Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет інформаційної безпеки та електронних комунікацій
(повне найменування факультету)

Кафедра інформаційної безпеки та наноелектроніки
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
магістра
(ступінь вищої освіти)

на тему Дослідження впливу генеративних мовних моделей на створення фішингових повідомлень та розробка системи їх детектування
(назва теми)

Виконав студент 2 курсу, групи БК-714м
Спеціальності 125 Кібербезпека та захист інформації
(код і найменування спеціальності)
Освітня програма (спеціалізація)
Системи технічного захисту інформації, автоматизація її обробки
МАРЮХА В.Г.
(ПРІЗВИЩЕ та ініціали)

Керівник КОРОЛЬКОВ Р. Ю.
(ПРІЗВИЩЕ та ініціали)

Рецензент ЛИТВИЦЬКИЙ О. П.
(ПРІЗВИЩЕ та ініціали)

2025

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет інформаційної безпеки та електронних комунікацій
Кафедра інформаційної безпеки та наноелектроніки
Ступінь вищої освіти магістр
Спеціальність 125 Кібербезпека та захист інформації
(код і найменування)
Освітня програма (спеціалізація) Системи технічного захисту інформації, автоматизація її обробки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
Завідувач кафедри ІБтаН
Андрій КОРОТУН
«____» _____ 2025року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

МАРЮХИ Віталія Григоровича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження впливу генеративних мовних моделей на створення фішингових повідомлень та розробка системи їх детектування
Research on the impact of generative language models on the creation of phishing messages and the development of a system for their detection
керівник проєкту (роботи) к.т.н., доцент КОРОЛЬКОВ Роман Юрійович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «26»листопада 2025 року № 530

2. Строк подання студентом проєкту (роботи) 19.12.2025

3. Вихідні дані до проєкту (роботи) наукові та нормативні джерела з фішингу, соцінженерії й генеративних мовних моделей; ML/DL інструменти (Python, PyTorch, Hugging Face Transformers); попередньо навчені трансформери

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) огляд і систематизація джерел щодо ГММ та фішингу; проєктування й реалізація прототипу; експериментальне оцінювання за метриками якості та аналіз результатів; висновки й практичні рекомендації. .

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) Презентація доповіді (в MS PowerPoint), 12 слайдів.

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1 – 3	КОРОЛЬКОВ Р. Ю., доцент кафедри ІБтаН	04.09.25	16.12.2025
Нормоконтроль	КОРОЛЬКОВ Р. Ю., доцент кафедри ІБтаН		19.12.2025

7. Дата видачі завдання «04» вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз літературних джерел за тематикою дослідження.	04.09.25–18.09.25	Виконано
2	Формулювання мети, завдань, об’єкта та предмета дослідження, уточнення структури магістерської роботи.	19.09.25–30.09.25	Виконано
3	Аналіз фішингових атак та методів їх детектування, обґрунтування вибору архітектури системи.	01.10.25–15.10.25	Виконано
4	Проєктування та реалізація програмного прототипу системи детектування фішингових повідомлень.	16.10.25–15.11.25	Виконано
5	Проведення експериментальних досліджень і збір результатів.	16.11.25–30.11.25	Виконано
6	Аналіз та узагальнення результатів експериментів, формування висновків і рекомендацій.	01.12.25–15.12.25	Виконано
7	Оформлення матеріалів магістерської роботи.	16.12.25–19.12.25	Виконано

Студент(ка) _____ Віталій МАРЮХА
(підпис) (Ім’я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Роман КОРОЛЬКОВ
(підпис) (Ім’я ПРІЗВИЩЕ)

АНОТАЦІЯ

Пояснювальна записка до магістерської роботи: 71 с., 5 табл., 13 рис., 2 дод., 32 джерела.

ГЕНЕРАТИВНІ МОВНІ МОДЕЛІ, КЛАСИФІКАЦІЯ ТЕКСТІВ, ТРАНСФОРМЕРИ, ФІШИНГ, DISTILBERT, HUGGING FACE, LLM, LLM-AWARE FINE-TUNING, PYTORCH, ROC-AUC.

Актуальність теми. Фішинг і соціальна інженерія залишаються одними з найбільш суттєвих кіберзагроз для організацій, а поява генеративних мовних моделей підвищує масштабованість і переконливість текстових атак, ускладнюючи їх виявлення традиційними засобами.

Об'єктом дослідження є генеративні мовні моделі та їх роль у створенні фішингових повідомлень, використання методів машинного навчання та глибокого навчання у виявленні фішингових повідомлень. У дипломному проєкті розглянуті такі питання, як структура та функціонування генеративних мовних моделей (LLM), характеристики фішингових повідомлень та тенденції їхнього розвитку, а також недоліки існуючих методів запобігання.

Предметом дослідження є методи аналізу, класифікації та детектування фішингових повідомлень, згенерованих за допомогою генеративних мовних моделей, оцінка ефективності алгоритмів детектування та розробка практичної системи для виявлення таких атак, включно з тестуванням, масштабуванням та інтеграцією в корпоративну інфраструктуру. Предмет дослідження охоплює як теоретичний аналіз характеристик текстів, так і практичну реалізацію моделі, що дозволяє підвищити точність і надійність систем захисту.

Метою дипломного проєкту є дослідження впливу генеративних мовних моделей на формування фішингових повідомлень та розробка ефективної системи їхнього детектування. Для досягнення мети передбачено: аналіз архітектури генеративних мовних моделей (LLM) і принципів їх роботи; класифікацію та характеристику сучасних фішингових атак; огляд та порівняння традиційних і сучасних методів детектування; проєктування, реалізацію та тестування системи детектування; оцінку ефективності та можливостей масштабування і інтеграції розробленого рішення.

Практична цінність роботи полягає у створенні відтворюваного програмного прототипу детектора фішингових повідомлень та методики перевірки його стійкості до текстів, сформованих генеративними мовними моделями, що може бути використано для подальшого вдосконалення систем захисту корпоративної електронної пошти.

ABSTRACT

Explanatory note to the master's thesis: 71 pp., 5 tabs., 13 figs., 2 app., 32 refs.

GENERATIVE LANGUAGE MODELS, TEXT CLASSIFICATION, TRANSFORMERS, PHISHING, DISTILBERT, HUGGING FACE, LLM, LLM-AWARE FINE-TUNING, PYTORCH, ROC-AUC.

Relevance of the topic. Phishing and social engineering remain among the most significant cyber threats to organizations. The emergence of generative language models has increased the scalability and persuasiveness of text-based attacks, thereby complicating their detection by traditional security mechanisms.

Object of the research. The object of the study is generative language models and their role in creating phishing messages, the use of machine learning and deep learning methods in detecting phishing messages. The thesis project examines issues such as the structure and functioning of generative language models (LLMs), the characteristics of phishing messages and trends in their development, as well as the shortcomings of existing prevention methods.

Subject of the research. The subject of the research comprises methods for the analysis, classification, and detection of phishing messages generated using generative language models, the evaluation of detection algorithm effectiveness, and the development of a practical system for identifying such attacks, including testing, scalability assessment, and integration into corporate infrastructure. The subject area encompasses both the theoretical analysis of textual characteristics and the practical implementation of models, enabling improved accuracy and robustness of protection systems.

Purpose of the study. The aim of the thesis is to investigate the impact of generative language models on the formation of phishing messages and to develop an

effective system for their detection. To achieve this aim, the study involves: analyzing the architecture and operating principles of generative language models (LLMs); classifying and characterizing contemporary phishing attacks; reviewing and comparing traditional and modern detection methods; designing, implementing, and testing a phishing detection system; and evaluating the effectiveness, scalability, and integration potential of the proposed solution.

Practical significance. The practical value of the work lies in the development of a reproducible software prototype for phishing message detection and a methodology for assessing its robustness against texts generated by generative language models, which can be used to further enhance corporate email security systems.

ПЕРЕЛІК СКОРОЧЕНЬ

ГММ — генеративні мовні моделі;

BEC — Business Email Compromise (компрометація ділової електронної пошти);

BERT — Bidirectional Encoder Representations from Transformers (двонапрямні енкодерні подання з трансформерів);

DL — Deep Learning (глибоке навчання);

HTML — HyperText Markup Language (мова гіпертекстової розмітки);

LLM — Large Language Models (великі мовні моделі);

ML — Machine Learning (машинне навчання);

NLP — Natural Language Processing (обробка природної мови);

ROC-AUC — Area Under the ROC Curve (площа під ROC-кривою);

SMS — Short Message Service (служба коротких повідомлень);

TF–IDF — Term Frequency–Inverse Document Frequency (частота терміна – обернена частота документа);

URL — Uniform Resource Locator (уніфікований покажчик ресурсу).

ЗМІСТ

ВСТУП

Фішингові атаки є однією з найбільш поширених загроз інформаційній безпеці сучасних інформаційно-комунікаційних систем. Вони широко застосовуються зловмисниками для отримання конфіденційної інформації, компрометації облікових записів користувачів, здійснення несанкціонованих фінансових операцій та подальшого розвитку складніших багатоступеневих атак. Незважаючи на постійне вдосконалення технічних і організаційних заходів захисту, фішинг залишається ефективним інструментом атак через активне використання соціальної інженерії, експлуатацію довіри користувачів та швидку адаптацію сценаріїв атак до нових умов. Згідно з аналітичними звітами ENISA, фішинг і соціальна інженерія стабільно входять до переліку найбільш значущих кіберзагроз для державних установ, фінансових організацій і корпоративних інформаційних систем у країнах Європейського Союзу.

Сучасні фішингові кампанії характеризуються поєднанням технічних і психологічних механізмів впливу, використанням різноманітних каналів доставки повідомлень та багатоканальних сценаріїв взаємодії з потенційними жертвами. Успішність таких атак значною мірою зумовлена можливістю імітації легітимної комунікації, використанням правдоподібних формулювань і контекстів, а також залученням інформації з відкритих джерел і попередніх витоків даних. Огляди ENISA та рекомендації CISA свідчать, що саме поєднання технічних засобів із соціально-психологічними компонентами є ключовим фактором ефективності сучасних фішингових атак. Крім того, складність розпізнавання фішингових повідомлень користувачами зумовлює необхідність кількісної оцінки ризику людської помилки, для чого в наукових і прикладних дослідженнях застосовується підхід NIST Phish Scale.

Суттєвий вплив на еволюцію фішингових атак має стрімкий розвиток генеративних мовних моделей. Сучасні великі мовні моделі здатні автоматично формувати граматично коректні, стилістично природні та контекстно релевантні

тексти, що значно підвищує переконливість фішингових повідомлень. Використання таких моделей дозволяє масштабувати фішингові кампанії, адаптувати зміст повідомлень до конкретних адресатів і зменшувати кількість очевидних ознак підробки, на які орієнтуються традиційні антифішингові засоби. У результаті ефективність методів, заснованих на чорних і білих списках, статичних евристичних правилах або простому контент-аналізі, суттєво знижується в умовах появи LLM-згенерованих текстів.

У зв’язку з цим у сучасних дослідженнях зростає інтерес до застосування методів машинного та глибокого навчання для детектування фішингових повідомлень. Особливу увагу приділено трансформерним архітектурам, які завдяки механізмам самоуваги здатні враховувати семантичні та контекстні залежності в тексті та виявляти приховані шаблони, характерні для фішингових повідомлень. Використання таких моделей відкриває можливості для підвищення стійкості систем детектування до нових сценаріїв атак, зокрема до повідомлень, сформованих або модифікованих генеративними мовними моделями.

Таким чином, дослідження впливу генеративних мовних моделей на формування фішингових повідомлень та розробка ефективних методів їх детектування є актуальним науково-практичним завданням у галузі кібербезпеки. Розв’язання цієї проблеми сприяє підвищенню рівня захищеності корпоративних поштових систем і інформаційних ресурсів, а також зменшенню ризиків, пов’язаних із людським фактором у процесі обробки електронних повідомлень.

1 ТЕОРЕТИЧНІ ОСНОВИ ГЕНЕРАТИВНИХ МОВНИХ МОДЕЛЕЙ ТА ЕВОЛЮЦІЯ ФІШИНГОВИХ АТАК

1.1 Архітектура та принципи функціонування генеративних мовних моделей

Генеративні мовні моделі — це системи нейронних мереж, які використовують статистичне прогнозування для створення текстових послідовностей. Ці моделі побудовані на архітектурі трансформера, яка замінила рекурентні мережі в сучасних методах, оскільки вона може обробляти довгі залежності та одночасно обробляти дані. Відмовившись від послідовної обробки тексту, вдалося поліпшити властивості узагальнення моделей і зробити навчання більш продуктивним [1].

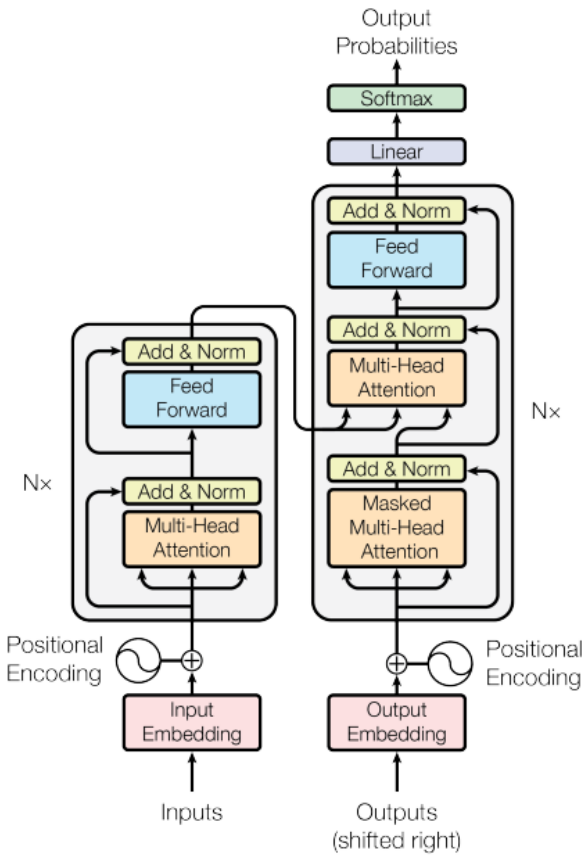

Рисунок 1.1 – Архітектурна модель трансформера [1]

Механізм самоуваги є найважливішою частиною трансформера. Він дозволяє визначити, які частини вхідної послідовності мають найбільший вплив на поточний токен. Для кожного токена створюються три вектори: запит, ключ та значення.

Їх взаємодія забезпечує перерозподіл інформації між елементами послідовності з урахуванням їхньої ваги без будь-яких проміжних рекурентних станів. Це означає, що модель може зберігати структурні залежності між словами на будь-якій відстані без втрати контексту [1, 2].

Трансформери мають здатність розбивати та аналізувати вхідні дані на різних, більш детальних рівнях одночасно, і ця здатність називається багатоаспектною увагою.

