ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА

Факультет математики, фізики та інформаційних технологій

Кафедра комп'ютерних систем і технологій

Кваліфікаційна робота

на здобуття рівня вищої освіти «бакалавр»

Розробка інформаційної моделі аналізу емоційного забарвлення текстових повідомлень із застосуванням методів інтелектуального аналізу даних та глибинного навчання

Development of an information model for analyzing the emotional coloring of text messages using data mining and deep learning methods

Виконав: студент денної форми навчання
спеціальності 122 – Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

Костенко Данило Русланович
(прізвище, ім'я, по-батькові)

Керівник д.т.н професор Приходько С.Б.
(науковий ступінь, вчене звання, прізвище та ініціали, підпис)

Рецензент к.ф.-м.н доцент Шугайло Ю.Б.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:

Протокол засідання кафедри

№____від «___»_____2025 р.

Завідувач кафедри

_____ Юрій ГУНЧЕНКО
(підпис) (ім'я, прізвище)

Захищено на засіданні ЕК № ____

протокол № ___від «___» _____2025 р.

Оцінка _____/_____/_____
(за національною шкалою, шкалою ECTS, бали)

Голова ЕК

_____ Микола МАЛАКСІАНО
(підпис) (ім'я, прізвище)

Одеса - 2025

АНОТАЦІЯ

У кваліфікаційній роботі розробляється тема «Розробка інформаційної моделі аналізу емоційного забарвлення текстових повідомлень із застосуванням методів інтелектуального аналізу даних та глибинного навчання».

Мета роботи – розробити та оцінити модель мульти-міткової класифікації емоцій на основі трансформерних архітектур у текстових даних, використовуючи повний датасет GoEmotions. Застосовуючи 27 відібраних емоційних категорій та сучасні трансформерні архітектури, дослідження прагне вийти за межі наявних рішень, заснованих на меншій кількості міток або одно-мітковому підході, та досягнути якомога кращого результату з-поміж обраного емоційного спектра.

В результаті проведених в роботі експериментів з використання різних інструментів тренування та тонкого налаштування моделі обробки природної мови було досягнуто задовільного результату успішності прогнозування моделлю заданих емоційних категорій.

ABSTRACT

The qualification work develops the topic “Development of information model for analyzing emotional coloring of text messages with application of data mining and deep learning methods”.

The aim of the work is to develop and evaluate a model of multi-label emotion classification based on transformational architectures in text data using the full GoEmotions dataset. By applying 27 selected emotional categories and state-of-the-art transformational architectures, the study aims to go beyond the limits of existing solutions based on fewer labels or a single-label approach and achieve the best possible result among the selected emotional spectrum.

As a result of the experiments conducted in this work on the use of various training tools and fine-tuning of the model of natural language processing, a satisfactory result of the success of the model in predicting the given emotional categories was achieved.

ЗМІСТ

ВСТУП

На сьогодні обсяг опублікованого у вільний доступ користувачами текстового контенту, від будь-яких публікацій в соціальних мережах чи текстових повідомлень в особистому спілкуванні до відгуків на продукти та дискусій на форумах, значно зріс. Подібна неструктурована текстова інформація часто насичена різноманітним емоційним забарвленням, яке, якщо правильно визначене та інтерпретоване, має можливість надати цінні висновки для покращення різних сфер діяльності людини, таких як сервісна підтримка користувачів, модерація контенту, цільова реклама, чи соціальні дослідження. Традиційні моделі глибокого навчання, що працюють з одно-мітковим аналізом не завжди можуть повністю покрити спектр навантаження тексту, та хоча дослідження моделей, заснованих на мульти-мітковому принципі, продовжується, вони зазвичай обмежені в кількості класів, що оглядаються.

Перспектива використання великих датасетів як GoEmotions, з двадцяти семи можливими мітками та вручну анотованими текстовими даними, а також швидкий розвиток мовних моделей постійно відкривають нові шляхи в сфері класифікації емоцій, зокрема для класифікації з використанням декількох міток одночасно. Тим не менше, наявні дослідження фокусуються на зменшеній кількості оглянутих міток, або з початку назначають кожному екземпляру даних лише одну домінантну емоцію. Такий підхід дозволяє значно підвищити точність моделей у результаті, і хоча декілька успішних варіантів вже було створено, як наслідок вони не цілісно відтворюють складну природу людської комунікації.

Тому, розробку і тестування моделі з фокусом на більшу кількість потенційних міток можна назвати вчасним наступним кроком. Досягнення визначених тестових метрик, як макро-F1 більше за 0.6, не тільки доведе якість моделі, але й допоможе сфері рухатися далі в сторону більш широких і контекстуалізованих моделей.

Об’єктом дослідження є процес автоматичного розпізнавання емоцій у написаному тексті. Точніше, це охоплює повний алгоритм збору даних, препроцесингу тексту, дизайн моделі, тренування або тонке налаштування, валідацію та її оцінку. Фокусуючись на більш широкому феномені емоційно наповненого текстового контенту, дослідження охоплює частини глибинного навчання, зокрема, інтелектуальний аналіз даних, обробки природної мови та афективних технологій.

Предмет дослідження – інформаційна модель мульти-міткової класифікації на основі трансформера налаштована на датасет GoEmotions, що включає 58000 вручну позначених прикладів, класифікованих з-поміж 27 можливих емоційних категорій. Дослідження окремо зосереджене на визначенні ефективності тонко налаштованих трансформерних архітектур (BERT, RoBERTa) для точного визначення декількох емоційних категорій в кожному варіанті даних одночасно. Основні аспекти включають оптимізацію технік препроцесингу, вибір стратегій ембедингу та підбір параметрів архітектури для досягнення якомога кращого показника макро-F1.

Метою дослідження є розробити та оцінити модель мульти-міткової класифікації емоцій на основі трансформерних архітектур у текстових даних, використовуючи повний датасет GoEmotions. Застосовуючи 27 відібраних емоційних категорій та сучасні трансформерні архітектури, дослідження прагне вийти за межі існуючих рішень, заснованих на меншій кількості міток або одно-мітковому підході, та досягнути якомога кращого результату з-поміж обраного емоційного спектру.

Для досягнення поставленої мети перед дослідженням стоять такі завдання:

- Аналіз датасету GoEmotions. Необхідно перевірити розподіл даних, баланс класів поміж мітками, та визначити проблеми що вимагають індивідуальних підходів.

- Препроцесинг текстових даних. Залучити чистку даних, токенізацію та нормалізацію. Налаштувати класи згідно з їх репрезентативністю для кращого результату
- Дизайн та тонке налаштування трансформерних архітектур. Адаптувати сучасні натреновані моделі для наявних даних та оптимізувати гіперпараметри тренування.
- Визначити критерії оцінки. Для мульти-міткових завдань найбільш репрезентативним є критерій макро-F1, що зважує всі мітки окремо. Визначити цільові пороги валідації міток.
- Аналіз помилок та порівняльний аналіз. Визначити найбільш проблемні класи емоцій, найпоширеніші помилки класифікацій та порівняти модель з базовими підходами.
- Синтез результатів і рекомендацій. Визначити, чи модель відповідає поставленим цілям, її використання в практичних задачах і можливі напрямки майбутнього розвитку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Еволюція підходів

Розпізнавання емоцій в тексті пройшло декілька окремих фаз, кожна з яких була викликана просуваннями в лінгвістичній теорії, збільшенням обчислювальних можливостей комп’ютерів, та наявністю даних. Ранні спроби покладалися на засновані на лексиконі і правилах системи, в яких вручну створені словники визначали індивідуальні слова чи фрази до окремих емоційних категорій. Такі методи забезпечували зрозумілість та низьке навантаження на обчислювальні машини, але були мало чутливі до контексту, полісемії, та тонкого прояву людей.

Зі зростальним інтересом в статистичному моделюванні, сфера зрушилась ближче до класичних методів машинного навчання. Традиційні алгоритми навчання з учителем, включаючи наївний баєс, метод опорних векторів, та дерева рішень, довгий час застосовувались для задачі емоційної класифікації тексту. За цими підходами кожне повідомлення, або будь-який фрагмент даних, перетворюється на вектор ознак фіксованої довжини часто використовуючи модель bag-of-words, ваги TF-IDF, чи вручну задані лінгвістичні ознаки, і потім передається в класифікатор, що навчається відповідності цих векторів до одної або декількох міток емоцій.

Моделі наївного Байєса оцінюють ймовірність кожної емоції, враховуючи наявність (або частоту) слів у вхідних даних, припускаючи умовну незалежність між ознаками. Незважаючи на свою простоту і сильні припущення про незалежність, наївний Байєс часто забезпечує напрочуд надійну базову продуктивність і вимагає мінімального часу на навчання.

Метод опорних векторів (SVM) шукає оптимальну гіперплощину, яка відокремлює екземпляри різних класів емоцій у високорозмірному просторі ознак. Використовуючи функції ядра (наприклад, лінійні або RBF), SVM

може фіксувати нелінійні межі рішень, що робить їх добре придатними для задач, де емоції демонструють складні шаблони в текстових ознаках.

Дерева рішень рекурсивно розбивають простір ознак на основі порогових значень окремих ознак (наприклад, наявність певного слова або фрази), створюючи дерево правил прийняття рішень, яке легко інтерпретувати. Хоча окремі дерева можуть бути надмірно пристосованими, ансамблі, такі як випадкові ліси або дерева з градієнтним підсиленням, пом'якшують цю проблему, об'єднуючи кілька дерев для покращення узагальнення.

Кожен з цих методів значною мірою залежить від якості вхідних ознак: ретельний відбір ознак, нормалізація та обробка дисбалансу класів є критично важливими для досягнення високої продуктивності в умовах мульти-міткової класифікації.

Також, результативність класифікаторів значно залежить від якості ознак визначених з тексту, процесу відомого як конструювання ознак. Зазвичай він включає конвертування необроблених текстових даних в структуровано репрезентовану числами, що відповідають необхідній семантичній, стилістичній, чи синтаксичній інформації.

До поширених ознак можна віднести bag-of-words, що охоплюють присутність і частоту слів в різних контекстах не звертаючи уваги на його значення чи порядок, ваги TF-IDF, що зважують слова згідно з їх важливістю відносно всієї інформаційної одиниці, визначені частини мови, що надають контекст до відповідної структури, оцінки сентименту та метрики читабельності, використання пунктуації чи довжини повідомлення, що також можуть ідентифікувати емоційний тон.

Хоча такі ознаки можуть бути ефективними для використання в окремих випадках, вони мають критичні обмеження. Ручне визначення факторів/ознак потребує великого обсягу часу і часто працює лише у вузько-напрямлених напрямках, що ускладнює перенесення моделей на інші датасети чи мови. Тому такі ознаки мають складності з визначенням більш

глибокої семантики і нюансів контексту природної мови, що є необхідним при виконанні задачі емоційної класифікації.

Ці обмеження конструювання ознак і мотивували перехід до підходів глибокого навчання, що автоматично вивчали ієрархічні й орієнтовані на контекст представлення з необробленого тексту без необхідності використання вручну заданих правил чи словників, що своєю чергою призводило до зменшення залежності від ручного препроцесингу.

Саме нещодавно глибоке навчання стало основною парадигмою в тренуванні моделей природної мови, зокрема в задачах емоційної класифікації тексту, в основному через здатність автоматичного аналізування ознак необроблених текстових даних і моделювання складних нелінійних відносин між точками даних. На відміну від традиційного машинного навчання, що покладалося на кероване створення подібних ознак вручну, моделі глибокого навчання виділяють абстракції високого рівня напряму з тексту під час тренування.

Рекурентні нейронні мережі, в особливості архітектури LSTM (довготривала короткочасна пам’ять) та GRU (керовані рекурентні блоки), були одними з найперших глибоких моделей використаних для задачі емоційного аналізу. Ці мережі здатні моделювати послідовні дані та виділяти залежності слів протягом часу, що робить їх кваліфікованими для репрезентації емоційного тону речень або повідомлень. Тим не менш, рекурентні нейронні мережі обробляють текст послідовно, що обмежує паралелізацію і може призводити до втрати ефективності на великих об’ємах даних. Це і призвело до переходу до використання трансформерів.

Поява трансформерних архітектур ознаменувала стрімку зміну в підходах до обробки природної мови, оскільки механізми самоуваги дозволили моделям вивчати контекстне представлення слів у спосіб, що добре піддається паралелізації. На відміну від попередніх рекурентних або згорткових підходів, трансформери обробляють всі вхідні послідовності одночасно, фіксуючи довгострокові залежності без серйозних проблем з

градієнтом, які характерні для рекурентних нейронних мереж. За останні кілька років спільнота дослідників розробила численні великомасштабні, попередньо навчені трансформерні моделі, зокрема BERT, RoBERTa та DeBERTa, які можна ефективно налаштовувати для подальших задач, включаючи класифікацію емоцій за кількома мітками.

Формально, такі моделі складаються зі стека кодерів (і, в деяких варіантах, стека декодерів), що складається з декількох рівнів самоуваги та підрівнів зворотного зв'язку. Кожен рівень самоуваги обчислює оцінки уваги між кожною парою токенів, створюючи зважену суму ембедингів токенів, яка відображає контекстні зв'язки. Потім ці репрезентації проходять через позиційні мережі прямого поширення, після чого відбувається нормалізація шарів і встановлення залишкових зв'язків. Оскільки механізм самоуваги добре масштабується на графічних процесорах, великі трансформерні моделі можна тренувати на великих масивах даних, вивчаючи нюанси використання мови.

1.2 Теоретичні особливості мульти-міткової класифікації

В одно-мітковій класифікації, кожній одиниці даних надається одна відповідна мітка зі заздалегідь визначеного набору міток. Формально, з вхідного тексту x прогнозується єдина категорія,$y \in \{1, \dots, C\}$де C – загальне число класів. До поширених прикладів входять полярні сентименти (позитивний, негативний, нейтральний) чи класифікація теми тексту (спорт, політика, ін.)

На відміну, мульти-міткова класифікація дозволяє відмітити кожен екземпляр даних будь-яким числом міток одночасно. В цьому випадку, кожній вхідній одиниці даних x протиставляється вектор,$y = [y_1, y_2, \dots, y_C]$ де кожний,$y_i \in \{0, 1\}$що означає присутність чи відсутність класу у відповідній одиниці даних. Таке формулювання є необхідністю у задачах, де категорії перетинаються чи з’являються одночасно, зокрема у емоційному аналізі.

