Міністерство освіти і науки України НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем Факультету інформатики

КУРСОВА РОБОТА

на тему «Порівняльне дослідження моделей обробки природної мови для класифікації текстів українською мовою»

за спеціальністю 121 «Інженерія програмного забезпечення»

Керівник курсової роботи
ст. викладач Кузьменко Д. О.
(прізвище та ініціали)

_____
(підпис)
“____” _____ 2025 р.

Виконала студентка Качинська І.В.
(прізвище та ініціали)
“____” _____ 2025 р.

Тема: Порівняльне дослідження моделей обробки природної мови для класифікації текстів українською мовою.

Календарний план виконання роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	7.10.24	
2.	Огляд наукової літератури та формування теоретичного підґрунтя за темою роботи.	26.01.25	
3.	Проведення дослідження.	10.02.25	
4.	Аналіз результатів та початок написання текстової частини.	25.03.25	
4.	Оформлення та підготовка до захисту.	04.04.25	
5.	Захист курсової роботи.		

Студентка Качинська І.В.

Керівник Кузьменко Д.В.

“_____” _____

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

NLP – Natural Language Processing

DL – Deep Learning

LM – Language Model

LLM – Large Language Model

BOW – Bag of Words

TF – Term frequency-inverse document frequency

ANN – Artificial Neural Networks

RNN – Recurrent Neural Networks

LSTM – Long Short-Term Memory Networks

GRU – Gated Recurrent Units

Seq2Seq – Sequence to Sequence

MCC – Matthews Correlation Coefficient

TP – True positives

TN – True negatives

FP – False positives

FN – False negatives

ВСТУП

У сучасному інформаційному суспільстві обробка природної мови (NLP) відіграє ключову роль у розробці інтелектуальних систем, здатних працювати з великими обсягами текстових даних. Технології NLP активно застосовуються в автоматичному перекладі, аналізі настроїв, пошукових системах, чат-ботах та в багатьох інших сферах. Одним із основних завдань є текстова класифікація, зокрема — багатокласова класифікація повідомлень.

Попри значний прогрес у розвитку моделей для англійської мови, для української мови обсяг доступних ресурсів та адаптованих моделей значно більш обмежений. Це створює потребу в оцінці ефективності існуючих рішень для роботи з українськомовним текстом.

У рамках цієї курсової роботи проведено порівняльний аналіз двох підходів до класифікації українських текстів:

- використання локально доступних трансформерних моделей (наприклад, BERT, XLM-R, RoBERTa), які можна донавчати на специфічних даних;
- використання великих мовних моделей (LLM) через API-платформи (зокрема Groq), які забезпечують високий рівень генеративних можливостей без потреби у локальних обчислювальних ресурсах.

Метою дослідження є порівняння точності, швидкодії та практичних переваг цих моделей при вирішенні задачі багатокласової класифікації українськомовних повідомлень. Для цього було використано штучно збалансований датасет з 1000 текстів, поділених за категоріями.

Актуальність цієї роботи зумовлена зростанням потреби у створенні надійних інструментів автоматичного аналізу українського тексту, зокрема для медіа, освітніх платформ та служб модерації контенту. Оцінка ефективності різних архітектур дозволяє сформулювати рекомендації щодо вибору моделей у практичних застосунках.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ

1.1 Обробка природної мови

Обробка природної мови (NLP) - це напрямок штучного інтелекту, метою якого є надання комп'ютеру здатності розуміти, генерувати та маніпулювати людською мовою [1]. Ключова задача NLP полягає в нівеляції розриву між складною і варіативною людською мовою та логічними і структурованими числовими представленнями, зрозумілими для комп'ютера. Хоча дослідження в галузі NLP розпочались ще в 1950-их роках, після винаходу цифрових комп'ютерів, але значного прогресу вони досягли нещодавно, завдяки розвитку машинного навчання (ML), зокрема глибокого навчання (Deep Learning). Моделі глибинного навчання здатні виділяти складні мовні патерни за допомогою багатошарових нейронних мереж, які навчаються на великих наборах неструктурованих даних (Детальніша інформація про передумови до їх розробки та особливості архітектури надана в наступних розділах роботи). Варто також зазначити, що побудова глибоких нейронних мереж вимагає значних обчислювальних потужностей та великої кількості часу. Для сучасних моделей обсяг навчальних даних може сягати десятків ГБ, а сам процес тренування розтягуватись на тижні. Це викликає додаткові ускладнення при впровадженні їх в практичні застосунки.

1.2 Transfer learning та fine-tuning

Основною технікою, яка робить можливим широке використання глибоких нейронних мереж в NLP є transfer learning. Її суть полягає в тому, щоб знання, що були отримані в рамках одного завдання чи датасету, використовувати для покращення продуктивності моделі для іншого пов'язаного завдання та/або датасету (Рисунок 1.1). Іншими словами, трансферне навчання використовує отримані знання з одного середовища для покращення узагальнення в іншому середовищі [2].

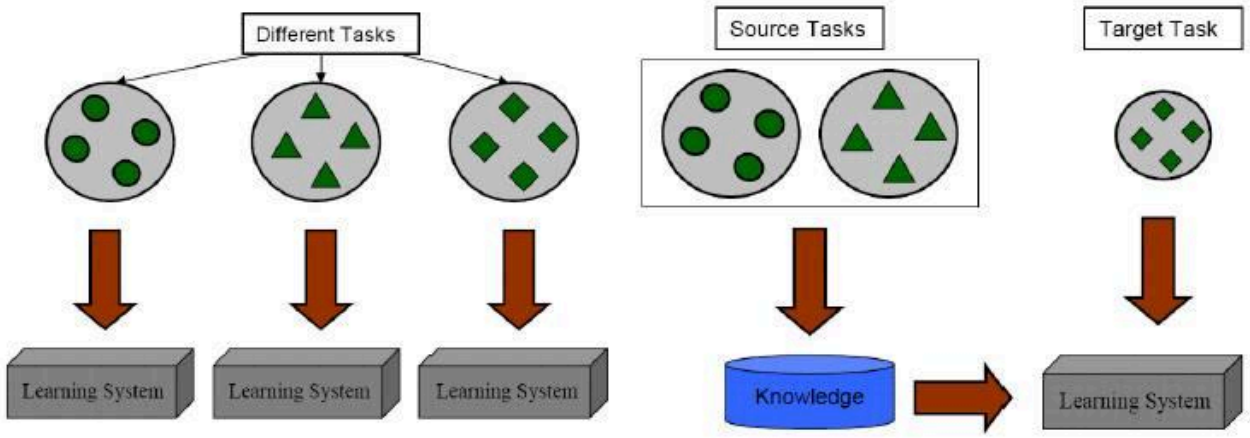


Рисунок 1.1 Порівняння класичного машинного навчання та transfer learning

Найпростішим видом трансферного навчання є fine-tuning. За цього підходу ми використовуємо вже попередньо навчену (pre-trained) на великих корпусах даних модель та адаптуємо її для вирішення нової конкретної задачі, задіявши при цьому значно менший набір даних для навчання, ніж цього потребувала б нова модель. Теоретично, цей набір даних можна було б використовувати і для початкового навчання, проте в такому випадку існував би великий ризик перенавчання (overfitting). За такого сценарію, модель демонструє хорошу точність на навчальній вибірці, але отримує поганий результат, опрацьовуючи нові дані.

Таким чином, fine-tuning забезпечує використання широких знань, отриманих внаслідок попереднього навчання на величезному наборі даних, та вдосконалює розуміння моделлю більш конкретних концепцій внаслідок донавчання, без потреби у значній кількості ресурсів [3]. За використання цього підходу сформувались цілі екосистеми вже готових pre-trained моделей (наприклад, доступних через платформу Hugging Face), які є у відкритому доступі та можуть бути використані розробниками для інтеграції у практичні задачі [1].

1.3 Класифікація текстів як задача NLP

З кожним днем можливості NLP розширюються і наразі галузь охоплює широке коло завдань, зокрема, такі як робота віртуальних асистентів, оптимізація пошукових систем (SEO), автоматична фільтрація спаму, машинний переклад текстів, сумаризація довгих документів, аналіз тональності та навіть перевірка граматики та орфографії [4]. Однією з фундаментальних та широко застосовуваних задач в NLP є класифікація текстів. За своєю суттю, класифікація тексту - це процес автоматичного аналізу певного відкритого текстового фрагмента та призначення йому однієї або декількох заздалегідь визначених категорій [5]. Під час цього дослідження розглядається саме багатокласова класифікація, за якої кожному фрагменту присвоюється одна мітка з фіксованого набору. Ручна класифікація є дороговартісним та неточним процесом, тому застосування моделей машинного навчання для цієї задачі дозволяє відносно швидко та якісно отримати аналітику даних.

1.4 Еволюція мовних моделей

Для глибшого розуміння можливостей, обмежень та принципів роботи сучасних мовних моделей, таких як трансформери та LLM, важливо розглянути ключові етапи розвитку моделей загалом. Це допоможе усвідомити, як еволюція архітектур і методів навчання вплинули на сучасні методи обробки природної мови.

1.4.1 Rule-Based and Retrieval-Based Models

Першою програмою, розробленою з використанням NLP, вважається ELIZA. Цей чат-бот використовував розпізнавання патернів (pattern recognition) для симуляції розмов. Вхідні дані від користувача перетворювались на питання, після чого здійснювалась генерація відповіді на основі раніше визначених правил [18]. Такі Rule-Based моделі були обмежені своєю базою знань та набором правил, визначених програмою, що ускладнювало їх розробку та масштабування, проте вони стали основою для подальших розробок.

Також, до ранніх належать моделі на основі пошуку (Retrieval-Based models). Такі системи обирали відповідь на запит користувача з заздалегідь підготовленої бази даних або набору шаблонів. Сам вибір ґрунтувався на зіставленні патернів вхідного повідомлення з парами “запит-відповідь” у базі [19]. Хоча відповіді retrieval-based моделей були досить точними (в межах бази знань), проте через обмежену кількість наперед заданих відповідей, діалог з моделлю був шаблонним та повторюваним.

1.4.2 Statistical Models and Techniques

Наступним етапом розвитку LMs стало використання статистичних підходів. Вони відмовились від чітких правил і натомість аналізували великі обсяги реальних текстових даних, виявляючи ймовірнісні закономірності мови.

Першими та одними з найпоширеніших серед статистичних моделей були N-грамні моделі (N-Grams). Вони використовувались для передбачення ймовірності появи наступного N-нного слова в послідовності, базуючись на N-1 попередніх словах [19].

Ще однією поширеною технікою для моделювання мови є “мішок слів” (bag of words). Він вважається найпростішим способом перетворення тексту в числове представлення, використовуючи частоту слів для представлення документу у вигляді вектора фіксованої довжини.

Окрему увагу отримав показник TF-IDF (частота терміна, обернена до частоти документа), який оцінює релевантність слова для певного набору документів. Він, порівняно з BOW, враховує дві метрики: частоту слова в одному документі та частоту входжень у всьому корпусі.

Попри те, що статистичні моделі вимагали наявності значних обчислювальних ресурсів та великих наборів даних, саме вони (зокрема N-Grams) продемонстрували важливість врахування контексту, тим самим заклавши основи для подальших, більш просунутих технік [18].

1.4.3 Neural Network-Based Models

Розробка штучних нейронних мереж (ANN) стала новим потужним кроком в обробці природної мови. ANN були створені за прототипом нейронних зв’язків в мозку людини. Загальну структуру цих мереж можна охарактеризувати так:

- Мережі містять штучні нейрони (units). Серії нейронів утворюють шари мережі.
- Кожен шар може мати від десяти до кількох мільйонів нейронів.
- Будь-яка мережа має вхідний, вихідний та один або кілька прихованих шарів (Рисунок 1.2).
- Вхідний шар приймає початкові дані, які далі нелінійно проходять через приховані шари, де модель вивчає особливості та складні закономірності даних, після чого з вихідного шару формується результат.
- Шари нейронів пов’язані між собою. Кожен зв’язок має вагу, яка визначає як нейрон впливає на інші нейрони. Під час навчання мережа змінює ваги зв’язків та шарів, аби досягти мінімальних втрат [20].