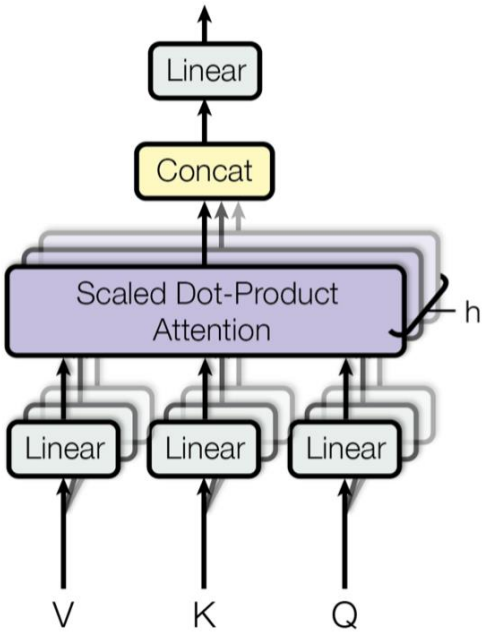

Рисунок 1.2 – Схема роботи багатоаспектної самоуваги в трансформерній архітектурі [1]

Ці групи моделі допомагають повністю зрозуміти суть і функцію кожного слова. Модель інтегрує характерні внески кожної групи і здатна оцінювати та детально ознайомлюватися з рівнем інформації в тексті в академічний спосіб. Більше того, така формація дозволяє їм використовувати багато параметрів

одночасно і при цьому діяти так само продуктивно та ефективно, як і раніше, оскільки, наприклад, кількість параметрів може бути досить великою [1, 3].

Оскільки в трансформерах відсутня рекурсія, порядок слів у реченні встановлюється на основі позиційного кодування. Модель вводить спеціальні позиційні елементи в представлення лексем, щоб охарактеризувати відносне або абсолютне положення кожного елемента. Такий підхід пом'якшує відсутність внутрішньої послідовної структури і дозволяє мережі правильно ідентифікувати структуру речення [1, 4].

Окрім уваги, інші шари в мережі трансформера включають повністю прямі з'єднання. Ця мережа є універсальною та нелінійною. Вона обробляє окремі вектори для всіх токенів. Ця мережа має здатність змінювати характеристики після того, як увага спрацювала. В результаті нового прихованого розміру прямі з'єднання покращують характеристики та розширюють можливості розширення текстового контексту [1, 5].

Для стабілізації процесу навчання застосовуються нормалізація шару та залишкові з'єднання. При використанні залишкових з'єднань стає можливим подолати проблему погіршення якості з урахуванням глибини мережі шляхом підтримки градієнтного потоку через пряме з'єднання, яке обходить складні перетворення. Нормалізація усуває нерівномірність у плані масштабу активації та забезпечує рівну стабільність для кожного рівня в мережі [6].

Найпоширеніші генеративні моделі працюють на основі авторегресійного принципу. Текст створюється послідовно. Тобто для кожного токена він покладається на всі інші токени, що передують йому. Для цього застосовується маскування уваги, щоб гарантувати, що прогнози не робляться з використанням токенів, які не зустрічалися. Це триває доти, доки модель не згенерує ціле речення. Також можна контролювати характер цього тексту за допомогою параметрів вибірки та температури [7].

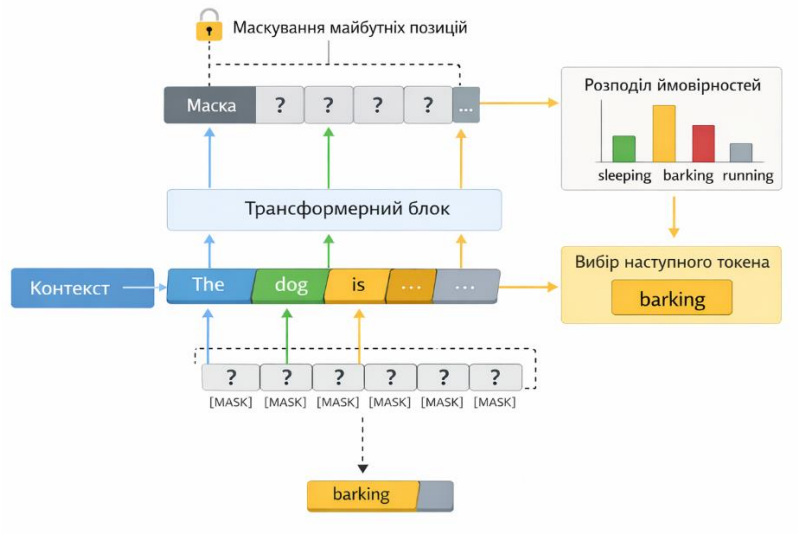

Рисунок 1.3 – Авторегресивне генерування

Сучасні великомасштабні мовні моделі масштабуються шляхом додавання більшої кількості шарів, параметрів та обсягу даних. Це дозволяє їм краще фіксувати складні лінгвістичні структури, виконувати логічні міркування та будувати складні текстові композиції з різними елементами. Масштабування трансформера продемонструвало свою ефективність порівняно з рекурентними нейронними мережами з точки зору точності та адаптивності до нових лінгвістичних моделей для різних завдань [8, 9].

1.2 Класифікація та основні вектори сучасних фішингових атак

Фішинг-атаки – це різновид атак із використанням соціальних інженерних методів, під час яких зловмисник видає себе за довірену особу, щоб виманити конфіденційну інформацію або спонукати користувачів вчинити дії проти їхньої волі. У переважній більшості випадків фішинг-атаки базуються на підроблених повідомленнях, які виглядають так, ніби їх надіслали банки, державні органи, системи електронної пошти (або інші корпоративні машини) та веб-сайти.

Не тільки ENISA визнала цю проблему – серед основних кіберзагроз в ЄС фішинг і соціальна інженерія в цілому є двома найбільш згадуваними проблемами, що негативно впливають на випадки порушення безпеки даних [10].

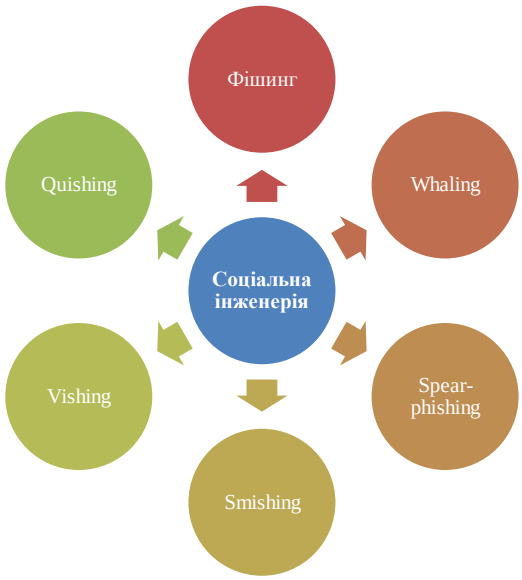

Рисунок 1.4 – Загальне місце фішингу в ландшафті загроз

Класифікація фішингових атак зазвичай здійснюється за кількома ознаками:

- за каналом доставки (електронна пошта, SMS, месенджери, соціальні мережі, телефонні дзвінки, QR-коди тощо);
- за рівнем таргетованості (масові кампанії, цілеспрямований spear-phishing, атаки на керівництво – whaling, бізнес-компрометація e-mail – BEC);
- за цілями (крадіжка облікових даних, фінансове шахрайство, розповсюдження шкідливого ПЗ, примус до здійснення платежів або зміни реквізитів);
- за складністю сценарію (простий фішинг із типовим шаблоном, складні мультиетапні атаки з елементами попередньої розвідки, підробленими доменами та багатоканальною взаємодією). Такий багатовимірний підхід дозволяє точніше оцінювати ризики та підбирати відповідні заходи протидії [10, 11].

Рисунок 1.5 – Основні критерії класифікації фішингових атак

Серед елементарних форм фішингових атак є масові фішингові атаки. Вони зачіпають різних людей. Зазвичай повідомлення, що використовуються в масових фішингових кампаніях, мають форму шаблонів. Ці повідомлення можуть містити такі слова, як «блокування рахунку», «непогашені рахунки», «перевірка безпеки». Щоб продовжити, повідомлення просить особу, яка отримала посилання, надати свої облікові дані. Звичайно, масові фішингові атаки передбачають надсилання великої кількості технічно простих електронних листів.

Незважаючи на низький рівень успішності, зловмисники досягають своєї мети. Статистика з доповіді Антифішингової робочої групи за четвертий квартал 2024 року вказує на високу частоту масових фішингових атак. За три місяці зареєстровано приблизно мільйон випадків фішингу, що свідчить про зростання цього показника протягом останніх шести місяців року [11].

Однак більш шкідливою формою фішингу є spear-phishing, також відомий як особисті атаки на певних осіб. У цьому випадку фішингове електронне повідомлення створюється з урахуванням контексту, в якому відбуватиметься фішингова атака. Це означає, що фішингове електронне повідомлення може мати вигляд повідомлення від керівника відділу про певні проєкти. Згідно зі статистикою Verizon DBIR, багато атак соціального інжинірингу в діловому світі пов'язані з претекстуванням. В основному, хакер придумує якийсь план комунікації перед тим, як надіслати перше електронне повідомлення [12].

Рисунок 1.6 – Порівняння масового фішингу та spear-phishing

Атаки типу whaling на керівників, членів правління та бізнес-лідерів вимагають особливої уваги. Такі електронні листи зазвичай імітують повідомлення, що мають юридичне значення, наприклад, електронні листи від банків, аудиторів, урядових установ та ділових партнерів. Оскільки довіра до цього засобу комунікації дуже висока, а керівники не витрачають багато часу на читання електронних листів, вплив атаки типу whaling (цільовий фішинг на керівництво) з точки зору фінансових втрат або крадіжки конфіденційної ділової інформації зазвичай є значним. Іноді зловмисники поєднують процес полювання з компрометацією ділової електронної пошти (BEC), під час якої зловмисники протягом тривалого часу спілкуються від імені бізнес-лідерів, змінюючи умови оплати або вкладені файли [10, 12].

Поряд із традиційним фішингом за допомогою електронних листів, розробляються нові методи фішингу:

- smishing (SMS-фішинг) — фішингові повідомлення, що надсилаються через SMS або месенджери, які містять посилання на фішингові сторінки та/або підроблені сторінки оплати.

- vishing (голосовий фішинг) — дзвінки, під час яких зловмисник видає себе за співробітника банку, інженера з технічної підтримки, державного службовця.

- quishing (QR-фішинг) — процес встановлення або розсилки посилань, що містять QR-коди для фішингових сайтів.

Спільні сценарії, в яких користувач отримує SMS-повідомлення, дзвінок та електронне повідомлення, що доповнюють одне одного. Згідно з аналізом агентства ENISA на тему IT-загроз у фінансовому секторі, використання як smishing, так і quishing в останні роки зростає, особливо щодо клієнтів банків [13].

Рисунок 1.7 – Канали реалізації фішингових атак

Що стосується цілей атак у фішингових повідомленнях, найпоширенішими є отримання доступу до облікових даних користувача, здійснення небажаних фінансових операцій та розповсюдження шкідливого програмного забезпечення.

Слід зауважити, що зловмисники хочуть, щоб одержувач надав у повідомленні свої облікові дані, паролі, одноразові паролі підтвердження або інші секретні дані для автентифікації, вдаючи, що це звичайна процедура авторизації.

Отже, основною метою зловмисників є змусити жертву здійснити платіж, змінити спосіб оплати, підтвердити переказ коштів або підтвердити «компенсацію/повернення коштів», ініційоване банком або постачальником послуг.

Варто також наголосити, що використання фішингових повідомлень для розповсюдження шкідливого програмного забезпечення, посилання

перенаправляє відвідувача на веб-сторінку, яка містить експлойт/завантаження виконуваного файлу. Крім того, вкладення електронного листа містить документ, який містить макрос/шкідливий код файлу.

Індивідуально для кожної конкретної спроби атаки фішинг-атака служить першим ланцюжком у ланцюжку сценаріїв атак, які створюють середовище для втручання в інформаційні системи організації [11, 12].

Також велика увага приділяється складності фішингових повідомлень з точки зору психології людини. Так, наприклад, підхід «NIST Phish Scale» пропонує оцінювати фішингові повідомлення з точки зору складності їх розпізнавання користувачем, враховуючи різні параметри, пов'язані з очевидними ознаками підробки повідомлення, відповідністю «очікуванням», використанням «термінового» мовлення, імітацією природних процесів у бізнес-організації в повідомленні тощо. Це допомагає класифікувати фішингові повідомлення не тільки з технічної точки зору, але й з точки зору психологічного впливу [14].

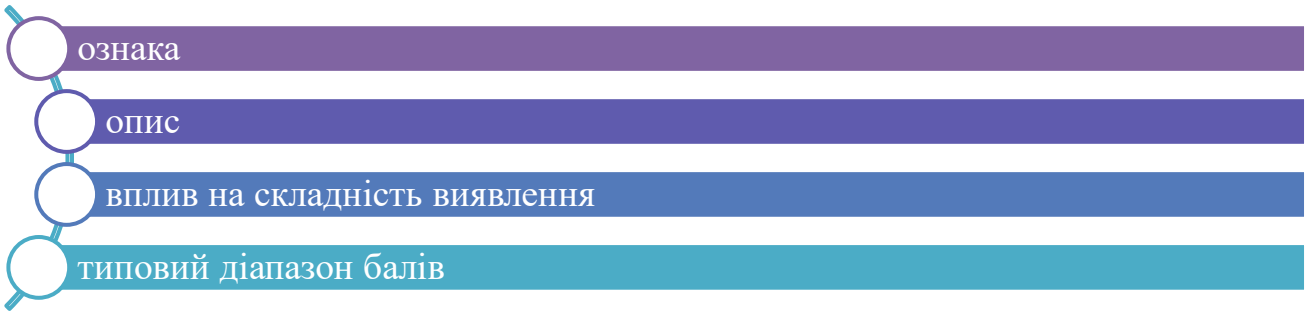

Рисунок 1.8– Приклад оцінки складності фішингового листа за NIST Phish Scale

Звичайно, сучасні фішинг-атаки використовують одночасно різні вектори. Так, наприклад, повідомлення можуть також містити дані з попередніх витоків інформації в Інтернеті, соціальних мережах, а також результати розвідки доменів та бізнес-інфраструктури. Такий складний сценарій поєднує в собі різні технологічні елементи (підроблені домени, SSL-сертифікати, обхід фільтрів) та ретельно розроблену соціальну психологію. Згідно з аналізом огляду ENISA та рекомендаціями CISA, успіх сучасних фішингових атак

полягає в поєднанні різних технологій з потужними психологічними компонентами [10, 15].

На інфраструктурному рівні однією з помітних тенденцій є те, що зловмисники використовують як «сумнівні», так і дуже авторитетні хостингові компанії для розміщення фішингових сторінок і розсилки електронних листів. Це можна спостерігати в аналізі походження фішингового мережевого трафіку. Було відзначено, що багато фішингових електронних листів розміщуються дуже авторитетними хмарними та електронними хостинговими компаніями. Це фактично ускладнює блокування повідомлень системою залежно від їхнього походження [16].

