Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)
Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження мовних моделей для виявлення текстів, згенерованих штучним інтелектом
(тема)

Виконав:
здобувач 2 року навчання
групи ІПЗм-23-3
Олександра ТКАЧЕНКО
(власне ім'я, прізвище)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Інженерія програмного забезпечення
(повна назва освітньої програми)

Керівник проф. Кирило СМЕЛЯКОВ
(посада, власне ім'я, прізвище)

Допускається до захисту
Зав. кафедри _____ Кирило СМЕЛЯКОВ
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук

Кафедра програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)

Тип програми освітньо-наукова програма
(освітньо-професійна або освітньо-наукова)

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 20___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Ткаченко Олександрі Олексіївні
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження мовних моделей для виявлення текстів, згенерованих штучним інтелектом»

затверджена наказом університету від 15 квітня 2025 р. № 290Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 13 червня 2025 р.

3. Вихідні дані до роботи *відкрита інформація про моделі обробки природної мови для виявлення AI-текстів, засоби для виконання теоретичного аналізу та експериментального дослідження, мова програмування Python, бібліотеки NLP: transformers, fasttext, scikit-learn, середовище розробки Google Colab*

4. Перелік питань, що потрібно опрацювати у роботі *аналіз предметної галузі і постановка задачі, огляд та аналіз літературних джерел з дослідження, постановка задачі, теоретичне дослідження, практичне дослідження*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	16.04.2025	*виконано*
2	Аналіз предметної галузі і постановка задачі	17.04.2025	*виконано*
3	Огляд існуючих моделей	19.04.2025	*виконано*
4	Теоретичне дослідження	28.04.2025	*виконано*
5	Підготовка до апробації результатів дослідження. Публікація матеріалів	01.05.2025	*виконано*
6	Практичне дослідження	08.05.2025	*виконано*
7	Підготовка пояснювальної записки	20.05.2025	*виконано*
8	Підготовка презентації та доповіді	25.05.2025	*виконано*
9	Перевірка на плагіат	01.06.2025	*виконано*
10	Нормоконтроль	03.06.2025	*виконано*
11	Рецензування	04.06.2025	*виконано*
12	Попередній захист	08.06.2025	*виконано*
13	Занесення диплома в електронний архів	09.06.2025	*виконано*
14	Допуск до захисту у зав. кафедри	10.06.2025	*виконано*

Дата видачі завдання 16 квітня 2025р.

Здобувач _____
(підпис)

Керівник роботи _____ проф. Кирило СМЕЛЯКОВ
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ / ABSTRACT

Звіт: 71 с., 4 рис., 13 табл., 21 джерело.

КЛАСИФІКАЦІЯ, КОНТЕКСТ, МАШИННЕ НАВЧАННЯ, МОВНІ МОДЕЛІ, РОЗПІЗНАВАННЯ ТЕКСТУ, ТРАНСФОРМЕРИ, ШТУЧНИЙ ІНТЕЛЕКТ, AI, BERT, MISTRAL 7B.

Об’єктом дослідження є мовні моделі, їх ефективність та можливості застосування для розпізнавання текстів, створених штучним інтелектом.

Метою роботи є аналіз сучасних мовних моделей для ідентифікації текстів, згенерованих ШІ, та оцінка їхньої ефективності за ключовими критеріями.

Методи дослідження включають класифікацію мовних моделей за типами, побудову векторного опису моделей за критеріями, нормування оцінок і застосування згортки для виявлення оптимальних моделей. Обрані моделі пройшли подальше тестування на репрезентативному наборі даних з метою оцінки їхньої практичної ефективності за допомогою стандартних метрик якості класифікації.

У результаті дослідження було визначено оптимальні мовні моделі для задачі розпізнавання AI-згенерованих текстів. Їхній вибір базувався на порівнянні моделей за кількісними критеріями, застосуванні згортки оцінок і емпіричній перевірці на реальному наборі даних.

Практична цінність роботи полягає у розробці та апробації методики оцінювання мовних моделей для задачі ідентифікації AI-згенерованих текстів. Запропонований підхід дозволяє обґрунтовано обирати ефективні моделі для застосування у реальних умовах, що сприятиме підвищенню надійності систем виявлення штучно згенерованого контенту та розширенню можливостей їх використання в різних галузях.

CLASSIFICATION, CONTEXT, MACHINE LEARNING, LANGUAGE MODELS, TEXT RECOGNITION, TRANSFORMERS, ARTIFICIAL INTELLIGENCE, AI, BERT, MISTRAL 7B.

The object of this research is language models, their effectiveness, and their applicability for detecting texts generated by artificial intelligence.

The aim of the study is to analyze modern language models for identifying AI-generated texts and to evaluate their performance based on key criteria.

The research methods include classifying language models by type, constructing a vector-based description of the models according to defined criteria, normalizing the scores, and applying aggregation techniques to identify the most effective models. The selected models were further tested on a representative dataset to assess their practical performance using standard classification metrics.

As a result of the study, optimal language models for the task of detecting AI-generated texts were identified. Their selection was based on a comparison of models across quantitative criteria, the application of a convolution (aggregation) and empirical validation on a real dataset..

The practical value of this work lies in the development and testing of a methodology for evaluating language models in the context of identifying AI-generated texts. The proposed approach enables the well-grounded selection of effective models for real-world applications, contributing to the reliability of AI-generated content detection systems and expanding their applicability across various domains.

Завідувачу кафедри
ПІ_____
(скорочена назва кафедри)
проф. Кирилу СМЕЛЯКОВУ_
(вчене звання, сласне ім'я, прізвище)

ЗАЯВА
щодо самостійності виконання кваліфікаційної роботи та можливості її публікації (та/або публікації анотації кваліфікаційної роботи) в електронному архіві відкритого доступу ElAr KhNURE

Я, Ткаченко Олександра Олексіївна_____
(прізвище, ім'я, по батькові)

здобувач вищої освіти на другому (магістерському) рівні вищої освіти академічної групи___ІПЗм-23-3_____

кафедра_____ програмної інженерії _____,
(повна назва кафедри)

заявляю: моя кваліфікаційна робота на тему _«Дослідження мовних моделей для_____ виявлення текстів, згенерованих штучним_ інтелектом»_____,
(назва роботи)

що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в репозиторії "ElArKhNURE". Погоджуюся з авторським договором, відповідно до Положення про репозиторій ХНУРЕ "ElArKhNURE". Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений (а) з вимогами академічної доброчесності, згідно з якими виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

Дата Підпис

ЗМІСТ

ВСТУП

Упродовж останніх років використання текстів, згенерованих штучним інтелектом, стало масовим явищем. Генерація текстів, які майже неможливо відрізнити від написаних людиною, відкриває нові можливості у багатьох сферах, але водночас створює виклики. Питання забезпечення академічної доброчесності, боротьби з дезінформацією, захисту інтелектуальної власності та інформаційної безпеки набувають особливої важливості у сучасному цифровому середовищі. Ідентифікація текстів, створених штучним інтелектом, стає ключовим завданням для мінімізації ризиків, пов'язаних із їх некоректним використанням.

У центрі уваги сучасних досліджень, пов’язаних з аналізом текстів згенерованих штучним інтелектом – використання мовних моделей. Завдяки здатності працювати з контекстом та великими обсягами даних, мовні моделі є провідним інструментом для вирішення завдань класифікації та ідентифікації текстів, створених ШІ. Їх розвиток і вдосконалення дозволяють підвищити точність аналізу та зробити процес ідентифікації більш надійним.

Основна мета цієї роботи полягає в аналізі та оцінюванні сучасних мовних моделей з метою виявлення найбільш ефективних для розпізнавання текстів, згенерованих штучним інтелектом.

Об’єктом дослідження є автоматизовані процеси аналізу текстового контенту для визначення його походження. Предметом дослідження виступають мовні моделі, здатні виявляти специфічні ознаки текстів, створених штучним інтелектом.

Методи дослідження включають класифікацію мовних моделей, побудову їх векторного опису за кількісними критеріями, згортку оцінок і подальше тестування відібраних моделей на наборі даних із реальними та згенерованими текстами.

Отримані результати мають практичну цінність для створення інструментів аналізу текстів, підтримки академічної доброчесності та протидії фейковим матеріалам. Результати роботи були опубліковані у вигляді статті на конференції «СУЧАСНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ MIT@AIS-2025».

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Опис предметної області

Мова – це складна та багатогранна система людського спілкування, що базується на граматичних правилах.

Розробка алгоритмів штучного інтелекту, які здатні розуміти та використовувати мову, є складним викликом. Одним із ключових підходів до розробки алгоритмів, що активно досліджуються протягом останніх двох десятиліть, є мовне моделювання, яке еволюціонувало від статистичних моделей до нейронних.

Нещодавно були запропоновані попередньо навчені мовні моделі (PLMs), які базуються на архітектурі Transformer та навчаються на великих текстових корпусах. Ці моделі показали високу ефективність у вирішенні різноманітних завдань обробки природної мови (NLP). Дослідники виявили, що масштабування моделей, зокрема збільшення кількості параметрів, значно покращує їх продуктивність, і почали детальніше вивчати цей ефект, поступово збільшуючи розмір моделей.

При досягненні певного масштабу параметрів мовні моделі не лише демонструють суттєве покращення продуктивності, але й виявляють нові, несподівані можливості, яких немає у моделей меншого масштабу. Для позначення таких моделей дослідницька спільнота ввела термін "великі мовні моделі" (LLM). Останнім часом LLM стали предметом активних досліджень як у наукових колах, так і в індустрії. Одним із найбільш значущих досягнень стало створення ChatGPT, який привернув увагу широкої аудиторії.[1]

Великі мовні моделі (LLM) стали важливим досягненням у розвитку технологій обробки природної мови. Завдяки навчанню на великих масивах текстових даних, вони демонструють високі результати у генерації структурованого та зрозумілого тексту, що робить їх універсальними інструментами для широкого спектра завдань, включаючи машинний переклад, системи питання-відповідь і автоматичну класифікацію текстів.

Однак широкі можливості цих моделей супроводжуються низкою проблем. Їх здатність генерувати текст високої якості створює ризики зловживань, наприклад, для поширення неправдивої інформації, що негативно впливає на суспільну довіру та спотворює уявлення про реальність. Також значною є загроза плагіату, коли створені моделями тексти використовуються як авторські, що підриває академічну доброчесність. Крім того, виникають питання, пов’язані з порушенням авторських прав, оскільки контент, згенерований на основі чужих даних, може використовуватися без належного дотримання прав інтелектуальної власності.

У великому систематичному аналізу, що охопив 950965 статей, опублікованих з січня 2020 по лютий 2024 року на платформах arXiv, bioRxiv та у журналах Nature, було оцінено поширеність контенту, суттєво зміненого або створеного за допомогою великих мовних моделей (LLM).

Для цього використовувалася статистична модель на рівні всього корпусу текстів, що забезпечила більшу надійність порівняно з аналізом окремих випадків.[2]

Результати дослідження показали стабільне зростання використання LLM, причому найбільше та найшвидше зростання спостерігалося у статтях з комп’ютерних наук (до 17,5%).

Для порівняння, статті з математики та публікації в журналах Nature демонстрували найнижчий рівень модифікації за допомогою LLM (до 6,3%). Також було встановлено, що більший рівень використання LLM характерний для статей, автори яких частіше публікують препринти, працюють у більш насичених дослідницьких галузях та пишуть коротші за обсягом роботи.[2]

Результати свідчать про широке використання LLM у наукових текстах, що підкреслює необхідність подальших досліджень їхнього впливу на академічну практику. Ці виклики змушують шукати нові способи впровадження і використання мовних моделей. Зокрема, важливо навчитися визначати тексти, створені штучним інтелектом, щоб запобігти неправомірному викоританню їхніх можливостей.

1.2 Існуючі підходи

Розвиток штучного інтелекту, особливо великих мовних моделей (LLM), призвів до необхідності розробки методів для виявлення машинно-згенерованого тексту. На сьогодні дослідники застосовують кілька ключових підходів до розв'язання цієї задачі: традиційні статистичні методи, класифікатори, а також методи на основі водяних знаків. [3]

Традиційні статистичні методи базуються на виявленні відмінностей між текстами людини та машинно-згенерованими текстами на основі лінгвістичних характеристик. До них належать такі метрики, як ентропія, перплексія та частота n-грам. Ці методи працюють ефективно для простіших моделей, проте зі зростанням складності LLM вони стають менш надійними.

Новітня розробка, DetectGPT (Mitchell et al., 2023), демонструє значний прогрес у цьому напрямку. Метод працює на основі логарифмічної ймовірності тексту, визначаючи, що машинно-згенеровані тексти часто розташовуються у зонах негативної кривизни розподілу ймовірностей моделі. DetectGPT генерує кілька модифікацій вихідного тексту та порівнює їх для ухвалення рішення, показуючи високі значення AUC.

Методи на основі класифікаторів використовують машинне навчання для побудови систем, які здатні розрізняти тексти людини та AI. Ці методи є популярними для виявлення фейкових новин та боротьби з дезінформацією. У цьому підході активно використовуються сучасні нейронні мовні моделі: BERT, RoBERTa, GPT та інші трансформерні моделі.