Ключові відмінності:

- В задачах одно-міткової класифікації наявність одного класу виключає можливість наявності інших, що не є правдою для задач мульти-міткової класифікації.
- Найчастіше рішення задач мульти-міткової класифікації використовують індивідуальний поріг для кожної з визначених міток, в той час як для задач одно-міткової класифікації частіше використовують softmax для всіх класів.
- Для задач одно-міткової класифікації важливіші такі метрики як проста точність чи зважений F1, в той час, як макро-F1 надасть більш точну інформацію рівномірно зваживши всі класи для задач мульти-міткової класифікації

Для емоційного аналізу тексту природно більше підходить мульти-мітковий підхід, бо один емоційний тон не виключає наявності іншого. Іноді одне текстове повідомлення може передати декілька навіть протилежних станів.

Проблема формалізації задач мульти-міткової класифікації

Позначимо датасет як колекцію n пар вхід-вихід,$\{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$ де кожен *x* репрезентує текстову одиницю даних, а *y* – вектор міток, що їй відповідає. В умовах задачі мульти-міткової класифікації з *C* можливим числом міток, кожен *y* буде бінарним вектором розмірності *C*:

$$y^{(i)} = \left[y_1^{(i)}, y_2^{(i)}, \dots, y_C^{(i)}\right], y_j^{(i)} \in \{0,1\}, \quad (1.1)$$

де $y_j^{(i)} = 1$ означає наявність мітки *j*;

$y_j^{(i)} = 0$ – її відсутність.

Кожен вхід $x^{(i)}$ відображається у вектор ознак $h^{(i)} \in R^d$ за допомогою функції вбудовування. $f_{emb}: x \mapsto h$ В архітектурах на основі трансформерів f_{emb} може складатися з вбудовувань токенів, позиційних кодувань та

контекстних шарів, які створюють об'єднане представлення для всього тексту. Наступні рівні класифікації використовують $h^{(i)}$ для прогнозування ймовірностей міток.

Щоб організувати мітки для всіх n екземплярів даних використовується матриця бінарних міток $Y \in \{0,1\}^{n*C}$, де i-ий рядок Y це вектор $y^{(i)}$.

В такій матриці кожна колонка протиставляється одному емоційному класу, а кожен рядок – одному екземпляру даних. Таке представлення даних дозволяє побачити повний аналіз розподілення міток по датасету і їх відносин (наприклад, частоти зустрічі).

Аналіз структури Y часто включає такі обчислення

- Кардинальність міток – середнє число міток на один екземпляр

$$Cardinality(Y) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{C} y_j^{(i)}. \quad (1.2)$$

Більша кардинальність означає, що кожен екземпляр даних в середньому обозначений декількома мітками.

- Щільність міток – кардинальність нормалізована по C

$$Density(Y) = \frac{Cardinality(Y)}{C}. \quad (1.3)$$

Ця метрика дозволяє побачити стандартизовану міру розрідженості або наповненості призначених міток.

- Частота зустрічі – частота з якої мітки з'являються разом, частіше за все репрезентована через $C \times C$ матрицю зустрічності.

$$M_{jk} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{I}(y_j^{(i)} = 1 \wedge y_k^{(i)} = 1), \quad (1.4)$$

де I – функція індикації.

M_{jk} підраховує як часто мітки j та k зустрічаються разом, що демонструє поширені пари міток, як наприклад «радість + здивування» чи «сум + злість»

Якщо визначити задачу як декілька бінарних, то мульти-міткову проблему можна

Стратегії мульти-міткової класифікації

Binary relevance – це одна з найбільш прямих та зрозумілих стратегій мульти-міткової класифікації. Вона розбиває проблему на C паралельних задач бінарної класифікації. Для кожної мітки j вивчається окреме граничне рішення, щоб відрізнити позитивні екземпляри від негативних.

Формально, вивчається C таких функцій:

$$g_j(h) = \sigma\left(w_j h + b_j\right), j = 1, \dots, C, \quad (1.5)$$

де σ – це функція сигмоїдної активації;

w_j та b_j – параметри для мітки j.

Під час інференції кожна $g_j\left(h^{(i)}\right)$ зрівнюється з порогом τ_j для визначення $y_j^{(i)} = 1$ чи 0.

Для кожної емоції j тренується окремий класифікатор h_j для визначення її наявності (y_j = 1) чи відсутності (y_j = 0). Під час інференції, всі C класифікаторів працюють паралельно на одному векторі ознак h, та їх результати комбінуються в один фінальний бінарний вектор $\hat{y} = [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_C]$.

До переваг такого підходу можна віднести:

- Відносну простоту його імплементації за допомогою будь-якого готового бінарного класифікатора (наприклад, логістичної регресії чи SVM).

- Легке скалювання, оскільки кожна модель може навчатися незалежно, що дозволяє паралелізацію та більш легке налаштування гіперпараметрів.
- Гнучкість, адже нові мітки можна додавати без повторного тренування класифікаторів, що існують.

  Але, прямота бінарної класифікації призводить до деяких обмежень.

- Бінарна класифікація оглядає кожен емоційний клас як ізольований екземпляр, що може призводити до ігнорування потенційної кореляції між ними (наприклад, «злість» може часто зустрічатись разом з «сум»).
- Мітки, що з'являються в даних ріже і мають недостатню кількість позитивних екземплярів можуть призводити до нестабільних класифікаторів.
- Не беручи до уваги залежності між мітками, бінарна класифікація може призводити до маловірогідних комбінацій міток, або тих що не мають сенсу. Наприклад, «радість» та «сум» ніколи не помічені разом в наборі тренувальних даних, але класифіковані моделлю.

Ланцюги класифікаторів (Classifier Chains) доповняють бінарну класифікацію моделюючи залежність міток через ланцюг C бінарних класифікаторів. В заздалегідь обраному порядку кожен класифікатор h_j отримує не тільки вектор ознак h, а й бінарні прогнозування всіх попередніх класифікаторів. Формально, до h_j надходять.$[h, \hat{y}_1, \hat{y}_2, \dots, \hat{y}_{j-1}]$В момент інференції, прогнозування відбувається послідовно: перший класифікатор прогнозує результат, що додається на вхід другого класифікатору, і т.д.

Переваги:

- Через те, що кожен класифікатор покладається на попередній результат, весь ланцюг охоплює залежність між мітками, що призводить до більш смислових результатів.
- У багатьох випадках моделювання залежності міток покращує кінцеву точність чи оцінки F1 моделі, особливо коли одні емоції майже точно сигналізують появу інших.

Обмеження:

- Якість прогнозування може сильно варіюватись залежно від обраного порядку міток. Неоптимальний або випадково обраний порядок може знизити успішність моделі.
- Помилки спочатку ланцюга мають каскадний ефект, сильніше впливаючи на класифікатори, шо йдуть слідом.
- Вимоги до обчислювальних машин та використання пам’яті зростають бо кожен наступний класифікатор займає більше місця ніж попередній.

Для того, щоб уникнути упередженості викликаної порядком міток часто використовують ансамблі моделей з різними випадково обраними порядками, осереднюючи результат з-посеред декількох ланцюгів для стабілізації прогнозування.

Label Powerset (LP) – метод, що трансформує проблему мульти-міткової класифікації в проблему класифікації єдиного мульти-класу ставлячись до кожного унікального набору міток, що наявний в тренувальному наборі даних, як «супермітку». Конкретно, якщо тренувальний набір даних має m унікальних наборів міток,$\{L_1, L_1, \dots, L_m\}$LP протиставляє кожному з цих класів свій екземпляр «супермітки», і далі тренує модель як стандартний мульти-класовий класифікатор. В момент прогнозування, кожен індекс прогнозованого класу повертає назад відповідний набір міток.

Переваги:

- За визначенням, LP охоплює залежності між мітками, тому що кожний клас протиставляється відповідному набору міток.
- Через те, що лише один класифікатор оцінюється в момент прогнозування, це надає можливість дістати повний набір міток в один крок.

Обмеження:

- Кількість класів (можливих наборів міток) експоненційно зростає зі зростанням C кількості міток, що призводить до високих вимог обчислювальної до обчислювальної техніки.
- LP не може прогнозувати набори міток, що не були надані в тренувальному наборі даних, навіть якщо їх комбінація потенційно можлива.

Можна сказати, що BR потенційно найбільш гнучкий підхід, бо кожна мітка моделюється незалежно і нові мітки можуть бути додані без перетренування наявної моделі. CC зберігає велику частину подібної гнучкості, при цьому додаючи умовні залежності. LP найменш гнучкий з трьох, оскільки працює лише з наборами міток, що з'являються під час тренування.

З точки зору обчислювальної складності BR – найлегший метод, бо всі C натренованих моделей залишаються малими й незалежними. CC має схожий потенціал паралелізації, але збільшує час необхідний на тренування через залежність кожного наступного вектора від попереднього. LP, з іншого боку, може бути досить важким методом, якщо початкове число міток і їх комбінацій велике.

BR з-поміж трьох пропонує повну незалежність міток, що може нашкодити ефективності моделі якщо існують чіткі залежності між мітками. CC розроблений для моделювання залежностей, що покращує смислове навантаження результуючого набору міток, але може пропагувати хибнопозитивні помилки. В той час як LP повність кодує кожен спостережений набір окремо, але ризикує мати надмірну упередженість до конкретного набору міток.

На практиці зазвичай найкращим варіантом є гібридний або ансамблевий підхід, що дозволяє використати сильні сторони методів і одночасно позбутися помилок, що їм притаманні.

Функції активації та втрати

В задачах мульти-міткової класифікації необхідність вихідного шару працювати з декількома можливими мітками, а не лише з одним нормалізованим розподілом з-поміж взаємозаперечних класів прямо впливає на вибір функції активації і відповідної втрати та є критичним для того, щоб забезпечити незалежне визначення ймовірності появи кожної мітки та для ефективного тренування моделі з невеликого числа позитивних екземплярів даних.

Сигмоїдальна функція активації з бінарною крос-ентропійною функцією втрат застосовується поелементно до вихідних логітів, в результаті для кожної мітки *j*:

$$\hat{p}_j = \sigma(z_j) = \frac{1}{1+\exp(-z_j)}, \tag{1.6}$$

де z_j – логіт, що відповідає мітці j.

Область сигмоїду (0, 1) робить його природнім вибором для обчислення $\hat{p}_j$ як ймовірності того, що мітка j присутня незалежно від інших міток. Для тренування моделі бінарна кросс-ентропійна функція втрат обчислюється окремо для кожної мітки і потім сумується або зважується з-поміж всіх міток:

$$L_{BCE} = -\frac{1}{C}\sum_{j=1}^{C}[y_j \log(\hat{p}_j) + (1-y_j)\log(1-\hat{p}_j)]. \tag{1.7}$$

Така поелементна функція втрат понижує як хибно позитивні, так і хибно негативні появи класів міток, та покращує калібрацію $\hat{p}_j$.

Функція активації softmax змушує всі прогнозовані ймовірності по всіх класах в результаті сумуватися до 1:

$$\hat{p}_j = \frac{\exp(z_j)}{\sum_{k=1}^{C}\exp(z_k)}. \tag{1.8}$$

Хоча ця функція один з кращих варіантів для задач одно-міткової класифікації, softmax не є доречною для використання в мульти-міткових задач, бо вона штучно поєднує результати, тобто підвищення ймовірності появи одної мітки автоматично понижує ймовірність появи всіх інших. В мульти-міткових задач наявність одної мітки не виключає наявності інших, тому нормалізоване обмеження softmax напряму погіршує здатність модель обирати декілька міток одночасно та викривляє індивідуальні ймовірності міток.

Мульти-міткові датасети часто мають нерівномірний розподіл міток, деякі з'являються частіше ніж інші. Без врахування такого дисбалансу класів, модель може може недотренуватися на нечастих мітках, пріоритизуючи більш репрезентовані класи для того, щоб зменшити втрати. Дві поширені стратегії спрямовані на те, щоб виправити цю проблему:

- Кожній мітці назначається вага w_j, яка скалює участь мітки в результаті обчислення відповідно до її рідкості:

$$L_{weighted} = -\frac{1}{C}\sum_{j=1}^{C} w_j[y_j \log(\hat{p}_j) + (1 - y_j)\log(1 - \hat{p}_j)]. \qquad (1.9)$$

  Ваги можуть бути обернено пропорційні частоті появи міток чи визначені більш витонченими шляхами.

- Визначення окремого порогу для кожної мітки допомагає контролювати компроміс між точністю та відгуком на незбалансованих класах. Додатково, такі техніки як передискретизація рідких міток чи недодискретизація частих можуть призвести до більш збалансованих результатів.

Разом, комбінація сигмоїдної активації, бінарної крос-ентропії, порогів та стратегій дискретизації призводить до швидких стрибків в покращенні метрик моделі для мульти-міткових проблем зокрема. Така конфігурація зберігає незалежність прогнозу міток, дозволяє тонкий контроль над

тренуванням кожного класу та дозволяє використання датасетів, в яких частота міток сильно варіюється.

Мульти-мітковий підхід в задачі емоційної класифікації текстових даних

Людське вираження емоцій в тексті природно складний процес: єдине повідомлення може слугувати провідником декількох емоцій одночасно, тому строге категоризування емоцій у взаємозаперечні категорії ризикує спростити задачу за межі реального застосування. Мульти-мітковий підхід напряму відповідає на цей виклик дозволяючи кожній одиниці даних переносити декілька емоційних тегів, тобто моделює тонкий підхід до людської комунікації.

На відміну від одно-міткових систем, мульти-міткові дозволяють емоційним класам з’являтись одночасно. Наприклад, повідомлення відсвяткування може нести як радість, так і очікування, а рефлективна публікація може мати суміш суму та ностальгії. Прибираючи обмеження взаємозаперечення, мульти-міткові класифікатори можуть назначати кожну мітку на єдиний екземпляр даних, що підходить вхідному тексту, що збільшує відгук для складного емоційного контенту та уникає вимушених рішень, які спотворюють намір автора.

Масив даних GoEmotions складається з більш ніж 58,000 коментарів із соцмережі Reddit, кожен з яких помічений одною або декількома класами з 27 запропонованих, що дозволяє відстежити залежність між мітками, що часто з’являються разом. Аналіз одночасних появ показує, що:

- Злість та сум часто з’являються разом в повідомленнях, що згадують скорботу та несправедливість.
- Радість та здивування з’являються в публікаціях, що описують неочікувані позитивні події.
- Страх та огида іноді поєднуються в описах огидного чи небезпечного досвіду.

