірки пояснювальної записки унікальність роботи становить **96,5%**

29/32

НАПРЯМКИ ПОДАЛЬШОЇ РОБОТИ

1. Оптимізація використання ALBERT
2. Удосконалення оброблення емодзі
3. Застосування механізму уваги та BiLSTM
4. Зниження залежності від обчислювальних ресурсів
5. Мультимовна підтримка
6. Розширення доступності через Web API та GUI

ВИСНОВКИ

1. Виконано аналіз існуючих підходів до автоматичної класифікації природномовних текстів за наявністю агресії, зокрема методів сентимент-аналізу та сервісів онлайн-класифікації агресивних висловлювань.
2. Реалізовано багаторівневий метод автоматичного виявлення агресії в англомовних текстах, який поєднує контекстне векторне представлення на основі моделі ALBERT, векторизацію емодзі, а також структурні, лексичні та морфологічні ознаки.
3. Розроблено програмну реалізацію методу у вигляді консольного застосунку мовою Python, із використанням сучасних бібліотек глибокого навчання та обробки природної мови.
4. Проведено аналіз ефективності роботи запропонованого методу, за результатами якого визначено приріст точності від 2% до 4% за різними метриками у порівнянні з базовою реалізацією ALBERT, завдяки поєднанню глибинного контекстного аналізу з додатковими лінгвістичними ознаками.
5. Визначено обмеження запропонованого підходу, зокрема високе споживання обчислювальних ресурсів та відсутність мультимовної підтримки.
6. Окреслено напрями подальшої роботи, зокрема оптимізацію швидкодії за допомогою легких моделей, навчання на нових мовах та створення Web API й графічного інтерфейсу для зручності користувачів.

Дякую за увагу!

Питання?