Компанія OpenAI нещодавно вдосконалила GPT-модель, донавчивши її на великих наборах даних, що включають Wikipedia, WebText та інші. Такий підхід поєднує можливості класичного класифікування з експертною оцінкою, що дозволяє підвищити точність визначення машинного походження тексту.

Класифікатори демонструють значний потенціал для запобігання поширенню AI-згенерованої дезінформації та підвищення надійності текстових даних у суспільному просторі.

Методи на основі водяних знаків. Водяні знаки є інноваційним підходом до ідентифікації машинно-згенерованого тексту. Спочатку водяні знаки використовувались у сфері захисту зображень та відео для запобігання крадіжці інтелектуальної власності. Нещодавні дослідження демонструють успішну адаптацію цих методів для текстів, згенерованих LLM.

Був розроблений метод м’якого водяного знакування, який використовує ймовірності логітів моделі для вбудовування маркерів у текст. Ці маркери залишаються непомітними для людини, але дозволяють верифікувати походження тексту. Водяні знаки категоризують слова у спеціальні "зелені" та "червоні" списки на основі попередніх токенів, забезпечуючи точну ідентифікацію AI-згенерованого тексту та захист авторських прав.

Окрім того, для ефективного виявлення текстів, згенерованих штучним інтелектом, розглядають класифікацію мовних моделей (LM) за етапами їх розвитку та функціональними можливостями. Це дозволяє краще зрозуміти, які моделі використовуються у різних підходах до розпізнавання машинно-згенерованого тексту. Розвиток мовних моделей можна умовно поділити на чотири етапи, що відображають їх еволюцію та здатність вирішувати завдання різної складності. [1]

Перший етап – статистичні мовні моделі (SLM), що з’явилися у 1990-х роках. Ці моделі базуються на статистичних підходах, наприклад, n-грамні моделі, які передбачають наступне слово на основі попереднього контексту. Вони широко застосовувалися для задач пошуку інформації та обробки природної мови. Однак, через обмежений обсяг контексту та проблему "вимірності" вони не здатні ефективно працювати із сучасними текстами, створеними складними LLM.

Другий етап – нейронні мовні моделі (NLM), що використовують нейронні мережі, такі як рекурентні нейронні мережі (RNN) та багатошарові персептрони. Важливим досягненням стало введення розподілених представлень слів (наприклад, Word2Vec), які дозволили моделювати текст на рівні векторів слів. Цей підхід значно підвищив ефективність задач обробки природної мови, але все ще був обмежений у контекстному розумінні тексту.

Третій етап – попередньо навчені мовні моделі (PLM). Серед них можна виділити такі моделі, як BERT та ELMo, що запровадили парадигму попереднього навчання та подальшого точного налаштування для конкретних задач. Ці моделі здатні створювати контекстно-залежні представлення слів завдяки великим обсягам даних та архітектурам, як Transformer. PLM стали основою для класифікаторів, що дозволяють розрізняти тексти людини та машинно-згенеровані тексти.

Четвертий етап – великі мовні моделі (LLM), такі як OpenChat-3.5, Mistral 7B та ChatGPT, які є найбільш сучасним етапом розвитку. Масштабування моделей за розміром параметрів та обсягом даних призвело до появи emergent abilities – здатності моделей виконувати складні завдання без додаткового навчання. LLM демонструють унікальні можливості у генерації текстів, що майже не відрізняються від людських, і водночас активно використовуються для їх ідентифікації.

1.3 Існуючі обмеження

Попри значний прогрес у розробці методів виявлення AI-згенерованого тексту, існують певні обмеження та вразливості, які ускладнюють ефективну ідентифікацію тексту:

- моделі штучного інтелекту постійно вдосконалюються, наближаючись до людського стилю написання, що ускладнює їх розпізнавання;
- традиційні статистичні методи, такі як ентропія та перплексія, демонструють низьку ефективність проти сучасних великих мовних моделей;
- детектори залежать від якості навчальних даних і не справляються з текстами нових або рідкісних моделей;
- відсутність доступу до внутрішніх параметрів комерційних моделей, наприклад, лог-імовірностей, обмежує ефективність методів виявлення;
- атаки перефразування дозволяють змінювати текст так, щоб він обходив більшість існуючих детекторів;

- схожість розподілів текстів людини та машинного тексту робить їх майже неможливими для розрізнення;
- обмеження в ресурсах та високі обчислювальні витрати знижують доступність детекторів для широкого використання.

1.4 Існуючі тенденції та перспективи

Існуючі методи виявлення здебільшого зосереджені на бінарній класифікації, тобто розрізненні текстів, згенерованих ШІ, від текстів, написаних людиною. Більшість із них використовують мовні моделі, такі як BERT та RoBERTa, для вирішення цієї проблеми. Останні дослідження, які аналізують спільні тексти людини та ШІ, намагаються або класифікувати текст на декілька класів, або розділити його на сегменти й визначити межі між ділянками різного авторства.

Класифікатори, що базуються на статистичних властивостях згенерованого тексту, є логічними підходами до вирішення задачі, оскільки вони намагаються захопити шаблони у тексті, які виникають як прямий результат процесу генерації. Однак точна оцінка функції ймовірностей вимагає доступу до генеративної LLM, що часто неможливо.

Деякі дослідники стверджують, що оскільки LLM наближаються до рівня генерації текстів людини, єдиним дієвим підходом до виявлення стануть водяні знаки. 7 лютого 2024 року OpenAI оголосила про використання водяних знаків в зображеннях, створених за допомогою DALL-E 3 і ChatGPT, а також про публічний інтерфейс для їх виявлення. Проте ми не можемо гарантувати, що тексти, зібрані в інтернеті з невідомих джерел, міститимуть такі маркери. Таким чином, хоча водяні знаки є активною сферою досліджень та важливим аспектом виявлення, вказується, що вони навряд чи відіграватимуть ключову роль у сучасних стратегіях, доки не відбудеться ширше їх впровадження у промисловості.[5]

2 ОГЛЯД Й АНАЛІЗ ЛІТЕРАТУРНИХ, НАУКОВИХ ДЖЕРЕЛ

У межах дослідження було проаналізовано низку наукових робіт, присвячених сучасним мовним моделям, їхній архітектурі, ефективності та практичному застосуванню.

Однією з ключових робіт, що стала основою для розвитку сучасних мовних моделей, є "Attention Is All You Need" (2017) Васвані та співавторів. У цій роботі було запропоновано архітектуру Transformer. Цей підхід дозволив ефективно моделювати довготривалі залежності у текстах та значно підвищивши продуктивність у задачах, таких як генерація тексту та машинний переклад.

Робота "How to Detect AI-Generated Texts?" (2023) Нгуєна та співавторів висвітлює проблему автоматичного виявлення текстів, створених штучним інтелектом. Автори запропонували два підходи: інженерію ознак для машинного навчання та аналіз схожості текстів. Використання цих підходів дозволило досягти високої точності класифікації текстів, створених AI.

Дослідження "A Survey of Large Language Models" (2024) Чжао та співавторів детально розглядає розвиток великих мовних моделей (LLM). Автори класифікували моделі за їх можливостями, починаючи від статистичних підходів до LLM. Робота підкреслює технічні досягнення, що дозволили LLM виконувати складні завдання, та описує етичні виклики, пов’язані з їхнім використанням.

У роботі "Mapping the Increasing Use of LLMs in Scientific Papers" (2024) Лян та співавтори проаналізували вплив LLM на наукову комунікацію. Дослідження показало, що використання LLM у написанні наукових статей значно зросло за останні роки, особливо в галузі комп’ютерних наук. Це свідчить про зміну підходів до створення наукових текстів та підкреслює необхідність розробки методів оцінки якості таких матеріалів.

На основі аналізу представлених робіт можна зробити висновок, що сучасні мовні моделі не лише забезпечили прорив у задачах обробки природної мови, а й викликали нові питання щодо їхнього впливу на академічну та наукову сферу. Їхнє використання відкриває широкі можливості, але водночас вимагає створення нових етичних і методологічних підходів для запобігання можливим ризикам.

3 ПОСТАНОВКА ЗАДАЧІ

Основною метою роботи є дослідження мовних моделей для розпізнавання текстів, створених штучним інтелектом, а також аналіз їх ефективності.

Для цього потрібно виконати такі завдання:

- визначити основні мовні моделі, які будуть досліджуватися, та обґрунтувати їх вибір;
- розробити методику аналізу, включаючи опис критеріїв оцінки моделей;
- оцінити обмеження в застосуванні мовних моделей для задач розпізнавання текстів та запропонувати підходи для їхнього вдосконалення;
- визначити необхідні програмні засоби та ресурси для реалізації поставлених завдань;
- оцінити практичну цінність запропонованих методів та можливість їхнього впровадження.

Для реалізації поставлених завдань основною мовою програмування обрано Python завдяки його широкому набору бібліотек і інструментів, таких як TensorFlow, PyTorch, Hugging Face Transformers, які дозволяють працювати з сучасними мовними моделями. Крім того, Python забезпечує ефективну інтеграцію з бібліотеками для аналізу даних (Pandas, NumPy) та візуалізації результатів (Matplotlib, Seaborn), що є важливим для представлення результатів дослідження.

Для тестування та навчання моделей будуть використан набір даних із відкритої платформи Hugging Face, який включає реальні й синтетичні тексти.

4 ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ

4.1 Аналіз тексту згенерованого ШІ

У цьому розділі розглянемо більш детально мовні моделі для кожної категорії: статистичні мовні моделі (SLM), нейронні мовні моделі (NLM), попередньо навчені мовні моделі (PLM), великі мовні моделі (LLM).

4.2 Статистичні мовні моделі

Чисті статистичні мовні моделі були одними з перших підходів до обробки природної мови. Вони базуються на ймовірнісних методах, які намагаються оцінити ймовірність послідовності слів у тексті. Одним із найпоширеніших підходів є n-грамні моделі, вони базуються на припущенні, що ймовірність наступного слова у послідовності залежить лише від фіксованого вікна попередніх слів. Якщо враховується тільки одне попереднє слово, модель називається біграмною; якщо два слова — триграмною; якщо $n - 1$ слів — n-грамною моделлю.

Послідовності слів довжиною n називаються n-грамами, наприклад:

– для біграм: "штучний інтелект";
– для триграм: "текст згенерований ШІ".

Для задачі розпізнавання текстів, згенерованих штучним інтелектом, серед статистичних мовних моделей було використано підхід на основі n-грам із згладжуванням Modified Kneser-Ney, який забезпечує точнішу оцінку ймовірностей появи послідовностей слів. На відміну від простіших методів, таких як Лапласове чи Good-Turing згладжування, Modified Kneser-Ney враховує частотність рідковживаних слів та контекст їх появи, що є критично важливим для виявлення структурних аномалій у тексті.

Ця модель застосовується для обчислення перплексії, яка є мірою передбачуваності тексту: AI-згенеровані тексти зазвичай мають нижчу перплексію, тоді як людські тексти демонструють більшу варіативність. Згладжування Modified Kneser-Ney переважає інші статистичні підходи завдяки здатності ефективно працювати з великими корпусами текстів, виявляти шаблони в послідовностях слів

і забезпечувати збалансовану оцінку ймовірностей навіть для незнайомих контекстів.

Перед початком аналізу корпус текстів обробляється для побудови частотного розподілу n-грам**.** Частоти кожної n-грам записуються у вигляді таблиці, наприклад: $c\left(w_{i-n+1}^{i}\right)$ — частота появи n-грам; $c\left(w_{i-n+1}^{i-1}\right)$ — частота (n-1)-грам, які передують слову w_i.

Modified Kneser-Ney використовується для уникнення нульових ймовірностей, які виникають для рідковживаних або невідомих n-грам. Формула виглядає так 4.1:

$$P\left(w_i|w_{i-n+1}^{i-1}\right) = \frac{\max\left(c\left(w_{i-n+1}^{i}\right) - D, 0\right)}{c\left(w_{i-n+1}^{i-1}\right)} + \lambda\left(w_{i-n+1}^{i-1}\right) * P(w_i|w_{i-n+2}^{i-1}) \quad (4.1)$$

де $c\left(w_{i-n+1}^{i}\right)$ - частота n-грам,

$P(w_i|w_{i-n+2}^{i-1})$ - ймовірність (n-1)-грам,

$\lambda\left(w_{i-n+1}^{i-1}\right)$ - коефіцієнт нормалізації, який забезпечує збалансованість між частинами формули,

$\max\left(c\left(w_{i-n+1}^{i}\right) - D, 0\right)$ - знижує ймовірність пропорційно до D, але ніколи не дає негативних значень.

Якщо n-грам невідомий, аналіз зводиться до (n-1)-грам. Процес триває доти, доки не досягне уніграми (одного слова), для якої ймовірність береться з абсолютної частоти в корпусі.