1.3 Огляд існуючих традиційних методів детектування фішингу та їхні обмеження

Традиційно методи виявлення фішингових атак почали розвиватися як вдосконалення інших методів, що використовуються для захисту електронної пошти, веб-трафіку та браузерів. Оскільки фішинг розглядався в основному як спам у будь-якій його формі (бажаний або небажаний), перші рішення полягали у спробі просто видалити небажані повідомлення шляхом фільтрування їх за дуже простими критеріями, що стосуються їх властивостей. Загалом, методи виявлення фішингу в літературі класифікуються на методи, засновані на списках (чорні/білі списки), евристичні рішення, методи, що враховують зміст повідомлень, та клієнтські антифішингові програмні рішення, а також організаційні заходи, такі як симуляції фішингу [17, 18].

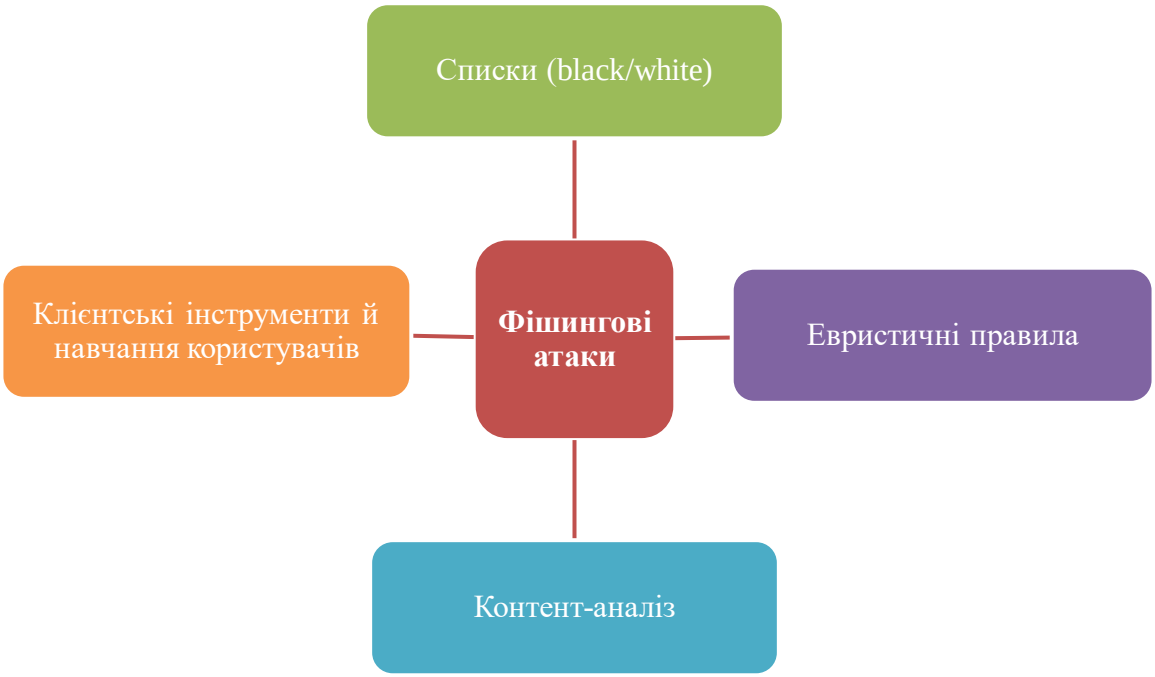

Рисунок 1.9 – Узагальнена схема традиційних підходів до детектування фішингу

Одним із найстаріших і найпростіших методів є підхід із використанням чорного/білого списку. У цьому випадку домени/IP-адреси/URL-адреси вважаються фішинговими, якщо вони містять відомі фішингові ресурси, і навпаки, якщо вони містять надійні ресурси. Ці ресурси підтримуються різними компаніями з безпеки, браузерами, антивірусним програмним забезпеченням та поштовими клієнтами. Чорні списки також регулярно оновлюються на основі інцидентів, про які повідомляють користувачі, а також на основі інформації від партнерських компаній. Цей підхід простий у реалізації, оскільки процес перевірки полягає лише в перевірці відомих адрес у їхній базі даних. Однак у деяких оглядах повідомляється, що механізм блокування за допомогою чорного списку стає неефективним у випадках, коли фішингові домени швидко змінюються, а зловмисники використовують їх на скомпрометованих сайтах [17, 19].

Евристичні або засновані на правилах методи також стали більш досконалими, а зусилля спрямовані на кодифікацію відмінностей між фішингом і надійними ресурсами за допомогою заздалегідь визначеного набору ознак. На рівні веб-сайту до таких ознак зазвичай належать аномалії

URL-адреси, такі як надмірна довжина, використання цифрових IP-адрес замість доменних імен, дивні імена піддоменів або занадто багато параметрів рядка запиту, наявність індикаторів перенаправлення веб-сторінок, підозрілі параметри SSL-сертифіката або суперечності між доменним ім'ям і брендингом веб-сторінки, що відображається. На рівні електронної пошти фактори включають відмінності між адресою відправника повідомлення та доменами, знайденими в URL-адресах будь-яких гіперпосилань, дивні формати вкладень до повідомлень або поєднання загрозливої мови з проханнями про здійснення фінансової транзакції. Ці моделі, ймовірно, розроблені на основі минулого досвіду експертів у сфері фішингу або попередніх досліджень аналізу інцидентів; проте вони залишаються нестійкими, оскільки невеликі зміни в структурі повідомлення або веб-сторінки можуть легко обійти заздалегідь визначені евристичні алгоритми, як видно з оглядів літератури, присвяченої виявленню фішингу [18, 20, 23].

Окрема область традиційного аналізу фішингу пов'язана з аналізом повідомлень і структурою веб-сайтів. У найпростішому випадку це полягає в пошуку характерних фраз, пов'язаних з тим, що вважається «блокуванням облікового запису», «терміновим оновленням даних» або «перевіркою безпеки». Більш складний підхід передбачає аналіз HTML-структури сторінки, наявність форм для подання інформації про обліковий запис, спосіб відображення логотипів і графіки або ступінь їх схожості. У деяких публікаціях детально описано використання підходів комп'ютерного зору, де знімки екрана фішингових сторінок порівнюються з заздалегідь визначеними зображеннями реальних веб-ресурсів [18]. Однак, коли фішингова сторінка використовує такі практики, як заплутування коду, динамічне завантаження елементів або невеликі навмисні коригування структури сторінки чи тексту, ефективність аналізу повідомлень значно знижується [23].

Це клієнтське програмне забезпечення для захисту від фішингу, яке включає розширення для браузерів, модулі перевірки посилань, панелі інструментів або панелі попередження в клієнтському програмному

забезпеченні електронної пошти. Ці рішення поєднують в собі підходи, засновані на списках, евристичні або орієнтовані на вміст, пропонуючи користувачеві додаткову візуальну інформацію про сумнівні форми введення даних, невідповідності між адресним рядком і показаною адресою або інші дивні елементи ресурсу. Ключовою сильною стороною цих рішень є їх безперебійна інтеграція в робочий контекст. Водночас їхня ефективність значною мірою залежить від якості внутрішніх баз даних, а надто велика кількість помилкових тривог може заважати користувачам реагувати на попередження [17, 21].

Таблиця 1.1 – Основні традиційні підходи до детектування фішингу та їхні обмеження

Підхід	**Короткий опис**	**Переваги**	**Типові обмеження**
Списки (black/white)	Порівняння доменів/URL/відправників із базами «заборонених» (blacklist) або «дозволених» (whitelist) значень.	Простота реалізації; висока точність для вже відомих загроз; низькі обчислювальні витрати.	Погано працює проти нових (zero-day) фішингових ресурсів; залежність від актуальності списків; можливі хибні блокування або пропуски через схожі домени/редіректи/скорочувачі посилань.
Евристичні методи	Правила та «сигнали ризику» (наприклад, підозрілі патерни URL, невідповідність домену, аномальні заголовки листа, підозрілі вкладення).	Швидкі; можуть виявляти частину невідомих атак; гнучко налаштовуються під політики організації.	Високий ризик хибнопозитивних/хибнонегативних спрацювань; обхід шляхом адаптації шаблонів атак; потреба в регулярному перегляді правил.

Продовження таблиці 1.1.

Підхід	Короткий опис	Переваги	Типові обмеження
Контент-орієнтований аналіз	Аналіз тексту/візуального вмісту (ключові слова, стилістика, заклики до дії, бренд-імітація, аналіз HTML/форм, структури сторінки/листа).	Більш «змістовне» виявлення; ефективний проти підробок брендів і соціальної інженерії; може працювати без готових списків.	Складність обробки мультимодального контенту (текст+зображення); обхід через обфускацію, зображення замість тексту, динамічний контент; потреба в якісних ознаках/моделях і даних.
Клієнтські антифішингові інструменти	Механізми в браузерах/поштових клієнтах: попередження про небезпечні сайти, перевірка посилань, sandbox для вкладень, блокування скриптів.	Масштабованість на рівні кінцевого користувача; зручність і автоматизація; зниження ризику «людської помилки».	Нерівномірна ефективність залежно від налаштувань і оновлень; обхід через соціальну інженерію («дозвольте», «натисніть усе одно»); обмежена видимість контексту (особливо в корпоративних сценаріях).
Організаційні заходи (навчання)	Тренінги, симуляції фішингу, політики безпеки, процедури верифікації запитів, двофакторна автентифікація, культура звітування.	Зменшує вплив людського фактору; універсально для різних каналів атак; підсилює технічні засоби.	Ефект залежить від регулярності та якості навчання; «втома» користувачів від тренінгів; не гарантує захист від таргетованого spear-phishing; потребує часу й ресурсів.

Окрім технічних рішень, організаційні заходи, спрямовані на покращення розуміння користувачами, є ключовим кроком у запобіганні фішингу. Організації проводять навчання користувачів практикам кібергігієни, проводячи тренінги, імітуючи фішингові атаки, під час яких організація аналізує колективні помилки користувачів, щоб відповідно вдосконалити власні політики кібербезпеки. У своїх рекомендаціях для малих підприємств NIST чітко зазначає, що фішинг є однією з головних загроз, з якою необхідно боротися за допомогою поєднання технічного фільтрування, навчання, чіткого повідомлення про інциденти та зрозумілих критеріїв перевірки

підозрілих електронних листів [22]. Для оцінки ризику виявлення певного фішингового електронного листа було запропоновано підхід NIST Phish Scale, який дозволяє класифікувати сценарії на основі ризику людської помилки, щоб відповідно планувати відповідні навчальні заходи [14].

Незважаючи на наявність різних традиційних рішень, аналіз оглядової літератури показує, що існують певні підводні камені, які можуть перешкоджати їхній ефективності в нинішніх умовах загроз. Рішення на основі білих списків все ще є реактивними, очікуючи на появу нових доменів або URL-адрес, які потрібно своєчасно додавати, через що вони відстають від нових фішингових матеріалів. Евристика або вміст вимагають ряду заздалегідь визначених правил або сигнатур, які потребують частого втручання людини для оновлення, а їх ефективність серйозно страждає від навмисних спроб змінити схеми атак. Хоча навчання користувачів мінімізує випадки наївних людських помилок, воно не забезпечує довгострокового захисту, особливо щодо атак типу spear-phishing та whaling-кампаній, в яких кожне окреме повідомлення адаптоване для конкретних одержувачів [17, 23].

Систематичний аналіз методів боротьби з фішингом показує, що поєднання технічних і соціально-інженерних підходів, багатоканальна доставка атак і швидка мінливість інфраструктури атак унеможливлюють ефективність статичних правил, списків або суто евристичних рішень [17, 24]. Ця проблема створює потребу в більш динамічних методах, здатних автоматично розпізнавати приховані шаблони текстів, форматування повідомлень та пов'язані метадані, самостійно розробляти нові шаблони атак та адаптуватися до мінливої поведінки зловмисників. Відповідно, це призводить до природного впровадження машинного навчання, моделей глибокого навчання або сучасних генеративних мовних моделей для розробки більш гнучких систем виявлення фішингових повідомлень, як обговорюється в наступних розділах [23, 24].

2 АНАЛІЗ ВПЛИВУ ГЕНЕРАТИВНИХ МОВНИХ МОДЕЛЕЙ НА СТВОРЕННЯ ФІШИНГОВИХ ПОВІДОМЛЕНЬ

2.1 Огляд методів машинного навчання та глибокого навчання для класифікації текстів

Звертаючи увагу на підрозділ 1.3, можна зауважити, що традиційні антифішингові рішення, засновані на чорних списках і статичних евристичних правилах з перевіркою конкретного вмісту, виявилися менш надійними в умовах динамічної поведінки зловмисників і зміни інфраструктури атак. Як наслідок, зріс інтерес до рішень, заснованих на аналізі поведінки зловмисників за допомогою широкого спектру показників і моделей машинного навчання на великих наборах даних, отриманих з минулих атак.

У сучасних передових рішеннях ідентифікація фішингу формулюється як проблема класифікації, де кожен розглянутий елемент (URL, повідомлення та веб-сторінка) може бути описаний за допомогою вектора показників, а техніка машинного навчання або глибокого навчання класифікує їх як фішингові або нефішингові повідомлення [18, 24].

Таблиця 2.1 – Процес детектування фішингових повідомлень на основі методів машинного та глибокого навчання

Етап	Компонент системи	Характеристика та функціональне призначення
1	Вхідні дані	Формування набору даних для аналізу, що включає URL-адреси, текст електронних листів або SMS-повідомлень, HTML-структуру вебсторінок, а також технічні метадані (WHOIS, SSL-сертифікати, час відправлення, IP-адреси тощо).
2	Попередня обробка	Очищення та нормалізація даних, видалення шуму, токенізація тексту, усунення стоп-слів, лематизація або стемінг, обробка спеціальних символів і HTML-тегів.

Продовження таблиці 2.1.

Етап	Компонент системи	Характеристика та функціональне призначення
3	Побудова ознак / ембеддингів	Формування числових представлень даних шляхом вилучення статистичних, лексичних, структурних та семантичних ознак, а також створення контекстних ембеддингів за допомогою моделей глибокого навчання.
4	Модель машинного або глибокого навчання	Класифікація повідомлень із використанням традиційних алгоритмів машинного навчання (логістична регресія, SVM, Random Forest) або нейронних мереж (CNN, LSTM, трансформерні архітектури типу BERT).
5	Прийняття рішення	Формування фінального результату класифікації, що визначає повідомлення як фішингове або легітимне, з можливістю подальшої інтеграції результату в системи інформаційної безпеки.

У традиційних налаштуваннях машинного навчання основна увага приділяється створенню інформативного набору ознак, який потім можна використовувати в поєднанні з традиційними класифікаторами машинного навчання. Для класифікації URL-адрес можна отримати набір ознак на основі лексичних характеристик, таких як довжина URL-адреси, використання спеціальних символів, наявність підозрілих токенів в URL-адресах, структура доменних і піддоменних імен, інформація WHOIS та інформація про репутацію хостингу. Для веб-сторінок аналіз можна проводити на основі структури DOM, наявності форм входу, використання скриптів перенаправлення, наявності вбудованих фреймів та використання сторонніх скриптів. У налаштуваннях електронної пошти можна генерувати текстові векторні представлення, такі як матриці bag-of-words або TF-IDF, разом з технічною інформацією, що міститься в заголовках, та інформацією про вкладення. Для цих представлень можуть бути ефективними такі методи, як логістична регресія, машини опорних векторів, дерева рішень, випадкові ліси та градієнтне підсилення. При правильній побудові цих функцій ці методи можуть легко класифікувати з прийнятним рівнем точності, але вони схильні до еволюції моделей атак і потребують постійних оновлень [18, 25, 26].