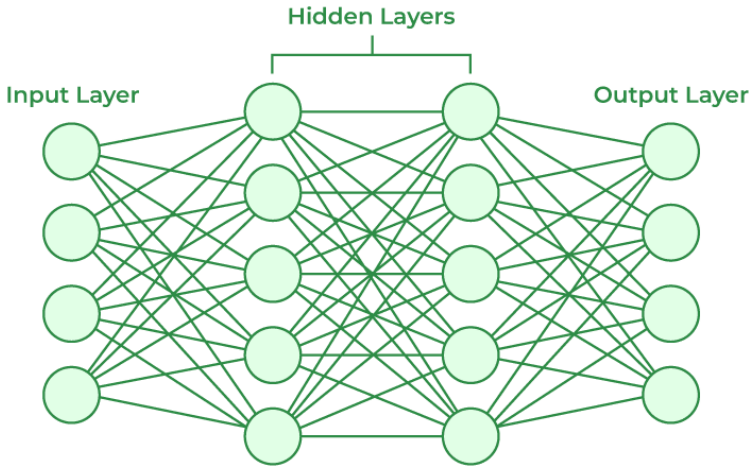


Рисунок 1.2 Архітектура ANN

Окремим типом ANN є рекурентні нейронні мережі (RNN), які були розроблені для обробки послідовних даних і числових рядів. За допомогою feedback loop механізму (вихідний результат повертався до нейрона як вхідний

для наступного часового кроку) мережа набула здатності розуміти попередній контекст і робити більш точні передбачення [21].

Проте при обробці довгих послідовностей RNN стикались з проблемою зникаючого градієнта, через яку їм було складно зберігати довгострокові залежності. Цю проблему частково вирішили мережі з довгою короткотривалою пам'яттю (LSTM). Вони мають спеціальні вентилі (gates), які "вирішують", які дані потрібно зберігати, а які можна забути [22]. Такий механізм навчання суттєво покращив якість моделей, проте є дорожчим з точки зору обчислень. Пізніше були запропоновані вентильні рекурентні одиниці (GRU), які є спрощеним варіантом LSTM з меншою кількістю параметрів та простішою архітектурою (2 вентилів замість 3, та відсутність спеціальної комірки пам'яті). Це зробило їх швидшими у навчанні та менш вимогливими до ресурсів.

1.4.4 Sequence to Sequence Models

Наступним етапом розвитку мовних моделей стали sequence to sequence models (seq2seq). Це тип нейронних мереж, ключовими компонентами архітектури якого є encoder та decoder. Encoder має RNN-based архітектуру. Він поелементно читає вхідну послідовність та інкапсулює значення у векторне представлення фіксованої довжини (context vector). Декодер (також реалізований на основі RNN) приймає цей контекстний вектор і поступово формує вихідну послідовність. На кожному кроці роботи генерується розподіл ймовірностей для потенційних елементів вихідної послідовності. Потім декодер вибирає з цього розподілу наступний елемент послідовності і поступово формує вивід [18].

Seq2seq моделі були дуже популярними для задач машинного перекладу. Проте, незважаючи на свій успіх, вони теж мали обмеження, що переважно стосуються RNN-based архітектури. Основною проблемою була неспроможнійсть моделі виділяти важливі частини послідовності з-поміж інших частин і, відповідно, належним чином їх засвоювати. Тож цей недолік і

став передумовою для наступного прориву – attention-based models та, зокрема, архітектури трансформерів.

1.4.5 Attention-Based Models

Щоб подолати обмеження рекурентних архітектур (RNN, LSTM, GRU), потрібно було знайти принципово нове рішення. Ним став механізм уваги (attention mechanism), який спочатку був запропонований для покращення seq2seq моделей. Основна ідея полягала в тому, щоб, маючи доступ до всієї вхідної послідовності, фокусуватись лише на найбільш релевантних її частинах. Фактично, механізм обчислював вагові коефіцієнти уваги, які і визначали відносну важливість кожного фрагменту для задачі. Потім ці коефіцієнти застосовувались для регулювання впливу кожної з частин на модель [23]. Завдяки цьому більше не було потреби стискати всю інформацію у вектор фіксованої довжини, що покращило якість обробки довгих речень.

Проте справжній прорив стався після представлення моделі трансформера у фундаментальній статті “Attention is All You Need” [24]. Трансформери стали першими моделями, що повністю відмовилися від рекурентних і згорткових шарів на користь механізму уваги (зокрема, self-attention) для створення глобальних залежностей між вхідними та вихідними послідовностями [24]. Однією з основних переваг такого підходу є можливість паралельної обробки токенів, що значно пришвидшує процес навчання у порівнянні з послідовною обробкою в RNN. Наразі архітектура трансформера вважається однією з фундаментальних моделей в галузі штучного інтелекту [19].

1.4.6 Large Language Models

Крайнім етапом еволюції мовних моделей наразі є великі мовні моделі (Large Language Models). Вони навчаються на надзвичайно великих обсягах даних і здатні виконувати широкий спектр задач NLP без додаткового навчання, що робить їх практично універсальними. LLM мають мільярди параметрів, завдяки чому можуть легко визначати складні закономірності і демонструвати

відносно хороший результат роботи. Та, незважаючи на ефективність цих моделей, вони все ще мають низку обмежень, які потребують подальшого вдосконалення.

1.5 Архітектура трансформера

Оскільки архітектура трансформера лежить в основі більшості сучасних мовних моделей, зокрема і тих, що розглядаються в цій роботі, критично важливим є розуміння ключових принципів її роботи.

1.5.1 Компоненти трансформера

Як вже згадувалось в попередніх розділах, особливістю архітектури Трансформер є механізм уваги (attention mechanism). Він фіксує зв’язки та залежності між всіма токенами вхідної послідовності, незалежно від відстані між ними. Механізм присвоює вагові коефіцієнти кожному елементу вхідної послідовності, цим самим пріоритезуючи їх [25]. Архітектура трансформерів включає в себе 5 типів механізмів уваги: scaled dot-product attention, multi-head attention, self-attention, encoder decoder attention та masked self-attention. Ключовим серед них є self-attention механізм, який при обробці кожного токена у послідовності дозволяє динамічно оцінювати важливість усіх інших токенів і формувати представлення поточного токена в контексті цієї інформації.

Більшість трансформерних моделей складаються з двох основних блоків – стеку енкодерів (Encoder) та стеку декодерів (Decoder), як показано на Рисунку 1.3.

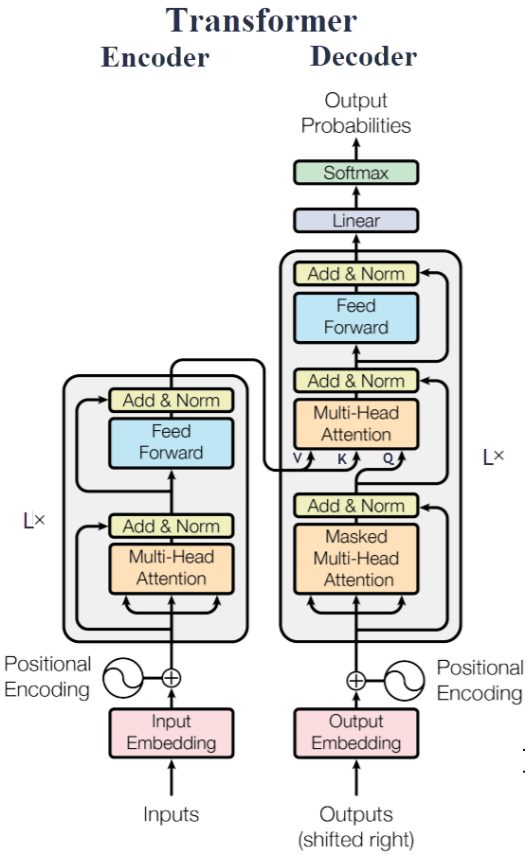


Рисунок 1.3 Класична архітектура моделі Трансформер

Енкодер відповідальний за обробку вхідної послідовності та створення на її основі послідовності контекстуалізованих векторних представлень. Кожен з шести шарів енкодера зазвичай містить два основні підшари. Перший - це multi-head self-attention механізм, який дозволяє зважувати важливість слів у послідовності, а другий – повнозв'язна нейронна мережа (feed-forward network), яка незалежно застосовується до кожного представлення слова [24].

Структура декодера є дещо схожою, він теж має шість шарів, але кожен з них має три основні підшари. До двох вже згаданих додається masked multi-head attention. Цей механізм гарантує, що передбачення кожного токена залежить тільки від попередньо згенерованих токенів. Таким чином відбувається послідовна генерація тексту, що і є головним завданням декодера.

1.5.2 Застосування трансформерів для задач класифікації

Базовим принципом для розв'язання задач класифікації тексту є використання Encoder-only models. Це моделі, які використовують лише стек енкодерів з архітектури трансформерів. Тобто, такі моделі здатні розуміти

вхідну послідовність, але не можуть генерувати текст (через відсутність декодер частини) [26].

Першим етапом класифікації є токенізація вхідного тексту, а потім – його передача на вхід трансформеру (загальна схема зображена на Рисунку 1.4). На виході енкодера утворюється послідовність контекстуалізованих векторів, проте, щоб отримати передбачення категорії, необхідно використати класифікаційну голівку (classification head).

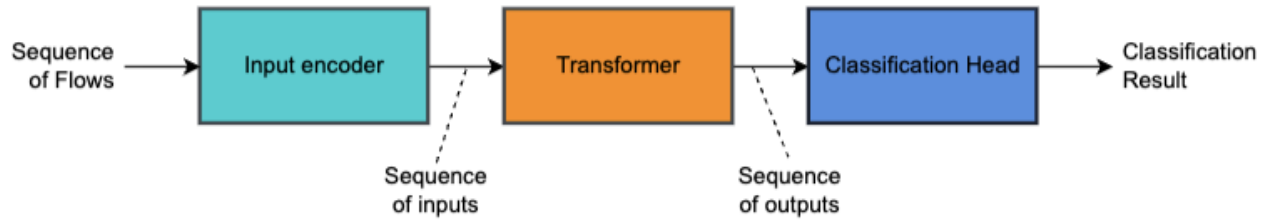


Рисунок 1.4. Схема класифікації тексту з використанням архітектури Трансформер

Основне завдання голівки полягає в агрегуванні вихідної послідовності трансформера у вектор фіксованого розміру і перетворенні його в передбачення класу [27]. У найпопулярніших моделях, таких як BERT, для агрегування використовується спеціальний токен [CLS], який слугує своєрідним плейсхолдером для збереження контексту всієї послідовності після проходження шарів трансформера [28].

Підсумовуючи, задача класифікації за допомогою моделей на архітектурі Трансформер зводиться до обробки модифікованих вихідних векторів за допомогою класифікаційної голівки. Варто також зазначити, що такий підхід використовується не лише в класичних трансформерних моделях, але і в великих мовних моделях.

1.6 Особливості роботи з LLM

1.6.1 Визначення та загальні характеристики LLM

Large Language Models - це моделі глибокого навчання з мільярдами параметрів, які навчаються на величезних корпусах даних. В основі більшості

LLM лежить вже згадана вище архітетура трансформерів (encoder та decoder), проте від класичних трансформерів вони відрізняються масштабом та можливостями.

Впродовж навчання мовна модель ітеративно змінює значення своїх параметрів, поки не навчиться правильно визначати наступний токен з послідовності. Таке можливо завдяки механізму самонавчання, який показує моделі як потрібно коригувати параметри для максимізації ймовірності правильного передбачення в навчальних прикладах [29].

Великі мовні моделі здатні узагальнювати знання, генерувати текст, подібний до створеного людиною, та ефективно виконувати різні задачі NLP без додаткового специфічного навчання.

1.6.2 Парадигма спонукання (Prompting)

Завдяки можливості LLM виконувати завдання без етапу fine-tuning, виникла нова парадигма у взаємодії з моделями – контекстне навчання (in-context learning). Цей підхід не змінює параметрів моделі, натомість завдання подається у вигляді запиту-інструкції (prompt) [30]. Модель аналізує вхідний prompt і виділяє контекстні закономірності (наприклад, очікувану структуру виходу), на основі яких відбувається генерація відповіді. Контекстне навчання більше стосується розпізнавання шаблонів, аніж вивчення нових знань.

Існують різні підходи до формування промптів [31]:

- Zero-Shot Prompting: на вхід моделі подається лише опис завдання та вхідні дані без прикладів виконання. За цього підходу модель лише з інструкції повинна зрозуміти яким чином сформувати відповідь та її зміст.
- Few-Shot Prompting: до інструкції та вхідних даних додається декілька коректно оформлених прикладів виконання. Це задає додатковий контекст, який здатен покращити якість відпрацювання.

- Chain-of-Thought (CoT) Prompting полягає в розбитті завдання на менші частини. При цьому підході на вхід моделі подається перелік дій, які потрібно виконати для формування коректної відповіді. Також заохочується пояснення ходу міркувань моделі на кожному з цих кроків. Такий підхід полегшує розуміння “логіки” моделі та спрощує оцінку її роботи.

Варто зазначити, що LLM можна донавчати звичним способом, для адаптації до певної специфічної задачі, або для покращення здатності слідувати інструкціям (instruction fine-tuning), проте, через масштаб моделей, такий підхід є досить ресурсоємним процесом. У межах цієї роботи взаємодія з LLM відбувалась лише через prompting, без попереднього донавчання.