Після визначення ймовірностей n-грам для всього тексту обчислюється перплексія, що є мірою передбачуваності тексту, формула 4.2:

$$Perplexity = 2^{-\frac{1}{N}\Sigma_{i=1}^{N}\log_2 P(w_i|w_{i-n+1}^{i-1})} \quad (4.2)$$

де N - кількість слів у тексті,

$P(w_i|w_{i-n+1}^{i-1})$ - ймовірність n-грам.

AI-згенеровані тексти зазвичай мають нижчу перплексію, бо їхня структура більш передбачувана. AI часто використовують повторювані структури, що призводить до завищених ймовірностей для деяких n-грам.

Людські тексти мають вищу перплексію через більшу лексичну та синтаксичну варіативність. Такі тексти можуть включати унікальні послідовності слів, які важко передбачити для AI.

4.3 Нейронні мовні моделі

Розвиток мовних моделей продовжився завдяки появі нейронних мереж. Одним із головних обмежень статистичних моделей було те, що вони погано масштабувалися для великих обсягів тексту та не могли враховувати довготривалі залежності між словами. Це призвело до виникнення нейронних мовних моделей (NLM), які використовують глибокі нейронні мережі для створення більш складних і контекстуально залежних представлень тексту. Завдяки використанню векторизації слів та рекурентних архітектур, такі моделі змогли враховувати залежності в межах усього речення, а не лише у фіксованому вікні слів, як це було в n-грамних підходах.

FastText є однією з найефективніших моделей для векторизації слів. Вона базується на простій архітектурі нейронної мережі, яка враховує не лише слово як одиницю, але й його субслова.

FastText демонструє високу продуктивність у задачах класифікації тексту завдяки своїй здатності враховувати морфологічні особливості слів і швидко обробляти великі обсяги даних.

У контексті розпізнавання тексту, згенерованого ШІ, модель може бути використана для виявлення характерних ознак, які відрізняють "штучний" текст від створеного людиною.

FastText розширює підхід Word2Vec, представляючи кожне слово не як єдиний токен, а як сукупність субсловів — коротких послідовностей символів фіксованої довжини n (n-грам).

Наприклад, для слова "машина" із n=3 (3-грамами) модель генерує такі субслови: <ма, маш, аши, шин, ина, на>. Спеціальні символи < і > позначають початок і кінець слова, що допомагає моделі враховувати позицію символів у слові.

Векторне представлення слова $v(w)$ розраховується як сума векторів усіх його n-грам, формула 4.3:

$$v(w) = \sum_{g \epsilon G(w)} z\,(g) \tag{4.3}$$

де $G(w)$ — множина всіх n-грам слова w,

$z(g)$ — векторне представлення кожної n-грами g.

Навчання моделі відбувається за допомогою оптимізації функції втрат, яка спрямована на передбачення контексту слів. Модель максимізує ймовірність того, що вектор слова-цілі w_t є близьким до векторів слів-контекстів w_c, і одночасно мінімізує ймовірність того, що випадкові слова (негативні семпли) відповідають цьому ж контексту. Функція втрат записується так, формула 4.4:

$$\mathcal{L} = -\,log\,\sigma\,(u_{w_c}^{T} v(w_t)) - \sum_{n=1}^{N} log\,\sigma\,(-u_{w_n}^{T} v(w_t)) \tag{4.4}$$

де σ — сигмоїдна функція,

$v(w_t)$ — вектор слова-цілі w_t,

$u_{w_c}^{T}$ — транспонований вектор слова-контексту w_c,

$u_{w_n}^{T}$ — транспонований вектор негативного семплу w_n.

Ця функція забезпечує навчання моделі, де вектори слів, що знаходяться поруч у тексті, стають схожими, а вектори далеких або випадкових слів — несхожими.

Для задачі розпізнавання текстів, створених штучним інтелектом, модель FastText може використовуватися як класифікатор. У цьому випадку кожен текст

подається як набір векторів, створених на основі його слів, і модель навчається передбачати, до якого класу належить текст: створений людиною чи згенерований ШІ.

Для цього оптимізується функція втрат на основі бінарної крос-ентропії, формула 4.5:

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}[y_i log\, p\,(y_i|x_i) + (1-y_i)log(1-p(y_i|x_i))] \quad (4.5)$$

де y_i — мітка класу тексту x_i, що дорівнює 1, якщо текст згенерований, і 0, якщо текст створений людиною,

$p(y_i|x_i)$ — ймовірність, представляє впевненість моделі у віднесенні тексту до класу y_i.

Після завершення навчання FastText готова до класифікації нових текстів. Модель об'єднує вектори всіх слів у тексті, створюючи векторне представлення всього тексту. Отриманий вектор подається в класифікаційний шар, який було налаштовано під час навчання. На основі параметрів цього шару модель обчислює ймовірність належності тексту до кожного класу.

Ймовірність для кожного класу розраховується за допомогою сигмоїдної функції, яка перетворює результат класифікації у значення від 0 до 1.

Формула 4.6 сигмоїдної функції виглядає так:

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad (4.6)$$

де x — вхідне значення (лінійна комбінація векторів і параметрів моделі),

e — основа натурального логарифм.

Якщо ймовірність класу "згенерований ШІ" перевищує порогове значення, наприклад 0.5, текст вважається створеним штучним інтелектом. В іншому випадку текст класифікується як написаний людиною.

4.4 Попередньо навчені мовні моделі

Попередньо навчені мовні моделі (PLM) є продовженням розвитку нейронних мовних моделей (NLM) із суттєвими покращеннями. Базуючись на тих самих принципах, PLM вирізняються тим, що проходять етап попереднього навчання на великих корпусах текстів, завдяки чому засвоюють загальні закономірності мови. Це дозволяє моделям швидко адаптуватися до конкретних задач, таких як розпізнавання текстів, згенерованих штучним інтелектом, без потреби навчання "з нуля".

Однією з найважливіших моделей у розвитку попередньо навчених мовних підходів є ELMo. Її унікальність полягає у здатності створювати контекстно-залежні векторні представлення слів, враховуючи як попередній, так і наступний контекст. Це дозволяє моделі захоплювати тонкі семантичні та синтаксичні взаємозв'язки, які не враховують моделі типу Word2Vec чи FastText. Завдяки цій архітектурі ELMo є одним із ключових підходів у розпізнаванні текстів.

ELMo будує свої векторні представлення на основі багаторівневої двонаправленої рекурентної нейронної мережі (BiLSTM), яка обробляє текст як зліва направо, так і справа наліво. Завдяки цьому кожне слово в тексті отримує представлення, що залежить як від попереднього, так і від наступного контексту. Для слова w_i у реченні модель створює векторне представлення через комбінацію прихованих станів на всіх рівнях мережі, формула 4.7:

$$v_{ELMo}(w_i) = \gamma \sum_{k=1}^{L} s_k h_{k,i} \tag{4.7}$$

де L — кількість шарів у BiLSTM,

$h_{k,i}$ — прихований стан k-го шару для слова w_i,

s_k — ваги шарів, що адаптивно комбінують інформацію з різних рівнів,

γ — параметр масштабування для адаптації до конкретної задачі.

Навчання ELMo відбувається на великому корпусі текстів за допомогою задачі мовного моделювання. Модель прогнозує наступне слово на основі попередніх (прямий контекст) і попереднє слово на основі наступних (зворотний контекст). Це дозволяє створювати вектори, які враховують як граматичні, так і семантичні зв'язки між словами. Функція втрат на цьому етапі, формула 4.8:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[log\, P\,(w_i|w_1, \dots, w_{i-1}) + log\, P\,(w_i|w_{i+1}, \dots, w_N)] \qquad (4.8)$$

де w_i — слово, для якого обчислюється ймовірність,

N — кількість слів у тексті,

$P(w_i| \dots)$ — ймовірність слова w_i у заданому контексті.

Для розпізнавання тексту, згенерованого ШІ, ELMo використовується як компонент для генерації векторів, а класифікація відбувається за допомогою зовнішньої моделі. Спочатку ELMo створює контекстно-залежні вектори для кожного слова у тексті. Потім ці вектори комбінуються у векторне представлення тексту, наприклад, через обчислення середнього або зваженого середнього векторів слів.

Отриманий вектор тексту подається в класифікаційний шар, наприклад, логістичну регресію чи багатошарову нейронну мережу, яка прогнозує ймовірність належності тексту до класу "згенерований ШІ" або "людський текст"

Хоча ELMo і FastText відрізняються своєю внутрішньою архітектурою і підходами до створення векторів, на етапі класифікації обидві моделі вирішують одну й ту саму задачу: передбачити ймовірність того, що текст належить до певного класу. Тому вони використовують одну й ту саму функцію втрат.

Трансформери стали наступним етапом у розвитку мовного моделювання, усунувши обмеження послідовної обробки тексту, властиві рекурентним нейронним мережам. Архітектура трансформера базується на інтеграції кількох ключових компонентів, які працюють у тісному взаємозв'язку для забезпечення ефективної обробки тексту. Основна структура складається з двох частин: кодера

(encoder) та декодера (decoder). Кодер перетворює вхідну послідовність символів у набір векторних представлень, тоді як декодер використовує ці представлення для генерації вихідного тексту. Кожен шар кодера включає два основні елементи: багатоголову самоувагу (multi-head self-attention) та повнозв'язну мережу (feed-forward network). Самоувага дозволяє моделі аналізувати залежності між усіма словами у тексті, а повнозв'язна мережа застосовується до кожного слова окремо для обробки його семантичного значення.

У центрі роботи трансформера лежить механізм уваги, відомий як Scaled Dot-Product Attention. Він оцінює зв'язок між кожною парою слів у тексті, обчислюючи вагу впливу одного слова на інше. Цей процес формалізується через функцію, де запити (queries), ключі (keys) і значення (values) представлені у вигляді матриць. Формула 4.9 виглядає так:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (4.9)$$

де Q, K і V — це матриці запитів, ключів та значень,

d_k— розмірність ключів, яка забезпечує стабільність обчислень.

Цей підхід дозволяє моделі визначати, наскільки важливим є кожне слово в контексті інших, навіть якщо вони знаходяться далеко одне від одного.

Щоб забезпечити ще більшу точність аналізу, трансформери використовують багатоголову увагу. Цей механізм дозволяє обчислювати кілька незалежних "голів" уваги, кожна з яких аналізує різні аспекти залежностей між словами. Обчислення результатів кожної "голови" виконується паралельно, а потім вони об'єднуються у спільне представлення за допомогою навчальних матриць.

Оскільки трансформери обробляють текст паралельно, їм необхідно враховувати порядок слів у послідовності. Для цього використовується позиційне кодування, яке додає інформацію про позицію кожного слова до його векторного представлення. Це реалізується за допомогою синусоїдальних функцій, формула 4.10:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right), PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \quad (4.10)$$

де pos — це позиція слова,

i — індекс виміру,

d — розмірність ембедингів.

Завдяки позиційному кодуванню трансформери можуть враховувати порядок слів, навіть обробляючи їх паралельно.

Ці компоненти взаємодіють, створюючи потужну архітектуру (див. рис. 4.1), яка здатна ефективно аналізувати текст, враховуючи його контекст та структуру.

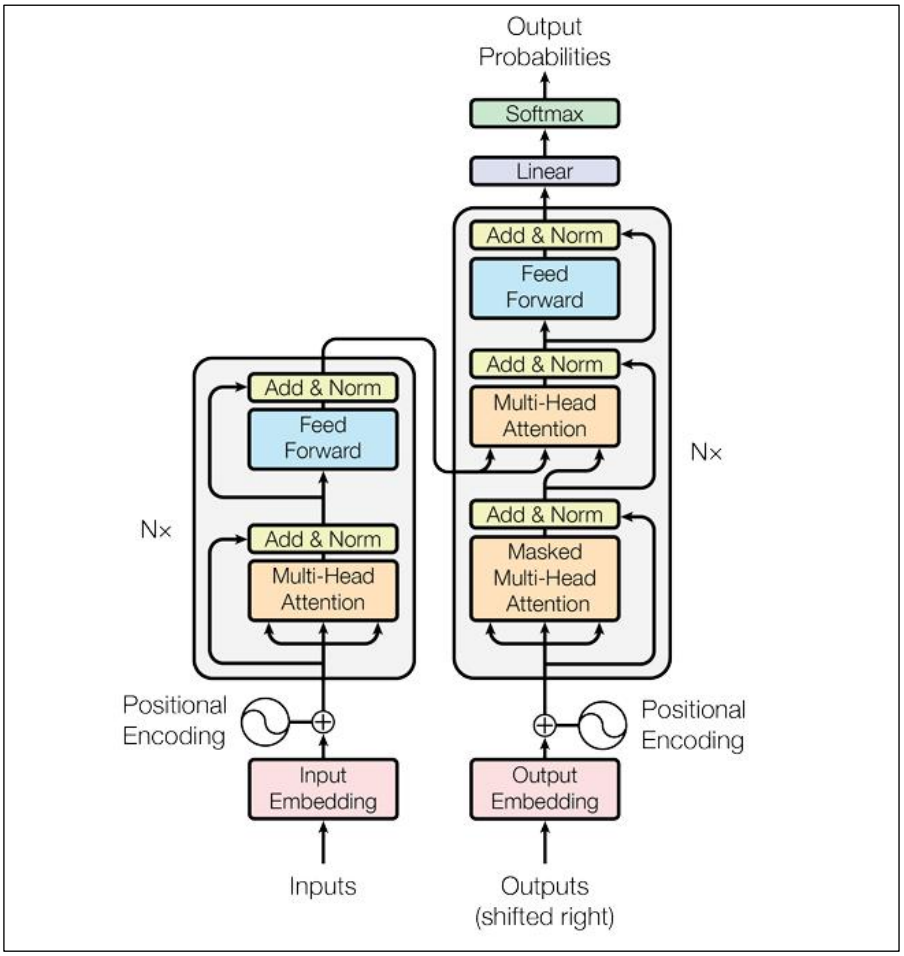

Рисунок 4.1 – Архітектура трансформерів [6]

Завдяки цьому трансформери стали основою для багатьох сучасних моделей обробки природної мови, таких як BERT та GPT, які демонструють передові

результати у задачах класифікації текстів, машинного перекладу та розпізнавання текстів, створених штучним інтелектом.