Ці пари емоцій разом складають велику частину мульти-міткових екземплярів датасету. Наприклад, понад 12% коментарів позначених «радість» також позначені класом «сюрприз», що підкреслює важливість охоплення поєднаних емоційних сигналів замість нав’язування єдиного вибору.

Точне моделювання спільної присутності міток має вирішальне значення для подальших застосувань:

- Системи рекомендацій, що враховують емоції, можуть точніше адаптувати рекомендації, коли вони розуміють, що відгук користувача виражає як хвилювання, так і занепокоєння.
- Інструменти моніторингу психічного здоров'я виграють від розрізнення тривоги (страху) в поєднанні з безнадією (сумом) і лише страху, що призводить до більш тонкої оцінки ризиків.
- Аналіз відгуків клієнтів у галузевих умовах набуває глибини, коли повідомлення, які виражають розчарування (гнів), також розкривають приховане розчарування (смуток), що дає змогу розробити більш ефективні втручання.

Крім того, правильно інтерпретовані багатозначні результати роблять можливою багатшу візуалізацію, наприклад, теплові карти емоцій або мережі спільних проявів, надаючи чіткіше уявлення про тенденції колективних настроїв.

Застосовуючи стратегію класифікації за кількома мітками, моделі, навчені на GoEmotions, можуть одночасно виявляти всі релевантні емоційні нюанси, присутні в тексті. Ця комплексна можливість виявлення не тільки покращує метричні показники (наприклад, макро-F1 за кількома мітками), але й забезпечує більш достовірне уявлення про те, як емоції проявляються в природній мові. Така точність має вирішальне значення для будь-якої реальної системи, яка має на меті розуміти, реагувати або прогнозувати людські почуття на основі текстових даних.

1.3 Метрики для оцінки якості моделей глибокого навчання

Попри те, що кожен екземпляр даних в мульти-міткових проблемах може бути асоційований з декількома мітками, за своєю суттю модель ставиться до кожної мітки як до окремого бінарного рішення. Відповідно, стандартні метрики класифікації, такі як точність, відгук чи оцінки F1, залишаються фундаментальними інструментами оцінки роботи моделі. З усім тим, підхід до їх обчислення має бути обережно обраний з огляду на набір міток.

Точність (Precision) визначається для окремої мітки як відношення справжніх позитивних прогнозів (TP_i) до всіх позитивних прогнозів для цієї мітки ($TP_i + FP_i$).

Для мітки L_i:

$$Precision_i = \frac{TP_i}{TP_i+FP_i}. \quad (1.10)$$

Висока оцінка точності означає, що модель зазвичай правильно прогнозує появу мітки, тобто робить небагато хибно позитивних тверджень. В контексті емоційного аналізу висока точність для «злості» означає, що більшість екземплярів появи цієї мітки правильно помічені моделлю.

Відгук або чутливість (Recall) виміряє пропорцію справжніх позитивних прикладів, які модель успішно ідентифікувала.

Для мітки L_i:

$$Recall_i = \frac{TP_i}{TP_i+FN_i}. \quad (1.11)$$

Високий відгук для мітки означає, що модель пропускає небагато її екземплярів. Наприклад, високий відгук для «радості» означає, що більшість прикладів появи цієї мітки правильно відмічені моделлю. Технічно, якщо

абсолютно всі екземпляри будуть помічені міткою, то відгук для неї буде максимальним, на відміну від точності. На практиці максимізація відгуку забезпечує прогнозування недорепрезентованих, рідких або критичних міток.

Оцінка F1 – це метрика, що створена для усереднення точності та відгуку, вона балансує їх за допомогою обчислення їх гармонійного середнього.

$$F1_i = 2 * \frac{Precison_i * Recall_i}{Precision_i + Recall_i}. \quad (1.12)$$

Значення гармонійного середнього стрімко зменшиться зі збільшення дисбалансу класів, тобто якщо будь-яка з оцінок буде малою, F1 теж буде малою. Така поведінка робить F1 найкращою метрикою для задач мульти-міткової класифікації, зокрема для емоційного аналізу.

Але, частіше за все під час дослідження є корисним звертати увагу на всі метрики окремо, бо покращення різних метрик відбувається в різний спосіб. Для задачі мульти-міткового емоційного аналізу зокрема всі ці метрики, обчислені для кожної мітки, дають комплексне уявлення про поведінку моделі у різноманітному емоційному спектрі.

Агреговані метрики F1

При роботі з задачами мульти-міткової класифікації кінцева оцінка результативності моделі зазвичай вимагає збору оцінок для всіх міток у значущі глобальні показники. Агреговані метрики F1 є одним з найпоширеніших інструментів для досягнення цієї мети, вони показують загальну ефективність моделі з урахуванням дисбалансу класів і структурних особливостей даних.

Для розширення F1 на всі k метрик в мульти-мітковому контексті в основному використовуються дві основні стратегії:

- Макроусереднення обчислює метрику окремо для кожної мітки і потім обчислює їх незважене середнє. Тобто, при такому підході кожна мітка має однаковий вплив на результат, незалежно від її репрезентованості в

наборі даних, що робить його більш інформативним, коли продуктивність моделі на рідких мітках є критичною.

$$F1_{macro} = \frac{1}{k}\sum_{i=1}^{k} F1_i, \quad (1.13)$$

де k – загальна кількість міток.

Зокрема для задачі емоційної класифікації, класи як «скорбота» чи «полегшення» скоріше за все будуть зустрічатися набагато рідше, за, наприклад, «нейтральний». Високе значення макро-F1 означає, що модель добре справляється як з частими, так і з рідшими мітками, що робить його важливою метрикою для збалансованої оцінки.

- Мікроусереднення, навпаки, ставиться до всіх індивідуальних бінарних рішень рівноцінно, за допомогою сумування справді позитивних, хибно позитивних, та хибно негативних результатів по всім міткам перед сумуванням метрики. Такий підхід в результаті зважує кожну пару екземпляр-мітка однаково, тому на обчислення більше впливають мітки з великою кількістю екземплярів даних.

$$F1_{micro} = \frac{2*\sum TP}{2*\sum TP+\sum FP+\sum FN}. \quad (1.14)$$

Тобто, чим краще модель справляється з прогнозуванням більш популярних міток, тим вище буде значення метрики. Хоча мікро-F1 може запропонувати високу оцінку навіть якщо модель погано справляється з рідшими класами, вона є корисною для того, щоб побачити загальний рівень успішності моделі, коли стоїть ціль максимізувати число правильних прогнозів.

- Зважене усереднення – це компроміс між мікро-F1 та макро-F1. Вона обчислюється за допомогою зваження оцінки F1 кожної мітки відповідно до кількості екземплярів появи мітки.

$$F1_{weighted} = \sum_{i=1}^{k} \frac{n_i}{N} * F1_i, \quad (1.15)$$

де n_i – кількість істинних екземплярів мітки;

N – загальна кількість екземплярів в датасеті.

Такий підхід забезпечує більший вплив для більш частих міток, але надає кожній мітці можливість брати участь в результаті оцінки.

Вибір між цими метриками залежить від характеристик датасету, з яким працює модель, та від специфіки її застосування. В задачах емоційної класифікації, де деякі класи міток часто є недорепрезентованими, макро-F1 вважається кращою метрикою для вибору моделі, в той час, як мікро-F1 є корисною для визначення її загальної надійності. Зважена міра F1 особливо актуальна в умовах, де часті емоції (наприклад, «радість», «зацікавленість») повинні бути в пріоритеті, не ігноруючи менш поширені класи.

Метрики множин міток

Часто є важливим оцінити не тільки прогнозування мітки окремо, а також здібність моделі прогнозувати правильну комбінацію міток для конкретного екземпляра. Для оцінки моделі на рівні наборів, що відображає «цілісність» призначення міток для кожного екземпляру даних, можна використати такі метрики:

- Втрата Хаммінга (Hamming Loss) вимірює частину міток, які прогнозуються неправильно, беручи до уваги як хибно позитивні так і хибно негативні результати. На відміну від метрик, що концентруються на мітках окремо, втрата Хаммінга відображає середнє помилок з-поміж всіх міток і екземплярів.

$$Hamming\ Loss = \frac{1}{N*k} \sum_{j=1}^{N} \sum_{i=1}^{k} 1\left[\hat{y}_{j,i} \neq y_{j,i}\right], \quad (1.16)$$

де N – кількість екземплярів даних;

k – кількість міток;

$\hat{y}_{j,i}$ – прогнозоване бінарне значення для мітки i екземпляру j.

Менше значення втрати Хаммінга означає, що в середньому менша кількість прогнозованих міткою є неправильною. Ця метрика особлива корисна, коли всі помилки по всім міткам розглядаються як рівноцінні, попри спостережені шаблони зустрічності.

- Точність підмножин (Subset Accuracy) є найбільш строгою метрикою на рівні наборів. Вона підраховує частину екземплярів для якої модель прогнозує повний набір міток ідеально:

$$Subset\ Accuracy = \frac{1}{N}\sum_{j=1}^{N} 1\left[\{\hat{y}_{j,*}\} = \{y_{j,*}\}\right]. \quad (1.17)$$

де $\{\hat{y}_{j,i}\}$ та $\{y_{j,*}\}$ означають набори прогнозованих та істинних міток відповідно для екземпляра j.

Так як навіть одна неправильно прогнозована мітка позначає весь приклад як неправильний, точність підмножин дає досить строгу оцінку продуктивності. Вона є найбільш інформативним, коли точні комбінації міток є критично важливими, наприклад, у програмах, де відсутність або додавання будь-якої мітки суттєво змінює подальшу інтерпретацію.

- Індекс Джаккара (Jaccard Index) обчислює перетин прогнозованих та істинних наборів міток, що дозволяє йому бути помірно строгим.

$$Jaccard(j) = \frac{[\{\hat{y}_{j,*}\}\cap\{y_{j,*}\}]}{[\{\hat{y}_{j,*}\}\cup\{y_{j,*}\}]},\ Jaccard\ Index = \frac{1}{N}\sum_{j=1}^{N} Jaccard(j) \quad (1.18)$$

Ця метрика нагороджує частково правильне прогнозування: прогнозовані мітки впливають позитивно на результат, в той час як надлишкові чи відсутні мітки зменшують його. Таким чином, середнє значення індексу Джаккара для всіх випадків відображає загальну

здатність моделі апроксимувати справжні набори міток, що робить її популярним вибором для мульти-міткового оцінювання.

Поєднуючи втрату Хаммінга, точність підмножин та індекс Джаккара, можна отримати детальну картину поведінки моделі: втрата Хаммінга визначає середній рівень помилок на кожну мітку, точність підмножин вимірює точні прогнози набору, а індекс Джаккара фіксує ступінь перекриття. Разом ці метрики забезпечують ретельну оцінку мульти-міткової послідовності у задачах класифікації.

ROC-AUC для мульти-міткової класифікації

Крива ROC (Receiver Operating Characteristic) і пов'язана з нею оцінка площі під кривою (AUC – Area Under The Curve) є широко використовуваними незалежними від порогу метриками для оцінки бінарних класифікаторів. У мульти-міткових контекстах ROC-AUC надає додаткову перспективу до F1 або втрат Хаммінга, кількісно оцінюючи здатність моделі ранжувати позитивні екземпляри вище, ніж негативні, в масиві порогів прийняття рішень.

Для кожної мітки L_i будується ROC-крива, яка відображає відношення частоти істинних спрацьовувань (TPR) до частоти хибних спрацьовувань (FPR) при різних порогах класифікації. Значення AUC для мітки L_i, позначене AUC_i, представляє ймовірність того, що випадково вибраний позитивний приклад для L_i отримає вищу оцінку, ніж випадково вибраний негативний. Значення AUC_i 0,5 вказує на відсутність дискримінаційної здатності (випадкове вгадування), тоді як значення близько 1,0 сигналізує про відмінну відокремлюваність. Для задачі емоційної класифікації AUC для кожної мітки показує, наскільки добре модель класифікує повідомлення, що містять певну емоцію, порівняно з тими, що її не містять.

Щоб отримати загальний показник, який однаково враховує всі мітки, обчислюють незважене середнє значення AUC для кожної з них:

$$AUC_{macro} = \frac{1}{k}\sum_{i=1}^{k} AUC_i, \tag{1.19}$$

де k – кількість міток.

Макро-AUC підкреслює здатність моделі правильно прогнозувати складні або рідкі мітки, бо вона зважує всі міти рівноцінно.

Відповідно, мікро-AUC об'єднує всі прогнози на основі міток в єдину об'єднану ROC-криву, підсумовуючи істинні та хибні спрацьовування для всіх міток, перш ніж обчислювати TPR та FPR. Як і з мікро-F1, ця метрика розглядає кожне індивідуальне рішення однаково, надаючи більшої ваги загальним емоціям, які домінують у наборі даних.

В основі як макро-, так і мікростратегій лежить оцінка «один проти решти» (OvR): кожна мітка оцінюється проти всіх інших як негативна. Це дає змогу легко поширити бінарний ROC-аналіз на мульти-міткові проблеми, не вимагаючи принципово нової математики. Однак OvR передбачає незалежність міток, що може не врахувати складні шаблони спільної присутності, притаманні даним про емоції.

ROC-AUC залишається стійким до дисбалансу класів, оскільки він оцінює якість ранжування, а не абсолютну кількість класифікацій. Ця властивість особливо цінна, коли певні емоції з'являються нечасто, але їх все одно потрібно надійно розрізняти. З іншого боку, AUC може маскувати проблеми з калібруванням і не відображає безпосередньо продуктивність при певних порогових значеннях класифікації, що може бути критично важливим для додатків із суворими обмеженнями на помилкові спрацьовування.

На практиці, використовуючи ROC-AUC в поєднанні з іншими метриками, дослідники отримують детальне розуміння поведінки моделі в заданому спектрі, не обмежуючись одним пороговим значенням.

Вибір метрик та порогів класифікації

Зазвичай в задачах мульти-міткової класифікації результатом прогнозування є набір ймовірностей для кожної з міток. Для того щоб виділити, які мітки позначити як істинні для відповідного прикладу даних, необхідно виставити порогове значення ймовірності, результати вище якого

будуть вважатися правильними, нижче – хибними. Відповідно, визначення метрик, які будуть оптимізуватися під час навчання моделі, впливає на визначення порогу істинності та загальну поведінку системи.