РОЗДІЛ 2. МЕТОДОЛОГІЯ ЕКСПЕРИМЕНТІВ

2.1 Датасет

Вибір якісного та репрезентативного набору даних є надзвичайно важливим для порівняльного дослідження моделей обробки природної мови. Оскільки саме якість, розмір та характеристики навчальних визначають здатність моделей до узагальнення і подальшої роботи з новими даними. Повнота тестової вибірки теж важлива, оскільки при тестуванні LLM без попереднього донавчання якість вмісту запиту безпосередньо впливає на релевантність отриманого результату.

2.1.1 Опис та обґрунтування вибору набору даних

В цій курсовій роботі для порівняння моделей було використано датасет FIdo-AI/ua-news, який був створений неприбутковою студентською організацією Fido.ai Національного університету “Києво-Могилянська Академія”. Доступ до даних відбувався за допомогою Hugging Face API [17]. Вибір саме цього набору даних було зумовлено низкою факторів:

- По-перше, датасет складається з статей новин повністю українською мовою та містить дані відносно високої якості, що є важливим фактором, враховуючи обмежену кількість публічно доступних корпусів для української мови.
- По-друге, всі записи датасету належать до однієї з 5 категорій: бізнес, новини, технології, політика та спорт. Це робить його придатним для вирішення задачі багатокласової класифікації.
- По-третє, значний обсяг (близько 150 тисяч записів) набору є достатнім для ефективного донавчання сучасних потужних моделей NLP.

2.1.2 Структура даних та попередня обробка

Початковий набір даних містив 4 поля для кожної статті:

– title (заголовок)

– text (основний текст)

– tags (перелік ключових слів, не використовувався при проведенні експериментів)

– target (істинна категорія новини).

Для подальшої ідентифікації записів програмно було додано колонку id, з унікальним послідовним номером для кожної зі статей.

Оскільки заголовок часто містить стиснуту інформацію про зміст всієї статті, колонки title і text було конкатеновано в одну колонку content (з використанням роздільника “. ”). Зміст цієї колонки, власне, і виконував роль вхідного тексту для моделей в подальших експериментах. Завдяки здатності сучасних LM виявляти ключові для виконання завдань патерни на відносно необроблених текстових даних, інших специфічних обробок тексту (наприклад, лематизації, видалення стоп-слів) не проводилось.

Також, важливим етапом попередньої обробки даних стало перетворення текстових міток класів з колонки target на числовий формат. Кожній категорії було присвоєно унікальний індекс від 0 до 4 (Рисунок 2.1). Отримання числових міток є необхідним для донавчання трансформерних моделей та уніфікованого підрахунку метрик оцінки ефективності.

Рисунок 2.1. Зміст json-файлу з перетвореннями текстових міток в числові

2.1.3 Розмір вибірок та розподіл класів

Загальний обсяг даних, що було використано для донавчання трансформерних моделей у цьому дослідженні, налічує 120 417 статей новин.

Для того, щоб уникнути перенавчання (overfitting) та забезпечити коректність оцінки процесу навчання, цей набір даних було розділено на навчальну та валідаційну вибірки у співвідношенні 90% до 10%. Тому кінцева навчальна вибірка складається з 108 375 записів, а валідаційна, в свою чергу, містить 12 042 записи. Розділення відбувалось за допомогою стратифікованої випадкової вибірки (з фіксованим станом генератора seed = 42 для відтворюваності результатів), що дозволило зберегти початкові пропорції класів в обох наборах. Як видно з Таблиці 2.1, навчальні та валідаційні дані мають природний дисбаланс класів (наприклад, новин з класом “політика” значно більше, ніж з класом “технології”). Це створює додаткове навантаження при навчанні моделей, а також є важливим фактором для обрання метрик їх оцінки.

Таблиця 2.1 – Розподіл категорій у навчальній та валідаційній вибірці

Категорія	**Кількість**	**Відсоток (%)**
бізнес	14 759	12.3
новини	25 209	20.9
політика	40 364	33.5
спорт	28 438	23.6
технології	11 647	9.7
Всього	**120 417**	**100.0**

Фінальна оцінка всіх протестованих моделей у дослідженні здійснювалась на тестовому датасеті, що містить 1000 записів. Він був створений з штучним розподілом даних, який проілюстровано в Таблиці 2.2. Це було зроблено з метою перевірки здатності моделей переносити вивчені патерни на вибірку, що має інші пропорції класів, а також слугує своєрідною імітацією реальних даних, які рідко бувають збалансованими.

Таблиця 2.2 – Розподіл категорій у тестовій вибірці

Категорія	**Кількість**	**Відсоток (%)**
бізнес	90	9.0

новини	180	18.0
політика	420	42.0
спорт	270	27.0
технології	40	4.0
Всього	**1000**	**100.0**

Важливо зазначити, що тестова вибірка не брала участі в етапі налаштування чи навчання моделей, а валідаційна вибірка використовувалась лише для моніторингу процесу навчання та вибору найкращої версії донавчених моделей.

2.2 Обрані моделі для дослідження

Важливим етапом методології даного дослідження є вибір репрезентативного набору мовних моделей для аналізу ефективності на задачі класифікації новин українською мовою. Для досягнення цієї мети було обрано сім моделей, які відрізняються за архітектурою, розмірами, особливостями перед-навчання (pre-training) та специфікацією. Зокрема, три локально розгорнуті трансформерні моделі сімейств BERT та RoBERTa та чотири великі мовні моделі (LLM), що належать до сімейств Llama, Qwen та Mistral. Такий різноманітний набір дозволяє оцінити які фактори впливають на якість обробки текстів саме українською мовою. Нижче наведено детальні характеристики кожної моделей та обґрунтування їх вибору.

1. BERT-base-multilingual-cased.

BERT (Bidirectional Encoder Representations from Transformers) – це нейронна мережа, яка була розроблена компанією Google AI в 2018 році. Вона побудована на основі архітектури трансформерів, має 12 шарів (layers), 12 незалежних голів уваги, а кожен токен представляється вектором розмірності 768 (для base версії). Модель використовує двонаправлений механізм уваги, який дозволяє обробляти і лівий і правий контекст кожного токена.

Bert-base-multilingual-cased містить 177 857 285 параметрів і може оперувати 104-ма мовами, серед яких є і українська. Її навчання відбувалось на великих мультимовних корпусах даних таких як Wikipedia (~ 2.5 мільярди слів) та BooksCorpus (~800 мільйонів слів) [32]. Для локального використання модель вимагає приблизно 3.5-4 ГБ відеопам'яті (VRAM). Що стосується FLOPs-оцінки інференсу - це приблизно 66 GFLOPs.

Саме ця модель була обрана для дослідження, оскільки вона є стандартним та визнаним багатомовним baseline і слугує важливим орієнтиром для порівняння ефективності класифікації українських текстів іншими багатомовними моделями.

2. XLM-RoBERTa-base.

Це ще одна багатомовна трансформерна модель, яка була розроблена у 2020 році компанією Meta AI і поєднує в собі переваги від двох своїх попередників: XLM, який розширив метод маскованого мовного моделювання на декілька мов, і RoBERTa, що являє собою вдосконалену версією BERT, відмовившись від завдання прогнозування наступного речення (next sentence prediction) та використовуючи покращені гіперпараметри [33].

XLM-RoBERTa-base має 278 047 493 параметрів, 12 шарів, 12 attention heads та таку ж як у BERT розмірність векторів – 768. Модель була навчена на CommonCrawl корпусі, який містить близько 2.5 ТБ даних, та підтримує 100 різних мов, зокрема й українську. Завдяки великим обсягам навчальних даних модель демонструє вищу якість багатомовної репрезентації, особливо на низькоресурсних мовах. FLOPs-оцінка інференсу складає приблизно 66 GFLOPs, а для локального використання потрібно близько 4.5–5 ГБ VRAM.

Цю модель було обрано для дослідження, оскільки вона є однією з найсильніших багатомовних трансформерів з відкритим кодом. Крім того, важливо було оцінити, як покращена, порівняно з BERT, архітектура впливає на якість класифікації тексту. Також, однією з причин була наявність в дослідженні трансформерної моделі, донавченої спеціально для роботи з українською

мовою. Тож порівняння цих двох моделей є особливо цінним в контексті задачі цієї курсової роботи.

3. Ukr-RoBERTa-base

Ukr-RoBERTa є адаптованою версією RoBERTa, донавченою спеціально для української мови командою YouScan. Модель має таку ж саму архітектуру, що і roberta-base: 12 шарів, 12 голів уваги, 768 прихованих одиниць та 125 981 957 параметрів.

Ukr-roberta була натренована на великих україномовних корпусах (загальним обсягом 2.59 мільярди слів), зокрема українській вікіпедії, Ukrainian OSCAR deduplicated dataset, а також на наборі даних, зібраному в соцмережах [34]. Навчання моделі тривало 85 годин на 4 графічних процесорах NVIDIA V100. FLOPs-оцінка інференсу становить близько 66 GFLOPs, а обсяг необхідної для локального використання VRAM – 3-3.5 ГБ.

Використання саме цієї моделі дозволяє оцінити, наскільки корисним є донавчання моделі під конкретну мову, навіть за використання значно меншої кількості параметрів порівняно з універсальними багатомовними моделями.

4. Mistral-7B-Instruct-Ukrainian

Згадана вище модель є адаптованою для української мови версією базового трансформера Mistral-7B-v.2 і була розроблена командою Sherlock Assistant. Вона має приблизно 7 мільярдів параметрів та підтримує довжину контексту до 32 768 токенів [35].

Джерелами для донавчання є UA-SQUAD, українська вікіпедія, Ukrainian StackExchange, UAlpaca, Ukrainian Subset з Belebele та XQA та корпус ЗНО з UNLP 2024. Завдяки такій різноманітності навчальних даних модель може виконувати широкий спектр задач, наприклад, генерацію тексту чи надання інструкційних відповідей. Для локального запуску цієї моделі необхідно близько 14.4 ГБ VRAM.

Цю модель було включено до дослідження як приклад сучасної LLM з фокусом лише на українську мову. Це дозволило порівняти її якість з багатомовними LLM.

5. LLaMA-3.3-70b-versatile

Ця модель є однією з найпотужніших у сім'ї LLaMA та була розроблена в 2024 році компанією Meta AI. Вона побудована на основі архітектури Трансформер з використанням лише стака декодерів. Модель має близько 70 мільярдів параметрів, підтримує контекстну довжину до 8-ми тисяч токенів та реалізує Grouped-Query Attention механізм, що забезпечує покращену якість генерації. Хоча в переліку офіційно підтримуваних мов немає української, проте завдяки використанню величезних мультимовних корпусів для навчання, модель здатна обробляти запити українською мовою. Для локального запуску цієї LLM потрібно приблизно 45-65 ГБ VRAM.

LLaMA-3.3-70b була обрана для дослідження як еталонна модель великого масштабу. Її використання дозволяє оцінити роботу LLM загального призначення на задачі класифікації без попередньої адаптації до мови чи тематики завдання.

6. Qwen-2.5-coder-32b

Qwen- це велика мовна модель, орієнтована на генерацію коду, що була розроблена компанією Alibaba Cloud у 2024 році. Попри свої середні, як для LLM, розміри, модель демонструє високу якість логічних міркувань та здатність слідувати інструкціям. Архітектурно qwen-2.5-coder-32b є decoder-only трансформером, який включає 64 шари, 32.5 мільярди параметрів та підтримує довжину контексту до 128 тисяч токенів. Окрім мов програмування, розробники зазначають підтримку ще 29 природних мов. Для локального розгортання модель потребує орієнтовно 30-40 ГБ VRAM.

Модель була обрана для дослідження завдяки своїй здатності вирішувати складні логічні задачі. Вона показала один з найкращих серед LLM результатів, при виборі попередньому тестуванні через Groq Playground. Таким чином, в контексті цієї робити, qwen - це приклад LLM, оптимізованої під певну задачу, яка водночас демонструє хороші результати для інших завдань без додаткової адаптації.

7. Mistral-saba-24b

Mistral-saba-24b - це спеціалізована велика мовна модель, розроблена для регіонального використання з фокусом на мови Близького Сходу. Вона побудована на основі decoder-only трансформера та підтримує Grouped Query і Sliding-Window Attention, що надає змогу працювати з послідовностями довжиною до 32 тисяч токенів.

Модель має 24 мільярди параметрів і, за словами розробників, демонструє високу точність в завданнях, де потрібно враховувати регіональну специфіку. Хоча українська мова не входить до офіційного переліку підтримуваних мов, але в процесі тестування модель продемонструвала навички її використання. Для локального розгортання необхідно орієнтовно 30-40 ГБ VRAM.