BERT (Bidirectional Encoder Representations from Transformers) є однією з найвідоміших мовних моделей, заснованих на архітектурі трансформерів. Її унікальність полягає в тому, що вона використовує двонаправлений підхід для аналізу тексту, тобто одночасно враховує як контекст слів, які йдуть перед заданим словом, так і тих, які йдуть після нього.

Архітектура BERT ґрунтується виключно на кодерній частині трансформера. Кожен шар кодувальника містить багатоголову самоувагу і повнозв'язну нейронну мережу, що дозволяє обробляти текст паралельно та враховувати складні залежності між словами. Завдяки двонаправленості, модель може розуміти контекст більш глибоко, що особливо важливо для задач, де значення слова чи фрази залежить від їхнього оточення в реченні.

Для навчання BERT використовується двоетапний підхід: попереднє навчання (pre-training) і тонке налаштування (fine-tuning). На етапі попереднього навчання модель проходить через дві ключові задачі. Перша задача — Masked Language Model (MLM), де частина слів у тексті замінюється спеціальним маркером [MASK], і модель намагається передбачити ці слова на основі їхнього контексту. Формально задача формулюється так, формула 4.11:

$$\mathcal{L} = -\sum_{i \in masked} log\, P(w_i | w_1, \dots, w_{i-1}, w_{i+1}, \dots, w_N) \tag{4.11}$$

де w_i — слово, яке потрібно передбачити,

N — кількість слів у тексті.

Цей підхід дозволяє BERT навчитися розуміти як локальні, так і глобальні залежності між словами.

Друга задача — Next Sentence Prediction (NSP), яка вчить модель розуміти відносини між реченнями. На цьому етапі модель отримує пару речень і повинна

визначити, чи друге речення логічно продовжує перше. Ця задача допомагає моделі враховувати більш високорівневу семантику тексту.

Після завершення попереднього навчання BERT можна адаптувати до конкретних задач за допомогою тонкого налаштування. Це досягається додаванням класифікаційного шару до вихідного представлення моделі та повторним навчанням на невеликому наборі даних, специфічному для задачі. У задачі розпізнавання текстів, створених штучним інтелектом, BERT може використовуватися для класифікації текстів на "згенеровані ШІ" або "людські". Вхідний текст подається на модель, яка генерує представлення для кожного слова та речення в цілому. Після цього класифікаційний шар передбачає ймовірність належності тексту до конкретного класу.

RoBERTa (Robustly Optimized BERT Approach) — це вдосконалена версія моделі BERT, яка фокусується на покращенні процесу навчання та максимальному використанні її архітектури. Основні зміни стосуються збільшення обсягу навчальних даних до 160 ГБ (порівняно з 16 ГБ у BERT) та оптимізації навчальних задач. Зокрема, було виключено задачу Next Sentence Prediction (NSP), яка передбачала оцінку зв’язку між реченнями. Дослідження показали, що NSP незначно впливає на продуктивність, тому її замінили вдосконаленою реалізацією Masked Language Modeling (MLM). У цьому підході модель передбачає замасковані слова в тексті, аналізуючи їхній контекст. Формула 4.11 втрат для цієї задачі залишається незмінною.

Одним із ключових удосконалень RoBERTa стало використання динамічного маскування. На відміну від BERT, де замасковані слова залишаються незмінними протягом усього навчання, у RoBERTa вони змінюються при кожному проході тексту через модель. Це дозволяє створювати різноманітні навчальні приклади, що підвищує здатність моделі до генералізації. Крім того, RoBERTa оптимізувала гіперпараметри навчання, зокрема збільшила розміри пакетів і тривалість навчальних епох. Завдяки цим змінам модель демонструє значно вищу продуктивність у задачах класифікації текстів, таких як розпізнавання текстів, створених штучним інтелектом.

DeBERTa (Decoding-enhanced BERT with disentangled attention) є продовженням розвитку моделей на основі BERT і RoBERTa, що запроваджує кілька важливих інновацій. Основною особливістю DeBERTa є розділена увага (disentangled attention)**.** У традиційних трансформерах позиційна інформація додається безпосередньо до векторних представлень слів, що ускладнює їхню роздільну обробку. DeBERTa вирішує цю проблему, обробляючи семантичне представлення слова і його позицію окремо. Формула 4.12 уваги у DeBERTa виглядає так:

$$Attention(Q, K, V) = softmax\left(\frac{QW^Q * KW^K}{\sqrt{d_k}} + P\right)V, \qquad (4.12)$$

де P — це матриця позиційної уваги, яка враховує залежності між позиціями слів.

Такий підхід дозволяє моделі точніше враховувати залежності між словами, особливо у довгих текстах зі складними структурами.

DeBERTa-v3 також використовує вдосконалений процес навчання. Модель проходить попереднє навчання на великих текстових корпусах із використанням Masked Language Modeling. Завдяки розділеній увазі та більш ефективним оптимізаторам DeBERTa досягає високих результатів навіть на задачах із невеликими обсягами даних. Крім того, вдосконалення торкнулися способу обробки контексту: модель точніше враховує семантичні зв’язки між словами, що покращує її продуктивність у складних мовних задачах.

4.5 Великі мовні моделі

Великі мовні моделі (LLM, Large Language Models) — це сучасний етап у розвитку попередньо навчених мовних моделей, який характеризується використанням величезної кількості параметрів, що досягають сотень мільярдів. Вони здатні вирішувати широкий спектр задач обробки природної мови, включаючи генерацію текстів, класифікацію, аналіз емоцій, переклад, створення коду та багато іншого. Їхній успіх ґрунтується на кількох ключових факторах:

масштабному навчанні, ефективній архітектурі трансформерів та вдосконалених оптимізаційних методах.

Масштабність LLM забезпечується завдяки розподіленому навчанню на кластерних системах із використанням GPU і TPU. Багатоголова увага (multi-head attention) дозволяє ефективно обробляти текст, розподіляючи обчислення між різними компонентами, а системи оптимізації, такі як мікробатчі, знижують потребу у великих обсягах пам’яті. Завдяки цим механізмам моделі здатні обробляти сотні мільярдів параметрів і генерувати точні представлення текстів.

OpenChat-3.5 — це відкрита мовна модель, побудована на архітектурі GPT і орієнтована на генерацію тексту за допомогою причинного мовного моделювання (causal language modeling). Вона використовує лише декодери трансформера, що дозволяє їй ефективно передбачати наступне слово на основі контексту.

Модель підтримує few-shot і zero-shot навчання, що дає змогу розв’язувати нові задачі без попереднього донавчання. Завдяки відкритості та хорошій якості генерації тексту, OpenChat-3.5 є зручним інструментом для експериментального використання в задачах виявлення AI-згенерованих текстів.

Модель Mistral 7B орієнтована на оптимізацію ресурсів і використовує ефективні блоки трансформерів для зменшення обчислювальної складності. Вона була створена Google Research з акцентом на оптимізацію використання ресурсів. Модель має лише 7 мільярдів параметрів, але використовує вдосконалений механізм багатоголової уваги та ефективну нормалізацію параметрів, що дозволяє досягати продуктивності, порівнянної з більшими моделями.

Модель також інтегрує нормалізацію параметрів (parameter efficiency techniques), яка мінімізує втрати інформації під час навчання. Ці особливості роблять Mistral придатною для використання у середовищах із обмеженими обчислювальними ресурсами.

Phi-2 запроваджує покращений механізм обробки контексту, який дозволяє глибше аналізувати семантичні залежності у текстах. Ця модель орієнтована на складні сценарії, наприклад, технічну документацію або довгі наукові тексти. Її архітектура містить розширені шари уваги, які інтегрують позиційне кодування з

динамічним масштабуванням, забезпечуючи гнучкість у роботі з довгими текстами. Phi-2 також включає адаптивні алгоритми тренування, що знижують споживання ресурсів під час навчання.

Orion-14B має 14 мільярдів параметрів і використовує модифіковану архітектуру трансформерів із підтримкою глобальної уваги (global attention)**.** Це дозволяє моделі ефективно обробляти довгі тексти, зберігаючи їхню послідовність і контекст. Унікальною особливістю Orion є адаптивне позиційне кодування, яке оптимізує обробку текстів із різною довжиною. Крім того, модель використовує гібридний підхід до нормалізації шарів, що знижує ризик перенавчання та підвищує стабільність роботи.

4.6 Проведення теоретичного дослідження

Серед численних доступних мовних моделей було відібрано новітні та найпопулярніші представники кожного напрямку:

- Modified Kneser-Ney – метод згладжування для n-грамних моделей, удосконалена версія Kneser-Ney для великих текстових корпусів;
- FastText (Meta AI, 2016) – модель, що враховує субсловні одиниці для роботи з рідкісними словами, джерело інформації [11];
- ELMo (AllenNLP, 2018) – контекстно-залежні вектори слів, створені за допомогою рекурентних нейронних мереж, джерело інформації [12];
- BERT (Google, 2018) – двонаправлена модель на основі трансформерів для контекстного аналізу тексту [13];
- RoBERTa (Meta AI, 2019) – покращена версія BERT з оптимізованим навчанням [14];
- DeBERTa-v3 (Microsoft, 2021) – модель із розділеною увагою для точного врахування позицій слів [15];
- Phi-2 (Microsoft, 2024) – дуже нова модель, що забезпечує високу продуктивність на компактному розмірі [16];
- Orion-14B (2024) – нова багатомовна модель з відкритим кодом, орієнтована на широкий спектр NLP-завдань [17];

- Mistral 7B (Mistral AI, 2023) – компактна модель з ефективною архітектурою, що показує високу продуктивність [18]
- OpenChat-3.5 (OpenChat, 2023) – відкрита модель для генерації та аналізу тексту, розроблена на базі архітектури GPT; доступна для локального використання і модифікації [19].

Для порівняння моделей було виділено 5 критеріїв.

Перший – кількість параметрів. Кількість вагових параметрів, які використовуються для навчання та прогнозування. Тип шкали – порядкова. Вимірюється кількість вагових параметрів моделі, які використовуються для навчання та прогнозування. Більша кількість параметрів підвищує здатність моделі працювати зі складними текстовими залежностями, але розриви між значеннями не є рівномірними.

Значення:

- маленькі мовні моделі: до 1 000 000 параметрів (статистичні та нейронні мовні моделі);
- середні моделі: 1 000 000 - 1 000 000 00 параметрів (попередньо навчені мовні моделі);
- великі моделі: понад 1 000 000 000 (великі мовні моделі).

Другий – врахування контексту. Тип шкали – шкала відносин. Максимальна кількість токенів, які модель може обробити одночасно. Чим більше значення, тим краще модель враховує контекст. Великий контекст дозволяє краще аналізувати довгі тексти та складні залежності між словами.

Третій – необхідна пам’ять. Тип шкали – шкала відносин. Обсяг оперативної пам'яті або ресурсів GPU (у гігабайтах), необхідний для збереження параметрів моделі та її використання. Великі моделі потребують більше пам'яті для навчання та роботи.

Четвертий – розмір вбудованого словника. Тип шкали – шкала відносин. Кількість унікальних слів або токенів, які модель може враховувати у своїй роботі. Більший словник забезпечує роботу з різноманітним контентом, включаючи рідкісні та технічні терміни.

П’ятий – призначена для класифікації тексту. Тип шкали – порядкова. Визначає, чи була модель створена для класифікації тексту або може бути адаптована для класифікації за допомогою додаткових засобів.

Складемо векторний опис обраних для дослідження моделей за визначеними критеріями, результат наведений у таблиці 4.1.