Перед тим як розпочинати налаштування порогів, важливим кроком є визначення які агреговані чи поміткові метрики пріоритизувати. Наприклад, якщо хибно негативні результати для міток «страх» чи «сум» несуть високий ризик у реальному застосуванні (наприклад, в задачах моніторингу психічного здоров'я), дослідники можуть визначити «відгук за умови фіксованої точності» як цільову метрику. Альтернативно, коли збалансована продуктивність по всім міткам це бажаний результат, то оптимізація макро-F1 може бути кращим вибором. Визначення подібних цілей направляє процес пошуку порогів та дозволяє уникнути довільного або занадто широкого рішення.

Поширена стандартна практика – це використання одного уніфікованого порогу для всіх міток (часто це 0.5). І хоча це досить прямий підхід, він упускає нюанси задачі та датасету. Зокрема для задач емоційної класифікації є притаманним дисбаланс класів, що призводить до стабільно нижчих оцінок ймовірності істинності. Для вирішення цієї проблеми застосовується стратегія підбору окремого порогу для кожної з міток. Після тренування моделі сканується масив можливих значень порогу (наприклад, від 0.1 до 0.9 з інкрементом 0.01) для кожної мітки окремо на обраному тестувальному наборі даних та обирається значення яке максимізує обрану метрику (наприклад, макро-F1)

Добре відкалібровані оцінки ймовірності сприяють більш надійному встановленню порогу. Методи калібрування (наприклад, шкалювання Платта або ізотонічна регресія) можуть бути застосовані після навчання для узгодження прогнозованих ймовірностей з істинними ймовірностями. Після калібрування вибір порогу стає більш стабільним та легшим для інтерпретування. Крім того, можна застосувати прості правила постпроцесингу, наприклад, вимагати, щоб принаймні одна емоція була

передбачена для кожного випадку, щоб забезпечити узгодженість з обмеженнями предметної області.

З практичної точки зору:

- Дуже тонка настройка порогових значень може дати незначний приріст метрики за рахунок зниження надійності. Групування міток за поширеністю або профілем продуктивності може зменшити складність.
- Порогові значення, вибрані в автономному режимі, слід відстежувати на реальних даних і періодично переналаштовувати, якщо розподіл даних зміщується.
- Криві Precision-Recall і ROC для окремих міток допомагають зрозуміти компроміси при різних порогових значеннях.

Визначаючи чіткі цілі для метрик, використовуючи налаштування порогових значень для кожної мітки на основі даних валідації та застосовуючи методи калібрування, фахівці можуть узгодити вихідні дані моделі з реальними вимогами і гарантувати, що порогові значення відображають як статистичні показники, так і пріоритети предметної області.

Висновки до розділу 1

У першому розділі було пояснено еволюцію підходів до розпізнавання емоцій у тексті: від систем на основі словників і правил та класичних методів машинного навчання до сучасних архітектур глибокого навчання на основі трансформерів. Пояснено основні принципи мульти-міткової класифікації, як вона відрізняється від одно-міткової, та які труднощі можуть виникати при її реалізації. Детально описано набір метрик, які використовуються для комплексного оцінювання моделей у мульти-мітковому контекст. Цей теоретичний фундамент обґрунтовує вибір тонкого налаштування трансформерних архітектур як методу навчання моделі та визначені критерії оцінки успіху прогнозування.

2 ЗАСТОСОВАНІ МЕТОДИ

2.1 Вибір архітектур

Один із найперших і найвпливовіших попередньо навчених трансформерів, BERT, запровадив концепцію глибоких двонапрямних контекстних ембедингів. Його навчальні завдання, масковане мовне моделювання (MLM) і передбачення наступного речення (NSP), дозволяють моделі дізнатися, як слова зустрічаються в контексті і як речення пов'язані одне з одним. Зокрема, під час MLM 15% вхідних лексем випадковим чином маскуються, і модель повинна передбачити вихідну лексему, спираючись лише на її лівий та правий контекст. Під час NSP BERT отримує пари речень і вчиться передбачати, чи є друге речення справжнім продовженням першого. Після попереднього навчання на великих текстових масивах (наприклад, Вікіпедія, BookCorpus) BERT можна налаштувати, додавши простий вихідний шар для завдання класифікації. Для класифікації емоцій за кількома мітками остаточний об'єднаний вихідний вектор (що відповідає токену «[CLS]») зазвичай подається на повністю зв'язний шар із сигмоїдними активаціями, по одній для кожної мітки емоції, і навчається з бінарною втратою перехресної ентропії.

Переваги:

- Двонапрямне навчання BERT призводить до потужного процесу токенізації, що фіксує нюанси семантичної та синтаксичної інформації.
- Завдяки своїй популярності, існує багато документацій, навчальних посібників та широкий спектр попередньо навчених контрольних точок (наприклад, bert-base-uncased, bert-large-cased).
- BERT досягла найсучасніших або майже найсучасніших результатів у багатьох тестах NLP, включаючи аналіз настроїв і відповіді на запитання.

Обмеження:

- Навіть базова модель (110 мільйонів параметрів) зазвичай вимагає щонайменше 12-16 ГБ графічної пам'яті для точного налаштування при стандартних розмірах партій, що може бути серйозною вимогою для графічних процесорів споживчого класу з омеженим обсягом VRAM.
- Доведено, що NSP дає обмежену користь для багатьох завдань, і його видалення іноді може покращити результат.

RoBERTa базується на архітектурі BERT, але модифікує процес попереднього навчання для покращення вихідного результату. Основні відмінності полягають у наступному: видалення завдання передбачення наступного речення, навчання на більших і різноманітніших корпусах, використання динамічного маскування (тобто маскування відбувається по-різному в кожну епоху), і більші розміри партій і швидкість навчання під час попереднього навчання. Отримана модель roberta-base містить приблизно 125 мільйонів параметрів, що відповідає BERT-base за розміром, але часто перевершує її за багатьма показниками.

Переваги:

- Завдяки усуненню NSP і більш тривалому навчанню з динамічним маскуванням, RoBERTa часто дає вищі оцінки макро-F1 на завданнях класифікації, включаючи виявлення емоцій.
- Які і моделі BERT, моделі, як roberta-base і roberta-large, легко доступні через Hugging Face, що полегшує відтворюване тонке налаштування.
- Оскільки токени маскуються по-різному в кожну епоху, модель бачить більше різноманітних контекстів і уникає надмірного пристосування до одного шаблону маски.

Обмеження:

- Хоча архітектура RoBERTa ідентична архітектурі BERT, довший графік попереднього навчання та більші обсяги даних роблять модель важчою, що вимагає ретельного керування пам'яттю графічного процесора.

- На практиці, для досягнення максимальної продуктивності в задачах тренування може знадобитися додаткове налаштування гіперпараметрів і більша кількість епох.

DeBERTa представляє два основні архітектурні покращення порівняно з BERT та RoBERTa: роз'єднана увага, яка поділяє ембединги позицій та змісту та покращена структура декодування, яка використовує відносні позиційні упередження. У роз'єднаній увазі представлення кожної лексеми розкладається на «змістові» та «позиційні» компоненти, що дозволяє механізму самоуваги зосереджуватися на змісті, незалежно від позиції. Це призводить до більш тонкого моделювання порядку слів. DeBERTa-v3, зокрема, інтегрує покращену параметризацію, яка забезпечує чудову продуктивність на різноманітних тестах NLP.

Переваги:

- Емпіричні результати показують, що DeBERTa перевершує BERT і RoBERTa в таких завданнях, як GLUE, SQuAD і класифікація настроїв.
- Відокремлюючи позицію від змісту, DeBERTa може краще вловлювати відносний порядок слів, що має вирішальне значення для тонкого розрізнення емоцій.

Обмеження:

- Механізм розосередження уваги додає обчислювальних витрат, що може сповільнити етапи попереднього навчання та точного налаштування.
- Додаткові параметри та більш складні обчислення уваги зазвичай вимагають більший обсяг пам’яті для стандартних кроків тонкого налаштування, що часто робить DeBERTa-v3-base (приблизно 140 мільйонів параметрів) складною для використання на споживчих графічних процесорах без агресивного накопичення градієнта або методів змішаної точності.

- Хоча Hugging Face Transformers підтримує DeBERTa, деякі допоміжні бібліотеки та інструменти можуть відставати від підтримки BERT та RoBERTa.

Загалом був зроблений вибір протестувати кожну з цих трьох архітектур та обрати за результатом найкращу.

2.2 Інструменти розробки

Дослідження було проведено на операційній системі Windows 10, з основною мовою програмування Python 3.11.

Ключові використані бібліотеки:

- PyTorch: 2.6.0

  PyTorch була обрана через її нативну підтримку CUDA 12.x і покращену оптимізацію компілятора для трансформерних моделей.

- Transformers: 4.50.3

  Бібліотека Tansformers від Hugging Face запровадила доступ до готових трансформерних архітектур (BERT, RoBERTa, DeBERTa) та зручні класи для токенізації, конфігурації моделі та тренування.

- Datasets: 2.19.1

  Ще одна бібліотека від Hugging Face, Datasets, була використана для завантаження, використання та аугментації датасету GoEmotions, оптимізована для кешування та току даних.

- scikit-learn: 1.6.1

  Була використана для обчислення метрик оцінювання, таких як F1, precision та recall та утиліти для бінаризації міток.

- NumPy: 2.2.4

  Стандартна бібліотека Python, що використовується для фундаментальних обчислень, необхідних для прогнозування ймовірностей, визначення порогів та функцій втрат.

- tqdm: 4.67.1

Використовувалася для спостерігання прогресу при тренуванні моделі при проходженні через датасет.

Середовище Python керувалося через віртуальне середовище, створене за допомогою venv, що забезпечило ізоляцію залежностей бібліотек.

Через ефективне використання графічного процесора, центральний процесор фактично залишався пасивним для задачі тренування. Оскільки графічний процесор склав основне джерело обчислювальних можливостей, всі бібліотеки глибокого навчання були зкомпільовані для CUDA 12.x для максимальної результативності.

Під час проведення дослідження було використано дві основні зовнішні платформи для полегшення зберігання даних та тренування моделі: Dropbox та Hugging Face. Ці сервіси забезпечили надійний хостинг та можливість версіювання, що спростило робочий процес

Dropbox використовувався як централізоване рішення для зберігання наборів даних, проміжних артефактів попередньої обробки та контрольних точок навченої моделі. З огляду на обмеження розміру набору даних GoEmotions, особливо після доповнення тексту шляхом перефразування, зберігання декількох версій токенізованих наборів даних (у форматах NumPy і PyTorch) вимагало безпечної, доступної файлової системи за межами локального комп'ютера. Використання папок Dropbox дозволило забезпечити автоматичну синхронізацію між локальним навчальним середовищем і віддаленим репозиторієм:

- Попередньо оброблені дані (наприклад, токенізовані файли JSON) зберігалися як на локальних машинах, так і на хмарі.
- Доповнені набори даних архівувалися, що полегшувало використання різних версій у разі потреби.
- Навчені ваги моделі (збережені за допомогою *trainer.save_model()*) завантажувалися відразу після завершення, що переглядати результати без прямої передачі файлів між машинами.

Платформа Hugging Face була ресурсом для пошуку моделей та токенізаторів. Попередньо підготовлені архітектури трансформаторів (наприклад, roberta-base, ramsrigouthamg/t5_paraphraser) були отримані безпосередньо через API бібліотеки Transformers, яка взаємодіє з Hugging Face Hub. Хаб також надає версії кожної моделі, забезпечуючи відтворюваність, кожен експеримент посилається на точний ідентифікатор моделі (наприклад, roberta-base@sha256:...), щоб зафіксувати однакові ваги та конфігураційні файли.

Використовуючи Dropbox для стабільного резервного копіювання файлів і Hugging Face Hub для управління моделями/версіями, процес розробки підтримував відтворюваність.

2.3 Функції втрат

Як базову функцію було обрано бінарну крос-ентропію, але через те, що вона не розв'язує проблему дисбалансу класів, було обрано фокальну функцію втрати як наступну альтернативу.

Спочатку запропонована для виявлення щільності об’єктів, фокальна функція втрат модифікує бінарну крос-ентропію для зменшення ваги категорій, що добре розпізнаються моделлю.

$$L_{focal} = -\alpha_i(1 - \hat{y}_i)^{\gamma} \log(\hat{y}_i), \tag{2.1}$$

де $\hat{y}_i$ – прогнозована ймовірність істинності класу i;

γ – фокусувальний параметр, зазвичай в межах від 1 до 2;

α_i – значення ваги.

На початку фокусувальний параметр було визначено значенням 2 для того, щоб зменшити вплив репрезентованих категорій, та відповідно збільшити фокус на більш рідкісні категорії.

За необхідності, параметр альфа можна прибрати з рівняння, якщо зважження класів не призведе до покращення результатів тестування.

Тобто, базуючись на емпіричних результатах надалі параметри функції будуть налаштовуватись.

2.4 Стратегії боротьби з дисбалансом

Застосованою технікою семплінгу було виважене випадкове вибіркове виключення (Weighted Random Sampling). У PyTorch використовувався WeightedRandomSampler, що надавав категоріям ваги обернено пропорційні до кількості їх істинних міток. Це забезпечувало більш часту появу недорепрезентованих категорій в кожну епоху. На практиці чисті значення ваг було обмежено максимумом у три рази більшим, ніж середня вага, щоб уникнути перенавчання на одиничних прикладах.

Застосована фокальна функція втрат також вплинула на проблему дисбалансу класів.

Була спроба використання зважування за допомогою інверсної частоти позитивного класу:

$$w_i = \frac{n - n_i}{n_i + \varepsilon}, \tag{2.2}$$

де n_i – кількість позитивних прикладів класу i;

n – загальна кількість прикладів даних. $\varepsilon = 10^{-6}$

Спочатку таке значення ваг використовувалося в стандартній бінарній крос-ентропії, потім у фокальній функції втрат.

Первинною стратегією аугментації даних було перефразування за допомогою використання двох моделей:

- ramsrigouthamg/t5_paraphraser

  Ця модель базується на T5 (Text-to-Text Transfer Transformer) та була донавчена спеціально для перефразування англійських речень. У

ході спроб покращити генерацію аугментованих прикладів ця модель була першою використана для перефразування.