Mistral-saba-24b була включена до дослідження як базова багатомовна LLM для порівняння з Mistral-7B-Instruct-Ukrainian. Це дозволяє оцінити вплив спеціалізованого донавчання LLM на якість класифікації за наявності однакових архітектур. Крім того, вона є найменшою з трьох багатомовних LLM, протестованих у роботі, що робить її цікавим орієнтиром з точки зору балансу між ресурсами і точністю.

2.3 Експерименти

2.3.1 Налаштування та реалізація prompting для LLM

Як вже пояснено в Розділі 1, для тестування LLM можна використовувати як тонке навчання (fine-tuning), так і спонукання (prompting). Оскільки великі мовні моделі мають хорошу здатність до узагальнення знань, в цьому дослідженні тестування LLM відбувається лише за допомогою спонукання, щоб продемонструвати її ефективність на прикладі задачі класифікації українських новин.

Взаємодія з трьома багатомовними моделями (LLaMA-3.3, Qwen та Mistral) здійснювалась за допомогою хмарного сервісу Groq API [36], який

надає можливість безкоштовно надсилати запити до загальнодоступних моделей, хоча й має низку обмежень. Наприклад, для LLaMA-3.3-70b-versatile вони становлять:

– 30 запитів на хвилину;

– 1 000 запитів на день;

– 12 000 токенів на хвилину;

– 100 000 токенів на день.

На перший погляд, ліміти здаються не строгими, проте, на практиці, через великі обсяги статей новин, вони спричиняли значні труднощі при тестуванні.

В роботі використовувалась бібліотека aiohttp мови програмування Python, яка дозволяла асинхронно надсилати HTTP-запити до стандартного ендпоінту Chat Completion (/openai/v1/chat/completions), а також бібліотека groq для аутентифікації за допомогою персонального API ключа.

Тестування адаптованої до української мови ukr-mistral здійснювалось за допомогою бібліотеки Hugging Face transformers. Модель і відповідний токенізатор були завантажені з Hugging Face Hub і розгорнуті локально в середовищі Google Colab на графічному процесорі NVIDIA T4.

З метою забезпечення рівних умов для тестування, всім моделям було задано однакові параметри (Рисунку 2.2). Значення температури 0.6 робить відповіді моделей більш детермінованими, а параметр top_p визначає випадковість згенеровано відповіді. Високу максимальну кількість токенів встановлено для забезпечення коректної роботи при CoT підході. При інших методах задання запиту (наприклад, Zero-Shot Prompting) таке обмеження дозволяє відстежити наскільки модель слідує інструкціям запиту.

```
payload = {
        "model": model["model_id"],
        "messages": [
            {"role": "system", "content": system_prompt},
            {"role": "user", "content": user_prompt}
        ],
        "temperature": 0.6,
        "max_completion_tokens": 4096,
        "top_p": 0.95,
}
```

Рисунок 2.2 Параметри великих мовних моделей

Основним методом у цій роботі став Zero-Shot Prompting. Кожен запит формувався зі списку, який складається з системного повідомлення (role: “system”) з загальними інструкціями для моделі та повідомлення користувача (role: “user”), тобто, власне, тексту новини. Варто зазначити, що ukr-mistral приймає повідомлення лише з роллю “user”, тому системне повідомлення просто додавалось на початок тексту. Приклад наведено на Рисунку 2.3.

```
system_prompt = (
    "Віднеси наступний текст до однієї з 5 категорій: політика, спорт, новини, технології, бізнес.
     Відповідь подай одним словом без розділових знаків."
 )
```

Рисунок 2.3. Системне повідомлення для Zero-Shot Prompting

Відповіді всіх моделей після парсингу було збережено в окремий csv файл та перетворено на числові мітки для обчислення метрик.

Оскільки базовий Zero-Shot Prompting показав незадовільні результати, було випробувано ще три додаткові підходи:

- повернення додаткової категорії uncertain, якщо модель не впевнена в своєму висновку;
- Few-Shot Prompting;
- Chain-of-Thought (CoT).

Ці методи були застосовані до вибірки із 170-ти записів, які були неправильно класифіковані усіма моделями на попередньому етапі. Такий підхід одразу оцінити приріст точності для кожного з підходів. Приклади системних повідомлень, які надавались моделям, показані у Додатку 1.

2.3.2 Налаштування та реалізація fine-tuning для трансформерів

Оцінка ефективності трансформених моделей в цьому дослідженні є можливою лише після специфічного додаткового навчання на задачі класифікації українських новин. Як і у випадку промптингу, для забезпечення рівних умов, всі навчальні аргументи є однаковими та всі параметри моделей беруть участь в навчанні.

Весь процес відбувався в середовищі Google Colab (з використання графічного прискорювача NVIDIA T4). Завантаження моделей (Рисунок 2.4) та сам процес fine-tuning реалізовано з використанням бібліотеки transformers.

```
model = AutoModelForSequenceClassification.from_pretrained(
    MODEL_NAME,
    num_labels=num_labels,
    label2id=label2id,
    id2label=id2label
)
```

Рисунок 2.4. Доступ до pre-trained моделі за допомогою класу AutoModelForSequenceClassification

Перед тренуванням навчальна та валідаційна вибірки були токенізовані з вказанням параметру truncation=True, що гарантує обрізання вхідних послідовностей в разі, якщо вони перевищують ліміт моделі. Для оптимізованої роботи з батчами використовувався DataCollatorWithPadding.

Всі встановлені гіперпараметри проілюстровано на Рисунку 2.5. Загалом, навчання тривало 3 епохи, після чого найкраща (за метрикою f1-score) модель зберігалась як фінальна версія. Швидкість навчання (learning rate), яка

контролює оновлення ваг моделі, встановлена на рівні 3e-5. З метою відтворюваності результатів, було встановлено значення seed = 42.

```
BATCH_SIZE = 16

training_args = TrainingArguments(
    output_dir=OUTPUT_DIR_MODEL,
    logging_dir=LOGGING_DIR_MODEL,
    num_train_epochs=3,
    per_device_train_batch_size=BATCH_SIZE,
    per_device_eval_batch_size=BATCH_SIZE * 2,
    warmup_steps=500,
    weight_decay=0.01,
    learning_rate=3e-5,

    eval_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
    metric_for_best_model="f1_weighted",
    save_total_limit=2,

    fp16=True,

    logging_strategy="steps",
    logging_steps=100,
    report_to="tensorboard",
    seed=42
)
```

Рисунок 2.5. Параметри та налаштування процесу навчання

Після завершення процесу навчання, кожну з моделей було збережено на Google Диск, для можливості її подальшого використання.

Передбачення категорій для тестової вибірки відбувалось за допомогою метода predict (). Оскільки отримані результати одразу мають числовий формат, їх, без попередньої обробки, було збережено у csv-файл для подальшого розрахунку метрик.

2.4 Метрики оцінки моделей

2.4.1 Загальний огляд на задачу оцінки моделей

Оцінка якості моделей класифікації є фундаментальним завданням в науці про дані. Для цього використовуються метрики класифікації (classification metrics), що надають кількісну оцінку ефективності моделі та дозволяють

переконатися, що остання є надійною та відповідає потребам і вимогам конкретної задачі [6].

Використання різноманітного набору метрик є критично важливим в контексті цього дослідження, оскільки кожна з них оцінює модель з певного ракурсу, дозволяючи врахувати різні типи помилок (наприклад, хибнопозитивні (FP) та хибнонегативні (FN) спрацювання) і на основі цього робити висновки про якість її роботи. Орієнтація лише на один показник в оцінці, наприклад точність (Accuracy), може призвести до неправильних висновків стосовно ефективності моделі, особливо, в разі роботи з незбалансованими даними, де один або кілька класів зустрічаються частіше ніж інші. В такому випадку модель прогнозуватиме найчастіший клас і матиме високий показник точності, проте буде непридатною для розпізнавання інших класів і, відповідно, для використання на реальних даних [7]. Правильна інтерпретація результатів за різними метриками надає можливість обґрунтовано приймати рішення стосовно вибору моделі, її подальшого налаштування та оцінити її практичну цінність.

Для подальшого розуміння варто зазначити, якими спеціальними мітками (комбінаціями істинних та передбачених класів) оперують у обчисленні метрик:

- True Positives (TP) - кількість екземплярів, які належать до певного класу X і були коректно передбачені.
- True Negatives (TN) - кількість екземплярів, які не належать до класу X і були передбачені, як ті, що до нього не належать.
- False Positives (FP) - кількість екземплярів, які не належать до X, проте були передбачені, як екземпляри класу X.
- False Negatives (FN) - кількість екземплярів, які належать до класу X, проте помилково були передбачені як екземпляри іншого класу.

2.4.2 Огляд використаних метрик

У цьому дослідженні для оцінки та порівняння протестованих моделей було обрано такий набір метрик: Accuracy, Precision (Macro and Weighted),

Recall (Macro and Weighted), F1-Score (Macro and Weighted), Matthews Correlation Coefficient (MCC) та Confusion Matrix.

Варто зазначити, що оскільки роботі розглядається задача багатокласової класифікації (5 категорій новин), то для агрегованої оцінки таких метрик як Precision, Recall та F1-score потрібно застосовувати стратегії усереднення показників [9]. В цьому дослідженні було використано два стандартні підходи:

- Macro Average - розраховує відповідні метрики для кожного класу окремо та повертає середнє арифметичне цих значень. За такого підходу кожен клас, представлений у тестовій вибірці, має однакову вагу в фінальній оцінці, незалежно від кількості представлених записів. Таким чином метрика стає дуже чутливою до якості роботи моделі на незбалансованих даних.
- Weighted Average - також розраховує метрики для кожного класу окремо, проте при усередненні враховується кількість істинних екземплярів кожного класу [10]. Відповідно, класи, що мають більшу кількість входжень у тестову вибірку, мають більший вплив на кінцевий результат оцінки. Таким чином відображається загальна продуктивність моделі на всьому наборі даних, враховуючи при цьому дисбаланс класів.

Далі пропоную розглянути детальніше кожну з метрик:

1. Accuracy (Точність) - є однією з фундаментальних метрик для задач класифікації, яка визначає ступінь правильності прогнозів моделі. Загалом, точність - це кількість правильних прогнозів, поділена на загальну кількість прогнозів. Формула обрахунків accuracy для багатокласової класифікації виглядає так, як показано на Рисунку 2.5

$$Accuracy(y_i, Z_i)) = \frac{1}{N} \sum_{i=1}^{N} [[y_i = Z_i]]$$

Рисунок 2.5. Формула точності для багатокласової класифікації

де N - це кількість зразків; [[...]] - це дужка Айверсона, яка повертає 1, коли вираз у дужці істинний та 0 в протилежному випадку; yi та Zi це відповідно правильна та прогнозована мітка [8].

Як вже було вказано раніше, метрика показує загальний показник продуктивності моделі, тож для коректної інтерпретації результатів роботи моделі потрібно використовувати і інші метрики, які є більш чутливими до дисбалансу класів.

2. Precision (Macro and Weighted) - метрика, яка показує точність передбачення, визначаючи скільки з усіх позитивних відповідей моделі, виявились правильними.

   Для дослідження використовуються macro- та weighted- методи обчислення precision, формули яких показано на Рисунку 2.6. Умовним позначенням ni та n відповідають кількості екземплярів досліджуваного класу та загальній кількості екземплярів у вибірці.

$$Precision_{macro} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FP_i} \qquad Precision_{weighted} = \sum_{i=1}^{N}\frac{n_i}{n}\frac{TP_i}{TP_i + FP_i}$$

Рисунок 2.6. Формули розрахунку macro- та weighted- Precision

   Precision є корисною в разі, якщо є потреба виявити та мінімізувати частку FP результатів. Низькі результати підрахунку метрики (<0.5) можуть вказувати на погано налаштовані гіпер параметри моделі. Значення precision, що прямує до одиниці, означатиме, що модель майже не пропускає істинно позитивних результатів (TP) і здатна добре класифікувати правильне і неправильне маркування новин [11].

3. Recall (Macro and Weighted) вимірює здатність моделі ідентифікувати всі позитивні випадки (TP) з всіх фактично позитивних випадків (TP+FN) [12]. Вона показує здатність моделі уникати хибно негативних (FN) спрацювань, коли модель відносить новину одного класу до іншого. Математично Recall для мультикласової класифікації розраховується за формулою вказаною на Рисунку 2.7

$$Recall_{macro} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FN_i} \qquad Recall_{weighted} = \sum_{i=1}^{N}\frac{n_i}{n}\frac{TP_i}{TP_i + FN_i}$$

Рисунок 2.7. Формули розрахунку macro- та weighted- Recall

Високе значення цієї метрики означає, що модель може ефективно виявляти екземпляри цільового класу. Натомість низьке значення може свідчити про незбалансовану навчальну вибірку та не оптимальні параметри моделі [11].