Таблиця 4.1 – Векторний опис обраних моделей за визначеними критеріями

Модель	Кількість параметрів (у мільйонах)	Контекст (токени)	Необхідна пам’ять (GB)	Розмір словника (токени)	Призначення для класифікації тексту
Modified Kneser-Ney	Немає параметрів	3	1	20 000	Ні
FastText	Необмежена	10	4	Необмежена	Так
ELMo	30 (Маленька модель)	512	8	50 000	Ні
BERT	110 (Середня модель)	512	12	30 000	Так
RoBERTa	125 (Середня модель)	512	16	50 000	Так
DeBERTa-v3	184 (Середня модель)	512	16	128 000	Так
Phi-2	2700 (Велика модель)	2048	5.6	51 200	Ні
Orion-14B	14000 (Велика модель)	4096	29	84 608	Ні
Mistral 7B	7000 (Велика модель)	32K	14.4	32 000	Ні
OpenChat-3.5	7000 (Велика модель)	8192	14.4	32 002	Ні

Деякі з критеріїв оцінюються за якісною школою, для подальшого дослідження потрібно привести якісні шкали до кількісних.

Приводимо якісну шкалу кількість параметрів до кількісної, чим більше значення тим краще.

Значення:

– маленькі моделі: 1;

– середні моделі: 2;

– великі моделі: 3.

Приводимо якісну шкалу призначення для класифікації тексту до кількісної. Якщо модель була створена для класифікації тексту — присвоюємо значення 1, якщо ні — 0.

У n-грамної моделі з методом згладжування Modified Kneser-Ney немає парметрів, але оскільки в неї є здатність до прогнозування то поставимо значення – 1.

Для моделі FastText значення критерію кількість параметрів та розмір вбудованого словника необмежене, встановимо найбільше значення серед інших моделей – 3 та 200001 відповідно.

Складемо векторний опис обраних для дослідження моделей за визначеними критеріями за кількісними шкалами, результат наведений у таблиці 4.2.

Таблиця 4.2 – Векторний опис альтернатив за кількісними шкалами

Модель	Кількість параметрів	Контекст (токени)	Потреба у пам'яті (GB)	Розмір словника (токени)	Призначення для класифікації тексту
Modified Kneser-Ney	1	3	1	20 000	0
FastText	3	10	4	200 001	1
ELMo	1	512	8	50 000	0
BERT	2	512	12	30 000	1
RoBERTa	2	512	16	50 000	1
DeBERTa-v3	2	512	16	128 000	1
Phi-2	3	2048	5.6	51 200	0
Orion-14B	3	4096	29	84 608	0
Mistral 7B	3	32000	14.4	32 000	0
OpenChat-3.5	3	8192	14.4	32 002	0

Шкала обсягу пам’яті визначає кількість ресурсів (RAM або GPU VRAM), необхідних для роботи моделі. Оскільки чим менше значення, тим краще, то принцип оптимальності – мінімум. Шкалу слід привести до оптимальності за принципом максимуму: чим менше пам’яті споживає модель, тим вищий її бал за економією обсягу пам’яті. Результат приведення обсягу пам’яті до принципу оптимальності «за максимумом» можна побачити у таблиці 4.3.

Таблиця 4.3 – Приведення всіх шкал до принципу оптимальності «за максимумом»

Модель	Кількість параметрів	Контекст (токени)	Економія пам'яті (GB)	Розмір словника (токени)	Призначення для класифікації тексту
Modified Kneser-Ney	1	3	28	20 000	0
FastText	3	10	25	200 001	1
ELMo	1	512	21	50 000	0
BERT	2	512	17	30 000	1
RoBERTa	2	512	13	50 000	1
DeBERTa-v3	2	512	13	128 000	1
Phi-2	3	2048	23.4	51 200	0
Orion-14B	3	4096	0	84 608	0
Mistral 7B	3	32000	14.6	32 000	0
OpenChat-3.5	3	8192	14.6	32 002	0

За принципом Парето порівнюємо множину альтернатив, модель Phi-2 перевершує ELMo за всіма критеріями, що дозволяє виключити останню з розгляду. Модель DeBERTa-v3 перевершує RoBERTa за всіма критеріями. Результат наведений у таблиці 4.4.

Таблиця 4.4 – Результат використання на множині альтернатив принципу Парето

Модель	Кількість параметрів	Контекст (токени)	Економія пам'яті (GB)	Розмір словника (токени)	Призначення для класифікації тексту
Modified Kneser-Ney	1	3	28	20 000	0
FastText	3	10	25	200 001	1
BERT	2	512	17	30 000	1
DeBERTa-v3	2	512	13	128 000	1
Phi-2	3	2048	23.4	51 200	0
Orion-14B	3	4096	0	84 608	0
Mistral 7B	3	32000	14.6	32 000	0
OpenChat-3.5	3	8192	14.6	32 002	0

Проведемо нормування оцінок за шкалами.

Формула 4.13 для нормування:

$$f = \frac{f_{\text{вимір}} - f_{min}}{f_{max} - f_{min}} \quad (4.13)$$

де f_{min} – мінімальна з оцінок критерію;

f_{max} – максимальна з оцінок критерію;

$f_{\text{вимір}}$ – оцінка для якої відбувається нормалізація.

Результати отримані після нормування оцінок за шкалами можна побачити у таблиці 4.5.

Таблиця 4.5 – Нормування оцінок за шкалами

Модель	Кількість параметрів	Контекст	Економія пам'яті	Розмір словника	Призначення для класифікації тексту
Modified Kneser-Ney	0.0	0.0	1	0.0	0.0
FastText	1.0	0.00002	0.89286	1.0	1.0
BERT	0.5	0.01591	0.60714	0.05555	1.0
DeBERTa-v3	0.5	0.01591	0.46428	0.59999	1.0
Phi-2	0.5	0.06391	0.83571	0.17333	0.0
Orion-14B	1.0	0.12791	0.0	0.35893	0.0
Mistral 7B	1.0	1.0	0.52142	0.06666	0.0
OpenChat-3.5	1.0	0.25593	0.52142	0.06667	0.0

Для визначення найбільш відповідних критеріям моделей застосовано лінійну адитивну згортку з ваговими коефіцієнтами, оскільки деякі критерії мають значно більший вплив на ефективність та вибір моделі.

Формула 4.14 згортки:

$$Z^* = \max_{i=1,m} \sum_{j=1}^{n} a_j b_j a_{ij} \quad (4.14)$$

де a_j – нормуючий множник,

β_j – ваговий коефіцієнт.

Формула 4.15 нормуючого множника:

$$a_j = \frac{1}{\sum_{i=1}^{m} a_{ij}} \tag{4.15}$$

Для визначення найбільш підходящої моделі у задачі розпізнавання текстів, згенерованих штучним інтелектом, важливим критерієм є кількість параметрів, оскільки більша кількість параметрів дозволяє краще вловлювати складні закономірності та особливості тексту.

Врахування контексту також відіграє значну роль, цей критерій відображає кількість токенів, які модель може обробляти одночасно. Великий контекст особливо важливий при роботі з довгими текстами, де потрібно аналізувати глобальні залежності та шаблони. Проте його значення дещо зменшується у порівнянні з призначення для класифікації тексту моделі та кількістю параметрів, якщо тексти є відносно короткими або середнього розміру.

Третім важливим критерієм є розмір словника, що впливає на здатність моделі працювати з різноманітною лексикою, включаючи рідкісні та технічні терміни. Менш значущим, але важливим критерієм є призначення для класифікації тексту. Останнім за пріоритетністю є економія пам'яті, оскільки ресурсні обмеження є менш критичними у порівнянні з іншими характеристиками.

Отже, маємо такі ранги: кількість параметрів – 5; врахування контексту – 4; розмір словника – 3; призначення для класифікації тексту – 2; економія пам'яті – 1. У сумі виходить 15.

Тоді вагові коефіцієнти: кількість параметрів – 5/15; врахування контексту – 4/15; розмір словника – 3/15; призначення для класифікації тексту – 2/15; економія пам'яті – 1/15.

Розрахуємо корисності альтернатив за обраною загортковою моделлю. Розрахунки наведені у таблиці 4.6.

Таблиця 4.6 –Результати розрахунків корисності альтернатив

Модель	Кількість параметрів	Контекст	Економія пам'яті	Розмір словника	Призначення для класифікації тексту	Результат згортки
Modified Kneser-Ney	0.0	0.0	1	0.0	0.0	0.0144543
FastText	1.0	0.00002	0.89286	1.0	1.0	0.2024076
BERT	0.5	0.01591	0.60714	0.05555	1.0	0.0897989
DeBERTa-v3	0.5	0.01591	0.46428	0.59999	1.0	0.1346456
Phi-2	0.5	0.06391	0.83571	0.17333	0.0	0.0686771
Orion-14B	1.0	0.12791	0.0	0.35893	0.0	0.1142685
Mistral 7B	1.0	1.0	0.52142	0.06666	0.0	0.2557635
OpenChat-3.5	1.0	0.25593	0.52142	0.06667	0.0	0.1199842
Вагові коефіцієнти	0.33	0.27	0.07	0.20	0.13	
Нормуруючі множники	0.1818181	0.675863	0.206491	0.430825	0.333333	

Значення корисності моделей показують певний поділ на групи за результатами. FastText та Mistral 7B продемонстрували найвищі результати. Це пов'язано з їхньою здатністю обробляти великий обсяг контексту, значною кількістю параметрів та ефективним використанням ресурсів для вирішення складних задач.

DeBERTa-v3, OpenChat-3.5 та Orion-14B зайняли середні позиції. Вони демонструють збалансовані характеристики, забезпечуючи хороший рівень обробки контексту та продуктивності, проте поступаються лідерам за деякими критеріями, зокрема за розміром контексту та кількістю параметрів, проте можуть працювати швидше ніж моделі з високими значеннями.

Моделі Phi-2, BERT та n-грамна модель з методом згладжування Modified Kneser-Ney показали найнижчі результати. Основними причинами є невеликий контекст, менша кількість параметрів та обмежений розмір словника, що обмежує їхню ефективність у порівнянні з сучаснішими моделями.

Отже для подальшого дослідження будуть використані чотири моделі що показали найвищий результат, а саме: FastText, Mistral 7B, DeBERTa-v3, OpenChat3.5.

5 ПРАКТИЧНЕ ДОСЛІДЖЕННЯ

5.1 Мета дослідження

Метою практичного дослідження є експериментальна перевірка та порівняльний аналіз ефективності мовних моделей, визначених на основі результатів теоретичного аналізу, що наведений у розділі 5.

Завданнями практичного дослідження є:

- підготувати репрезентативний набір текстових даних, що включатиме тексти, створені людьми, а також тексти, згенеровані за допомогою штучного інтелекту;
- розробити методику проведення експерименту з чітким визначенням критеріїв оцінювання та метрик порівняння ефективності моделей;
- провести серію експериментів із залученням обраних моделей;
- виконати детальний аналіз отриманих результатів та сформулювати рекомендації щодо використання моделей у задачах визначення авторства тексту (людина чи ШІ).

Враховуючи висновки, отримані у теоретичному дослідженні (підрозділи 5.2–5.5), для подальшого експерименту обрано моделі FastText, Mistral 7B, DeBERTa-v3, OpenChat-3.5.

5.2 Підходи та інструменти

Для подальшого експерименту обрано моделі, що продемонстрували високі та середні результати за критеріями точності, ефективності роботи та потенціалу застосування у реальних задачах розпізнавання текстів, згенерованих штучним інтелектом, а саме: FastText, Mistral 7B, DeBERTa-v3, OpenChat-3.5.

Загальна схема експерименту:

- використання набору даних, поділеного на тренувальну, валідаційну й тестову вибірки;
- донавчання всіх моделей на тренувальній вибірці з використанням валідаційної для налаштування параметрів;

– тестування донавчених моделей на тестовій вибірці;

– оцінювання результатів за метриками Accuracy, Precision, Recall, F1-score.

Для проведення експериментів обрано Python (версія 3) із такими бібліотеками:

– transformers – робота з моделями, донавчання;

– datasets – завантаження та обробка набору даних;

– peft (LoRA) – ефективне донавчання великих моделей;

– pandas – аналіз результатів.

Середовище виконання – Google Colab із GPU Nvidia A100. Вибір середовища обумовлений необхідністю використання потужних графічних процесорів для швидкого й ефективного донавчання та тестування великих моделей.

5.3 Опис набору даних та підготовка даних

Для проведення експериментів використано набір даних ChatGPT-Research-Abstracts (створений Nicolai Thorer Sivesind), який є підмножиною набору «human-vs-machine» та доступний на платформі Hugging Face [20]. Набір даних містить 20 тисяч текстових записів англійською мовою у форматі «запитання–відповідь», де кожному запитанню (поле «title») відповідають дві текстові анотації («text»): одна, створена людиною (мітка 0 – human-produced); й одна, згенерована штучним інтелектом за допомогою моделі GPT-3.5-turbo-0301 (мітка 1 – machine-generated).

Кількість слів у кожному тексті варіюється від 27 до 594, що є важливим для коректної роботи вибраних моделей, оскільки FastText та DeBERTa-v3 мають обмеження максимальної довжини тексту до 500 токенів.

Набір тестових даних був попередньо розподілений на три вибірки у такому співвідношенні:

– тренувальна вибірка: 15 тисяч зразків;

– валідаційна вибірка: 2 тисячі зразків;

– тестова вибірка: 3 тисячі зразків.