- PEGASUS

  Після невдалих спроб застосування ramsrigouthamg/t5_paraphraser, модель PEGASUS показала значно кращий результат, балансуючи перефразовані варіанти між лексичним різноманіттям та збереженням семантичного контексту.

Основною метою було обрати кращий метод, або комбінацію кращих методів, що призвели до найбільших показників метрик.

2.5 Визначення цільових метрик

Згідно з протоколами оцінювання, встановлених в оригінальному бенчмарку GoEmotions та подальшому дослідженні EmoBERTa-X, було обрано набір метрик для охоплення головних аспектів ефективності мульти-міткової класифікації емоцій. Основними цілями були вимірювання як дискримінації за окремими мітками, так і загальної узгодженості набору міток. Були обрані такі цільові метрики:

- Макро-F1

  Основна метрика за якою визначається успішність моделі. Бля базової моделі було обране значення 0.42, продемонстроване в дослідженні GoEmtions, та 0.6, як цільове, що було досягнуто моделлю EmoBERTa-X при базовій імплементації.

- Мікро-F1

  Для оцінки впливу недорепрезентованих метрик на результат (при порівнянні з Макро-F1)

- Точність (precision)

  Для визначення істинно позитивних результатів прогнозування моделі.

- Відгук (recall)

Для оцінки того, як часто модель успішно згадує відповідну мітку незалежно від точності прогнозу.

Як і в EmoBERTa-X, макро-F1 розглядається як вирішальна метрика для налаштування гіперпараметрів і порівняння моделей, тоді як мікро-F1 і інші слугують вторинними перевірками загальної і суворої відповідності.

Висновки до розділу 2

У розділі 2 було розглянуто ключові інструменти та методи застосовані при навчанні моделі. Обрано три основних архітектури для порівняння, RoBERT, RoBERTa та DeBERTa. Описано застосовані інструменти та бібліотеки Python, та зовнішні ресурси.

Було розглянуто застосовані методи боротьби з дисбалансом класів, використані моделі перефразування та застосовані функції втрат.

Нарешті, визначено основні метрики оцінки моделі, зокрема прийняте рішення розглядати макро-F1 як вирішальну при виборі найуспішнішого варіанту моделі.

3 ВИБУДОВУВАННЯ ПРОЦЕСУ НАВЧАННЯ ТА НАЛАШТУВАННЯ МОДЕЛІ

3.1 Середовище розробки

Графічні процесори (GPU) пропонують набагато швидший процес тренування глибоких нейронних мереж завдяки кращій спроможності паралельного обчислення на відміну від центральних процесорів. Для тренування було застосовано NVIDIA GeForce RTX 3060 GPU із 6 гігабайтами GDDR6 відеопам’яті. Хоча ця карта споживчого класу здатна прискорити тензорні операції та пакетну обробку, сучасні трансформерні архітектури часто вимагають більшого обсягу пам'яті. Тому 6 ГБ виявилися обмежувальним фактором під час навчання, що вимагало менших розмірів пакетів, щоб вписати застосовані моделі в доступну пам'ять.

Основний цикл навчання та завдання попередньої обробки даних було майже повністю вивантажено на графічний процесор, що дозволило максимізувати пропускну здатність і скоротити загальний час виконання.

Було розглянуто альтернативний підхід – використання безкоштовних графічних процесорів Google Colab. Однак часті тайм-аути сеансів і суворі квоти на використання зробили Colab ненадійним для тривалих навчальних запусків. Тому вся розробка та налаштування моделі проводилася локально на RTX 3060, що забезпечило постійний доступ до апаратних ресурсів протягом усього проєкту.

3.2 Підготовка даних

Набір даних GoEmotions - це масштабний мульти-мітковий набір, розроблений Google Research для тонкої класифікації емоцій. Він містить 58000 англомовних коментарів з соцмережі Reddit, кожен з яких анотований однією або кількома з 27 емоційних міток, що робить набір даних цінним

ресурсом для задач розпізнавання емоцій. Окрім міток емоцій, кожен приклад містить унікальне текстове поле та відповідний список цілочисельних індексів, які представляють емоцію (емоції), приписані до цього коментаря.

Структура набору даних наступна:

- «text»: необроблений коментар на Reddit (рядок)
- «labels»: список індексів, що посилаються на мітки емоцій (цілі числа)
- «id»: ідентифікатор коментаря

Перший етап розробки охоплював обробку та аналіз набору даних.

При завантаженні датасету через бібліотеку datasets, повертається словник, що містить три частини: «train», «validation» і «test». Кожна з них є структурованим об'єктом (класу Dataset), який можна ітераційно переглядати або конвертувати в інші формати, такі як Pandas DataFrames, для полегшення аналізу та візуалізації.

GoEmotions з самого початку розроблений з метою мульти-міткової класифікації. Наприклад, коментар може бути позначений як захопленням і вдячністю, або і роздратуванням і розчаруванням. Тому він є ідеальним стартовим варіантом для початку роботи з моделлю. Для початку датасет проаналізовано на предмет того, як саме мітки появляються разом в екземплярах даних датасету (рис 3.1), де діагональ має стандартне значення 1.0, а інші області даних відповідний коефіцієнт зустрічності міток.

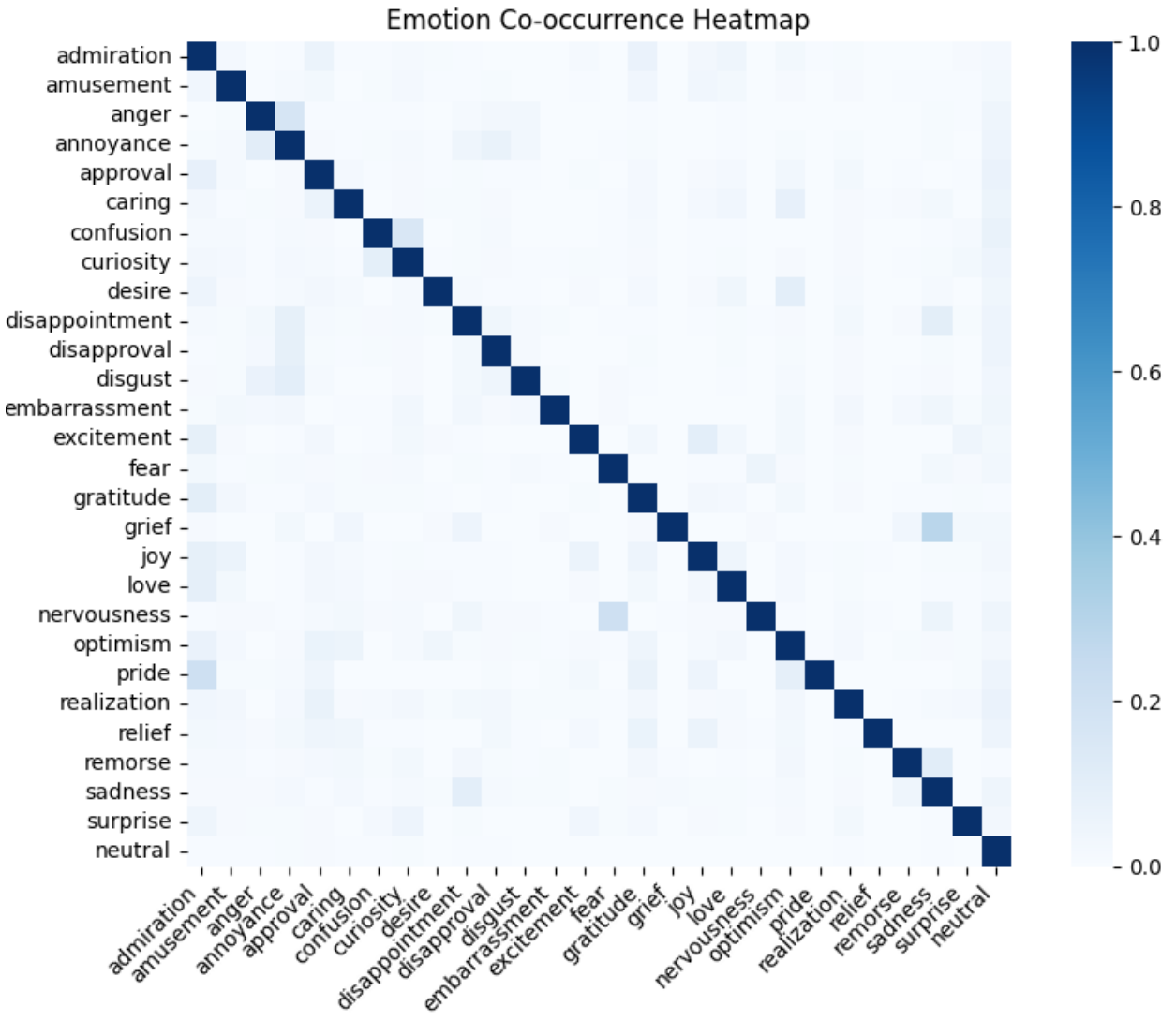

Рисунок 3.1 – «Теплова карта» зустрічності міток

Через велику кількість міток, вони зустрічаються рідко відносно стандартного значення, але все одно видно, що, наприклад, «сум» (sadness) та «скорбота» (grief) мають більший коефіцієнт зустрічності.

GoEmotions - це масштабний, багатомісний набір даних, який охоплює широкий спектр людських емоцій в онлайн-дискурсі. Розуміння його структури та розподілу міток є необхідним кроком перед тим, як переходити до препроцесингу та розробки моделі.

Аналіз дисбалансу класів

Після завантаження та первинного огляду набору даних, наступним кроком стало обчислення загального розподілу міток та визначення

дисбалансу класів, що міг негативно вплинути на тренування моделі. Спершу було обчислено кількість появи міток відповідних класів в екземплярах датасету. Отримано одномірний масив розміру 27, значення якого – кількість міток відповідного класу. Він показав, які мітки є недорезпрезентованими, і, можливо, потребують аугментації, а яких в наборі даних достатньо кількість. Природно, деякі емоції, як, наприклад, «радість» чи «розвага» з'являються набагато частіше ніж «скорбота» чи «нервозність» в контексті соцмережі Reddit. Для формалізації міток, що вважаються недорепрезентованими в датасеті, встановлено поріг в 1000 появ, нижче якого мітки вважатимуться «рідкими». Знання про репрезентованість міток можуть допомогти скерувати послідуючий шлях роботи з набором даних. Результат для базового датасету продемонстровано на рисунку 3.2.

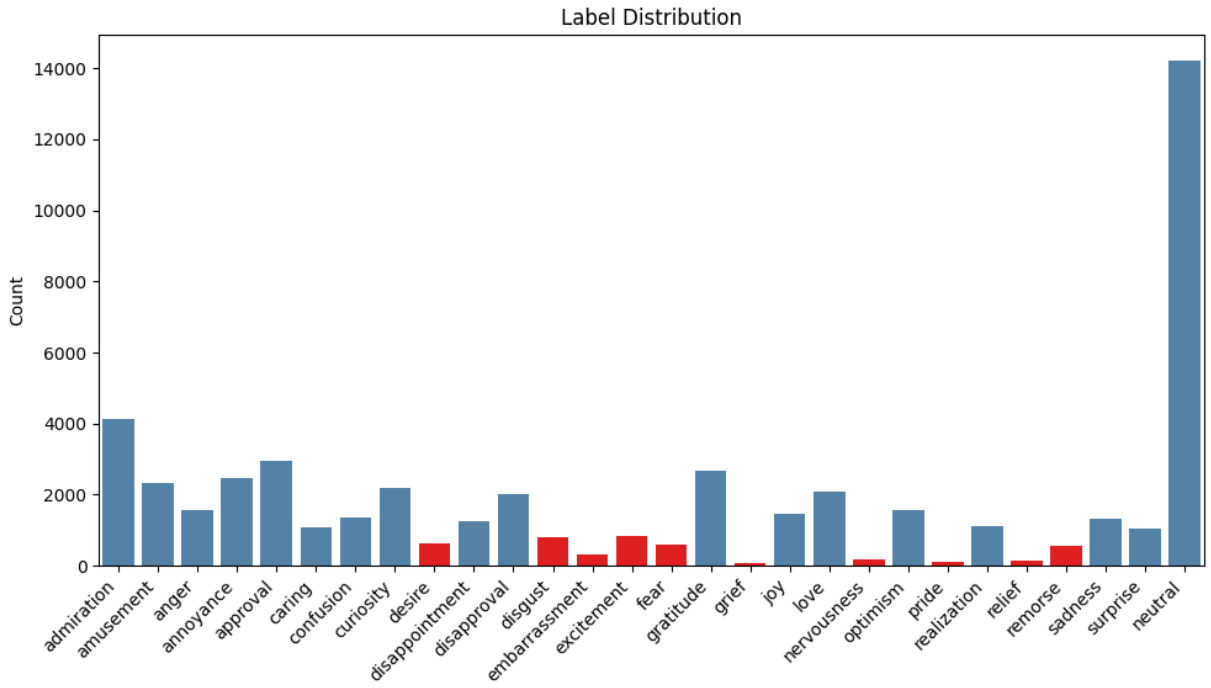

Рисунок 3.2 – Кількість міток відповідного класу в наборі даних.

Як видно, всього 10 міток попадають під визначення рідких з порогом 1000, коли деякі, як «скорбота», «нервовість», «гордість» та «полегшення», мають зовсім мало екземплярів.

Після ідентифікації рідких міток було обчислено пропорції частини екземплярів даних, які мають хоча б одну рідку мітку, проти тих, що не мають жодної. Розуміння даних та відношення на графіку (рис. 3.3) допомогло зрозуміти як часто та в якому об’ємі доведеться застосовувати інструменти зміни датасету.