4. F1-Score (Macro and Weighted) - це ключова метрика в оцінці якості класифікації, особливо при наявності перекосу даних. Вона представляє собою гармонійне середнє між Precision та Recall (Рисунок .2.8) і дозволяє збалансувати ці дві метрики.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Рисунок 2.8. Формула обчислення F1-Score

Високе значення F1 вказує на високу якість роботи моделі завдяки мінімізованим показникам FP та FN. В свою чергу, низький F1 вказує на те, що існує проблема, проте невідомо в чому вона полягає, саме тому необхідний окремий аналіз Precision та Recall [11]. Як і для попередніх метрик в даному дослідженні обчислюються macro- і weighted-узагальнення, задля забезпечення повноти аналізу моделей класифікації.

5. Confusion Matrix (Матриця плутанини) забезпечує детальний аналіз продуктивності моделей, оскільки являє собою табличне представлення результатів класифікації, порівнюючи істинні мітки класів з тими, що були передбачені моделлю [11]. Приклад вигляду матриці для задачі цього дослідження показано на Рисунку 2.9. Структурно, рядки матриці відповідають істинним класам, а стовпці – передбаченим. Тобто, кожна комірка (i, j) матриці містить кількість екземплярів класу i, яким було призначено клас j. Таким чином на головній діагоналі відображається

кількість правильно класифікованих екземплярів для кожного з класів (TP). Елементи поза головною діагоналлю репрезентують всі випадки помилкової класифікації, коли модель сплутала один клас з іншим. Матриця плутанини дозволяє значно глибше зрозуміти модель: виявити, які класи найлегше розпізнаються, а які плутаються, визначити слабкі місця моделі та проаналізувати характер помилок.

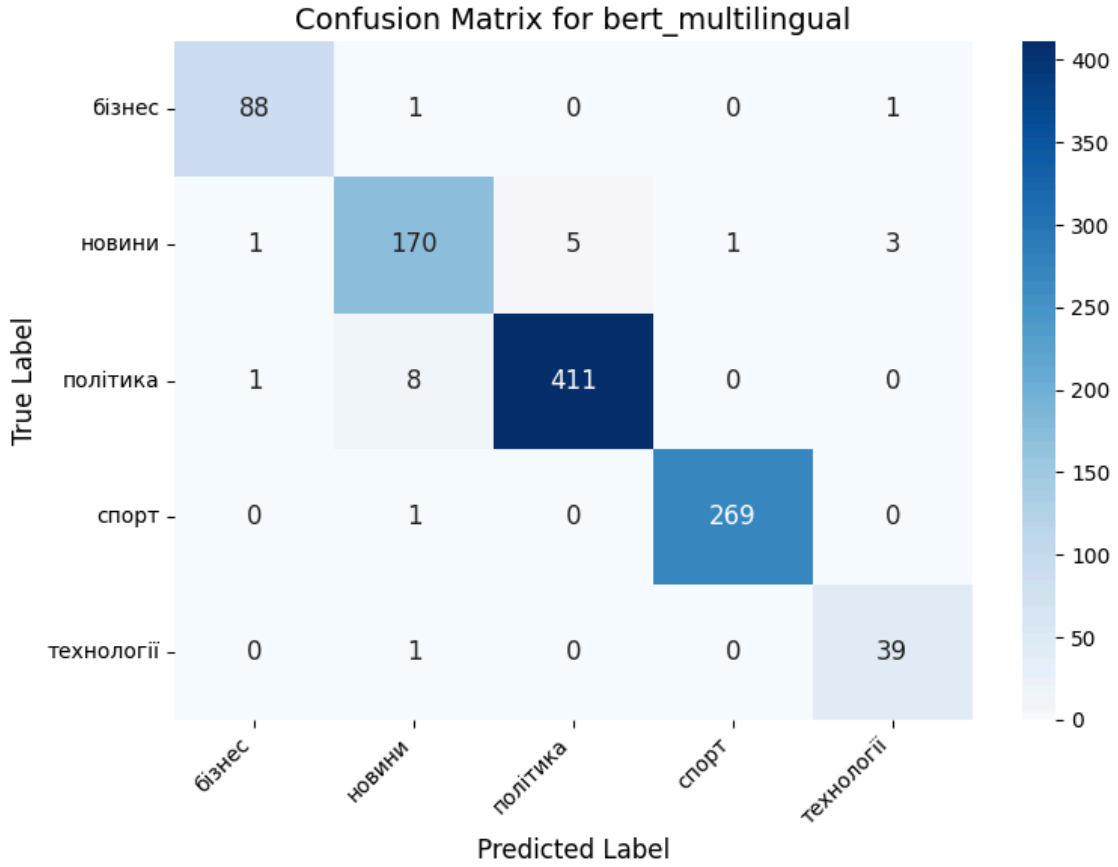


Рисунок 2.9. Приклад матриці плутанини

6. Matthews Correlation Coefficient (MCC) - це метрика, що враховує всі значення з матриці плутанини (TP, TN, FP, FN) і забезпечує комплексну числову оцінку моделі. Основною перевагою MCC є те, що його результати є інформативними навіть за наявності великої різниці розмірів класів у вибірці. Загалом, MCC - це коефіцієнт кореляції між істинними та передбаченими класами. Його значення знаходиться в діапазоні від -1 до 1. Чим ближче значення коефіцієнта до 1, тим вища якість роботи моделі. Результат 0 вказує на те, що загалом передбачення моделі близькі

до випадкових, а -1, в свою чергу, означає повну невідповідність та зворотне передбачення [13]. У контексті цього дослідження MCC використовується як стійкий до незбалансованості даних узагальнюючий показник ефективності моделей, що добре доповнює аналіз на основі F1-score.

2.4.2 Засоби обчислення метрик та візуалізації результатів

Для обчислення описаних вище метрик та візуалізації результатів було використано низку бібліотек мови програмування Python, які спеціалізовані для аналізу даних.

Для безпосереднього підрахунку метрик та побудови матриці плутанини використовувались відповідні функції (Рисунок 2.10) з модуля metrics бібліотеки scikit-learn [14]. Ця бібліотека є галузевим стандартом для задач машинного навчання і забезпечує коректність та стандартизованість обчислень.

```
from sklearn.metrics import (
    accuracy_score,
    f1_score,
    classification_report,
    confusion_matrix,
    matthews_corrcoef
)
```

Рисунок 2.10. Список функцій розрахунку метрик з модуля metrics

Для візуального представлення результатів обчислення метрик було використано бібліотеки matplotlib [15], яка є фундаментальною для створення графіків у Python та seaborn [16], що дозволяє створювати більш складні статистичні візуалізації, зокрема heatmaps, які добре підходять для матриць плутанини.

РОЗДІЛ 3. РЕЗУЛЬТАТИ ТА АНАЛІЗ

3.1 Оцінка якості класифікації

3.1.1 Загальні показники ефективності

Для кількісного порівняння ефективності семи протестованих мовних моделей на задачі класифікації новин українською мовою було розраховано низку метрик. Зведені результати кожної з них наведені у Таблиці 3.1.

Результати чітко демонструють перевагу трансформерних моделей, які були донавчені навчання, порівняно з великими мовними моделями. Очікувано, найвищі показники F1 та MCC має модель ukr-roberta, яка спеціалізована для виконання задач українською мовою.

Model name	Accuracy	F1 Weighted	F1 Macro	MCC	Precision Macro	Recall Macro	Precision Weighted	Recall Weighted
ukr_roberta	0.992	0.992	0.985	0.989	0.983	0.987	0.992	0.992
xlm_roberta	0.984	0.984	0.983	0.978	0.979	0.987	0.984	0.984
bert	0.977	0.977	0.968	0.968	0.962	0.974	0.977	0.977
mistral_ukr	0.805	0.808	0.712	0.725	0.756	0.701	0.827	0.805
qwen	0.79	0.797	0.729	0.712	0.716	0.749	0.808	0.79
mistral_groq	0.786	0.796	0.722	0.710	0.728	0.738	0.820	0.786
llama	0.774	0.784	0.712	0.703	0.707	0.736	0.817	0.774

Таблиця 3.1 Зведені результати оцінки моделей

Дуже близькі, проте все ж нижчі результати мають xlm-roberta та bert, порядок яких в таблиці теж є очікуваним, через покращену архітектуру першої. Результати чотирьох LLM, які тестувались за допомогою Zero-Shot Prompting виявились значно нижчими за показники інших моделей. Звісно, очікувано, що найкращий результат серед них матиме ukr-mistral, проте незадовільні результати потужної llama, навіть в порівнянні з орієнтованими арабські мови чи написання коду моделями, змушує сумніватись в ефективності zero-shot підходу та здатності LLaMA до розуміння української мови.

Для того, аби зрозуміти наскільки дисбаланс класів в навчальній і тестовій вибірці впливає на результат моделей, розглянемо різницю між показниками weighted- та macro- F1-Score. Для донавчених моделей вона є відносно невеликою, особливо для xml-roberta (0.001). Це свідчить про стабільну роботу моделей, навіть з урахуванням незбалансованих даних. В той час як для всіх LLM спостерігається великий розрив (до 0.096 одиниць). Причина криється в значно нижчому показнику F1 Macro, що може свідчити про труднощі моделі з класифікацією рідкісних класів (ймовірно “технології” та “бізнес”). Тенденції, виявлені за допомогою F1-Score підтверджує і MCC, який показує чіткий стрибок показників на межі LLM та трансформерів в таблиці.

3.1.2 Аналіз помилок класифікації

Важливо не лише кількісно оцінити якість виконання завдання, а й проаналізувати конкретні помилки. Це дасть розуміння про класи, які модель сплутала або не змогла розпізнати. Найкраще для такого аналізу підходять матриці плутанини. На Рисунку 3.1 та Рисунку 3.2 зображено confusion matrix для llama та ukr_mistral, які є достатньо репрезентативними для аналізу. Решта ілюстрації розміщені в Додатку 2.