Попередня обробка текстових даних здійснювалася з використанням бібліотек pandas, re та nltk. Її мета — уніфікувати формат тексту, зменшити кількість шумових елементів і покращити здатність моделей виявляти смислові закономірності, що зрештою сприяє підвищенню точності класифікації. Вона включала такі етапи:

– приведення текстів до нижнього регістру;
– видалення зайвих пробілів та перенесень рядків;
– очищення від спеціальних символів;
– видалення стоп-слів.

Нижче наведено код для попередньої обробки:

```
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))

# Function for preprocessing text
def preprocess_text(text):
    # Convert text to lowercase
    text = text.lower()
    # Remove extra spaces and newline characters
    text = re.sub(r"\s+", " ", text)
    # Keep only letters, digits, and basic punctuation marks
    text = re.sub(r"[^a-zA-Z0-9.,!?;:()\-\']", " ", text)
    # Remove stop-words
    text = " ".join([word for word in text.split() if word not in
stop_words])
    return text.strip()
```

Таким чином, було отримано підготовлений і очищений набір текстів для подальших експериментів.

5.4 Метрики оцінювання та критерії порівняння

Для оцінювання якості роботи моделей та забезпечення коректного порівняння результатів було використано кілька основних груп метрик та критеріїв, які дають змогу повноцінно охарактеризувати ефективність розпізнавання текстів, створених штучним інтелектом.

Accuracy (точність класифікації) – загальна частка правильно класифікованих текстів серед усіх прикладів тестової вибірки. Обчислюється за формулою:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5.1)$$

де TP (true positive) – кількість правильно визначених текстів, згенерованих ШІ;

TN (true negative) – кількість правильно визначених текстів, написаних людиною;

FP (false positive) – кількість текстів людини, помилково визначених як штучні;

FN (false negative) – кількість текстів ШІ, помилково визначених як людські.

Precision (точність) – частка правильно визначених штучно згенерованих текстів від загальної кількості текстів, які модель позначила як штучні:

$$Precision = \frac{TP}{TP + FP} \quad (5.2)$$

Recall (повнота) – частка правильно визначених текстів, створених штучним інтелектом, від загальної кількості таких текстів у вибірці:

$$Recall = \frac{TP}{TP + FN} \quad (5.3)$$

F1-score – гармонійне середнє Precision та Recall, що дозволяє оцінити баланс між ними і врахувати ефективність моделі при нерівномірному представленні класів:

$$F1\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (5.4)$$

Також використовувались два додаткових критерія порівняння моделей.

Перший – швидкість роботи, тобто оцінка часу, що витрачається на донавчання кожної моделі та класифікацію текстів на тестовій вибірці. Цей

критерій є важливим для практичного застосування моделей, особливо у разі великих обсягів даних. Другий – обчислювальні ресурси, оцінка обсягу пам’яті необхідної для збереження параметрів моделі та її роботи під час тестування. Це дозволяє визначити вимоги до обчислювальних ресурсів при розгортанні моделей у реальних умовах.

Для оцінювання використовувалась тестова вибірка обсягом 3000 текстів. Додатково проведено експерименти на підвибірках розміром 100, 700 та 1500 прикладів. Це дозволяє перевірити, чи залишаються результати стабільними при зміні кількості тестових даних, а також виключити можливість випадкових коливань метрик, пов’язаних із розподілом складних прикладів у вибірці.

5.5 Практична реалізація мовних моделей

5.5.1 FastText

FastText — це модель, яка навчається з нуля та не потребує попередньо збережених ваг. Вона не вимагає складної підготовки даних, такої як токенізація, нормалізація чи лематизація, що робить її простою у використанні. Завдяки невисоким обчислювальним вимогам FastText зручно використовувати для базової оцінки якості класифікації, особливо коли ресурси обмежені. У цьому дослідженні модель виступає відправною точкою для порівняння з більш потужними трансформерними архітектурами.

Для коректної роботи моделі обов’язковою є структура навчального корпусу у форматі .txt, де кожен рядок містить мітку класу й відповідний текст. Формат мітки задається згідно з вимогами бібліотеки: мітка повинна починатися з префікса __label__, після якого вказується назва класу. Під час проведення дослідження така структура була дотримана. Приклад правильного рядка можна побачити на рисунку 5.1.

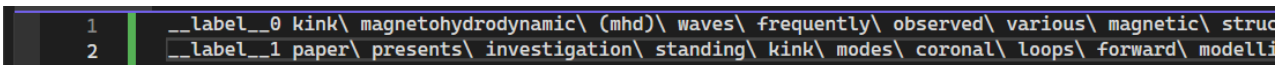

Рисунок 5.1 – Приклад структури тексту, яка необхідна для роботи з FastText (створено самостійно)

Навчання моделі здійснювалось за допомогою функції train_supervised з бібліотеки fasttext. Гіперпараметри були підібрані шляхом сіткового пошуку: для кожної конфігурації обчислювались метрики на валідаційній множині, після чого обирались значення, що давали найкращий баланс між точністю та повнотою. У підсумковій конфігурації було використано 25 епох навчання, розмір векторів 100, коефіцієнт навчання 0.5 та використання біграм:

```
model = fasttext.train_supervised(
    input="FastText_Main/train.txt",
    epoch=25,
    lr=0.5,
    wordNgrams=2,
    dim=100
)
model.save_model("FastText_Main/fasttext_model.bin")
```

Після завершення навчання модель зберігалась у файл формату .bin для подальшого використання.

Оцінювання на валідаційній вибірці виконувалось за допомогою функції test, яка повертала кількість протестованих прикладів, точність (precision) та повноту (recall):

```
result = model.test("FastText/validation.txt")
print(f"Precision: {result[1]}, Recall: {result[2]}")
```

На етапі тестування використовувались два файли: test.txt зі списком текстів та test.csv, що містив справжні мітки у вигляді окремого стовпця. Для кожного тексту обчислювалось передбачення, після чого воно порівнювалось із істинною міткою. Підрахунок правильно класифікованих прикладів дозволив визначити загальну точність моделі.

5.5.2 DeBERTa-V3

Навчання та тестування моделі DeBERTa-v3 проводились з використанням бібліотек TensorFlow, transformers, pandas, NumPy та tqdm.

У реалізації було використано модель microsoft/deberta-v3-base, що підтримує обробку до 512 токенів на вхід. Вхідні тексти попередньо токенізувались за допомогою AutoTokenizer. Було явно встановлено параметр max_length=512, що забезпечувало усічення надлишкових токенів і паддінг коротших послідовностей до фіксованої довжини:

```
train_encodings = tokenizer(
    train_texts,
    truncation=True,
    padding=True,
    max_length=512)
```

Модель використовувалась у форматі TFDebertaV2ForSequenceClassification без модифікації внутрішньої архітектури.

Особливістю реалізації є необхідність перетворення вхідних даних у формат tf.data.Dataset, що дозволяє ефективно керувати пам’яттю при обробці великих обсягів даних та організувати навчання пакетами із заданим розміром:

```
train_dataset = tf.data.Dataset
    .from_tensor_slices((inputs_dict, labels_tensor))
    .shuffle(N)
    .batch(16)
```

Навчання тривало протягом трьох епох із використанням оптимізатора Adam (learning rate = 5e-5) та функції втрат SparseCategoricalCrossentropy:

```
model.compile(
    optimizer=Adam(learning_rate=5e-5),
    loss=SparseCategoricalCrossentropy(from_logits=True),
    metrics=["accuracy"])
```

Після завершення навчання модель і токенізатор були збережені у відповідних директоріях для подальшого використання без необхідності повторної ініціалізації.

На етапі тестування передбачення виконувались поелементно з використанням збереженої моделі. Кожен текст токенізувався окремо, після чого результат класифікації зіставлявся з еталонною міткою. Усі передбачення були

автоматично збережені у текстовому файлі, що містив інформацію про відповідність або розбіжність між очікуваними та отриманими значеннями.

5.5.3 Mistral 7B

У межах практичного дослідження було використано модель Mistral-7B-v0.1, яка є відкрито доступною у репозиторії Hugging Face. Реалізація базувалась на бібліотеці transformers із застосуванням інтерфейсу Trainer, що забезпечує повну інтеграцію з механізмами навчання, перевірки та збереження результатів. Модель була ініціалізована за допомогою класу AutoModelForSequenceClassification, який автоматично підключає необхідні шари для виконання задачі класифікації. На відміну від базового класу AutoModel, ця версія містить додаткову вихідну лінійну голову (classification head), яка застосовується до представлення [CLS]-токена та дозволяє безпосередньо отримувати логіти для передбачення класу:

```
model = AutoModelForSequenceClassification.from_pretrained(
    model_name,
    num_labels=num_labels,
    trust_remote_code=True
)
```

Таким чином, використання AutoModelForSequenceClassification дозволяє уникнути ручного додавання лінійного шару поверх трансформерної архітектури.

Для адаптації моделі до задачі класифікації застосовувався підхід LoRA (Low-Rank Adaptation), який дозволяє ефективно донавчати великі мовні моделі шляхом оновлення лише невеликої кількості параметрів:

```
lora_config = LoraConfig(
    task_type="SEQ_CLS",
    r=32,
    lora_alpha=32,
    lora_dropout=0.1
)
model = get_peft_model(model, lora_config)
```

Оскільки архітектура моделі не містила pad-токена за замовчуванням, для забезпечення сумісності з архітектурою LLaMA-подібних моделей, його значення

було явно встановлено відповідно до кінцевого токена (eos_token). Токенізація застосовувалась до всього корпусу за допомогою методу .map() з подальшим перетворенням до формату torch, що дозволяло використовувати Trainer безпосередньо, без додаткових генераторів.

Токенізація корпусу виконувалась із фіксованою довжиною у 128 токенів. Використовувався векторизований підхід з використанням методу .map() над об’єктом Dataset:

```
def tokenize_function(examples):
    return tokenizer(
        examples['text'],
        truncation=True,
        padding="max_length",
        max_length=128
    )
train_dataset = train_dataset.map(tokenize_function, batched=True)
```

Параметри навчання задавались за допомогою об’єкта TrainingArguments, де було активовано 16-бітну арифметику (fp16) та gradient_checkpointing для оптимізації використання відеопам’яті:

```
training_args = TrainingArguments(
    output_dir="./results",
    num_train_epochs=6,
    per_device_train_batch_size=32,
    evaluation_strategy="epoch",
    fp16=True,
    gradient_checkpointing=True,
    logging_steps=1,
    report_to="none"
)
```

Модель навчалась за допомогою Trainer з автоматичним логуванням і перевіркою після кожної епохи:

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
)
trainer.train()
```

Після завершення навчання модель і токенізатор зберігались локально. Тестування виконувалось на тестовій вибірці із використанням методу predict, який повертав вектор передбачень.

5.5.4 OpenChat 3.5

У рамках практичного дослідження була використана модель openchat_3.5, доступна у відкритому репозиторії Hugging Face. OpenChat 3.5 належить до класу автогенеративних мовних моделей, вона не має вбудованої класифікаційної голови. Для її адаптації до задачі двокласової класифікації було реалізовано власну обгортку на основі nn.Module, яка включає лінійний класифікаційний шар і виклик функції втрат. У якості вхідного представлення для класифікатора використовувалось останнє приховане представлення послідовності, що відповідало останньому токену:

```
class OpenChatClassifier(nn.Module):
    def __init__(self, model_name, num_labels):
        super().__init__()
        bnb_config = BitsAndBytesConfig(
            load_in_4bit=True,
            bnb_4bit_compute_dtype=torch.float16,
            bnb_4bit_use_double_quant=True,
            bnb_4bit_quant_type="nf4",
        )
        self.base = AutoModelForCausalLM.from_pretrained(
            model_name,
            trust_remote_code=True,
            quantization_config=bnb_config,
            device_map="auto"
        )
        self.base = prepare_model_for_kbit_training(self.base)
        self.dropout = nn.Dropout(0.1)
        self.classifier = nn.Linear(
            self.base.config.hidden_size, num_labels)

    def forward(self, input_ids=None,
        attention_mask=None, labels=None):
        outputs = self.base.model(
            input_ids=input_ids,
            attention_mask=attention_mask,
            output_hidden_states=True,
            return_dict=True
        )
        hidden = outputs.hidden_states[-1]
        cls_token = hidden[:, -1, :]
        logits = self.classifier(self.dropout(cls_token))
        loss = None
```

```
        if labels is not None:
            loss = nn.CrossEntropyLoss()(
                logits.view(-1, logits.size(-1)), labels.view(-1))

        return {"loss": loss, "logits": logits}
```

Базова модель завантажувалась у 4-бітному форматі, що дозволяло зменшити споживання пам’яті та пришвидшити тренування. Перед донавчанням виконувався виклик функції prepare_model_for_kbit_training.

Як і в попередніх експериментах, було застосовано донавчання з використанням LoRA. У конфігурації, окрім основних параметрів, таких як r, alpha і dropout, додатково зазначались конкретні цільові модулі:

```
lora_config = LoraConfig(
    task_type="SEQ_CLS",
    r=8,
    lora_alpha=32,
    lora_dropout=0.1,
    target_modules=["q_proj", "v_proj"]
)
model.base = get_peft_model(model.base, lora_config)
```

Для токенізації використовувався стандартний підхід із паддінгом до 128 токенів. Навчання здійснювалось з використанням Trainer, параметри конфігурації задавались через TrainingArguments:

```
training_args = TrainingArguments(
    output_dir="./results",
    num_train_epochs=3,
    per_device_train_batch_size=256,
    per_device_eval_batch_size=256,
    evaluation_strategy="epoch",
    save_strategy="epoch",
    logging_dir='./logs',
    logging_steps=1,
    learning_rate=2e-5,
    weight_decay=0.01,
    fp16=True,
    load_best_model_at_end=True,
    report_to="none"
)
```

Після завершення навчання модель тестувалась на окремому наборі даних, а передбачення записувались у таблицю.

На відміну від реалізації з використанням Mistral, дана модель вимагала створення власної архітектури класифікації, ручного управління обчисленням втрат і адаптації генеративної моделі під класифікаційну задачу. Також застосовувалась квантизація, яка не використовувалась у попередніх експериментах. Інші етапи, зокрема організація навчального процесу та токенізація, відповідали структурі, реалізованій у попередніх підрозділах.

5.6 Аналіз результатів експерименту

Для кожної з досліджуваних моделей було отримано значення основних метрик якості класифікації на тестовій вибірці. У таблиці 5.1 наведено показники точності (Accuracy) та F1-міри.

Під час експерименту було виявлено, що значення Accuracy та F1-score практично збігаються для всіх моделей і вибірок. Це пояснюється збалансованістю класів та високою точністю класифікації.

Таблиця 5.1 - Accuracy та F1-score (macro avg) для різних обсягів вибірки

Модель	Обсяг 100	Обсяг 700	Обсяг 1500	Обсяг 3000
FastText	0.99	0.9829	0.9860	0.9807
DeBERTa-v3	0.9900	0.9900	0.9893	0.9887
Mistral-7B	0.8800	0.8357	0.8260	0.8270
OpenChat 3.5	0.9700	0.9743	0.9807	0.9800

Значення точності (Accuracy) та F1-міри залишаються стабільними для моделей DeBERTa-v3 і FastText у межах від 0.98 до 0.99 незалежно від обсягу тестової вибірки. Для моделі Mistral-7B спостерігається зниження точності зі збільшенням обсягу тестової вибірки: з 0.88 (на 100 прикладах) до 0.827 (на 3000 прикладах). Падіння становить 5.3 процентних пункти, що є найгіршим показником серед розглянутих моделей. Модель OpenChat 3.5 демонструє точність на рівні 0.97–0.98 при всіх обсягах вибірки, з максимальним значенням 0.9807 (на 1500 прикладах). Різниця між мінімальним і максимальним значенням становить лише 0.0107, що свідчить про стабільну ефективність.

Графік залежності точності класифікації від обсягу вибірки можна побачити на рисунку 5.2.

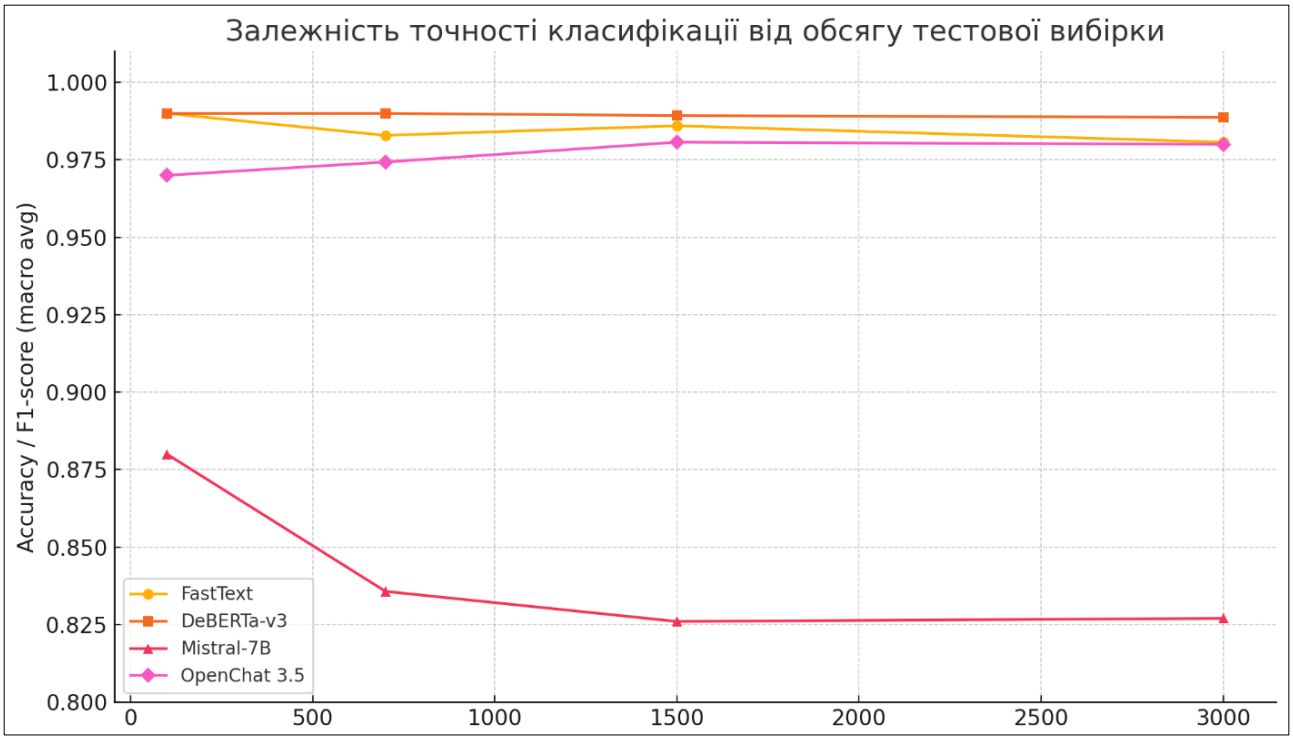

Рисунок 5.2 - Графік залежності точності класифікації від обсягу вибірки (створено самостійно)

У таблиці 5.2 наведено показники Precision та Recall отримані для 3000 прикладах.

Таблиця 5.2 - Precision та Recall по класах

Модель	Клас	Precision	Recall
FastText	0 human-produced	0.98	0.9813
	1 machine-generated	0.9813	0.98
DeBERTa-v3	0 human-produced	0.9887	0.9887
	1 machine-generated	0.9887	0.9887
Mistral-7B	0 human-produced	0.8187	0.84
	1 machine-generated	0.8357	0.814
OpenChat 3.5	0 human-produced	0.9762	0.984
	1 machine-generated	0.9839	0.976

Модель DeBERTa-v3 показала найкращі результати за всіма показниками: Precision = 0.9887 та Recall = 0.9887 для обох класів. FastText посіла друге місце: середні значення Precision ≈ 0.9806, Recall ≈ 0.9807. OpenChat 3.5 продемонструвала подібний рівень: Precision ≈ 0.98, Recall ≈ 0.98. Mistral-7B суттєво поступається іншим моделям за всіма метриками.

Для більшої наглядності була побудована діаграма, яку можна побачити на рисунку 5.3.

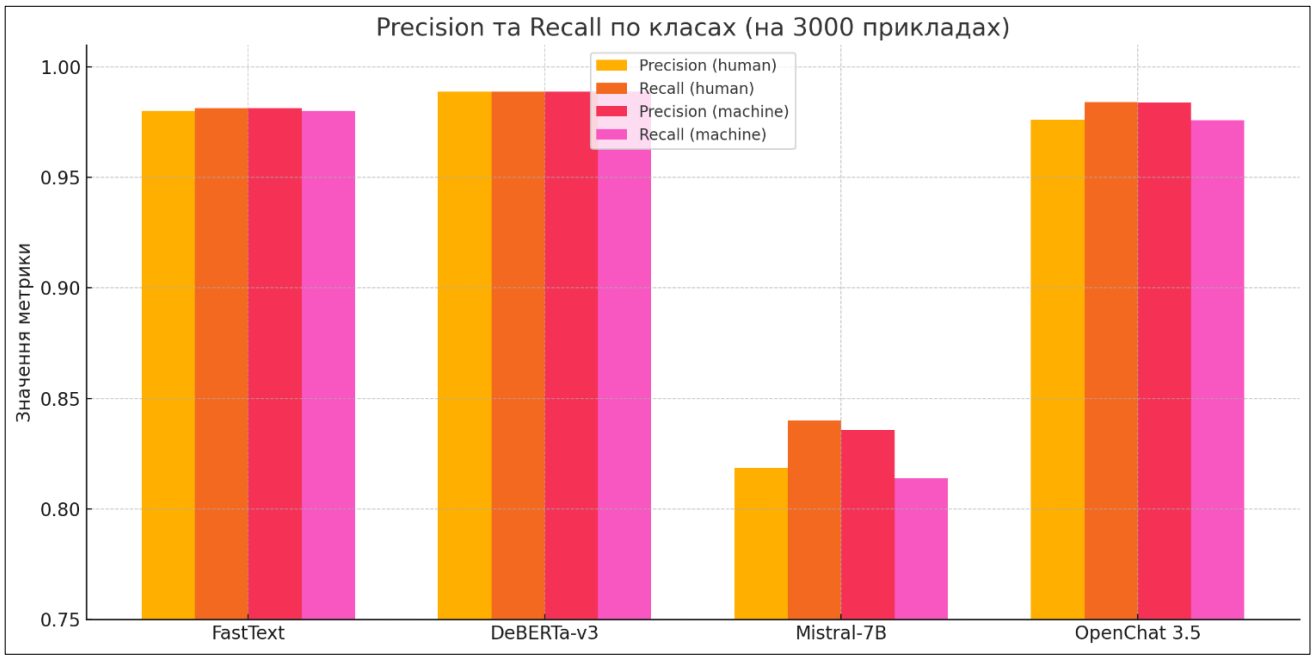

Рисунок 5.3 – Precision та Recall по класах, на 3000 прикладах (створено самостійно)

На основі результатів побудовано матриці помилок для кожної моделі, що дозволило дослідити характер неправильних класифікацій. Матриці можна побачити у таблицях 5.3-5.6

Таблиця 5.3 – Матриця помилок FastText (на 3000 прикладах)

	Передбачено: 0	Передбачено: 1
Фактично: 0	1472	28
Фактично: 1	30	1470

Таблиця 5.4 – Матриця помилок DeBERTa-v3 (на 3000 прикладах)

	Передбачено: 0	Передбачено: 1
Фактично: 0	1483	17
Фактично: 1	17	1483

Таблиця 5.5 – Матриця помилок Mistral-7B (на 3000 прикладах)

	Передбачено: 0	Передбачено: 1
Фактично: 0	1260	240
Фактично: 1	279	1221

Таблиця 5.6 – Матриця помилок OpenChat 3.5 (на 3000 прикладах)

	Передбачено: 0	Передбачено: 1
Фактично: 0	1476	24
Фактично: 1	36	1464

Аналіз матриць помилок показує, що найменшу кількість неправильних класифікацій допустила модель DeBERTa-v3 — лише 34 випадки з 3000 прикладів. Помилки рівномірно розподілені між класами, по 17 на кожен. Модель FastText демонструє порівнянну якість: 58 помилок, з яких 30 — у визначенні штучного тексту як людського, і 28 — навпаки. Модель OpenChat 3.5 має 60 помилок, розподіл також близький до рівномірного: 36 і 24 відповідно.

Найбільшу кількість помилок допустила модель Mistral-7B — 519 випадків, з них 279 при класифікації штучного тексту і 240 — людського. Це свідчить про істотне зниження здатності моделі до точного розрізнення класів, порівняно з іншими моделями.

Незважаючи на те, що окремі моделі демонструють високу точність, типові помилки не є спільними для всіх моделей. Це свідчить про те, що складність класифікації тексту значною мірою залежить від особливостей конкретної архітектури або навчальної парадигми, а не від об'єктивної складності тексту. Показово, що серед усіх прикладів лише чотири тексти були класифіковані

неправильно одночасно всіма чотирма моделями. Аналіз саме цих випадків має особливу цінність для виявлення потенційних обмежень навчального корпусу або архітектурної чутливості.

Текст 1 (індекс 689): "The Breuil-Mezard conjecture concerning the modularity of two-dimensional Galois representations over p-adic fields has been proven in the case of elliptic curves. We establish the validity of the conjecture for quaternion algebras...".

Текст 2 (індекс 1295): "We study the Technicolor (TC) contribution to lepton plus photon plus missing energy final states at the Tevatron in light of the recent CDF and D0 results. We find that TC models predict much smaller cross sections in this channel in comparison with the standard model…".

Текст 3 (індекс 2269): "The increasing frequency and sophistication of cyber attacks call for more advanced and realistic models to assess the likelihood of successful attacks and design effective defense mechanisms. In this paper, we introduce …".

Текст 4 (індекс 2947): "We establish a vanishing theorem for the direct image sheaves r i f omega x j(log d) on a compact k ahler manifold x , where f:x rightarrow y is a surjective holomorphic map to...".

Хоча ці тексти мали мітку штучного походження, всі моделі класифікували їх як людські. Такий збіг результатів свідчить про наявність спільних ознак, які вводять моделі в оману та сприяють помилковій інтерпретації.

Проаналізовані тексти характеризуються академічною структурою викладу, високою щільністю спеціалізованої термінології та чіткою логічною побудовою, що є типовими рисами наукового стилю. Один із прикладів присвячено доведенню гіпотези Брейля—Мезара у контексті двовимірних Ґалуаських зображень над p-адичними полями. Інший текст описує вплив теорії техніколору на результати експериментів у прискорювачі Tevatron і структурно нагадує типову статтю з фізики. Ще один приклад стосується моделювання кібератак у багатокористувацькому ігровому середовищі з багаторівневою аргументацією, формалізованим описом параметрів та чисельними результатами.

Попри те, що всі ці тексти були згенеровані мовною моделлю, вони не містили характерних ознак машинної генерації, зокрема повторів, стилістичної

шаблонності чи змістовної спрощеності. Високий рівень точності формулювань, коректне використання термінології і дотримання академічних норм, ймовірно, стали чинниками, що зумовили їхню хибну класифікацію.

Виходить, що навіть за умови збалансованої вибірки сучасні архітектури не завжди здатні коректно розпізнавати штучно згенеровані тексти, якщо останні формально відповідають структурі та стилю академічного письма. Це підкреслює наявні обмеження моделей при роботі з текстами, які дотримуються типових для академічного письма структурних і стилістичних норм — незалежно від того, чи написані вони людиною, чи згенеровані ШІ.

Також було проведено оцінку часових витрат і споживання пам’яті під час тестування моделей, результати якої подано в таблиці 5.7.

Таблиця 5.7 - Часові витрати і споживання пам’яті під час тестування моделей

Модель	Середній час тестування (3000 текстів)	Мінімальний обсяг відеопам’яті
FastText	< 1 с	0 гб
DeBERTa-v3	5 хв 40 с	4 гб
Mistral-7B	38 хв	18 гб
OpenChat 3.5	29 хв	16 гб

Великий обсяг пам’яті, необхідний для моделей Mistral-7B та OpenChat 3.5, пояснюється їхньою архітектурою та кількістю параметрів. Обидві моделі належать до класу великих мовних моделей (LLM), які мають мільярди параметрів і потребують значних ресурсів для зберігання, активації шарів і обробки тексту в режимі inference.

На відміну від них, FastText — це легка нейронна модель, яка представляє слова через символьні n-грамні вектори та не використовує трансформерну архітектуру. Завдяки цьому вона має мінімальні вимоги до оперативної пам’яті та може ефективно працювати навіть на обмежених апаратних ресурсах.