Поява технології глибокого навчання дозволила частково відійти від вибору характерних ознак, що дало можливість навчати модель на необроблених

послідовностях даних, таких як символи, слова або елементи HTML-коду. Спіральна нейронна мережа використовується для виявлення фішингових URL-адрес на основі локальних шаблонів у тексті цих URL-адрес та характерних сегментів зловмисної діяльності. Рекурентні архітектури у формі LSTM/GRU можуть використовуватися для роботи з текстом електронних листів та повідомлень завдяки їхній здатності описувати залежності на рівні, вищому за прості маркери, наприклад на рівні речень. Багато досліджень показують, що добре навчена модель може значно перевершувати традиційні алгоритми за точністю, особливо в ситуаціях, коли супротивник динамічно змінює структуру повідомлення та елементи сторінки [24, 26].

Окремий клас моделей базується на архітектурі трансформера та попередньо навчених моделях мовного представлення. Сімейство моделей BERT та їхні варіанти, такі як DistilBERT, RoBERTa та інші, налагоджені для фішингових електронних листів шляхом навчання їх на наборі електронних листів, позначених як фішингові та справжні. Завдяки самоувазі ці моделі мають потенціал для розуміння складних семантичних відносин у текстах, що є важливим для виявлення семантично добре продуманих, стилістичних і цільових фішингових атак. Експериментальний аналіз показує, що за допомогою точного налаштування гіперпараметрів можна значно підвищити точність і F1-показник на відміну від традиційних алгоритмів машинного навчання та більшості моделей глибокого навчання [24, 27, 28, 30].

У системах виявлення фішингу інтерпретованість моделей є новою сферою інтересу. Важливість ознак і локальні пояснення в традиційних моделях машинного навчання широко використовуються для оцінки ролі URL-адреси, тексту та метаданих у класифікації екземпляра як фішингового типу. При роботі з глибокими моделями для оцінки впливу тексту, структурної інформації HTML та мережевої активності на фішингову атаку використовуються методи пояснювальної штучної інтелекту, такі як LIME, SHAP та карти уваги. Пояснюваність є важливим кроком у впровадженні автоматизованих систем у реагування на інциденти, оскільки вони

дозволяють аналітикам безпеки оцінювати прогнози моделей та отримувати інформацію про виявлені загрози [22, 24, 29].

Таблиця 2.2 – Порівняльна характеристика основних груп методів детектування фішингових повідомлень

Група методів	**Тип представлення даних**	**Основні переваги**	**Типові обмеження**	**Поширені сценарії використання**
Традиційні алгоритми машинного навчання (логістична регресія, SVM, дерева рішень,)	Ручні ознаки, сформовані на основі статистичних структурних та URL-характеристик	Відносна інтерпретованість моделей, простота реалізації та навчання	Сильна залежність від якості та повноти ручних ознак, потреба в експертному налаштуванні	Масовий аналіз поштового трафіку, базовий фільтр фішингових повідомлень
Глибокі нейронні мережі (CNN, LSTM/GRU, гібридні моделі)	Векторні подання даних на основі ембеддингів слів, символів або послідовностей	Здатність автоматично виділяти складні нелінійні патерни, обробка послідовних і просторових залежностей	Потреба у великих обсягах розмічених даних, значні обчислювальні ресурси,	Моніторинг високоризикових транзакцій, захист фінансових сервісів, аналіз текстових і HTML-даних
Моделі на основі трансформерів (BERT, RoBERTa, спеціалізовані трансформери для фішингу)	Контекстуальні представлення токенів з урахуванням глобального контексту повідомлення	Глибоке моделювання семантики та контексту, висока точність у складних сценаріях, стійкість до варіативності атак	Висока обчислювальна вартість, складність впровадження та оптимізації, потреба в апаратному прискоренні	Виявлення таргетованих spear-phishing атак, аналіз складних комбінованих і багатоетапних фішингових кампаній

Машинне навчання та глибоке навчання забезпечують більш гнучкі та адаптивні засоби виявлення фішингу. Вони дозволяють поєднувати текстову інформацію, структуру веб-сторінок та технічні деталі в єдиній моделі, яка утримує все це разом. Такі моделі здатні ідентифікувати нові моделі загроз, які не обов'язково визначені в правилах або списках, і адаптуватися з часом, коли зловмисники вдосконалюють свої методи. Майбутнє залежатиме від

впровадження структури трансформерної моделі та генеративних мовних моделей. Ці два інструменти дозволять створювати реалістичні сценарії тестування, а також основу для високоточних детекторів фішингу. Особливості всіх цих концепцій та застосування трансформерної моделі в кібербезпеці будуть розглянуті в наступному розділі [24, 26, 27, 28, 30].

2.2 Вибір та обґрунтування архітектури моделі детектування

Аналіз сучасних рішень у галузі машинного навчання та глибокого навчання для класифікації текстів показує, що найуспішнішими є ті рішення, які здатні аналізувати інформацію на рівні слів, граматичну структуру повідомлення, а також його семантичний зміст. Як зазначено в підрозділі 2.1, традиційні рішення машинного навчання зазвичай орієнтовані на особливості і не здатні добре адаптуватися, якщо змінюються моделі атак, тоді як глибоке навчання та трансформерні мережі добре підходять для вирішення складних моделей у мові та структурі повідомлень [24, 26, 27, 28].

В рамках цього дипломного проєкту, модель детектора повинна відповідати кільком основним критеріям. По-перше, вона має бути функціональною для текстів, які створюються та/або змінюються цими генеративними мовними моделями, тобто вона повинна бути стійкою до граматично правильних і стилістично природних повідомлень, які можуть бути зловмисно використані для фішингових атак. Крім того, обов'язковою умовою для цієї моделі є підтримка декількох мов і доменів, таких як банківські повідомлення, ділове спілкування та технічні повідомлення, враховуючи, що багато сучасних фішингових атак охоплюють декілька доменів та/або рівнів. По-третє, ця модель повинна бути технологічно придатною для подальшого включення в прототипну систему, згадану в

розділі 3, з урахуванням обчислювальних можливостей та обмежень [24, 26, 27].

З огляду на всі перераховані вище вимоги, доцільно було б використовувати архітектурне рішення на основі попередньо навченої мовної моделі трансформера, такої як BERT або її варіації. Це було б застосовано для забезпечення кодування тексту, а потім налагоджено для завдання двокласової класифікації. Метод тонкої настройки дозволяє нам використовувати вже отримані знання з мови та коригувати їх відповідно до особливостей фішингових повідомлень на відносно невеликих обсягах даних у сфері спеціалізації завдання з виявлення фішингових повідомлень за допомогою архітектурних варіацій BERT [27, 28, 30].

Причиною віддати перевагу архітектурі трансформерної мережі замість традиційних моделей CNN і LSTM є те, що вона виправдана завдяки здатності обробляти віддалені залежності в тексті без будь-яких обмежень послідовного навчання, а також завдяки механізму самоуваги, який дозволяє мережі зосередитися на найбільш інформативній частині вхідних даних. Це особливо важливо в контексті завдань фішингу, оскільки індикативна інформація може міститися у фразах (таких як «оновіть свої облікові дані», «підтвердіть платіж», «ваш обліковий запис заблоковано»), назвах брендів або комбінаціях слів, які можуть бути розташовані по всьому повідомленню. Результати експериментів доводять, що продуктивність мережі трансформера незрівнянно краща порівняно з результатами, досягнутими при використанні функцій TF–IDF у лінійних моделях та моделях RNN [24, 26, 27, 28].

Однак у сучасних наукових дослідженнях також розглядаються підходи, що поєднують семантичний аналіз текстового вмісту повідомлень із використанням технічних характеристик, таких як тип домену, тривалість його реєстрації, невідповідність між доменом відправника та доменом, відображеним у гіперпосиланнях, а також загальні параметри масових електронних розсилок. Такі комбіновані підходи здатні підвищувати стійкість систем детектування до маніпуляцій семантичними ознаками та складних

сценаріїв атак. У межах цієї магістерської роботи зазначені технічні характеристики розглядаються як перспективний напрям подальшого розширення системи, тоді як експериментальна оцінка та реалізація зосереджені виключно на аналізі текстового вмісту фішингових повідомлень із використанням трансформерної моделі [18, 25, 26]. Тому пропонована архітектура ґрунтується на трансформерній моделі, яка здійснює аналіз текстового вмісту повідомлень та формує контекстні векторні подання, що передаються до класифікаційного шару для прийняття фінального рішення.

З практичної точки зору, архітектура є компромісом між точністю та обчислювальною складністю. З одного боку, використання трансформера зменшує залежність від великого обсягу навчальних даних, оскільки трансформер є ефективним у роботі з мовою. З іншого боку, обмеження кількості додаткових шарів та розміру повністю з'єднаних шарів допомагає забезпечити реалістичний час навчання та інференції, що є важливим не тільки для експериментів, але й для прототипування, пов'язаного з розробкою системи виявлення. Більше того, архітектура відповідає меті дослідницької роботи, яка полягає у вивченні того, як генеративні мовні моделі, що успішно генерують реалістичний фішинговий контент, можуть відігравати роль у розробці більш ефективних механізмів виявлення на основі подібних моделей текстового моделювання [24, 27, 30].

Обрана архітектура все ще є придатною для майбутнього розширення за допомогою використання генеративних моделей на етапах створення навчальних зразків та створення суперечливих зразків. Конкретне використання генеративної моделі дозволить створювати фішингові повідомлення, що відрізняються вибором слів та рівнем націленості. Обрана модель дозволить оцінити межу узагальнення, а також стійкість моделі до цільових атак, побудованих за допомогою сучасних LLM. Модель збереже структуру компактної моделі трансформера з класифікаційною головкою.

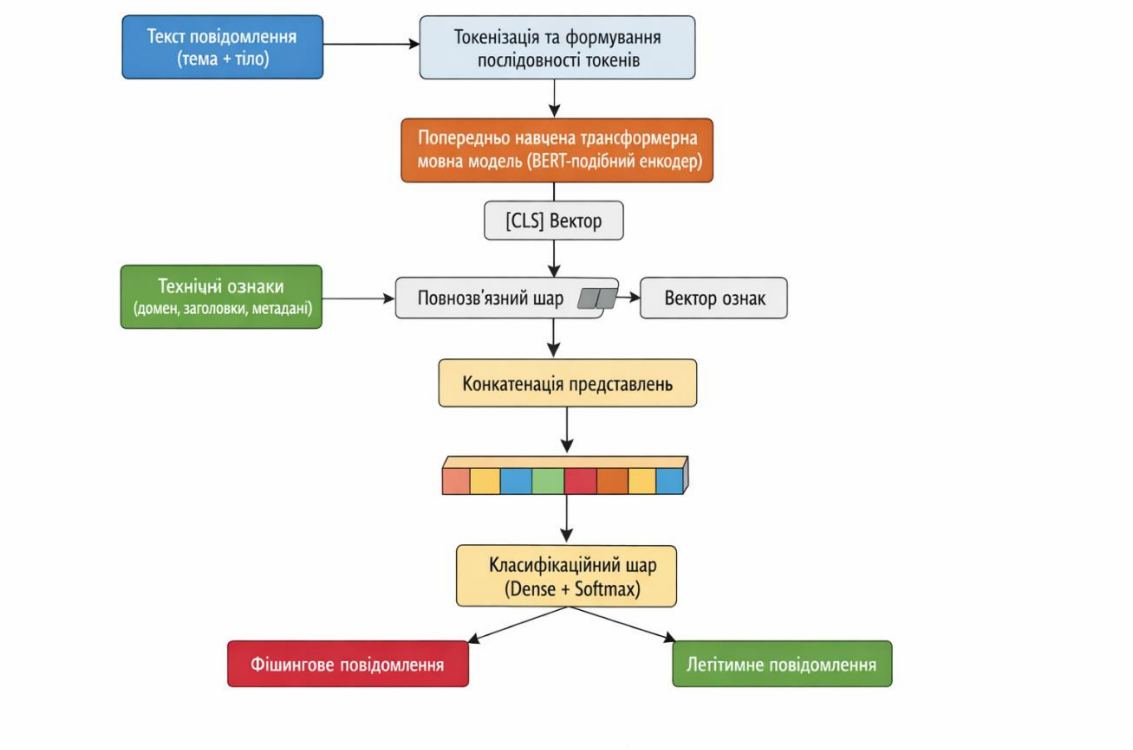

Рисунок 2.1 – Архітектура моделі детектування фішингових повідомлень на основі трансформерного енкодера

У подальших підрозділах наведена архітектура деталізується на рівні конкретної програмної реалізації, а також експериментально оцінюється її ефективність у сценаріях, де фішингові тексти повністю або частково сформовані генеративними мовними моделями різного класу.

3 РОЗРОБКА КІНЦЕВОЇ КОНСТРУКЦІЇ, ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ТЕСТУВАННЯ РОБОТИ

3.1 Проєктування архітектури системи детектування фішингових повідомлень із використанням генеративних мовних моделей

Архітектура системи виявлення фішингових повідомлень, описана нижче, необхідна для вирішення двох взаємопов'язаних проблем. По-перше, існує потреба у розробці та впровадженні детектора, що використовує трансформерну модель, здатну ефективно ідентифікувати фішингові повідомлення та інші справжні повідомлення. По-друге, розроблена система повинна мати можливість генерувати приклади фішингових повідомлень за допомогою генеративних мовних моделей, які використовуються для оцінки вразливості детектора до цих прикладів під час експериментів і процесу дослідження. Таким чином, архітектура буде включати модулі для обробки даних і генерації фішингових текстів.

З концептуальної точки зору, доцільно розрізняти дві основні частини системи: офлайн-контур навчання та дослідження і онлайн-контур виявлення. У цьому випадку офлайн-частина включає в себе діяльність, пов'язану зі збором та обробкою даних, створенням нових даних за допомогою гаусових змішаних моделей, навчанням та перенавчанням детектора, а також проведенням експериментів та оцінкою якості. З іншого боку, діяльність, пов'язана з розгортанням моделі для нових повідомлень та інтеграцією з іншими системами (поштова служба, веб-інтерфейс та служба інформаційної безпеки), відноситься до сфери онлайн-частини. Такий поділ допомагає визначити професійну межу між дослідницькою та оціночною роботою і роботою з впровадження, пов'язаною з цим проєктом, як це також рекомендується для впровадження рішень на основі ML для сценарію кібербезпеки [24, 26].