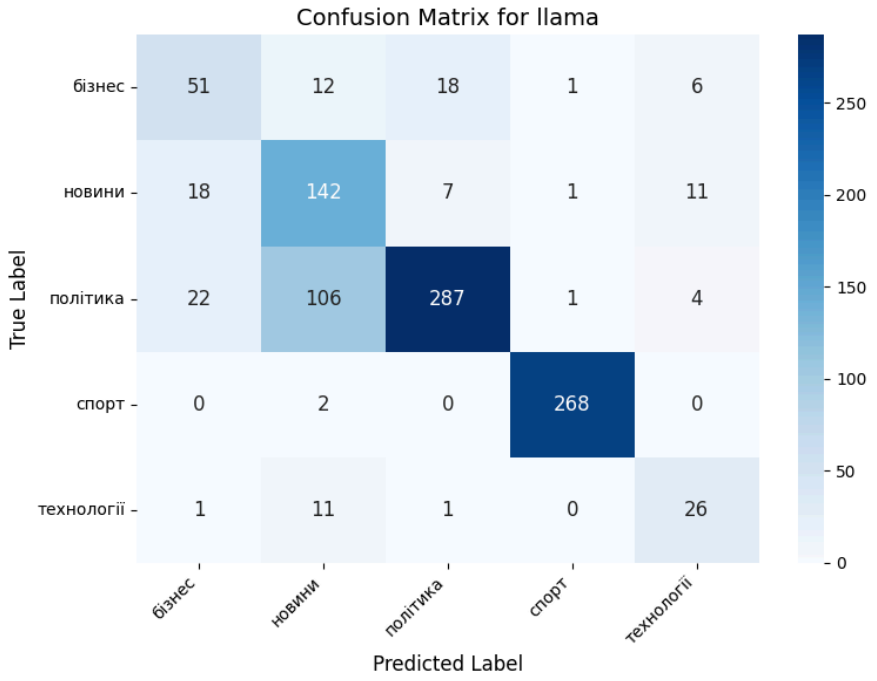


Рисунок 3.1. Матриця плутанини для LLaMA

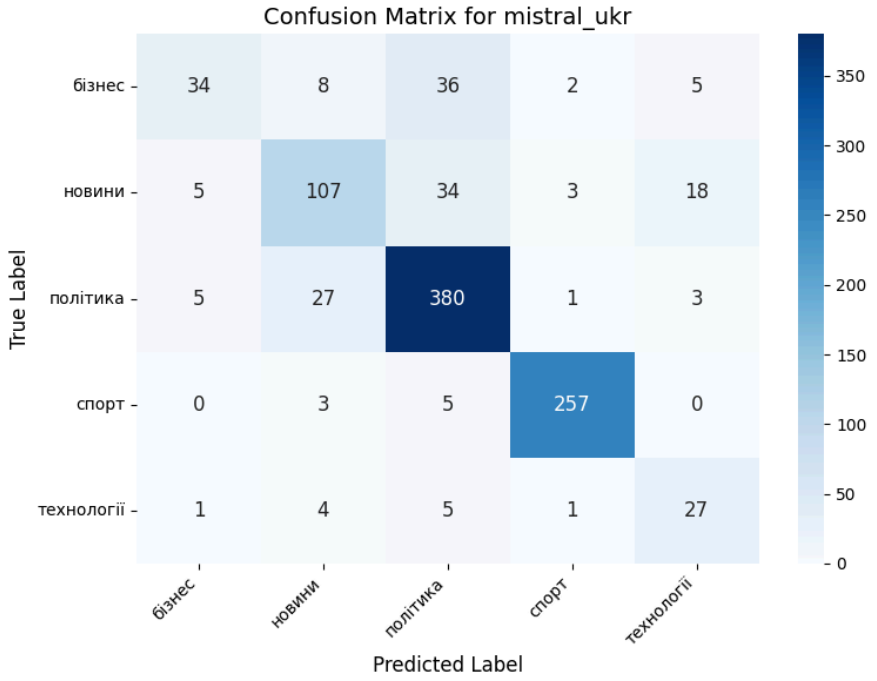


Рисунок 3.2. Матриця плутанини для ukr-mistral

Після детального аналізу матриць видно, що всі моделі рідко помиляються у класифікації спортивних новин. Також стало зрозуміло, що базові багатомовні моделі llama, qwen та mistral допускали однакові типи помилок в схожих пропорціях. Наприклад, “технології” плутались з “новинами” (переважно лише з ними), а “політику” часто відносили до “бізнесу” або “новин”. Це може свідчити про нестачу контексту. Що стосується ukr-mistral, ситуацію прояснює детальніший аналіз відповідей. Як видно з Рисунку 3.3, українізована LLM найчастіше призначала новинам невалідні категорії. Серед найпоширеніших – “економіка”, “культура”, “наука” тощо. Ukr-mistral навпаки, порівняно з багатомовними LLM, демонструвала надмірну обізнаність і намагалась присвоїти максимально точну категорію та пояснити свої міркування, навіть якщо це виходило за межі задачі дотримання інструкцій і знижувало сувору оцінку точності. Тож, завдяки аналізу помилок вдалося виявити проблемні і сильні сторони моделей, зокрема ukr-mistral. Отже, завдяки аналізу помилок вдалося виявити як проблемні, так і сильні сторони моделей, зокрема ukr-mistral.

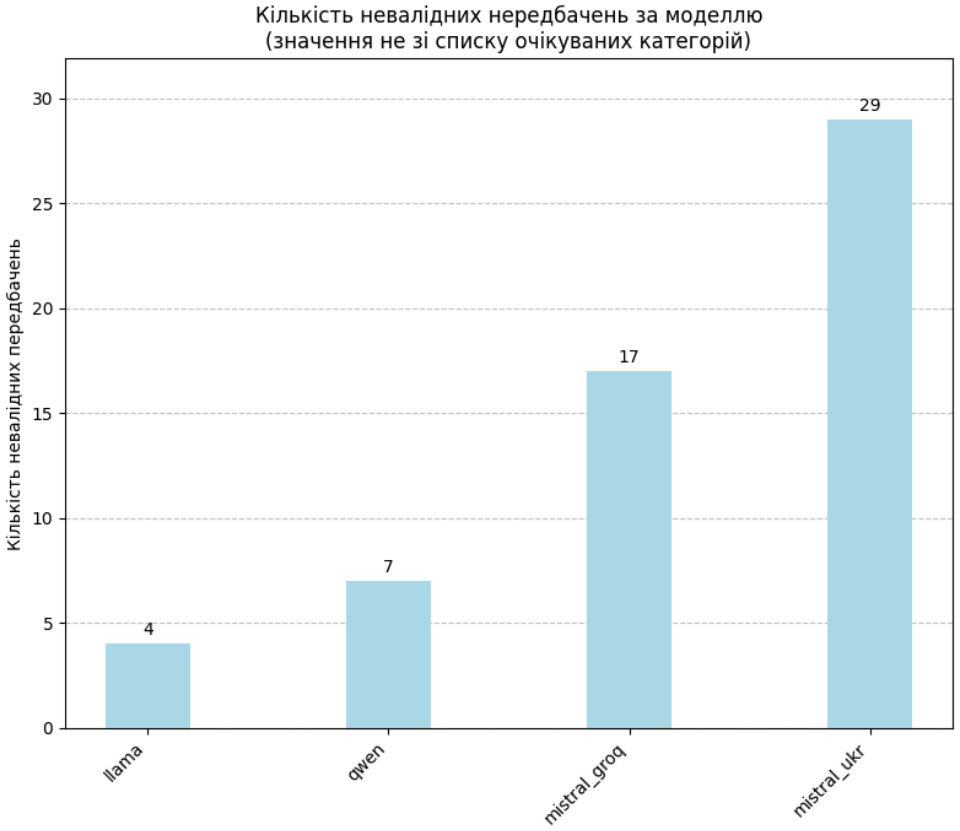


Рисунок 3.3 Кількість відповідей LLM, що не відповідають категоріям

3.1.3 Аналіз результатів prompting LLM

Як зазначалось в методології, для детальнішого вивчення можливостей LLM було проведено додаткові тестування з використанням більш просунутих підходів промптингу, результати яких наведено в Таблицях 3.2, 3.3 та 3.4. Варто зазначити, що додаткові експерименти проводились на складній для моделей вибірці (всі 4 моделі припустились помилок на цих прикладах при звичайному Zero-Shot Prompting).

Для встановлення базового рівня (baseline) для порівняння просунутих метрик Few-Shot Prompting і CoT використовується підхід з додаванням uncertain, в разі якщо модель не впевнена у своїй відповіді (Таблиця 3.2).

Model Name	Accuracy	F1 Weighted	F1 Macro	MCC	Precision Macro	Recall Macro	Precision Weighted	Recall Weighted
mistral_ukr	0.378	0.358	0.254	0.156	0.225	0.353	0.347	0.378
mistral_groq	0.371	0.325	0.208	0.083	0.180	0.276	0.294	0.371
llama	0.312	0.321	0.208	0.025	0.208	0.215	0.336	0.312

qwen	0.276	0.273	0.177	-0.021	0.169	0.196	0.275	0.276

Таблиця 3.2. Результати обчислення метрик. Zero-Shot Prompting + uncertain

Аналіз результатів Few-Shot Prompting показав, що такий підхід не призвів до покращення ефективності моделей. Вони залишились в тому самому ранжуванні що і при роботі з загальною тестовою вибіркою. До того ж, у всіх моделей (окрім ukr-mistral) MCC опустився нижче нуля, що свідчить про те, що відповіді моделей більше рандомні, аніж осмислені.

Model Name	Accuracy	F1 Weighted	F1 Macro	MCC	Precision Macro	Recall Macro	Precision Weighted	Recall Weighted
mistral_ukr	0.342	0.363	0.214	0.103	0.224	0.219	0.405	0.342
qwen	0.259	0.265	0.193	-0.040	0.193	0.202	0.285	0.259
mistral_groq	0.259	0.264	0.164	-0.057	0.164	0.174	0.273	0.259
llama	0.247	0.244	0.199	-0.016	0.205	0.228	0.293	0.247

Таблиця 3.3. Результати обчислення метрик. Few-Shot Prompting

Застосування Chain-of-Thought Prompting (Таблиця 3.4) продемонструвало неоднозначну динаміку. З одного боку, для обох моделей родини Mistral (mistral_ukr та mistral_groq) спостерігалося невелике покращення показників за основними метриками (F1 Weighted, MCC). Для моделі qwen результати залишилися та такому ж рівні, але особливо показовою є зміна показників у llama – її MCC став глибоко від'ємним. Це підтверджує попередні здогадки про її обмежені можливості для задачі класифікації українських текстів за допомогою промптингу.

Model Name	Accuracy	F1 Weighted	F1 Macro	MCC	Precision Macro	Recall Macro	Precision Weighted	Recall Weighted
mistral_ukr	0.397	0.386	0.237	0.159	0.249	0.259	0.407	0.397
mistral_groq	0.335	0.354	0.232	0.068	0.239	0.236	0.389	0.335
qwen	0.253	0.258	0.194	-0.044	0.194	0.201	0.277	0.253
llama	0.147	0.150	0.108	-0.182	0.121	0.115	0.189	0.147

Таблиця 3.3. Результати обчислення метрик. Chain-of-Thought

3.2 Оцінка ефективності навчання та швидкодії

Окрім аналізу якості класифікації також важливо враховувати технічні властивості моделей. Такі характеристики як кількість параметрів, обчислювальна складність (FLOPs) та швидкодія інференсу можуть надати додаткову вагу при виборі моделей для реальних практичних задач.

Проведений аналіз показав що всі три трансформені моделі мають оцінку FLOPs для інференсу приблизно 66 GFLOPs. Теоретично, це могло б вказувати на однакове обчислювальне навантаження. Проте вимірювання реальної швидкодії на GPU T4 виявили помітні відмінності.

За показником затримки найшвидшою зі всіх виявилась ukr-roberta (14.65 мс), яка також має найменшу кількість параметрів – приблизно 126 млн. Модель bert (~177 млн параметрів) показала середню затримку 18.44 мс. В той час як найбільша xml-roberta (понад 278 млн) має 17.04 мс.

Аналогічна тенденція спостерігається і для пропускної здатності (throughput). ukr-roberta теж посідає перше місце – 36 зразків/сек. У xml-roberta та bert пропускна здатність дорівнює 34.7 та 35.0 зразків/сек відповідно.

Тож, можна зробити висновок, що ukr-roberta окрім якісної класифікації забезпечує ще й найкращу ефективність.

РОЗДІЛ 4. ВИСНОВКИ

У даній курсовій роботі було проведено порівняльне дослідження двох класів сучасних мовних моделей – трансформерів з відкритим кодом (bert-base-multilingual-cased, xlm-roberta-base та ukr-roberta-base) та великих мовних моделей (LLaMA-3.3-70b-versatile, Qwen-2.5-coder-32b, Mistral-SABA-24b та Mistral-7B-Instruct-Ukrainian). Головна мета полягає в дослідженні того, як архітекура, масштаб та специфікація моделей впливає на їх здатність до обробки текстів українською мовою. Оцінка ефективності моделей відбувалась за результатами тестування на задачі багатокласової класифікації новин українською мовою.

Результати дослідження показали значну перевагу донавчених трансформерів у порівнянні з LLM, які використовують загальні знання для виконання завдання класифікації.

Дослідження також підтвердило важливість мовної спеціалізації моделей. Найкращі результати серед усіх моделей показала адаптована до використання української ukr-roberta-base (F1 Weighted ~0.992). Аналогічно, серед LLM, що тестувалися через prompting, найкраще себе показала донавчена модель Mistral-7B-Instruct-Ukrainian (F1 Weighted ~0.808). Це свідчить про те, що моделі, перед-навчені або донавчені безпосередньо на українських даних, мають перевагу при обробці українських текстів порівняно з багатомовними аналогами.

Аналіз роботи LLM за допомогою промптингу допоміг виявити низку проблем, зокрема брак контексту, невідповідність інструкціям та чутливість до способу задання запитів. При намаганнях покращити результати роботи через просунуті методи промптингу, було отримано неоднозначні результати, які вкотре підтвердили необхідність специфікації під конкретну задачу. Також, неочікуваним стало те, що не було виявлено чіткої залежності якості класифікації від розміру LLM.

Оцінка ефективності донавчений моделей показала, що найменша за кількістю параметрів ukr-roberta виявилась найточнішою і найшвидшою за показниками затримки та пропускної здатності інференсу.

Якщо порівнювати підходи до розгортання моделей, то використання Groq API надало доступ до найсучасніших LLM великого масштабу без потреби у значних обчислювальних ресурсах, проте якість класифікації все ж виявилася нижчою. Локальне донавчання моделей потребувало ресурсів, проте забезпечило високу точність виконання, тож вибір оптимального підходу до використання моделей залежить від вимог до точності та від наявності обчислювальних ресурсів.

Напрямки подальших досліджень можуть включати в себе тестування та аналіз роботи моделей на інших задачах, що включають в себе глибоке розуміння української мови, проте не мають суворих інструкційних обмежень. Також одним з напрямків досліджень може бути fine-tuning великих мовних моделей.