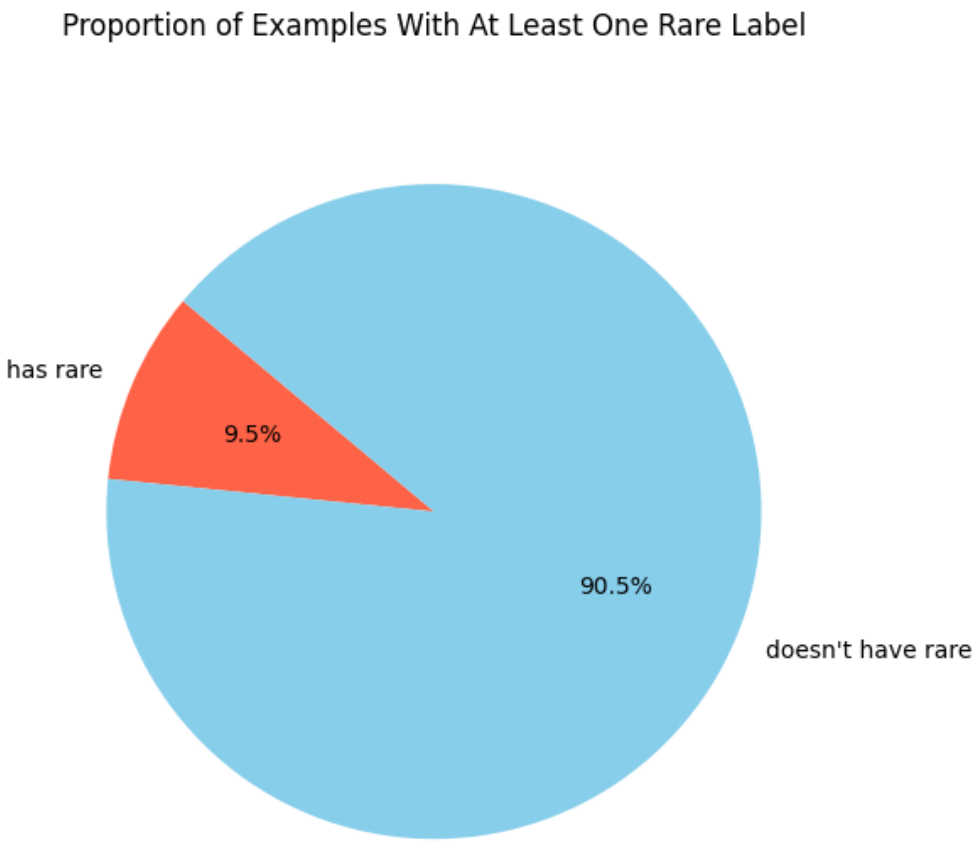

Рисунок 3.3 – Пропорційна кількість екземплярів з рідкими мітками

Лише 9.5% відсотків екземплярів повертають у відповідь рідкі мітки. Це знов спонукає до використання аугментації та ваг класів.

Нарешті, оскільки GoEmotions є мульт-мітковим, дисбаланс полягає не лише в кількості прикладів для кожної емоції, а й у тому, як рідкісні мітки поєднуються з більш поширеними. Обчислення та візуалізація частоти зустрічальності рідкісних та поширених міток (рис. 3.4) допомогло виявити такі закономірності, як «скорбота» часто з'являється поряд із «сумом», що впливає на вибір способу аугментації або архітектури моделі.

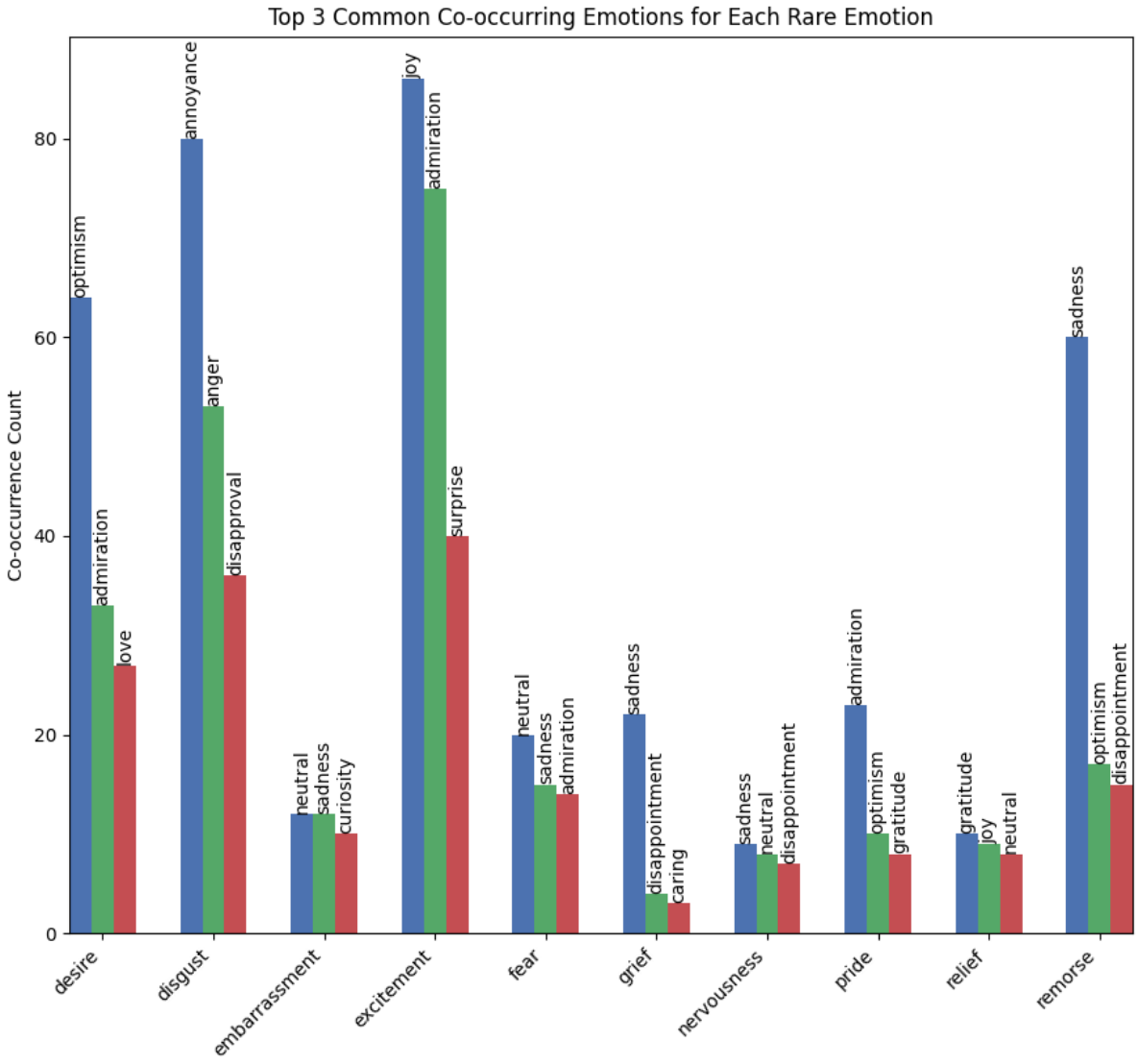

Рисунок 3.4 – Зустрічність рідкісних міток з частими

Також можна зробити попередні висновки щодо більш поширених міток, з якими модель надалі буде плутати рідкі.

Систематично вимірюючи кількість міток, виявляючи рідкісні мітки та візуалізуючи ці розподіли, вийшло створити чітку основу для подальших рішень щодо препроцесингу (наприклад, цілеспрямованої аугментації, зменшення ваги або стратегій вибірки), які спрямовані на усунення дисбалансу класів.

Аугментація даних

Щоб зменшити недорепрезентованість рідкісних емоційних міток, визначених у минулому пункті, було використано стратегію перефразування, яка генерує нові екземпляри від наявних, що містять рідкісні емоції. Цей підхід використовує попередньо навчену модель перефразування для створення альтернативних текстових варіантів зі збереженням оригінальних емоційних анотацій. Доповнюючи лише ті приклади, які містять принаймні одну рідкісну мітку, обчислювальні ресурси зосереджуються на збалансуванні набору даних, не вносячи шуму для емоцій, що часто зустрічаються.

Визначивши масив рідкісних міток, програма ітерує по кожному екземпляру тренувальних даних, генеруючи 1 фразу для кожного знайденого екземпляра рідкісних міток.

Для перефразування використано модель на основі T5 ramsrigouthamg/t5_paraphraser, яку налаштовано для генерації декількох варіантів перекладу англійською мовою. Ініціалізація відбулась на доступному пристрої GPU.

Для кожного вибраного прикладу було збудовано запит на введення виду "перефразувати: <оригінальний_текст>". Модель генерує один кандидат, як зазначено в гіперпараметрах:

- num_beams=5, щоб збалансувати якість і різноманітність
- num_return_sequences=1 для отримання одного варіанту
- max_length=128 токенів для перефразування на рівні речень

У самій функції перефразування циклічно переглядається кожен рідкісний приклад, відловлюються потенційні помилки генерації та накопичуються перефразовані тексти(поки що, згідно гіперпараметрів), поєднані з тими самими векторами міток.

Після того, як всі перефразовані варіанти зібрані в список, його перетворено у набір даних (Dataset «Hugging Face») і об'єднано з оригінальною навчальною вибіркою. Це призвело до нової навчальної

вибірки з підвищеною репрезентативністю раніше рідкісних емоцій. Об'єднаний набір даних збережено на диску для подальшого відтворення.

Далі дані знову виведено у вигляді гістограми для спостерігання змін в датасеті (рис. 3.5).

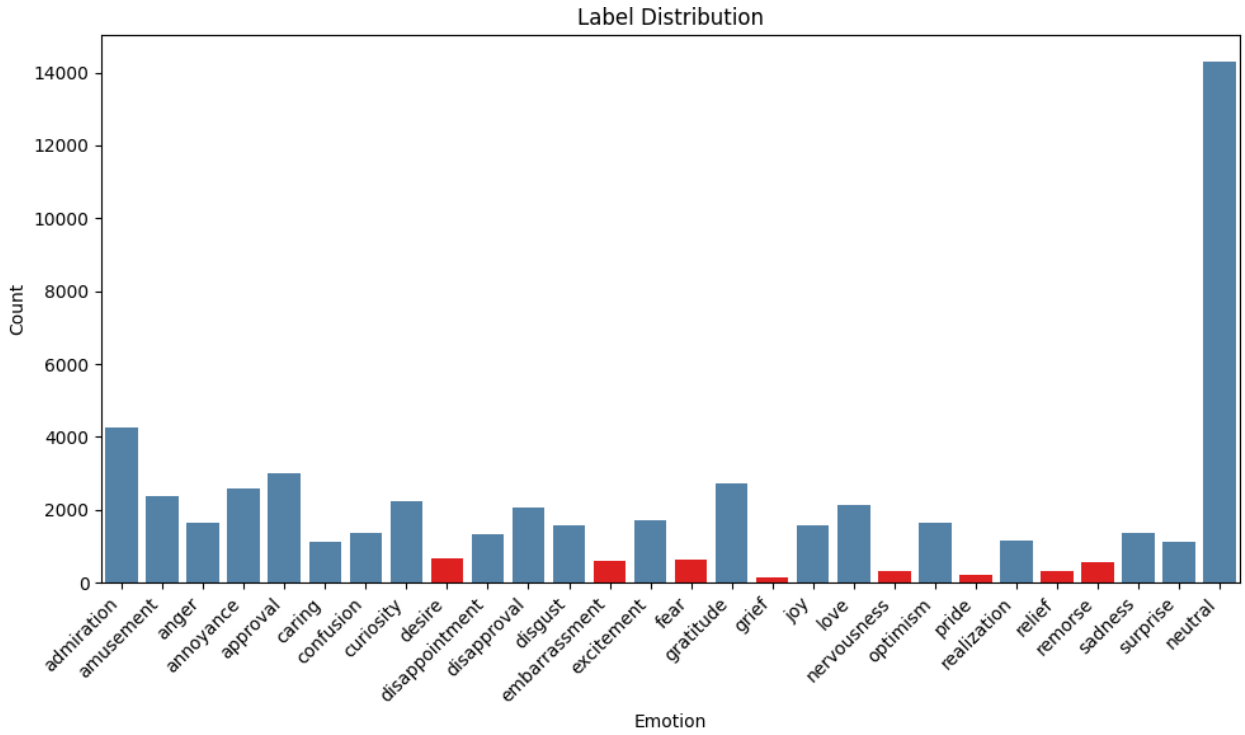

Рисунок 3.5 – Новий розподіл класів

Мітки все ще підпадають під визначення рідкісних, навіть після аугментації, але на гістограмі все одно видно збільшення деяких з них. Оскільки екземплярів з самого початку для роботи було небагато, аугментація в кращому випадку подвоїла їх кількість, аби не перенавантажувати набір даних згенерованими варіантами.

Далі переглянуто деякі зі згенерованих варіантів аби переконатися в їх якості (табл. 3.1).

Таблиця 3.1 – перефразовані екземпляри даних

Оригінальний	Перефразований
But really Vanilla Ice is scum RIP [NAME] and [NAME]	Is Vanilla Ice really scum?
Cheers. Never heard of any of them. Kids will do that to you...	Never heard of any of them. Kids will do that to you. Cheers. Never heard of any of them.
Agreed! I’m so excited for her story this season!	I'm so excited for her story this season!
Whew. I'm safe then.	I'm safe now.
This very thing got me diagnosed as ADD. I'm of the drugs now and feeling way much better.	I've been diagnosed with ADD. I'm on the drugs now and feeling way much better.

Але, більшість результатів не сильно відрізняються від оригіналу, або просто перероблені під питання. Це ставить під сумнів доцільність використання такої аугментації для продовження роботи, тому далі необхідно буде порівняти результат тренування моделі з натренованою на неаугментованому датасеті. Після цього можна буде вирішити, чи залишити такий варіант аугментації, налаштувати гіперпараметри, чи змінити модель перефразування.

Проте, використовуючи перефразування лише на прикладах, що містять рідкісні емоції, було отримано синтетичні дані, що відрізняються від їх попередників (в кращому випадку), які зберігають оригінальний розподіл міток, водночас посилюючи вплив моделі на недостатньо представлені категорії. Таке цілеспрямоване доповнення створило основу для більш збалансованого навчання на наступних етапах.

Препроцесинг тексту

Перед тим як передати текстові дані моделі на навчання, над ними провелося декілька кроків обробки, для того щоб забезпечити їх відносну

одноманітність, правильну довжину та меншу кількість шуму. Ці кроки включили очищення тексту, токенізацію з підставленням та обрізанням, а також підготовку масок уваги.