DeBERTa-v3, хоча і є трансформерною моделлю, потребує меншу кількість ресурсів порівняно з LLM через меншу кількість параметрів.

5.7 Підсумок практичного дослідження

У ході дослідження було встановлено, що найкращі результати класифікації показала модель DeBERTa-v3. Високу точність також продемонструвала FastText, яка при цьому майже не потребує ресурсів. OpenChat 3.5 показала схожі результати, але потребує більше часу та пам’яті для роботи. Найгірші результати спостерігалися у Mistral-7B — як за точністю, так і за необхідними ресурсами.

У процесі експерименту було зафіксовано суттєві відмінності у швидкості тестування й споживанні пам’яті. FastText завершила класифікацію 3000 текстів менш ніж за секунду без використання GPU. DeBERTa-v3 потребувала близько шести хвилин і 4 ГБ відеопам’яті.

Для OpenChat 3.5 і Mistral-7B час становив 29 і 38 хвилин відповідно, а мінімальні вимоги до пам’яті — понад 16 ГБ. Це обумовлено їхньою архітектурою: обидві є великими мовними моделями з мільярдами параметрів.

Попри свою масштабність, LLM не забезпечили переваги у якості класифікації.

Задача розпізнавання штучно згенерованого тексту не вимагає глибокого контекстуального розуміння або складної генерації мови, а натомість зводиться до аналізу стилістичних, лексичних і структурних особливостей. Такі ознаки ефективно фіксуються навіть компактними моделями. Крім того, великі моделі чутливі до довжини тексту, часто застосовують механізми “семплінгу” під час генерації та мають тенденцію до "гладких" відповідей, що не завжди корелює з потребами у точній класифікації. З огляду на це, використання LLM для таких задач є надмірним як у плані обчислювальних витрат, так і з огляду на фактичну продуктивність.

Для практичного застосування найбільш доцільним є використання FastText, якщо пріоритетом є швидкість і мінімальні ресурси, або DeBERTa-v3 — у випадках, коли важлива максимальна точність.

Великі мовні моделі доцільно використовувати для генерації тексту або аналізу відкритих мовних даних, де необхідне глибоке контекстуальне опрацювання.

У подальших дослідженнях варто розширити рамки експерименту й порівняти найкращі мовні моделі не лише між собою, а й з іншими підходами до класифікації текстів — наприклад, із використанням стилометричних ознак, синтаксичного аналізу чи методів машинного навчання на рівні структурних патернів. Це дозволить більш точно визначити оптимальні методи для розв’язання задачі розпізнавання штучного походження текстів.

Крім того у подальших дослідженнях варто також розглянути можливість генерації власного датасету, який би враховував специфіку завдання розпізнавання AI-згенерованих текстів. Це дало б змогу гнучко керувати тематикою, стилістикою та обсягом текстів, а також експериментувати з різними типами мовних моделей і рівнем обробки згенерованого контенту. Такий підхід дозволив би краще адаптувати методику до реальних сценаріїв застосування й отримати більш надійні результати.

ВИСНОВКИ

У межах цієї роботи був проведений аналіз сучасних мовних моделей, які можуть бути використані для розпізнавання текстів, створених штучним інтелектом. Розглянуті моделі було поділено на чотири основні типи — статистичні, нейронні, попередньо навчені та великі мовні моделі.

Теоретичне дослідження, що включало побудову векторної моделі на основі визначених критеріїв, засвідчило високий потенціал великих мовних моделей (LLM), таких як Mistral 7B та OpenChat-3.5. Їх архітектура дозволяє ефективно працювати з довгими контекстами та складними лінгвістичними залежностями, що формально виводить їх на перші позиції за можливостями.

Результати практичних експериментів показали, що моделі середнього масштабу — зокрема Deberta та FastText — у задачі розпізнавання AI-згенерованих текстів за стандартними метриками (точність, повнота, F1-міра, перплексія) стабільно демонстрували вищу або порівнянну якість класифікації при значно нижчих вимогах до ресурсів. Особливо добре себе зарекомендували Deberta а також FastText, що показали високу швидкість та ефективність. Це свідчить, що ефективність моделей залежить від умов використання, а більша кількість параметрів або складніша архітектура не гарантують кращих результатів.

На основі аналізу сформульовано рекомендації для для практичного застосування: використовувати Deberta коли пріоритет — висока точність, а FastText — для задач, де важлива швидкодія при мінімальних витратах ресурсів. Великі моделі доцільні лише за наявності відповідної інфраструктури й потреби працювати з довгими або складними текстами.

Таким чином, дослідження дозволило не лише визначити найрезультативніші моделі для задачі виявлення AI-згенерованих текстів, але й надати конкретні рекомендації для їх впровадження в реальні системи. Робота має як теоретичну, так і практичну цінність, і може слугувати основою для подальших досліджень у сфері виявлення штучно створеного контенту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Zhao, Y., Yang, J., Li, Z., & Wang, T. "A Survey of Large Language Models". / Journal of Artificial Intelligence Research, 2024. 25 с.

2. Liang, W., Zhang, Y., Wu, Z., et al. "Mapping the Increasing Use of LLMs in Scientific Papers". / Preprint, Stanford University, 2024. 25 с.

3. Chakraborty, S., Bedi, A. S., Zhu, S., An, B., Manocha, D., & Huang, F. "On the Possibilities of AI-Generated Text Detection". / Proceedings of the University of Maryland, 2024. 27 с.

4. Irina Tolstykh, Aleksandra Tsybina, Sergey Yakubson, Aleksandr Gordeev, Vladimir Dokholyan, Maksim Kuprashevich. "GigaCheck: Detecting LLM-generated Content". / SaluteDevices R&D Department, 2024. 30 с.

5. Fraser, K. C., Dawkins, H., & Kiritchenko, S. "Detecting AI-Generated Text: Factors Influencing Detectability with Current Methods". / National Research Council Canada, 2024. 45 с.

6. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. "Attention Is All You Need". / Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017). Long Beach, CA, USA, 2017. 15 с.

7. Сокорчук І.П. "Автоматизація повсякденних завдань викладачів за допомогою засобів штучного інтелекту: покращення робочого процесу та ефективності". // Матеріали конференції "Використання нових методів навчання у видавничо-поліграфічній галузі", ХНУРЕ, 2024, С. 330–331.

8. Каук В.І. "Генеративний штучний інтелект – креативний помічник дизайнера". // Матеріали конференції "Використання нових методів навчання у видавничо-поліграфічній галузі", ХНУРЕ, 2024, С. 283–294.

9. Smelyakov K., Kitsenko Y., Chupryna A. Deepfake Detection Models Based on Machine Learning Technologies. 2024 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream), Vilnius, Lithuania, 25 April 2024. 2024. URL: https://doi.org/10.1109/estream61684.2024.10542582 (date of access: 02.06.2025).

10. OpenAI. "GPT-4 Technical Report". OpenAI, 2024. URL: https://openai.com/research/gpt-4 (дата звернення: 25.12.2024).

11. Get started fastText. FastText. URL: https://fasttext.cc/docs/en/support.html (дата звернення: 01.04.2025).

12. Joint embedding VQA model based on dynamic word vector / Z. Ma та ін. PeerJ Computer Science. 2021. T. 7. C. e353. URL: https://doi.org/10.7717/peerj-cs.353 (дата звернення: 02.04.2025).

13. Hugging Face. BERT. URL: https://huggingface.co/docs/transformers/model_doc/bert (дата звернення: 01.04.2025).

14. Hugging Face. RoBERTa. URL: https://huggingface.co/docs/transformers/model_doc/roberta (дата звернення: 01.04.2025).

15. Hugging Face. URL: https://huggingface.co/microsoft/deberta-v3-base (дата звернення: 01.04.2025).

16. Phi 2 By microsoft: Benchmarks and Detailed Analysis. Insights on Phi 2. LLM Explorer. URL: https://llm.extractum.io/model/microsoft/phi-2,2YnPCnrDwMd2lY8uzQpoNk (дата звернення: 01.04.2025).

17. Orion 14B Base By orionstarai: Benchmarks and Detailed Analysis. Insights on Orion 14B Base. LLM Explorer. URL: https://llm.extractum.io/model/OrionStarAI/Orion-14B-Base,1WYDtkXPi4XtAGK73btC8P (дата звернення: 01.04.2025).

18. Mistral 7B Instruct V0.1 By mistralai: Benchmarks and Detailed Analysis. Insights on Mistral 7B Instruct V0.1. LLM Explorer. URL: https://llm.extractum.io/model/mistralai/Mistral-7B-Instruct-v0.1,3xB5LFPUJ46OqZY2vM0u82 (дата звернення: 01.04.2025).

19. Openchat 3.5 0106 By openchat: Benchmarks and Detailed Analysis. Insights on Openchat 3.5 0106. LLM Explorer. URL: https://llm.extractum.io/model/openchat/openchat-3.5-0106,3VG4c0r6tUXpfgI4qQPlaU (дата звернення: 01.04.2025).

20. Hugging Face. URL: https://huggingface.co/datasets/NicolaiSivesind/human-vs-machine (дата звернення: 01.04.2025).

21. Diploma-Oleksandra-Tkachenko URL: https://github.com/alexandratk/Diploma-Oleksandra-Tkachenko# (дата звернення: 03.06.2025)

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ ЗА НАУКОВИМИ НАПРЯМАМИ КЕРІВНИКА ТА НАУКОВЦІВ КАФЕДРИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

7. Сокорчук І.П. "Автоматизація повсякденних завдань викладачів за допомогою засобів штучного інтелекту: покращення робочого процесу та ефективності". // Матеріали конференції "Використання нових методів навчання у видавничо-поліграфічній галузі", ХНУРЕ, 2024, С. 330–331.

8. Каук В.І. "Генеративний штучний інтелект – креативний помічник дизайнера". // Матеріали конференції "Використання нових методів навчання у видавничо-поліграфічній галузі", ХНУРЕ, 2024, С. 283–294.

9. Smelyakov K., Kitsenko Y., Chupryna A. Deepfake Detection Models Based on Machine Learning Technologies. 2024 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream), Vilnius, Lithuania, 25 April 2024. 2024. URL: https://doi.org/10.1109/estream61684.2024.10542582 (date of access: 02.06.2025).