Таким чином, дана курсова робота провела порівняльний аналіз сучасних підходів до класифікації українських текстів, продемонструвавши поточні переваги донавчання спеціалізованих трансформерних моделей та висвітливши особливості й обмеження використання великих мовних моделей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Caroline Eppright. What Is Natural Language Processing (NLP)? [Електронний ресурс] / Oracle – Режим доступу до ресурсу: https://www.oracle.com/ua/artificial-intelligence/what-is-natural-language-processing/
2. Jacob Murel, Eda Kavlakoglu. What is transfer learning? [Електронний ресурс] / IBM – Режим доступу до ресурсу: https://www.ibm.com/think/topics/transfer-learning
3. Dave Bergmann. What is fine-tuning? [Електронний ресурс] / IBM – Режим доступу до ресурсу: https://www.ibm.com/think/topics/fine-tuning
4. Anastasia Rashevskaya. Exploring the World of Natural Language Processing (NLP) [Електронний ресурс] / Litslink – Режим доступу до ресурсу: https://litslink.com/blog/a-complete-guide-to-natural-language-processing-nlp
5. AWS. What is text classification? [Електронний ресурс] / AWS – Режим доступу до ресурсу: https://aws.amazon.com/what-is/text-classification/
6. Alooba. Classification Metrics. [Електронний ресурс] / Alooba – Режим доступу до ресурсу: https://www.alooba.com/skills/concepts/data-science/classification-metrics/
7. Venkatesh Kulkarni. Classification metrics. [Електронний ресурс] / Medium – Режим доступу до ресурсу: https://medium.com/@vyankatesh.kulkarni20/classification-metrics-63c635852995
8. Noam Bressler. How to Check the Accuracy of Your Machine Learning Model. [Електронний ресурс] / DeepCheck – Режим доступу до ресурсу: https://www.deepchecks.com/how-to-check-the-accuracy-of-your-machine-learning-model/#:~:text=Accuracy%20score%20in%20machine%20learning%20is%20an%20evaluation%20metric%20that

9. Sefidian Academy. Understanding Micro, Macro, and Weighted Averages for Scikit-Learn metrics in multi-class classification with example. [Електронний ресурс] / SefidianAcademy – Режим доступу до ресурсу: https://iamirmasoud.com/2022/06/19/understanding-micro-macro-and-weighted-averages-for-scikit-learn-metrics-in-multi-class-classification-with-example/
10. Jino Mathew, Rohit Kshirsagar, Dzariff Z Abidin та ін. A comparison of machine learning methods to classify radioactive elements using prompt-gamma-ray neutron activation data. [Електронний ресурс] / ResearchGate – Режим доступу до ресурсу: https://www.researchgate.net/publication/368257911_A_comparison_of_machine_learning_methods_to_classify_radioactive_elements_using_prompt-gamma-ray_neutron_activation_data
11. Aayush Bajaj. Performance metrics in Machine Learning. [Електронний ресурс] / Neptune.ai – Режим доступу до ресурсу: https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide
12. Prashanth Srinivasan Sarkar. Evaluating NLP Models for Text Classification and Summarization Tasks in the Financial Landscape. [Електронний ресурс] / IndiumTech – Режим доступу до ресурсу: https://www.indium.tech/blog/evaluating-nlp-models-financial-analysis-part-2/
13. Scikit-Learn. Compute the Matthews correlation coefficient (MCC) [Електронний ресурс] / Scikit-Learn.org – Режим доступу до ресурсу: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews_corrcoef.html
14. Scikit-Learn. Sklearn.metrics. [Електронний ресурс] / Scikit-Learn.org – Режим доступу до ресурсу: https://scikit-learn.org/stable/api/sklearn.metrics.html#module-sklearn.metrics
15. Matplotlib. Matplotlib: Visualization with Python. [Електронний ресурс] / Matplotlib – Режим доступу до ресурсу: https://matplotlib.org/

16. Seaborn. Seaborn: statistical data visualization. [Електронний ресурс] / Seaborn – Режим доступу до ресурсу: https://seaborn.pydata.org/
17. B. Ivanyuk-Skulsky, A. Zaliznyi, O. Reshetar та ін. UA-News. [Електронний ресурс] / Fido.ai – Режим доступу до ресурсу: https://github.com/fido-ai/ua-datasets/blob/main/examples/ua_news.md
18. Adria Cabello. The Evolution of Language Models: A Journey Through Time. [Електронний ресурс] / Medium – Режим доступу до ресурсу: https://medium.com/@adria.cabello/the-evolution-of-language-models-a-journey-through-time-3179f72ae7eb
19. Mohammad Raeini. The Evolution of Language Models: From N-Grams to LLMs, and Beyond. [Електронний ресурс] / ResearchGate – Режим доступу до ресурсу: https://www.researchgate.net/publication/376057398_The_Evolution_of_Language_Models_From_N-Grams_to_LLMs_and_Beyond
20. GeeksForGeeks. Artificial Neural Networks and its Applications. [Електронний ресурс] / GeeksForGeeks – Режим доступу до ресурсу: https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/
21. GeeksForGeeks. Introduction to Recurrent Neural Networks. [Електронний ресурс] / GeeksForGeeks – Режим доступу до ресурсу: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/
22. Harsh Bansal.Text Generation Using LSTM. [Електронний ресурс] / Medium – Режим доступу до ресурсу: https://bansalh944.medium.com/text-generation-using-lstm-b6ced8629b03
23. Dave Bergmann, Cole Stryker. What is an attention mechanism? [Електронний ресурс] / IBM – Режим доступу до ресурсу: https://www.ibm.com/think/topics/attention-mechanism
24. Ashish Vaswani, Noam Shazeer та ін. Attention Is All You Need. [Електронний ресурс] – Режим доступу до ресурсу: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

25. GeeksForGeeks. Transformer Attention Mechanism in NLP. [Електронний ресурс] / GeeksForGeeks – Режим доступу до ресурсу: https://www.geeksforgeeks.org/transformer-attention-mechanism-in-nlp/
26. Rishabh Singh. Types of Transformer Model. [Електронний ресурс] / Medium – Режим доступу до ресурсу: https://medium.com/@RobuRishabh/types-of-transformer-model-1b52381fa719
27. Liam Daly Manocchio, Siamak Layeghy та ін. FlowTransformer: A Transformer Framework for Flow-based Network Intrusion Detection Systems. [Електронний ресурс] / ResearchGate – Режим доступу до ресурсу: https://www.researchgate.net/publication/370417681_FlowTransformer_A_Transformer_Framework_for_Flow-based_Network_Intrusion_Detection_Systems
28. Aditya Raj. Understanding the [CLS] Token in BERT: A Comprehensive Guide [Електронний ресурс] / Medium – Режим доступу до ресурсу: https://aditya007.medium.com/understanding-the-cls-token-in-bert-a-comprehensive-guide-a62b3b94a941
29. AWS. What is LLM (Large Language Model)? [Електронний ресурс] / AWS – Режим доступу до ресурсу: https://aws.amazon.com/what-is/large-language-model/?nc1=h_ls
30. PromptHub. In Context Learning Guide. [Електронний ресурс] / Prompt Hub – Режим доступу до ресурсу: http://prompthub.us/blog/in-context-learning-guide
31. Patric Farley. Prompt engineering techniques. [Електронний ресурс] / Microsoft – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/prompt-engineering?tabs=chat
32. Larysa Katerynych, Maksym Veres and Eduard Safarov. Transformer-based Model for Text Classification in Ukrainian. [Електронний ресурс] / CEUR

Workshop Proceedings – Режим доступу до ресурсу: https://ceur-ws.org/Vol-3179/Short_6.pdf

33. Li Xu, Jun Zeng and Shi Chen. Fine-tune XML-RoBERTa for Hate Speech Identification. [Електронний ресурс] / CEUR Workshop Proceedings – Режим доступу до ресурсу: https://ceur-ws.org/Vol-2826/T2-28.pdf
34. Youscan. Ukr-roberta-base. [Електронний ресурс] / Hugging Face – Режим доступу до ресурсу: https://huggingface.co/youscan/ukr-roberta-base
35. LLM Explorer. Mistral 7B Instruct Ukrainian by Sherlock Assistant. [Електронний ресурс] / LLM Explorer – Режим доступу до ресурсу: https://llm.extractum.io/model/SherlockAssistant%2FMistral-7B-Instruct-Ukrainian,oPBxuski629mwQ02bHbmG
36. Groq Cloud. Groq API Documentation. [Електронний ресурс] / Groq Cloud – Режим доступу до ресурсу: https://console.groq.com/docs/overview

ДОДАТОК 1

```
system_prompt_uncertain = (
        "Твоя задача - класифікувати текст новини. "
        "Відповідай ТІЛЬКИ ОДНИМ СЛОВОМ з цього списку: [бізнес, новини, політика, спорт, технології]."
        "Якщо ти зовсім не впевнений у жодній з цих категорій, відповідай словом 'uncertain'. "
        "Жодних інших слів, пояснень чи розділових знаків у відповіді бути не повинно."
)
```

Додаток 1.1. Системне повідомлення для підходу з додаванням uncertain

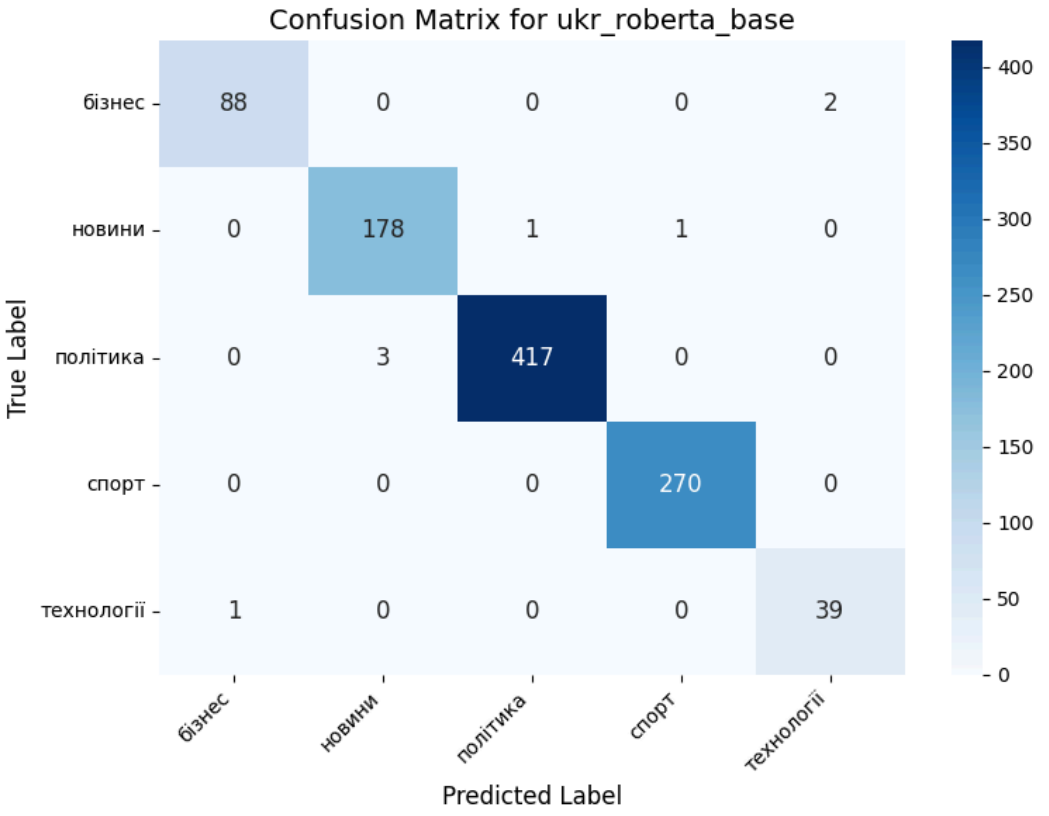
```
system_prompt_few_shot = (
    "Твоя задача - класифікувати текст новини. "
    "Відповідай ТІЛЬКИ ОДНИМ СЛОВОМ з цього списку: [бізнес, новини, політика, спорт, технології]. "
    "Жодних інших слів, пояснень чи розділових знаків у відповіді бути не повинно."
)

few_shot_examples = [
    {
      "text": "В Україні з’являться дві нові мережі - Miss Sixty і Pylones. Українська компанія
підписала договір про розвиток двох мереж з продажу одягу і подарунків на території країни.",
      "category": "бізнес"
    },
    {
      "text": "Новим прем'єром буде Гройсман. Народний депутат від БПП Вадим Денисенко заявив про те,
що після відставки Яценюка новим прем'єром стане Володимир Гройсман.",
      "category": "політика"
    },
    {
      "text": "Amazfit Neo — розумний годинник в класичному дизайні. Neo отримали 1,2-дюймовий
монохромний PK-дисплей без підтримки сенсорного введення і масивний пластиковий корпус.",
      "category": "технології"
    },
    {
      "text": "У четвер, 5 листопада, луганська «Зоря» в Запоріжжі зіграє матч третього туру групового
раунду Ліги Європи проти АЕКа.",
      "category": "спорт"
    },
    {
      "text": "Японія постраждала від потужного землетрусу магнітудою в 6,3 бала. Про це повідомляє
Європейський середземноморський сейсмологічний центр (EMSC).",
      "category": "новини"
    },
]
```