Хоча трансформерні токенізатори стійкі до пунктуації та регістру, стандартизація тексту може зменшити шум і покращити продуктивність моделі. Далі наведено конкретні кроки, що проводилися над текстом до його застосування:

- Перетворення всіх символів у малі літери для зменшення обсягу словника (наприклад, «Щасливий» у «щасливий»).
- Замінено серію пробілів, табуляцій або символів нового рядка одним пробілом, щоб уникнути неправильної токенізації.
- Видалено недруковані контрольні символи, які можуть з'являтися у коментарях Reddit. Цей крок зроблено дослідникам, що збирали датасет.
- Хоча надмірне видалення розділових знаків може зашкодити змісту, з тексту було вилучено лише певні послідовності розділових знаків (наприклад, кілька знаків оклику «!!!» у «!») або зайві символи, якщо вони присутні.
- Замінено URL-адреси або згадки про користувачів Reddit токенами-заповнювачами (<URL>, <USER>), щоб зберегти семантичну структуру, не роздуваючи словниковий запас. Цей крок також зроблено дослідникам, що збирали датасет.
- Кожний масив міток відповідний текстовому екземпляру перетворено у масив розмірності бінарних значень, де 1 означає наявність мітки, а 0 – її відсутність.

Далі очищені дані токенізовано за допомогою RobertaTokenizer від Hugging Face, налаштованого на вхідні дані фіксованої довжини відповідно до вимог моделі. Створена функція токенізації застосувалася для всього датасету.

Такий процес препроцесингу дозволив перетворити текстові дані на послідовності токенів фіксованого розміру, зменшити шум завдяки очищенню та зберегти позиції міток.

3.3 Навчання моделі

Через декілька тестів стало очевидним, що для тренування моделі класифікації емоцій найкращий результат показує архітектура RoBERTa (roberta-base), завдяки її високим показникам у оцінках NLP, доведеним можливостям тонкого налаштування для класифікації тексту, та її результатам після зрівнювання з BERT (bert-base-uncased) та DeBERTa (deberta-v3-base). RoBERTa розширює BERT, видаляючи завдання передбачення наступного речення і навчаючись на більших масивах даних, що часто призводить до покращення розуміння контексту даних.

Також, після проміжних валідаційних тестів RoBERTa стабільно досягала вищих показників Macro-F1 (рис. 3.6), особливо для менш частих емоційних міток. Тому зрештою було прийнято roberta-base як основну архітектуру.

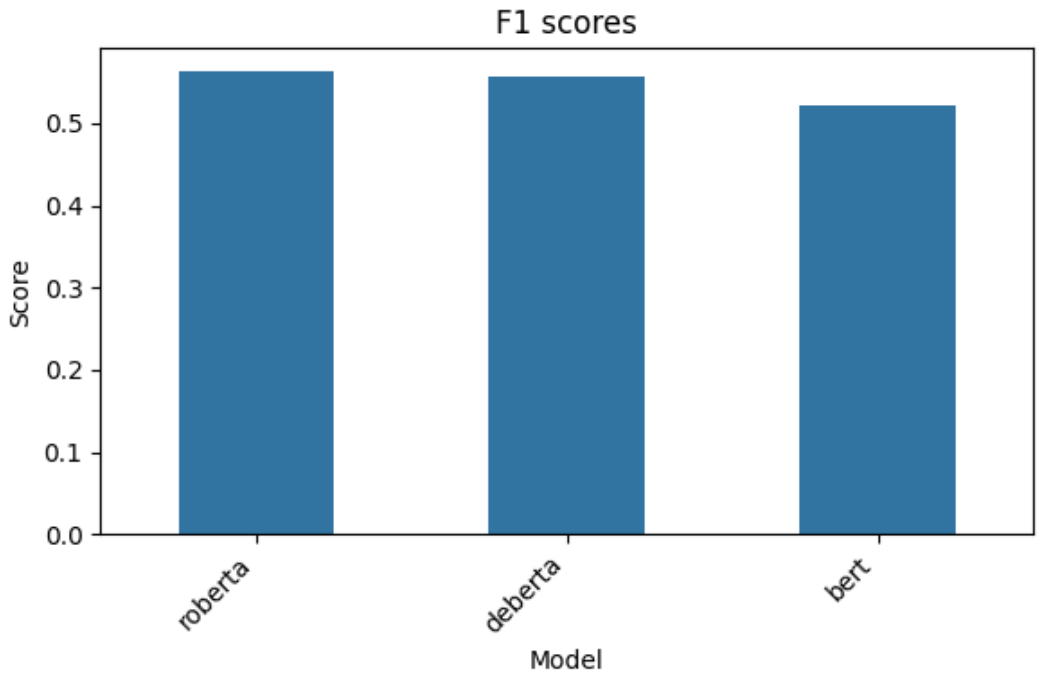

Рисунок 3.6 – Гістограма значень макро-F1 для трьох архітектур (для проміжного результату, з використанням постобробки порогів окремо для кожної мітки та застосуванням визначеної пізніше в процесі тренування функції втрат)

Конфігурація для мульти-міткової класифікації

Перші кроки другого етапу розробки, налаштування для адаптування RoBERTa під задачу мульти-міткової класифікації, включили виставлення правильного типу для подальшого використання класом Trainer Hugging Face та визначення механізму активації для прогнозування окремих оцінок ймовірності для кожної з 27 міток.

Виставлення правильного типу класифікації забезпечило, що Trainer ставиться до кожного вихідного логіту, як до окремого бінарного рішення, замість примусового виконання softmax по всім міткам. Внутрішньо це інструктує модель застосовувати сигмоїдну активацію по базовим логітам для кожної мітки. Сам класифікатор складається з повністю з'єднаного шару, який проєктує об'єднаний вихід RoBERTa (прихований розмір 768) на 27-мірний вектор логітів.

Після того, як енкодер RoBERTa обробив вхідні дані, фінальний результат проходить крізь класифікатор, який повертає 27 логітів $z = [z_1, z_2, \dots, z_{27}]$.

Ці логіти далі беруть участь в обчисленні поелементного сигмоїда:

$$\hat{p}_j = \sigma(z_j) = \frac{1}{1+e^{-z_j}}, j = 1,2, \dots ,27. \quad (3.1)$$

Результівні значення $\hat{p}_j$ репрезентують прогнозовані моделлю ймовірності того, що наданий текст має емоцію *j*.

Але, перед власне тренуванням моделі, необхідно виставити гіперпараметри навчання. Для початку, я обрав досить стандартні параметри, що є ефективною стартовою точкою для подальшої їх зміни:

- warmup_steps: 500

  Кількість тренувальних кроків, протягом яких значення learning rate (LR) лінійно підвищується від 0 до початкового рівня. Такий "прогрів" допомагає стабілізувати навчання на початкових етапах і уникнути занадто великих градієнтів одразу.

- learning_rate: 3e-5

  Початкова швидкість навчання оптимізатора. Значення $3*10^{-5}$ є типовим для тонкого налаштування трансформерних моделей, оскільки поєднує достатню швидкість збіжності з відносною стабільністю.

- per_device_train_batch_size: 16

  Розмір пакета (batch) на кожному пристрої (GPU) під час тренування. Пакет із 16 зразків обрано так, щоб вміститися у 6 ГБ VRAM, водночас забезпечити стабільну оцінку градієнтів.

- per_device_eval_batch_size: 16

  Розмір пакета на кожному пристрої під час оцінювання (валідації). Використання того ж розміру, що і для тренування, забезпечує послідовність дозволяє досягти об'єктивної точності моделі.

- num_train_epochs: 3

  Загальна кількість повних проходів (епох) по всьому тренувальному набору даних. Три епохи зазвичай достатні для досягнення прийнятної якості модельних прогнозів без надмірного перенавчання на середньому обсязі даних.

- load_best_model_at_end: True

  Прапорець, що вказує тренеру (Trainer) автоматично завантажити наприкінці тренування ту версію моделі, яка показала найкращі результати за обраною метрикою на валідаційному наборі. Забезпечує, що фінальна модель відповідає найвищій зафіксованій якості.

- metric_for_best_model: “f1”

  Визначає, що метрика F1 (макро-F1 по всіх мітках) використовується для вибору “найкращої” версії моделі. Під час кожної валідації тренер порівнює F1-значення та зберігає чекпоінт із максимальним показником.

Після найпершого раунду тренування моделі, було отримано стартовий результат моделі з оцінкою макро-F1 0.42 (таблиця 3.2). Не було застосовано ніяких методів постобробки для вибору порогів, тому стандартне значення 0.5 застосовувалося для всіх міток.

У таблиці 3.2 можна побачити успішність моделі та відповідні оцінки для кожної мітки окремо та загальний результат мікро, макро та зважуваного варіантів обчислень.

Таблиця 3.2 – значення оцінок для міток окремо та разом

	precision(точність)	**recall(відгук)**	**f1**
admiration	0.69	0.72	0.71
amusement	0.77	0.82	0.80
anger	0.61	0.41	0.49
annoyance	0.49	0.15	0.23
approval	0.54	0.32	0.40

Продовження таблиці 3.2

	precision(точність)	**recall(відгук)**	**f1**
caring	0.53	0.36	0.43
confusion	0.62	0.29	0.40
curiosity	0.55	0.54	0.54
desire	0.71	0.33	0.45
disappointment	0.65	0.07	0.13
disapproval	0.49	0.31	0.38
disgust	0.67	0.32	0.43
embarassment	0	0	0
excitement	0.64	0.26	0.37
fear	0.67	0.60	0.64
gratitude	0.94	0.89	0.92
grief	0	0	0
joy	0.66	0.52	0.58
love	0.80	0.81	0.80
nervousness	0	0	0
optimism	0.65	0.45	0.53
pride	0	0	0
realization	0.74	0.10	0.17
relief	0	0	0
remorse	0.63	0.70	0.66
sadness	0.71	0.42	0.53
surprise	0.61	0.50	0.55
neutral	0.72	0.56	0.63
macro	0.54	0.37	0.42
weighted	0.67	0.50	0.55

Можна побачити, що деякі категорії жодного разу не були визначені моделлю, та загалом велика кількість мають низькі оцінки метрик. Однак, з огляду на те, що це лише перший варіант моделі, без застосування додаткових інструментів зважування класів, обробки текстових даних, чи постобробки результатів, модель є гарною стартовою точкою для продовження роботи.

Висновки до розділу 3

В третьому розділі було детально описано процес побудови пайплайну тренування моделей та послідуючий отриманий первинний результат. Було розглянуто умови середовища в якому відбувалося тренування, інструменти, за допомогою яких проводилося тонке налаштування, початкові методи застосовані для покращення даних, та базові гіперпараметри застосовані для тренування моделі.

Було продемонстровано основні проблеми набору даних та обраного підходу з використанням усіх 27 міток, що в майбутньому могли б призвести до погіршення результатів прогнозування моделі.

Проаналізовано базовий натренований варіант моделі за допомогою обчислення встановлених метрик, що показало найпроблемніші категорії.

4 ПРОВЕДЕНІ ЕКСПЕРИМЕНТИ

4.1 Стандартна бінарна крос-ентропія як функція втрат

Визначення ваг відбулось за формулою інверсної частоти позитивного класу. Результати для метрик продемонстровано в таблиці 4.1.

Таблиця 4.1 – значення загальних оцінок

	precision(точність)	**recall(відгук)**	**f1**
micro	0.31	0.82	0.45
macro	0.28	0.82	0.40
weighted	0.42	0.82	0.52

Значення макро-F1 для всіх міток зменшилось на 0.02 пункти, це означає, що результат експерименту відкидається. Зваження класів показує гірший результат, через, ймовірно, велику схожість більш поширених міток з менш поширеними, що означає модель частіше плутає їх між собою.

4.2 Фокальна функція втрат без ваг альфа

Застосовані параметри:

- значення параметра гамма варіювалося між 1.0, 2.0, та 3.0
- ваги альфа були вимкнені для цього варіанту моделі
- загальні гіперпараметри залишилися незмінними з базового експерименту.

Найкращий результат продемонстрував варіант зі значенням гамма 2.0, що відображено у таблиці 4.2.

Таблиця 4.2 – значення оцінок для міток разом

	precision(точність)	**recall(відгук)**	**f1**
admiration	0.68	0.74	0.71
amusement	0.77	0.90	0.83
anger	0.57	0.47	0.52
annoyance	0.49	0.22	0.30
approval	0.55	0.35	0.43
caring	0.51	0.40	0.45
confusion	0.51	0.34	0.41
curiosity	0.57	0.47	0.52
desire	0.73	0.40	0.52
disappointment	0.48	0.19	0.27
disapproval	0.46	0.30	0.36
disgust	0.59	0.44	0.50
embarassment	0.67	0.38	0.48
excitement	0.53	0.39	0.48
fear	0.70	0.69	0.70
gratitude	0.94	0.90	0.92
grief	0	0	0
joy	0.69	0.61	0.65
love	0.77	0.87	0.82
nervousness	0.47	0.30	0.37
optimism	0.62	0.49	0.55
pride	0.86	0.38	0.52
realization	0.47	0.12	0.20
relief	1	0.27	0.43
remorse	0.64	0.68	0.66
sadness	0.59	0.55	0.57
surprise	0.63	0.48	0.54
neutral	0.73	0.55	0.63

Продовження таблиці 4.2

	precision(точність)	**recall(відгук)**	**f1**
micro	0.68	0.53	0.60
macro	0.62	0.46	0.51
weighted	0.66	0.53	0.58

У результаті показник макро-F1 зріс на 0.09 пунктів, порівняно з базовою моделлю, що показує значний приріст в точності прогнозування міток, хоча в деяких випадках сама метрика точності (precision) зменшилась, і навіть зменшився результат обчислення мікроточності для всієї моделі на 0.01 пункт.

4.3 Фокальна функція втрат з постобробкою порогів

Було перебрано всі можливі значення порогів від 0.1 до 0.9 з кроком 0.01, визначено найкращі, та застосовано їх у фінальному обчисленні метрик (табл. 4.3).