Архітектурні характеристики системи в цілому можна описати, розглядаючи її як сукупність різних логічно відокремлених модулів, які взаємодіють між собою через чітко визначені інтерфейси. Ці модульні частини та процеси архітектури системи складаються з:

- компонента системи управління даними;
- модуля для складання фішингових повідомлень за допомогою генеративної мовної моделі для складання тексту;
- модуля попередньої обробки для побудови ознак;
- модуля навчання та валідації для системи детектора;
- модуля служби виведення для онлайн-детекторів та модуля аналізу та візуального моніторингу для спостереження за результатами.

Для наочності взаємодію основних модулів подано на рисунку 3.1.

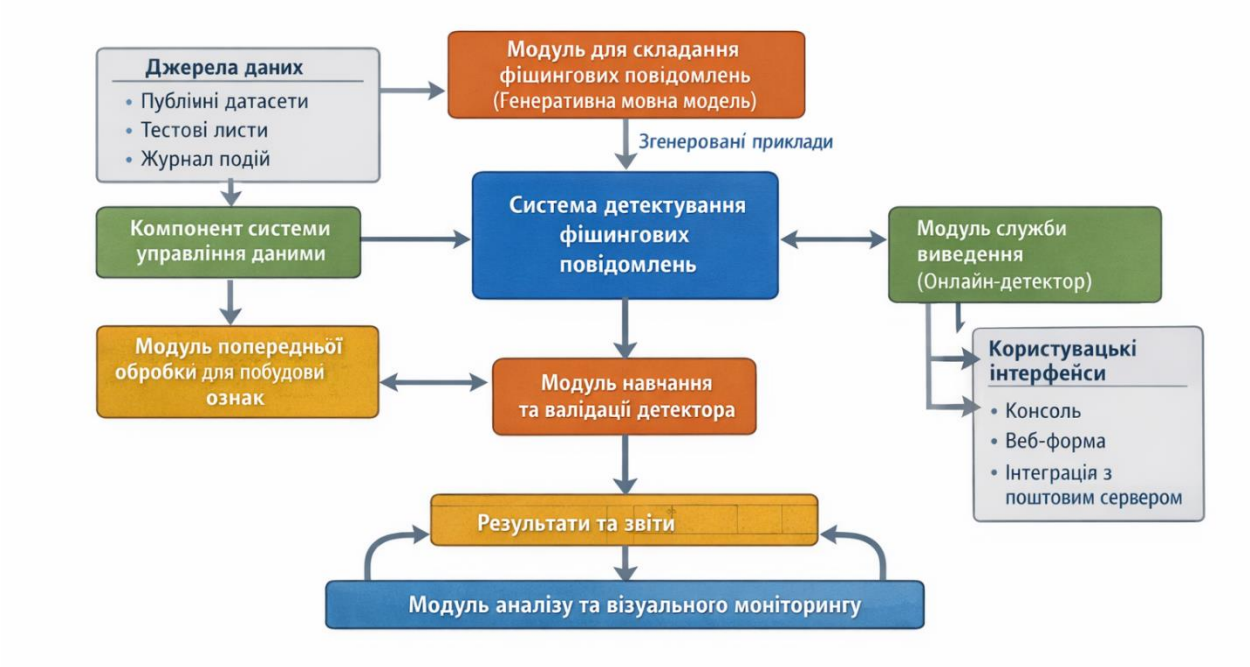

Рисунок 3.1 – Узагальнена модульна архітектура системи детектування фішингових повідомлень

Кожен з цих модулів виконує чітко визначений набір дій, а їхня злагоджена взаємодія забезпечує врахування всіх етапів життєвого циклу системи, починаючи від збору та обробки даних і закінчуючи онлайн-виявленням та аналізом результатів. Модуль управління даними відповідає за

централізоване завантаження та структурування наборів даних вхідних повідомлень. Саме на цьому етапі видаляються дублікати, нормалізується формат записів, анонімізуються конфіденційні дані та визначаються екземпляри для навчання, валідації та тестування, що є життєво важливим для збереження реалістичних співвідношень екземплярів, що належать до однієї з двох категорій — фішингових та легітимних комунікацій, а також для уникнення інформаційного потоку між підвибірками, як підкреслюється в сучасних дослідженнях, що обговорюють побудову систем машинного навчання в сучасному стані досліджень у галузі кібербезпеки [24, 26].

Модуль, який використовується для генерації фішингових повідомлень за допомогою генеративної мовної моделі, відіграє роль у генерації додаткових екземплярів, що імітують реальні ситуації атак. Модуль оснащений описом сценарію, що обслуговується (це може бути банківське повідомлення, лист, пов'язаний із службою підтримки, або будь-яке повідомлення компанії), а також змінними стилю. Сценарії генеруються кілька разів у текстовій формі та фільтруються для оцінки - ручної або напівавтоматичної — з метою перевірки відповідності вимогам до фішингових повідомлень. Перевірені екземпляри використовуються в спеціалізованих наборах даних для аналізу стійкості детектора до атак, в яких зловмисник може використовувати сучасні генеративні мовні моделі [24, 27, 30].

Модуль попередньої обробки з побудовою ознак використовує словник попередньо навченої моделі, обраної для токенізації тексту в повідомленнях і формування послідовності з включенням спеціальних токенів і маски уваги. Одночасно обчислюються технічні характеристики повідомлень, такі як довжина повідомлення, кількість гіперпосилань, частка цифрових символів і наявність типових ключових слів у повідомленні. Створений вихідний результат буде складатися з координованої послідовності тензорів, які потім будуть надіслані до модуля навчання та валідації детектора за допомогою стандартизованого інтерфейсу [26, 28].

Модуль компонента детектора, який відповідає за навчання та валідацію, використовує архітектуру трансформера, представлену в розділі 2.2, і розширює її за допомогою класифікаційного блоку, щоб відповідати задачі бінарної класифікації для розділення фішингових та легітимних електронних листів. Гіперпараметри, такі як розмір пакета, кількість епох, швидкість навчання та вибір оптимізатора, встановлюються у вищезазначеному компоненті. Для запобігання перенавчанню застосовуються методи регуляризації з ранньою зупинкою. Для набору валідації обчислюються такі метрики, як точність, прецизійність, показник F1 та ROC AUC, що дозволяє порівнювати різні версії моделей та випадки з синтезованими прикладами. Найкращі версії моделей зберігаються разом з їх конфігураційними файлами та описом використаних даних, що забезпечує відтворюваність експериментів [26, 28, 30].

Модуль вивідних послуг для онлайн-детекторів розроблений для роботи навченої моделі в режимі, близькому до реального часу. Ця послуга приймає вхідні повідомлення у вигляді тексту з можливістю подальшого розширення для підтримки технічних параметрів, повторно використовує методи попередньої обробки та передає тензори до моделі класифікації. Він надає вихідне повідомлення, що містить прогноз, який показує ймовірність того, що вхідні дані належать до класу фішингу, що дозволяє встановлювати порогові значення відповідно до конкретних політик безпеки, визначених в організації. Ці вихідні дані можуть бути передані на інтерфейс консолі, веб-сторінку або системи інформаційної безпеки, підключені до поштового сервера.

Частина моніторингу та аналізу — це місце, де всі дані журналу системи, показники навчання та валідації, а також результати виявлення збираються та консолідуються у вигляді зрозумілих графіків та звітів. Саме тут використовується весь аналіз ефектів генеративних моделей, і на основі попередніх досліджень можна вивести конкретні стратегії вдосконалення системи, включаючи ефекти генеративних моделей, зокрема [24, 27, 30].

Підсумовуючи, запропонована архітектура складається з різних модулів для управління даними, генерації фішингових текстів, попередньої обробки, навчання, онлайн-виявлення та моніторингу. Така архітектура забезпечує гнучкість для проведення експериментів та відповідного розширення запропонованої системи. Програмні засоби та деталі реалізації запропонованої архітектури для створення прототипу програмного забезпечення детально описані в розділі 3.2.

3.2 Реалізація програмного забезпечення системи детектування фішингових повідомлень

В рамках практичної частини цього дослідження було реалізовано прототип системи детектора фішингових повідомлень з використанням моделі трансформера, а також створено експериментальну схему для оцінки впливу прикладів з великої мовної моделі на ефективність детектора. Код був реалізований на Python в середовищі Windows з використанням бібліотеки PyTorch та бібліотеки Transformers від Hugging Face. Для кодування експерименту процес обчислення якості був формалізований у вигляді окремих скриптів у віртуальному середовищі, щоб зробити процес обчислення повністю відтворюваним. Щоб зробити конвеєр аналізу відтворюваним, кожна частина процесу аналізу була формалізована у вигляді окремих скриптів Python у зазначеному віртуальному середовищі.

Рисунок 3.2 – Структура конвеєру аналізу

Підготовка даних здійснювалась із приведенням текстових повідомлень до уніфікованого формату та формуванням підвибірок для навчання, валідації та тестування у вигляді CSV-файлів з полями text та label (0 — легітимне повідомлення, 1 — фішингове).

На цьому етапі забезпечувалась коректність розбиття даних, що дозволяє уникати змішування прикладів між підвибірками та забезпечує коректну оцінку узагальнювальної здатності моделі. Приклад запуску підготовки даних та сформованих підвибірок наведено у Додатку Б.1.

Базовий детектор реалізовано шляхом тонкого донавчання попередньо натренованої трансформерної моделі DistilBERT для задачі бінарної класифікації. Вхідні повідомлення токенізуються стандартним токенізатором обраної моделі з обмеженням максимальної довжини та побудовою тензорів input_ids і attention_mask, після чого подаються у класифікатор; на виході формується оцінка ймовірності належності повідомлення до класу фішингу. У процесі навчання контроль якості здійснювався за результатами валідації з використанням метрик precision, recall, F1-score та ROC-AUC. Логи навчання базової моделі наведено у Додатку Б.1.

Якість роботи базового детектора оцінено на відкладеній тестовій вибірці. Для цього реалізовано окремий сценарій оцінювання, який обчислює класифікаційний звіт, ROC-AUC, а також матрицю похибок із розкладом на TN/FP/FN/TP. Для базової моделі (збереженої у models/distilbert_nazario) отримано матрицю похибок TN=146, FP=4, FN=1, TP=156 та ROC-AUC = 0.9988. Відповідний вивід наведено у Додатку Б.2. Окремо реалізовано демонстраційний режим інференсу для одиночного повідомлення, який повертає вердикт (PHISHING/LEGIT) та значення P(phishing); приклади роботи наведено у Додатку Б.5.

Для дослідження впливу генеративних мовних моделей на стійкість детектора сформовано набір LLM-повідомлень та реалізовано контур їх оцінювання. Експериментальна логіка полягає у тому, що LLM-згенеровані повідомлення подаються на вхід уже навченого детектора, для кожного повідомлення обчислюється ймовірність p_phishing, а також аналізується частка повідомлень, що класифікуються як фішингові за заданим порогом. Додатково сформовано піднабори повідомлень, що відображають різні варіації представлення фішингового тексту (умовно “clean” та “artifacts”), і проведено аналіз чутливості системи до зміни порогу класифікації. Приклади запусків оцінювання LLM-наборів та порогового аналізу наведено у Додатку Б.4.

Наступним етапом виконано донавчання моделі з урахуванням LLM-згенерованих прикладів для підвищення здатності детектора реагувати на атаки, сформовані за допомогою сучасних генеративних моделей. Отримана модель збережена як models/distilbert_nazario_llmft і повторно оцінена на тестовій вибірці з використанням аналогічного сценарію оцінювання. Для моделі після LLM-aware fine-tuning отримано матрицю похибок TN=140, FP=10, FN=0, TP=157 та ROC-AUC = 0.9949; відповідний вивід наведено у Додатку Б.3. Таким чином, донавчання на LLM-прикладах може зменшувати ризик пропуску фішингових повідомлень (FN), однак супроводжується зростанням кількості хибних спрацювань (FP), що в практичних умовах потребує коректного вибору порогу та політики реагування.

Таблиця 3.1 – Порівняння якості детектора до та після донавчання на LLM-прикладах (тестова вибірка)

Модель	**Accuracy**	**ROC-AUC**	**TN**	**FP**	**FN**	**TP**
distilbert_nazario (базова)	0.9837	0.9988	146	4	1	156
distilbert_nazario_llmft (після LLM-aware fine-tuning)	0.9674	0.9949	140	10	0	157

Після донавчання на LLM-прикладах зменшується кількість пропусків фішингу (FN→0), однак зростає кількість хибних спрацювань (FP), що вказує на компроміс між чутливістю та специфічністю і потребує налаштування порогу прийняття рішення відповідно до політики безпеки.

3.3 Методика проведення експериментів та оцінка ефективності моделі

У межах роботи проведено серію експериментів для оцінювання ефективності моделі детектування фішингових повідомлень і аналізу впливу генеративних мовних моделей на стійкість класифікатора. Як базовий класифікатор використано трансформерну модель DistilBERT із доданою класифікаційною головкою для задачі двокласової класифікації (“phishing/legit”). Оцінювання виконувалось у двох взаємодоповнюючих сценаріях: (1) перевірка на реальному тестовому наборі повідомлень, (2) перевірка на наборі повідомлень, згенерованих/модифікованих генеративною мовною моделлю, що моделює сучасні фішингові тексти.

Для контролю якості на реальних даних використано відкладений тестовий набір data/processed/test.csv. Метрики оцінювання включали precision/recall/F1-score для кожного класу, загальну accuracy, ROC-AUC, а також матрицю помилок (confusion matrix), що дозволяє інтерпретувати співвідношення помилок першого

та другого роду. Оцінювання базової моделі models/distilbert_nazario виконано скриптом evaluate_base.py (додаток Б, рис. Б.2). Отримано такі результати: accuracy = 0.9902, ROC-AUC = 0.9997; матриця помилок має вигляд [147, 3, 0, 1570, 1570, 157], тобто TN=147, FP=3, FN=0, TP=157 (додаток Б, рис. Б.2). Це свідчить, що на реальному тесті базова модель забезпечує високий рівень виявлення фішингу при мінімальній кількості хибнопозитивних спрацювань.

Для оцінювання впливу генеративних мовних моделей сформовано LLM-набір із 48 текстів, а також піднабори “clean” та “artifacts”, що відрізняються наявністю характерних “артефактів” згенерованого тексту. Додатково реалізовано процедуру аналізу чутливості до порога класифікації (threshold sweep) у діапазоні значень 0.3–0.6. Для базової моделі на LLM-наборі спостерігалось заниження впевненості: частка повідомлень, віднесених до класу phishing при порозі 0.5, становила 0.375 для “LLM-clean”, а для “LLM-artifacts” частка залежала від порога та зростала при його зниженні (додаток Б, рис. Б.4). Це вказує на те, що частина LLM-згенерованих фішингових повідомлень маскується під легітимні та може проходити повз детектор без додаткової адаптації.