Додаток 1.2. Системне повідомлення та перелік прикладів для few-shot prompting

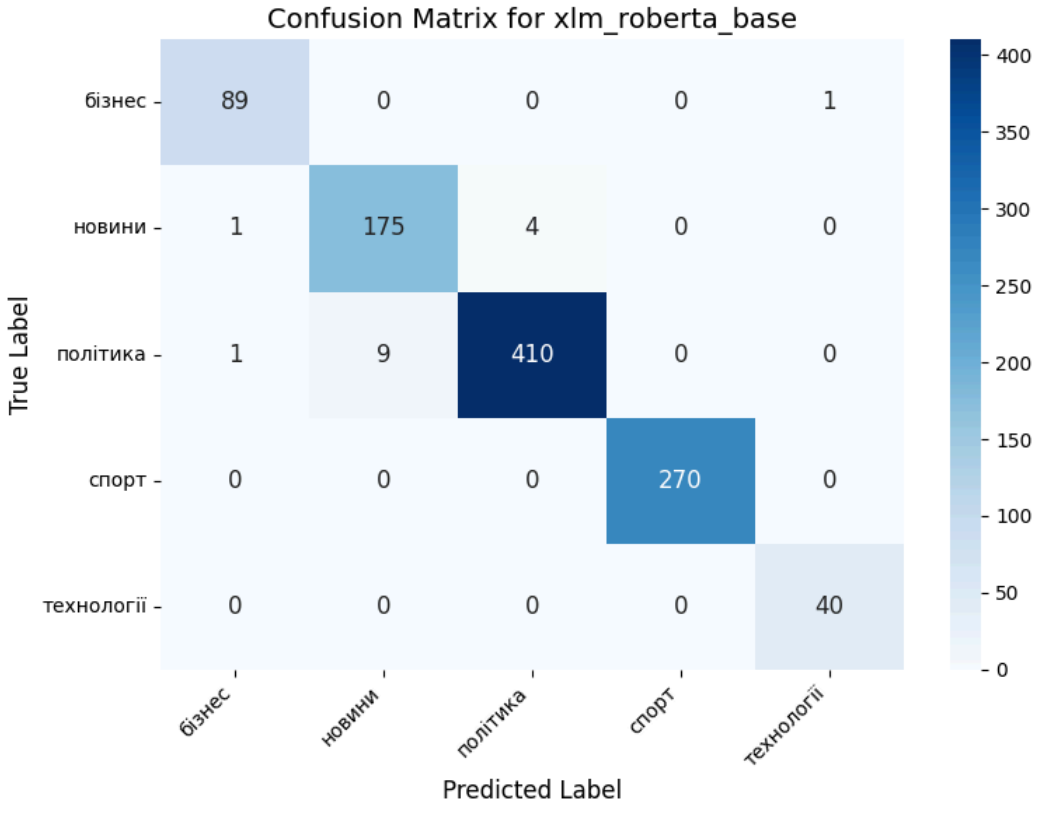
```
cot_user_instruction_template = """Виконай класифікацію тексту за такими кроками:
1.  Уважно прочитай текст.
2.  Визнач ключові слова, іменники та основні теми тексту.
3.  Проаналізуй, до якої з категорій [бізнес, новини, політика, спорт, технології] ці теми найбільше
стосуються. Коротко обґрунтуй.
4.  Зроби висновок щодо найбільш відповідної категорії.

Текст для класифікації:
{text}

Міркування крок за кроком:
1. Текст прочитано.
2. Ключові теми:"""
```

Додаток 1.3. Темплейт відповіді моделі при CoT

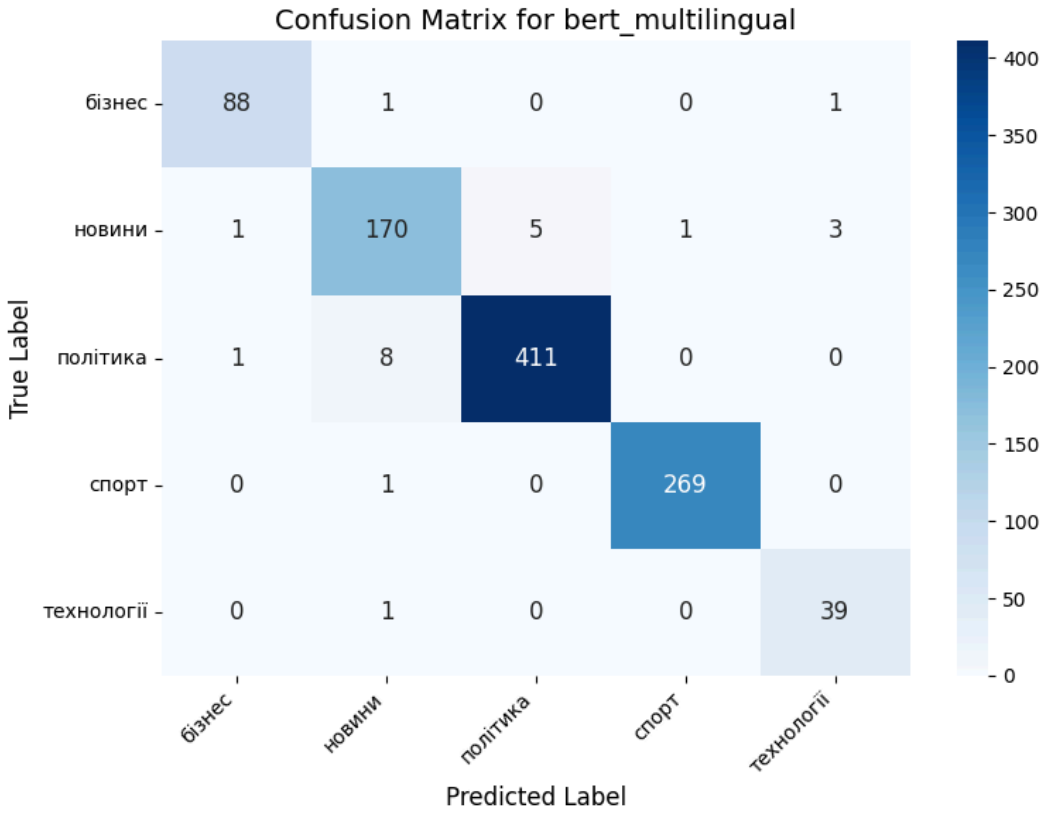
ДОДАТОК 2



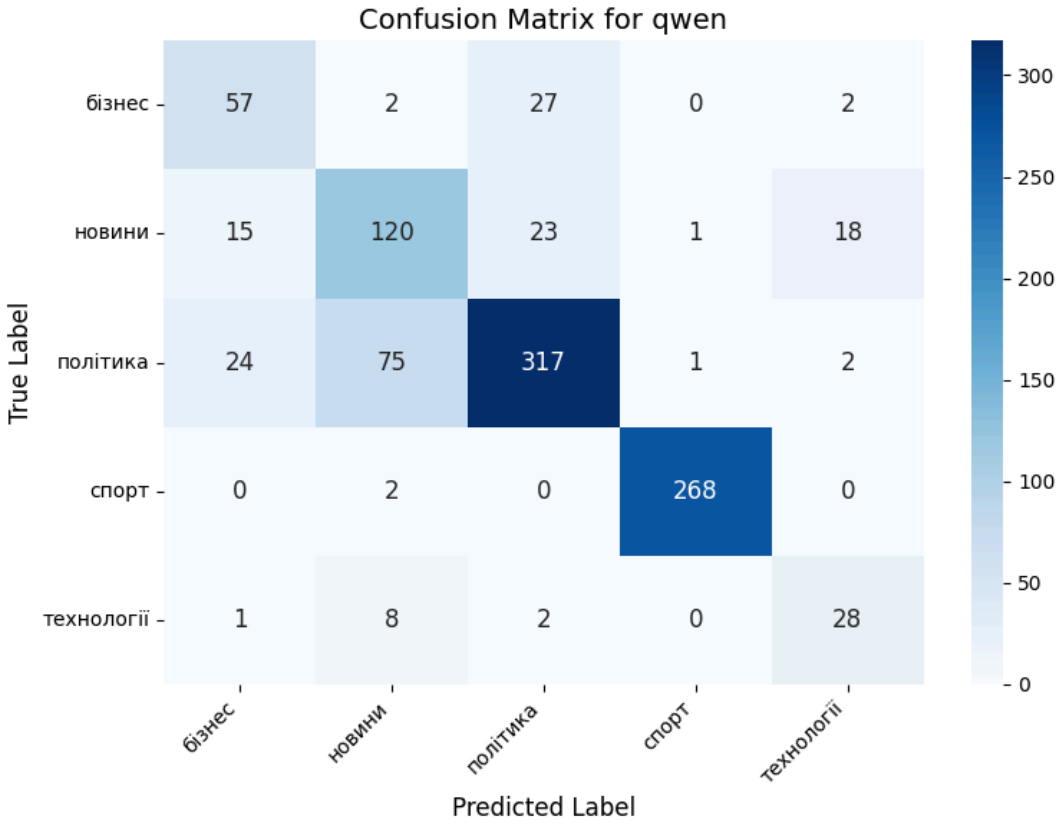
Додаток 2.1. Матриця плутанини для ukr-roberta



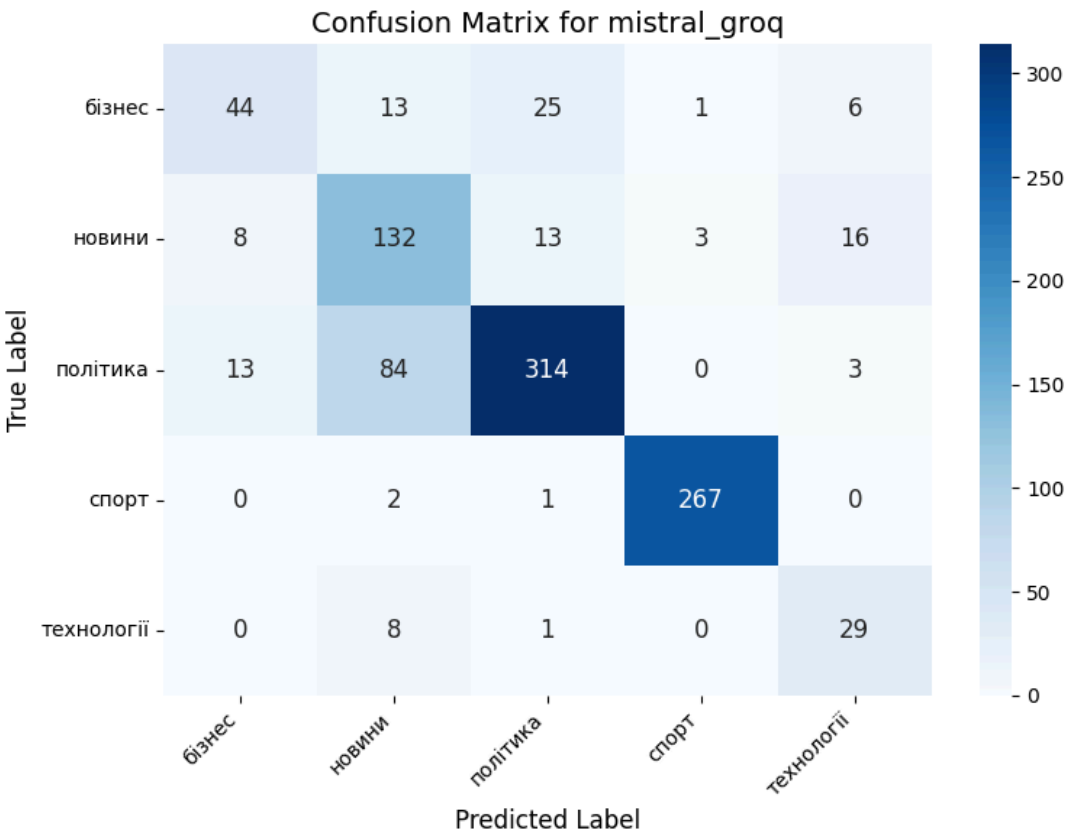
Додаток 2.2. Матриця плутанини для xml-roberta



Додаток 2.3. Матриця плутанини для bert



Додаток 2.4. Матриця плутанини для qwen



Додаток 2.1. Матриця плутанини для mistral