Таблиця 4.3 – значення оцінок для міток окремо та разом обчислених оцінок

	precision(точність)	**recall(відгук)**	**f1**
admiration	0.67	0.76	0.71
amusement	0.77	0.91	0.83
anger	0.54	0.51	0.53
annoyance	0.33	0.53	0.41
approval	0.39	0.51	0.44
caring	0.50	0.41	0.45
confusion	0.41	0.52	0.46
curiosity	0.48	0.75	0.59
desire	0.58	0.60	0.59

Продовження таблиці 4.3

	precision(точність)	**recall(відгук)**	**f1**
disappointment	0.41	0.34	0.38
disapproval	0.37	0.56	0.45
disgust	0.57	0.53	0.55
embarassment	0.62	0.43	0.51
excitement	0.67	0.35	0.46
fear	0.76	0.69	0.72
gratitude	0.97	0.88	0.92
grief	0.75	0.50	0.60
joy	0.69	0.61	0.65
love	0.77	0.87	0.82
nervousness	0.43	0.39	0.41
optimism	0.55	0.62	0.58
pride	0.86	0.38	0.52
realization	0.28	0.34	0.30
relief	0.83	0.45	0.59
remorse	0.65	0.80	0.72
sadness	0.58	0.56	0.57
surprise	0.59	0.57	0.58
neutral	0.60	0.82	0.69
micro	0.57	0.68	0.62
macro	0.59	0.58	0.57
weighted	0.58	0.68	0.62

Видно, як чуттєво впало значення точності та в той самий час зросло значення відгуку для багатьох міток. Через те, що для цієї задачі однаково важливі обидві метрики, а значення макро-F1 зросло на 0.06 пунктів загалом, результат можна назвати успішним.

4.4 Фокальна функція втрат з постобробкою порогів та аугментований перефразуванням датасет

Після невдалої первинної спроби аугментувати датасет перефразуванням було прийнято рішення змінити модель перефразування на PEGASUS, що принесло значно кращі результати (таблиця 4.4)

Таблиця 4.4 – порівняння оригінальних та перефразованих прикладів

Оригінальний	Перефразований
i can also post pics of the receipt if anyone doesn’t believe me!!!	If anyone does not believe me, I will post pictures of the receipt.
Nope! His real name starts with a G. Weird spelling of a common name, and pretty identifiable so I’ve never wanted to use it 😷	He has a weird spelling of his name and I have never wanted to use it.
i would feel really uneasy knowing i can never visit other countries, i think i would rather have the 50K	I think I would rather have the 50K and be able to visit other countries.
What was the original post? If it was as awful as this one my comment would still apply!	If the original post was as bad as this one, my comment would still apply.
Talking about scary foreign people spreading disease is a classic xenophobic trope. Vile.	Talking about foreign people spreading disease is a classic example of chauvinism.

Але, все одно можна помітити, що велика частина семантичного контексту втрачається при такому методі аугментації, що є критично важливим для задачі класифікації емоції, що і демонструють нові значення метрик (табл. 4.5)

Таблиця 4.5 – обчисленні значення метрик

	precision(точність)	**recall(відгук)**	**f1**
micro	0.55	0.69	0.61
macro	0.55	0.58	0.55
weighted	0.56	0.69	0.61

Значення макро-F1 понизилось на 0.02 пункти, що не є задовільним результатом для продовження роботи з аугментованим датасетом.

Висновки до роділу 4

У цьому розділі було проведено декілька кіл тренувань моделі на наборі даних GoEmotions з метою визначення підходів, які покращують результат прогнозування.

Використання стандартної бінарної крос-ентропії з обернено-частотними вагами класів дало помірний відгук (0.82), але низьку точність (мікро-F1 0.45, макро-F1 0.40). Зважені BCE дещо погіршили макро-F1, що вказує на те, що просте перезважування може сплутати семантично схожі категорії.

Введення фокальної втрати без α-ваг ($\gamma = 2.0$) суттєво покращило загальне прогнозування рідкісних класів. Було спостережено збільшення макро-F1 на 0.09 (до 0.51).

Застосування порогової оптимізації ще більше збалансувало точність і відгук. Макро-F1 збільшилась до 0.57 (+0.06).

Нарешті, збільшення навчальної вибірки за допомогою перефразування не дало чистого приросту. Хоча різноманітність вибірки зросла, макро-F1 впала на 0.02, підтверджуючи, що нерозбірливе перефразування тексту може спотворювати тонкі емоційні ознаки і вносити шум.

ВИСНОВКИ

Представлену роботу спрямовано на реалізацію засобів глибинного навчання для тренування інформаційної моделі мульти-міткової класифікації емоцій. Після вивчення трансформерних архітектур, оцінено три відомі попередньо навчені трансформатори (BERT, RoBERTa, DeBERTa) і обрано RoBERTa за її найкращий результат балансування точності та відгуку, обчислювальної ефективності та стабільності навчання.

Виконано поєднання декількох інструментів для покращення результат моделі. Щоб зменшити виражений дисбаланс класів у GoEmotions, запроваджено фокальну функцію втрат та індивідуальні порогові значення для окремих міток.

Експериментально було підвищено оцінку макро-F1 з 0,42 (базова імплементація) до 0.51 за допомогою фокальної втрати і далі до 0.57 завдяки налаштуванню порогових значень. Виявлено, що більш агресивне перефразування за допомогою не дало чистого приросту, що підкреслює важливість збереження емоційних нюансів та підводить до використання інших методів аугментації.

Отримана модель в теорії підходить для застосування в режимі реального часу, наприклад, для аналізу реакцій користувачів інтернету, моніторингу тенденцій та поглибленого вивчення настроїв у маркетингових дослідженнях. Але, у більшості сфер не має потреби у визначення емоцій з настільки великим рівним нюансності, який забезпечило використання 27 різних міток.

Перспективні розширення включають адаптивне зважування втрат за допомогою метанавчання, інтеграцію динамічного енсемблінгу моделей, та застосування менш агресивних методів аугментації даних. Розвиваючись у цих напрямках, майбутні системи стануть ще більш чутливими до складної, багатогранної природи людських емоцій у мові.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Cook J.A. When to Consult Precision-Recall Curves / J.A. Cook, V. Ramadas // SSRN Electronic Journal. – 2019. – Режим доступу: https://doi.org/10.2139/ssrn.3350582
2. Data Augmentation in NLP Using Back Translation With MarianMT. Towards Data Science. [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/data-augmentation-in-nlp-using-back-translation-with-marianmt-a8939dfea50a/
3. Demszky D. GoEmotions: A Dataset of Fine-Grained Emotions / D. Demszky, D. Movshovitz-Attias, J. Ko та ін. // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. – Stroudsburg, PA, USA, 2020. – PP. 4040-4054. – Режим доступу: https://doi.org/10.18653/v1/2020.acl-main.372
4. Devlin J. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / J. Devlin, M.-W. Chang, K. Lee та ін. // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. – 2019. – Vol. 1. – PP. 4171-4186. – Режим доступу: https://doi.org/10.48550/arXiv.1810.04805
5. Fine-tuning. Hugging Face – The AI community building the future. [Електронний ресурс]. – Режим доступу: https://huggingface.co/docs/transformers/training#multi-label-text-classification
6. Fine-Tuning and Evaluating Large Language Models: Key Benchmarks and Metrics. Towards AI. [Електронний ресурс]. – Режим доступу: https://towardsai.net/p/data-science/fine-tuning-and-evaluating-large-language-models-key-benchmarks-and-metrics
7. Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing / Pengcheng He, Xiaodong Liu, Jianfeng

Gao та ін. // Cornell University. – 2021. – 16 с. – Режим доступу: https://doi.org/10.48550/arXiv.2111.09543

8. Labib F.H. EmoBERTa-X: Advanced Emotion Classifier with Multi-Head Attention and DES for Multilabel Emotion Classification / F.H. Labib, M. Elagamy, S.N. Saleh // Big Data and Cognitive Computing. – 2025. – Т. 9, № 2. – С. 48. – Режим доступу: https://doi.org/10.3390/bdcc9020048
9. Language Modeling with nn.Transformer and torchtext – PyTorch Tutorials 2.7.0+cu126 documentation. PyTorch. [Електронний ресурс]. – Режим доступу: https://pytorch.org/tutorials/beginner/transformer_tutorial.html
10. Liu Y. RoBERTa: A Robustly Optimized BERT Pretraining Approach / Y. Liu, M. Ott, N. Goyal та ін. // arXiv preprint. – 2019. – Режим доступу: https://doi.org/10.48550/arXiv.1907.11692
11. RoBERTa. Hugging Face – The AI community building the future. [Електронний ресурс]. – Режим доступу: https://huggingface.co/docs/transformers/model_doc/roberta
12. Tokenizers. Hugging Face – The AI community building the future. [Електронний ресурс]. – Режим доступу: https://huggingface.co/docs/tokenizers/index
13. Transformers. Hugging Face – The AI community building the future. [Електронний ресурс]. – Режим доступу: https://huggingface.co/transformers/
14. Zhang J. PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization / J. Zhang, Y. Zhao, M. Saleh та ін. // Proceedings of the 37th International Conference on Machine Learning. – 2020. – Режим доступу: https://doi.org/10.48550/arXiv.1912.08777

ДОДАТОК А

Лістинг програмної реалізації пайплайну тренування моделі

```
import torch;
import numpy as np;
from sklearn.metrics import f1_score, precision_score, recall_score;
from datasets import load_dataset, Dataset, concatenate_datasets, load_from_disk;
from transformers import MarianMTModel, MarianTokenizer, BertTokenizerFast, RobertaTokenizer, RobertaForSequenceClassification, DebertaForSequenceClassification, DebertaTokenizer, RobertaConfig, BertForSequenceClassification, AutoModelForSeq2SeqLM, AutoModelForSequenceClassification, AutoTokenizer, TrainingArguments, Trainer, DataCollatorWithPadding;
from tqdm import tqdm
import os
import emoji
import re

class MultiLabelDataCollator(DataCollatorWithPadding):
    def __call__(self, features):
        batch = super().__call__(features)
        batch["labels"] = batch["labels"].float()
        return batch

class FocalLoss(torch.nn.Module):
    def __init__(self, gamma=2.0, alpha=None, reduction='mean'):
        super().__init__()
        self.gamma = gamma
        self.alpha = alpha
        self.reduction = reduction
```

```
        def forward(self, logits, targets):
            bce_loss = torch.nn.functional.binary_cross_entropy_with_logits(logits, targets, reduction="none")
            probas = torch.sigmoid(logits)

            pt = targets * probas + (1 - targets) * (1 - probas)
            focal_term = (1 - pt) ** self.gamma

            loss = focal_term * bce_loss

            if self.alpha is not None:
                alpha_t = self.alpha * targets + (1 - self.alpha) * (1 - targets)
                loss *= alpha_t

            if self.reduction == 'mean':
                return loss.mean()
            elif self.reduction == 'sum':
                return loss.sum()
            return loss

    class CustomTrainer(Trainer):
        def __init__(self, *args, gamma=2.0, alpha=None, **kwargs):
            super().__init__(*args, **kwargs)
            self.focal_loss = FocalLoss(gamma=gamma, alpha=alpha)

        def compute_loss(self, model, inputs, return_outputs=False, **kwargs):
            labels = inputs.pop("labels").float().to(model.device)
```

Лістинг програмної реалізації пайплайну тренування моделі, аркуш 2

```
            outputs = model(**inputs)
            logits = outputs.logits

            if self.focal_loss.alpha is not None and
self.focal_loss.alpha.device != model.device:
                self.focal_loss.alpha =
self.focal_loss.alpha.to(model.device)

            loss = self.focal_loss(logits, labels)

            return (loss, outputs) if return_outputs else loss


    emotions_dataset = load_dataset("go_emotions");
    #                 emotions_dataset["train"]                 =
load_from_disk(r"E:/datasets/goemotions_train_augmented_pegasus_
2");
    labels                                                        =
emotions_dataset["train"].features["labels"].feature.names;
    num_labels = len(labels);

    model = RobertaForSequenceClassification.from_pretrained(
        "roberta-base",
        num_labels=num_labels
    )
    model.config.problem_type = "multi_label_classification"
    tokenizer = RobertaTokenizer.from_pretrained("roberta-base")

    def preprocess_function(examples):
        for text in examples['text']:
            text = text.lower()
            text = emoji.demojize(text)
            text = re.sub(r"http\S+|www\S+|https\S+", '', text)
```

Лістинг програмної реалізації пайплайну тренування моделі, аркуш 3

```
            text = re.sub(r"<.*?>", '', text)
            text = re.sub(r'\s+', ' ', text).strip()
        tokenized = tokenizer(examples["text"], truncation=True,
padding="max_length", max_length=128);
        tokenized["labels"] = [
            [float(i in label_list) for i in range(num_labels)]
            for label_list in examples["labels"]
        ]
        return tokenized;

    tokenized_dataset                                          =
emotions_dataset.map(preprocess_function,          batched=True,
load_from_cache_file=True)

    data_collator = MultiLabelDataCollator(tokenizer=tokenizer)

    train_dataset = tokenized_dataset["train"]
    test_dataset = tokenized_dataset["validation"]

    label_matrix = np.array([example["labels"] for example in
train_dataset])
    label_counts   =   label_matrix.sum(axis=0)      #   shape:
[num_labels]

    N_total = label_matrix.shape[0]
    N_pos = label_counts
    N_neg = N_total - N_pos

    pos_weight = N_neg / (N_pos + 1e-6)

    alpha = pos_weight / (pos_weight + 1.0)

    alpha_tensor = torch.tensor(alpha, dtype=torch.float32)
```

Лістинг програмної реалізації пайплайну тренування моделі, аркуш 4

```
    def compute_metrics(eval_pred):
        logits, labels = eval_pred
        probs = 1 / (1 + np.exp(-logits))
        preds = (probs > 0.5).astype(int)

        f1 = f1_score(labels, preds, average="weighted")
        precision = precision_score(labels, preds,
average="weighted")
        recall = recall_score(labels, preds, average="weighted")

        return {"f1": f1, "precision": precision, "recall":
recall}

    training_args = TrainingArguments(
        output_dir="./results/v1.1",
        eval_strategy="epoch",
        save_strategy="epoch",
        warmup_steps=500,
        lr_scheduler_type="linear",
        learning_rate=3e-5,
        weight_decay=0.01,
        per_device_train_batch_size=16,
        per_device_eval_batch_size=16,
        num_train_epochs=5,
        logging_dir="./logs",
        logging_steps=10,
        save_total_limit=2,
        load_best_model_at_end=True,
        metric_for_best_model="f1",
        greater_is_better=True
    )
```

Лістинг програмної реалізації пайплайну тренування моделі, аркуш 5

```
    trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=train_dataset,
        eval_dataset=test_dataset,
        tokenizer=tokenizer,
        data_collator=data_collator,
        compute_metrics=compute_metrics,
        #gamma=2.0,
        #alpha=None
    )

    trainer.train();
    trainer.save_model("./emotions_model/weighted_5epochs/weight
ed_f1");
```

Лістинг програмної реалізації пайплайну тренування моделі, аркуш 6