Для підвищення стійкості класифікатора до LLM-згенерованих текстів виконано донавчання (fine-tuning) на LLM-наборі та збережено модель models/distilbert_nazario_llmft (додаток Б, рис. Б.1). Після донавчання повторно виконано оцінювання на реальному тестовому наборі скриптом evaluate_llmft.py (додаток Б, рис. Б.3). Результати: accuracy = 0.9674, ROC-AUC = 0.9949; матриця помилок [140,10, 0,1570, 1570,157], тобто TN=140, FP=10, FN=0, TP=157 (додаток Б, рис. Б.3). Порівняно з базовою моделлю збільшується кількість хибнопозитивних спрацювань, що проявляється у зниженні точності на класі “legit”, однак зберігається повне покриття фішингового класу на цьому тесті (FN=0).

Водночас на LLM-наборі після донавчання спостерігається різке зростання впевненості моделі: для “LLM-clean” та “LLM-artifacts” при порозі 0.5 частка повідомлень, класифікованих як phishing, становить 1.000, а також відсутня чутливість до зміни порога в межах 0.3–0.6 (додаток Б, рис. Б.4). Це означає, що

модель стала значно агресивнішою щодо LLM-згенерованих текстів і практично завжди відносить їх до класу phishing.

Практичну працездатність прототипу підтверджено сценарієм інференсу окремих повідомлень у CLI, де модель повертає клас та ймовірність належності до phishing (додаток Б, рис. Б.5). Таким чином, експерименти демонструють компроміс між підвищенням чутливості до LLM-згенерованого фішингу та зростанням хибнопозитивних спрацювань на реальних даних, що потребує додаткового налаштування (наприклад, балансування вибірок, калібрування ймовірностей або оптимізації порога за цільовою функцією).

3.4 Оцінка можливостей масштабування та інтеграції системи у корпоративну інфраструктуру

Розроблений прототип детектора фішингових повідомлень, реалізований у розділі 3.2, може бути використаний не лише як експериментальний стенд, а і як основа для прикладного сервісу в корпоративному контурі кіберзахисту. На практиці основними вимогами до такого рішення є відтворюваність результатів, контроль якості на відкладених наборах даних, керованість порогу прийняття рішення та можливість регулярного оновлення моделі відповідно до еволюції фішингових технік. В роботі ці аспекти вже частково підтверджені на рівні прототипу: наявні окремі сценарії навчання/оцінювання, пороговий аналіз і демонстраційний інференс, що наведено у додатку Б (рис. Б.1–Б.5).

З погляду розгортання в реальному середовищі доцільно розрізняти два режими застосування детектора: офлайн-контур (перенавчання, валідація, експерименти) та онлайн-контур (класифікація вхідних повідомлень у режимі, близькому до реального часу). Офлайн-контур відповідає за підтримку якості та відтворюваність: приклади підготовки вибірок і запуску навчання наведені у додатку Б (рис. Б.1). Онлайн-контур забезпечує прикладне використання моделі у

вигляді сервісу або утиліти інференсу; приклад роботи інференсу з поверненням вердикту PHISHING/LEGIT наведено у додатку Б (рис. Б.5). Такий поділ є практично важливим, оскільки дозволяє чітко розмежувати експериментальні процедури та експлуатацію моделі в продуктивному середовищі, де діють вимоги безперервності й мінімізації помилкових спрацювань.

3.4.1 Інтеграція з поштовим контуром та системами безпеки

Для інтеграції з корпоративною поштою найбільш типовими є два сценарії. Перший — використання детектора на рівні поштового шлюзу, коли повідомлення аналізується до доставки користувачу, а результат застосовується для блокування, карантину або маркування. Другий — інтеграція на рівні поштового клієнта або системи обробки звернень, де рішення детектора використовується як сигнал ризику для оператора або кінцевого користувача. У межах прототипу технічна готовність до такого сценарію підтверджується наявністю стабільного механізму оцінювання на тестовій вибірці (додаток Б, рис. Б.2 для базової моделі та рис. Б.3 для моделі після LLM-aware fine-tuning) і роботою інференсу для окремого повідомлення (додаток Б, рис. Б.5).

Важливим практичним параметром є поріг класифікації, оскільки саме він визначає баланс між чутливістю до фішингу та кількістю хибнопозитивних спрацювань. У роботі цей аспект досліджено через пороговий аналіз і оцінювання на LLM-наборах (додаток Б, рис. Б.4), що дозволяє обґрунтувати вибір порога під конкретну організаційну політику. Наприклад, для середовищ із високими ризиками допустимо застосовувати більш “жорстку” політику (нижчий поріг), тоді як для мінімізації навантаження на реагування на інциденти — більш “консервативну” (вищий поріг) з додатковими правилами ручної перевірки.

Для масштабування рішення в умовах значного потоку листів доцільно розгортати детектор як окремий сервіс із чітким API (наприклад, REST), який

приймає текст повідомлення (та, за необхідності, мінімальний набір технічних параметрів) і повертає ймовірність належності до класу фішингу. У такій архітектурі критичними стають стабільність часу відповіді, пропускна здатність, контроль використання ресурсів та керованість черги запитів у пікові періоди. Практично це означає, що в продуктивному середовищі слід стандартизувати параметри токенізації (максимальна довжина, формат подання тексту) і фіксувати версію моделі, щоб уникнути “тихих” змін результатів між релізами. У прототипі це вже частково враховано через відокремлення сценаріїв навчання, оцінювання та інференсу, що дозволяє повторно отримувати результати за однакових умов (див. додаток Б, рис. Б.2–Б.3).

Оскільки вхідні повідомлення можуть містити конфіденційну інформацію, при впровадженні необхідно обмежити логування повного тексту листів і замінювати його на технічні ідентифікатори або хеші, застосовувати контроль доступу до сервісу, а також захищати канал передавання даних. З погляду якості детектування важливим є постійний моніторинг метрик та виявлення деградації моделі, що може бути пов’язано як із зміною розподілу легітимних повідомлень, так і з появою нових фішингових шаблонів.

Особливої уваги потребує аспект, пов’язаний з генеративними мовними моделями. Експерименти роботи показують, що включення LLM-згенерованих прикладів у навчання здатне змінювати профіль помилок: зменшувати ризик пропуску фішингу, але збільшувати кількість помилкових спрацювань на легітимних повідомленнях. Тому оновлення моделі в експлуатації має виконуватись контрольовано: через повторне оцінювання на відкладеному тестовому наборі (див. додаток Б, рис. Б.2–Б.3) і перевірку поведінки на LLM-наборах із варіюванням порога (див. додаток Б, рис. Б.4). Це дозволяє приймати рішення про реліз моделі не лише за “середніми” метриками, а й за реалістичним сценарієм протидії атакам, які можуть бути сформовані сучасними генеративними моделями.

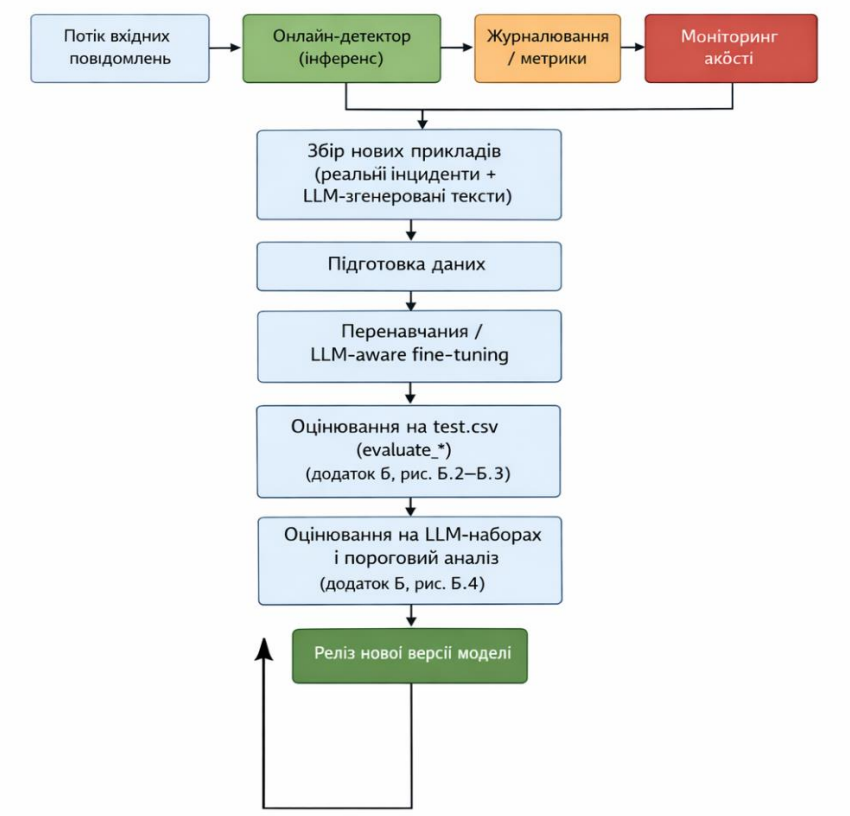

Рисунок 3.3 – Контур супроводу та оновлення детектора в умовах еволюції фішингових атак

Таблиця 3.2 – Сценарії інтеграції детектора та рекомендовані дії реагування

Сценарій	**Точка інтеграції**	**Тип реакції**	**Критичні налаштування**
Поштовий шлюз	MTA/SEG (pre-delivery)	reject/карантин, знешкодження URL/вкладень	пороги score, allowlist, fallback+timeout, аудит/логування
Поштова скринька/клієнт	правила/плагін, post-delivery	маркування/банер, переміщення в Junk, “Report”	пороги для UI, винятки VIP, privacy-redaction, фідбек для датасету
Helpdesk/SOC	тікет-система/SOC	автотікет, пріоритизація, ескалація	дедуп, SLA, шаблон полів (score/IOC), правила тригерів
SIEM/SOAR	конектор подій	кореляція, плейбуки (block/recall)	нормалізація полів, автодії лише high-conf, ліміти впливу, версія моделі

Підсумовуючи, розроблений прототип має достатній рівень завершеності для демонстрації практичної застосовності: він відтворювано оцінюється на тестовому наборі, підтримує перевірку на LLM-згенерованих повідомленнях та аналіз порога класифікації, а також забезпечує інференс для окремих повідомлень (додаток Б, рис. Б.2–Б.5). У поєднанні з результатами розділу 3.3 це дозволяє обґрунтувати ключовий висновок практичної частини: урахування LLM-згенерованих прикладів є дієвим інструментом підвищення стійкості детектора до сучасних атак, однак потребує контрольованого супроводу, вибору порога і періодичної переоцінки якості перед впровадженням нової версії моделі в продуктивне середовище.

ВИСНОВКИ

У магістерській роботі виконано дослідження впливу генеративних мовних моделей на формування фішингових повідомлень та розроблено прототип системи їх автоматизованого детектування. Актуальність теми зумовлена одночасним зростанням масштабів фішингових атак і появою нових можливостей їх персоналізації та стилістичного маскування за рахунок використання сучасних генеративних моделей, що ускладнює виявлення загроз традиційними засобами.

У межах теоретичної частини роботи систематизовано уявлення про архітектурні принципи та особливості функціонування генеративних мовних моделей, а також окреслено роль трансформерних підходів у сучасних задачах аналізу тексту. Розглянуто основні форми фішингу та канали поширення атак, включаючи електронну пошту, SMS-повідомлення, месенджери та голосові сценарії. Окрему увагу приділено традиційним підходам до детектування фішингу (спискові методи, евристики, контент-орієнтований аналіз, клієнтські засоби та організаційні заходи) та показано їхні типові обмеження в умовах швидкої адаптації зловмисників і появи “нульового дня” у контенті повідомлень.

Практична частина роботи зосереджена на побудові та апробації прототипу детектора фішингових повідомлень на основі трансформерної моделі-класифікатора. Реалізовано програмний контур підготовки даних, навчання, оцінювання та інференсу; результати ключових запусків і приклади роботи скриптів наведено у додатках (додаток Б, рис. Б.1–Б.5).

Експериментально оцінено якість базової моделі та моделі після донавчання із врахуванням LLM-згенерованих прикладів. Для базової моделі зафіксовано високі показники якості на тестовому наборі (зокрема наведено матрицю похибок та ROC-AUC у додатку Б.2), що підтверджує ефективність трансформерного підходу для задачі бінарної класифікації “phishing/legit” у межах обраного набору даних. Разом із тим, оцінювання на LLM-наборі показало, що частина згенерованих фішингових текстів може мати нижчу впевненість класифікації або

змінювати частку спрацювань залежно від порога прийняття рішення; відповідні результати та пороговий аналіз наведено в додатку Б.4. Це є важливим практичним підтвердженням того, що генеративні моделі здатні формувати повідомлення, які частково обходять детектори, навчені виключно на “традиційних” прикладах.

Для підвищення стійкості до таких сценаріїв реалізовано LLM-aware fine-tuning — донавчання моделі з урахуванням LLM-згенерованих повідомлень. Повторне оцінювання показало зміну профілю помилок: зменшення ризику пропуску фішингових повідомлень при одночасному збільшенні кількості хибнопозитивних спрацювань на легітимних повідомленнях (додаток Б.3). Таким чином, у роботі продемонстровано практично значущий компроміс між чутливістю до фішингу та специфічністю, що прямо впливає на організаційні політики впровадження (вибір порога, правила карантину/маркування, сценарії ручної перевірки).

Науково-практичний результат роботи полягає у побудові відтворюваної експериментальної схеми, яка дозволяє оцінювати детектори фішингу в умовах появи LLM-згенерованого контенту, а також у розробці прототипу детектора з можливістю перевірки на LLM-наборах і порогового аналізу. Запропонована архітектура та реалізація можуть бути використані як основа для подальшого розвитку прикладних рішень у корпоративних середовищах, зокрема для інтеграції у поштові шлюзи, системи моніторингу безпеки та контури реагування на інциденти.

Напрями подальших досліджень і розвитку системи включають розширення та різноманітнення наборів даних (у тому числі за рахунок більш репрезентативних прикладів цільових атак), вдосконалення методів калібрування ймовірностей і вибору порога за цільовими функціями організації, зменшення частки хибнопозитивних спрацювань після LLM-aware донавчання, а також поєднання текстових ознак із технічними характеристиками повідомлень і контекстом (заголовки, домени, URL-поведінка) для підвищення інтерпретованості та стійкості системи до адаптивних атак.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I. Attention Is All You Need // Advances in Neural Information Processing Systems (NIPS). – 2017.

2. Bahdanau D., Cho K., Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate // Proc. of the International Conference on Learning Representations (ICLR). – 2015.

3. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT). – 2019.

4. Gehring J., Auli M., Grangier D., Yarats D., Dauphin Y. N. Convolutional Sequence to Sequence Learning // Proc. of the 34th International Conference on Machine Learning (ICML). – 2017.

5. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition // Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2016.

6. Ba J. L., Kiros J. R., Hinton G. E. Layer Normalization. – Technical report, 2016.

7. Radford A., Narasimhan K., Child R., Sutskever I. Language Models as Unsupervised Multitask Learners. – OpenAI, 2019.

8. Brown T. B., Mann B., Ryder N. та ін. Language Models are Few-Shot Learners // Advances in Neural Information Processing Systems (NeurIPS). – 2020.

9. Touvron H., Lavril T., Izacard G. та ін. LLaMA: Open and Efficient Foundation Language Models. – Meta AI, 2023.

10. ENISA. ENISA Threat Landscape 2024. – European Union Agency for Cybersecurity, 2024.

11. Anti-Phishing Working Group. Phishing Activity Trends Report: 4th Quarter 2024. – 2025.

12. Verizon. 2024 Data Breach Investigations Report (DBIR). – Verizon, 2024.

13. ENISA. Threat Landscape – Finance Sector 2024. – European Union Agency for Cybersecurity, 2025.

14. Dawkins S., Jacobs J. NIST Phish Scale User Guide. – NIST Technical Note 2276. – National Institute of Standards and Technology, 2023. DOI: https://doi.org/10.6028/NIST.TN.2276 .

15. CISA. Counter-Phishing Recommendations for Federal Agencies. – Cybersecurity and Infrastructure Security Agency, 2023.

16. Luo E., Shuey D., Oest A., Thomas K., Gustafson E., Doupé A., Ahn G.-J. Characterizing the Networks Sending Enterprise Phishing Emails // Passive and Active Measurement (PAM 2025). – Lecture Notes in Computer Science. – 2025. – P. 437–466. – DOI: https://doi.org/10.1007/978-3-031-85960-1_18 .

17. Abdolrazzagh-Nezhad M., Langari N. Phishing Detection Techniques: A Review // Journal of Computational Artificial Intelligence (JoCAI).– 2025.

18. Safi A., Singh S. A Systematic Literature Review on Phishing Website Detection Techniques // Journal of King Saud University – Computer and Information Sciences.– 2023. – Vol. 35, Iss. 2. – P. 590–611. – DOI: https://doi.org/10.1016/j.jksuci.2023.01.004 .

19. Aboagye B. B., Agyemang F. O., Asare-Boateng K.Malicious URL Website Detection Using Ensemble Machine Learning Approach // International Journal of Science and Research Archive.– 2025. – Vol. 14, No. 3. – P. 1614–1622. – DOI: https://doi.org/10.30574/ijsra.2025.14.3.0857 .

20. Bhadani D. Heuristic-Based Phishing Site Detection. – Master's Thesis. – California State University, 2023.

21. Alghenaim M. F., Abu Bakar N., Abdul Rahim N. A. Anti-Phishing Tools: State of the Art and Detection Efficiencies // Applied Mathematics & Information Sciences. – 2022. – Vol. 16, No. 6. – P. 929–934. – DOI: https://doi.org/10.18576/amis/160609 .

22. NIST. Phishing – Cybersecurity for Small Business. – National Institute of Standards and Technology, 2021.

23. Abdul Aleem A., Huda N., Siddiqi M. Phishing Mitigation Techniques: A Literature Survey // International Journal of Network Security & Its Applications. – 2021– DOI: https://doi.org/10.5121/ijnsa.2021.13205 .

24. Kyaw P. H., Aung T. T., Oo Z. L., Tun T. T. A Systematic Review of Deep Learning Techniques for Phishing Detection // Electronics. – 2024. – Vol. 13, No. 19. – Art. 3823. – DOI: https://doi.org/10.3390/electronics13193823 .

25. Ahammad S. K. H. Phishing URL detection using machine learning methods // Advances in Engineering Software.– 2022. – Vol. 173. – Art. 103288. – DOI: https://doi.org/10.1016/j.advengsoft.2022.103288 .

26. Wilk-Jakubowski J. Ł., Pawlik Ł., Wilk-Jakubowski G., Sikora A. Machine Learning and Neural Networks for Phishing Detection: A Systematic Review (2017–2024) // Electronics.– 2025. – Vol. 14, No. 18. – Art. 3744. – DOI: https://doi.org/10.3390/electronics14183744 .

27. Jamal S., Wimmer H., Sarker I. H. An improved transformer-based model for detecting phishing emails // Security and Privacy.– 2024. – Vol. 7, Iss. 5. – e402. – DOI: https://doi.org/10.1002/spy2.402 .

28. Songailaitė M., Kankevičiūtė E., Zhyhun B., Mandravickaitė J.BERT-Based Models for Phishing Detection // CEUR Workshop Proceedings.– 2023. – Vol. 3575. – P. 34–44.

29. Calzarossa M. C., Giudici P., Zieni R.Explainable machine learning for phishing feature detection // Quality and Reliability Engineering International.– 2024. – Vol. 40, Iss. 1. – P. 362–373. – DOI: https://doi.org/10.1002/qre.3411 .

30. Uddin M. A., Sarker I. H.An Explainable Transformer-Based Model for Phishing Email Detection: A Large Language Model Approach. – SSRN, 2024.– DOI: https://doi.org/10.2139/ssrn.4785953 .

31. A. I. Champa, M. F. Rabbi, and M. F. Zibran, “Why phishing emails escape detection: A closer look at the failure points,” in *12th International Symposium on Digital Forensics and Security (ISDFS)*, 2024, pp. 1–6.

32. A. I. Champa, M. F. Rabbi, and M. F. Zibran, “Curated datasets and feature analysis for phishing email detection with machine learning,” in 3rd IEEE International Conference on Computing and Machine Intelligence (ICMI), 2024, pp. 1–7.

ДОДАТОК А

Лістинг програмного коду

Лістинг А.1 – data_prep.py:

```
import os
import pandas as pd
from sklearn.model_selection import train_test_split

RAW = r"data/raw/Nazario_5.csv"
OUT_DIR = r"data/processed"

def main():
    os.makedirs(OUT_DIR, exist_ok=True)

    df = pd.read_csv(RAW)

    df["subject"] = df["subject"].fillna("")
    df["body"] = df["body"].fillna("")

    df["text"] = (df["subject"].astype(str).str.strip() + "\n" +
df["body"].astype(str).str.strip()).str.strip()
    df["label"] = df["label"].astype(int)

    dataset = df[["text", "label"]].copy()

    train_df, temp_df = train_test_split(
        dataset, test_size=0.2, random_state=42, stratify=dataset["label"]
    )
    val_df, test_df = train_test_split(
        temp_df, test_size=0.5, random_state=42, stratify=temp_df["label"]
    )

    train_df.to_csv(os.path.join(OUT_DIR, "train.csv"), index=False)
    val_df.to_csv(os.path.join(OUT_DIR, "val.csv"), index=False)
    test_df.to_csv(os.path.join(OUT_DIR, "test.csv"), index=False)

    print("Saved:", train_df.shape, val_df.shape, test_df.shape)
    print("Train label dist:\n", train_df["label"].value_counts())
```

```
if name == "main":
    main()
```

Лістинг А.2 – train.py

```
import os
import pandas as pd
import torch
from torch.utils.data import Dataset, DataLoader
from transformers import AutoTokenizer, AutoModelForSequenceClassification
from transformers import get_linear_schedule_with_warmup
from sklearn.metrics import classification_report, roc_auc_score

MODEL_NAME = "distilbert-base-uncased"
TRAIN_CSV = r"data/processed/train.csv"
VAL_CSV = r"data/processed/val.csv"
OUT_DIR = r"models/distilbert_nazario"

MAX_LEN = 256
BATCH_SIZE = 8
EPOCHS = 2
LR = 2e-5


class TextDataset(Dataset):
    def init(self, df, tokenizer, max_len):
        self.texts = df["text"].astype(str).tolist()
        self.labels = df["label"].astype(int).tolist()
        self.tokenizer = tokenizer
        self.max_len = max_len

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

    def getitem(self, idx):
        enc = self.tokenizer(
            self.texts[idx],
            truncation=True,
            padding="max_length",
            max_length=self.max_len,
            return_tensors="pt",
        )
```

```
        item = {k: v.squeeze(0) for k, v in enc.items()}
        item["labels"] = torch.tensor(self.labels[idx], dtype=torch.long)
        return item


def evaluate(model, loader, device):
    model.eval()
    all_labels = []
    all_probs = []
    all_preds = []
    with torch.no_grad():
        for batch in loader:
            batch = {k: v.to(device) for k, v in batch.items()}
            outputs = model(**batch)
            probs = torch.softmax(outputs.logits, dim=1)[:,
1].detach().cpu().numpy()
            preds = outputs.logits.argmax(dim=1).detach().cpu().numpy()
            labels = batch["labels"].detach().cpu().numpy()

            all_labels.extend(labels.tolist())
            all_probs.extend(probs.tolist())
            all_preds.extend(preds.tolist())

    report = classification_report(all_labels, all_preds, digits=4)
    try:
        auc = roc_auc_score(all_labels, all_probs)
    except Exception:
        auc = None
    return report, auc


def main():
    os.makedirs(OUT_DIR, exist_ok=True)
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    print("Device:", device)

    train_df = pd.read_csv(TRAIN_CSV)
    val_df = pd.read_csv(VAL_CSV)

    tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
    model = AutoModelForSequenceClassification.from_pretrained(MODEL_NAME,
num_labels=2)
```

```
    model.to(device)

    train_ds = TextDataset(train_df, tokenizer, MAX_LEN)
    val_ds = TextDataset(val_df, tokenizer, MAX_LEN)

    train_loader = DataLoader(train_ds, batch_size=BATCH_SIZE, shuffle=True)
    val_loader = DataLoader(val_ds, batch_size=BATCH_SIZE, shuffle=False)

    optimizer = torch.optim.AdamW(model.parameters(), lr=LR)
    total_steps = len(train_loader) * EPOCHS
    scheduler = get_linear_schedule_with_warmup(
        optimizer, num_warmup_steps=int(0.1 * total_steps),
num_training_steps=total_steps
    )

    best_auc = -1.0
    for epoch in range(1, EPOCHS + 1):
        model.train()
        total_loss = 0.0

        for batch in train_loader:
            batch = {k: v.to(device) for k, v in batch.items()}
            outputs = model(**batch)
            loss = outputs.loss

            optimizer.zero_grad()
            loss.backward()
            optimizer.step()
            scheduler.step()

            total_loss += loss.item()

        avg_loss = total_loss / max(1, len(train_loader))
        print(f"Epoch {epoch}/{EPOCHS} - train loss: {avg_loss:.4f}")

        report, auc = evaluate(model, val_loader, device)
        print("Validation report:\n", report)
        print("Validation ROC-AUC:", auc)

        if auc is not None and auc > best_auc:
            best_auc = auc
            model.save_pretrained(OUT_DIR)
```

```
            tokenizer.save_pretrained(OUT_DIR)
            print("Saved best model to:", OUT_DIR)

    print("Done. Best val ROC-AUC:", best_auc)


if name == "main":
    main()
```

Лістинг А.3 – infer.py

```
import sys
import torch
from transformers import AutoTokenizer, AutoModelForSequenceClassification

MODEL_DIR = r"models/distilbert_nazario"
THRESHOLD = 0.5

def main():
    text = None
    if len(sys.argv) > 1:
        text = " ".join(sys.argv[1:])
    else:
        print("Paste email text (finish with Ctrl+Z then Enter):")
        text = sys.stdin.read()

    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    tokenizer = AutoTokenizer.from_pretrained(MODEL_DIR)
    model = AutoModelForSequenceClassification.from_pretrained(MODEL_DIR)
    model.to(device)
    model.eval()

    enc = tokenizer(text, truncation=True, padding=True, max_length=256,
return_tensors="pt")
    enc = {k: v.to(device) for k, v in enc.items()}

    with torch.no_grad():
        logits = model(**enc).logits
        prob_phish = torch.softmax(logits, dim=1)[0, 1].item()

    verdict = "PHISHING" if prob_phish >= THRESHOLD else "LEGIT"
```

```
    print(f"Verdict: {verdict}")
    print(f"P(phishing) = {prob_phish:.4f}")

if name == "main":
    main()
```

Лістинг А.4 – evaluate_base.py

```
import pandas as pd
import torch
from torch.utils.data import Dataset, DataLoader
from transformers import AutoTokenizer, AutoModelForSequenceClassification
from sklearn.metrics import classification_report, roc_auc_score,
confusion_matrix

MODEL_DIR = r"models/distilbert_nazario"
TEST_CSV = r"data/processed/test.csv"

MAX_LEN = 256
BATCH_SIZE = 16

class TextDataset(Dataset):
    def init(self, df, tokenizer, max_len):
        self.texts = df["text"].astype(str).tolist()
        self.labels = df["label"].astype(int).tolist()
        self.tokenizer = tokenizer
        self.max_len = max_len

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

    def getitem(self, idx):
        enc = self.tokenizer(
            self.texts[idx],
            truncation=True,
            padding="max_length",
            max_length=self.max_len,
            return_tensors="pt",
        )
        item = {k: v.squeeze(0) for k, v in enc.items()}
        item["labels"] = torch.tensor(self.labels[idx], dtype=torch.long)
        return item
```

```
def main():
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    print("Device:", device)

    df = pd.read_csv(TEST_CSV)
    tokenizer = AutoTokenizer.from_pretrained(MODEL_DIR)
    model = AutoModelForSequenceClassification.from_pretrained(MODEL_DIR)
    model.to(device)
    model.eval()

    ds = TextDataset(df, tokenizer, MAX_LEN)
    loader = DataLoader(ds, batch_size=BATCH_SIZE, shuffle=False)

    all_labels, all_probs, all_preds = [], [], []

    with torch.no_grad():
        for batch in loader:
            batch = {k: v.to(device) for k, v in batch.items()}
            out = model(**batch)
            probs = torch.softmax(out.logits, dim=1)[:, 1].detach().cpu().numpy()
            preds = out.logits.argmax(dim=1).detach().cpu().numpy()
            labels = batch["labels"].detach().cpu().numpy()

            all_labels.extend(labels.tolist())
            all_probs.extend(probs.tolist())
            all_preds.extend(preds.tolist())

    print("Test report:\n", classification_report(all_labels, all_preds,
digits=4))
    cm = confusion_matrix(all_labels, all_preds, labels=[0, 1])
    print("Confusion matrix (rows=true, cols=pred):\n", cm)
    tn, fp, fn, tp = cm.ravel()
    print(f"TN={tn} FP={fp} FN={fn} TP={tp}")
    print("Test ROC-AUC:", roc_auc_score(all_labels, all_probs))

if name == "main":
    main()
```

Лістинг А.5 – evaluate_llmft.py

```
import pandas as pd
import torch
```

```
from torch.utils.data import Dataset, DataLoader
from transformers import AutoTokenizer, AutoModelForSequenceClassification
from sklearn.metrics import classification_report, roc_auc_score,
confusion_matrix


MODEL_DIR = r"models/distilbert_nazario_llmft"
TEST_CSV = r"data/processed/test.csv"

MAX_LEN = 256
BATCH_SIZE = 16

class TextDataset(Dataset):
    def init(self, df, tokenizer, max_len):
        self.texts = df["text"].astype(str).tolist()
        self.labels = df["label"].astype(int).tolist()
        self.tokenizer = tokenizer
        self.max_len = max_len

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

    def getitem(self, idx):
        enc = self.tokenizer(
            self.texts[idx],
            truncation=True,
            padding="max_length",
            max_length=self.max_len,
            return_tensors="pt",
        )
        item = {k: v.squeeze(0) for k, v in enc.items()}
        item["labels"] = torch.tensor(self.labels[idx], dtype=torch.long)
        return item

def main():
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    print("Device:", device)

    df = pd.read_csv(TEST_CSV)
    tokenizer = AutoTokenizer.from_pretrained(MODEL_DIR)
    model = AutoModelForSequenceClassification.from_pretrained(MODEL_DIR)
    model.to(device)
```

```
    model.eval()

    ds = TextDataset(df, tokenizer, MAX_LEN)
    loader = DataLoader(ds, batch_size=BATCH_SIZE, shuffle=False)

    all_labels, all_probs, all_preds = [], [], []

    with torch.no_grad():
        for batch in loader:
            batch = {k: v.to(device) for k, v in batch.items()}
            out = model(**batch)
            probs = torch.softmax(out.logits, dim=1)[:, 1].detach().cpu().numpy()
            preds = out.logits.argmax(dim=1).detach().cpu().numpy()
            labels = batch["labels"].detach().cpu().numpy()

            all_labels.extend(labels.tolist())
            all_probs.extend(probs.tolist())
            all_preds.extend(preds.tolist())

    print("Test report:\n", classification_report(all_labels, all_preds,
digits=4))
    cm = confusion_matrix(all_labels, all_preds, labels=[0, 1])
    print("Confusion matrix (rows=true, cols=pred):\n", cm)
    tn, fp, fn, tp = cm.ravel()
    print(f"TN={tn} FP={fp} FN={fn} TP={tp}")
    print("Test ROC-AUC:", roc_auc_score(all_labels, all_probs))

if name == "main":
    main()
```

Лістинг А.6 – make_llm_subsets.py

```
import os
import pandas as pd
import random

IN_CSV = r"data/generated/llm_phish_clean.csv"
OUT_ART = r"data/generated/llm_phish_artifacts.csv"

random.seed(42)

ADD_TAILS = [
    "Action: sign in to the secure portal to continue. URL:
hxxps://example[.]com/verify",
```

```
    "Please confirm using a one-time code (OTP) in the portal. URL:
hxxps://example[.]com/auth",
    "Open the secure viewer to review the document. URL:
hxxps://example[.]com/view",
    "Verify your account details to avoid restriction. URL:
hxxps://example[.]com/login",
    "Update payment settings in the billing portal. URL:
hxxps://example[.]com/billing",
]

def main():
    df = pd.read_csv(IN_CSV)
    if "text" not in df.columns:
        raise ValueError("Input CSV must have a 'text' column")

    out = df.copy()
    new_texts = []
    for t in out["text"].astype(str).tolist():
        tail = random.choice(ADD_TAILS)
        new_texts.append(f"{t}\n\n{tail}")

    out["text"] = new_texts
    os.makedirs(os.path.dirname(OUT_ART), exist_ok=True)
    out.to_csv(OUT_ART, index=False, encoding="utf-8")
    print("Saved:", OUT_ART, "rows:", len(out))

if name == "main":
    main()
```

Лістинг А.7 – llm_eval.py

```
import os
import sys
import pandas as pd
import torch
from transformers import AutoTokenizer, AutoModelForSequenceClassification

MODEL_DIR = r"models/distilbert_nazario_llmft"
THRESHOLD = 0.5
MAX_LEN = 256

def score_file(in_csv: str):
    out_csv = os.path.splitext(in_csv)[0] + "_scored.csv"
```

```
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    print("Device:", device)
    print("Input:", in_csv)

    df = pd.read_csv(in_csv)
    if "text" not in df.columns:
        raise ValueError("CSV must have a 'text' column")

    tokenizer = AutoTokenizer.from_pretrained(MODEL_DIR)
    model =
AutoModelForSequenceClassification.from_pretrained(MODEL_DIR).to(device)
    model.eval()

    probs = []
    with torch.no_grad():
        for t in df["text"].astype(str).tolist():
            enc = tokenizer(t, truncation=True, padding=True, max_length=MAX_LEN,
return_tensors="pt")
            enc = {k: v.to(device) for k, v in enc.items()}
            logits = model(**enc).logits
            p = torch.softmax(logits, dim=1)[0, 1].item()
            probs.append(p)

    df["p_phishing"] = probs
    df["predicted"] = (df["p_phishing"] >= THRESHOLD).astype(int)

    n = len(df)
    share = df["predicted"].mean()
    print(f"LLM set size: {n}")
    print(f"Share predicted as phishing (thr={THRESHOLD}): {share:.3f}")
    print(f"Mean P(phishing): {df['p_phishing'].mean():.4f}")
    print(f"Median P(phishing): {df['p_phishing'].median():.4f}")

    df.to_csv(out_csv, index=False, encoding="utf-8")
    print("Saved scored file to:", out_csv)

def main():
    in_csv = sys.argv[1] if len(sys.argv) > 1 else
r"data/generated/llm_phish_clean.csv"
    score_file(in_csv)
```

```
if name == "main":
    main()
```

Лістинг А.8 – threshold_sweep.py

```
import pandas as pd

FILES = [
    ("LLM-clean",     r"data/generated/llm_phish_clean_scored.csv"),
    ("LLM-artifacts", r"data/generated/llm_phish_artifacts_scored.csv"),
]

THRESHOLDS = [0.3, 0.4, 0.5, 0.6]

def main():
    print("Threshold sweep (share predicted as PHISHING)")
    print("-" * 60)
    header = "set".ljust(14) + "".join([f"thr={t}".rjust(12) for t in
THRESHOLDS])
    print(header)

    for name, path in FILES:
        df = pd.read_csv(path)
        probs = df["p_phishing"]
        row = name.ljust(14)
        for t in THRESHOLDS:
            share = (probs >= t).mean()
            row += f"{share:12.3f}"
        print(row)

if name == "main":
    main()
```

Лістинг А.9 – split_llm.py

```
import os
import pandas as pd

IN_CSV = r"data/generated/llm_phish_artifacts.csv"
OUT_TRAIN = r"data/generated/llm_phish_artifacts_train.csv"
OUT_HOLD  = r"data/generated/llm_phish_artifacts_holdout.csv"

def main():
```

```
    df = pd.read_csv(IN_CSV)
    if "text" not in df.columns:
        raise ValueError("CSV must have a 'text' column")

    df = df.sample(frac=1.0, random_state=42).reset_index(drop=True)

    n = len(df)
    n_train = n // 2
    df_train = df.iloc[:n_train].copy()
    df_hold  = df.iloc[n_train:].copy()

    os.makedirs(os.path.dirname(OUT_TRAIN), exist_ok=True)
    df_train.to_csv(OUT_TRAIN, index=False, encoding="utf-8")
    df_hold.to_csv(OUT_HOLD, index=False, encoding="utf-8")

    print("Saved:", OUT_TRAIN, "rows:", len(df_train))
    print("Saved:", OUT_HOLD,  "rows:", len(df_hold))

if name == "main":
    main()
```

Лістинг А.10 – finetune_llm.py

```
import os
import pandas as pd
import numpy as np
import torch
from torch.utils.data import Dataset, DataLoader
from transformers import AutoTokenizer, AutoModelForSequenceClassification
from sklearn.metrics import classification_report, roc_auc_score

BASE_MODEL_DIR = r"models/distilbert_nazario"
PROC_TRAIN_CSV = r"data/processed/train.csv"      # має містити text,label
TEST_CSV       = r"data/processed/test.csv"       # для контролю якості
LLM_TRAIN_CSV  = r"data/generated/llm_phish_artifacts_train.csv"

OUT_MODEL_DIR  = r"models/distilbert_nazario_llmft"

MAX_LEN = 256
BATCH = 8
EPOCHS = 1
LR = 2e-5
```

```
class TextDS(Dataset):
    def init(self, texts, labels, tokenizer):
        self.texts = list(texts)
        self.labels = list(labels)
        self.tok = tokenizer
    def len(self): return len(self.texts)
    def getitem(self, i):
        enc = self.tok(
            str(self.texts[i]),
            truncation=True,
            padding="max_length",
            max_length=MAX_LEN,
            return_tensors="pt",
        )
        item = {k: v.squeeze(0) for k, v in enc.items()}
        item["labels"] = torch.tensor(int(self.labels[i]), dtype=torch.long)
        return item

@torch.no_grad()
def eval_on_csv(model, tokenizer, csv_path, device):
    df = pd.read_csv(csv_path)
    texts = df["text"].astype(str).tolist()
    y_true = df["label"].astype(int).tolist()

    probs = []
    preds = []
    model.eval()
    for t in texts:
        enc = tokenizer(t, truncation=True, padding=True, max_length=MAX_LEN,
return_tensors="pt")
        enc = {k: v.to(device) for k, v in enc.items()}
        logits = model(**enc).logits
        p = torch.softmax(logits, dim=1)[0, 1].item()
        probs.append(p)
        preds.append(1 if p >= 0.5 else 0)

    try:
        auc = roc_auc_score(y_true, probs)
    except Exception:
        auc = None
```

```
    print("\nEVAL:", csv_path)
    print(classification_report(y_true, preds, digits=4))
    if auc is not None:
        print("ROC-AUC:", auc)

def main():
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    print("Device:", device)

    # 1) LLM phishing train (yci label=1)
    llm = pd.read_csv(LLM_TRAIN_CSV)
    llm_phish = llm[["text"]].copy()
    llm_phish["label"] = 1

    # 2) Legit samples з основного train (label=0), стільки ж, як LLM phishing
    base_train = pd.read_csv(PROC_TRAIN_CSV)
    if not {"text", "label"}.issubset(base_train.columns):
        raise ValueError("data/processed/train.csv must have columns:
text,label")

    legit = base_train[base_train["label"] == 0].sample(n=len(llm_phish),
random_state=42)

    mix = pd.concat([llm_phish, legit[["text", "label"]]], ignore_index=True)
    mix = mix.sample(frac=1.0, random_state=42).reset_index(drop=True)
    print("Fine-tune set size:", len(mix), " (LLM phish:", len(llm_phish), ",
legit:", len(legit), ")")

    tokenizer = AutoTokenizer.from_pretrained(BASE_MODEL_DIR)
    model =
AutoModelForSequenceClassification.from_pretrained(BASE_MODEL_DIR).to(device)

    ds = TextDS(mix["text"].tolist(), mix["label"].tolist(), tokenizer)
    dl = DataLoader(ds, batch_size=BATCH, shuffle=True)

    optim = torch.optim.AdamW(model.parameters(), lr=LR)

    model.train()
    for ep in range(EPOCHS):
        total = 0.0
        for batch in dl:
            batch = {k: v.to(device) for k, v in batch.items()}
```

```
            out = model(**batch)
            loss = out.loss
            loss.backward()
            optim.step()
            optim.zero_grad()
            total += loss.item()
        print(f"Epoch {ep+1}/{EPOCHS} - train loss: {total/len(dl):.4f}")

    # контроль: на звичайному test (щоб не зламати базову якість)
    eval_on_csv(model, tokenizer, TEST_CSV, device)

    os.makedirs(OUT_MODEL_DIR, exist_ok=True)
    model.save_pretrained(OUT_MODEL_DIR)
    tokenizer.save_pretrained(OUT_MODEL_DIR)
    print("Saved fine-tuned model to:", OUT_MODEL_DIR)

if name == "main":
    main()
```

ДОДАТОК Б

Фрагменти запуску програмних модулів та результати експериментів

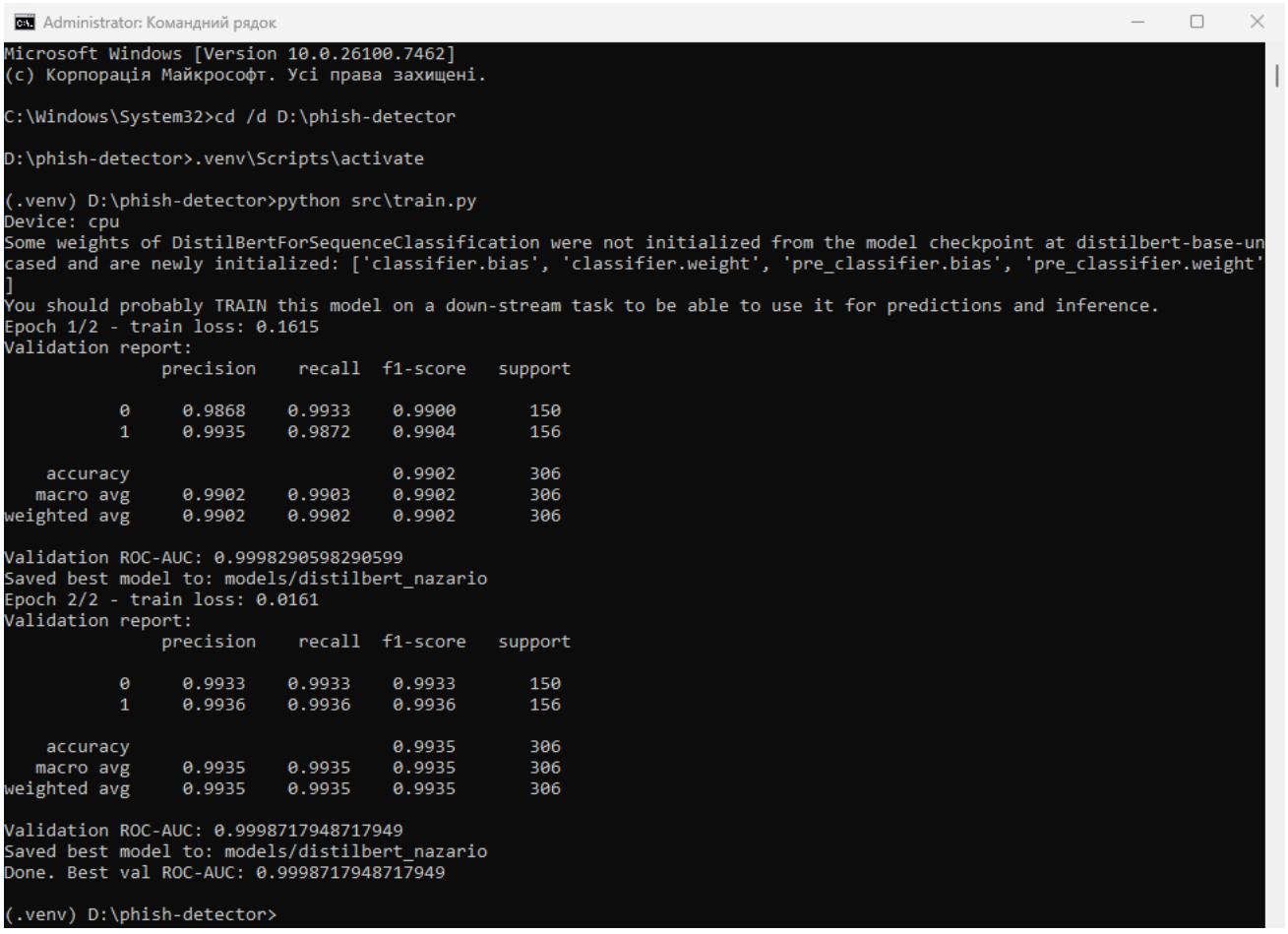

Рисунок Б.1 – Приклад журналу навчання та валідації моделі (train.py)

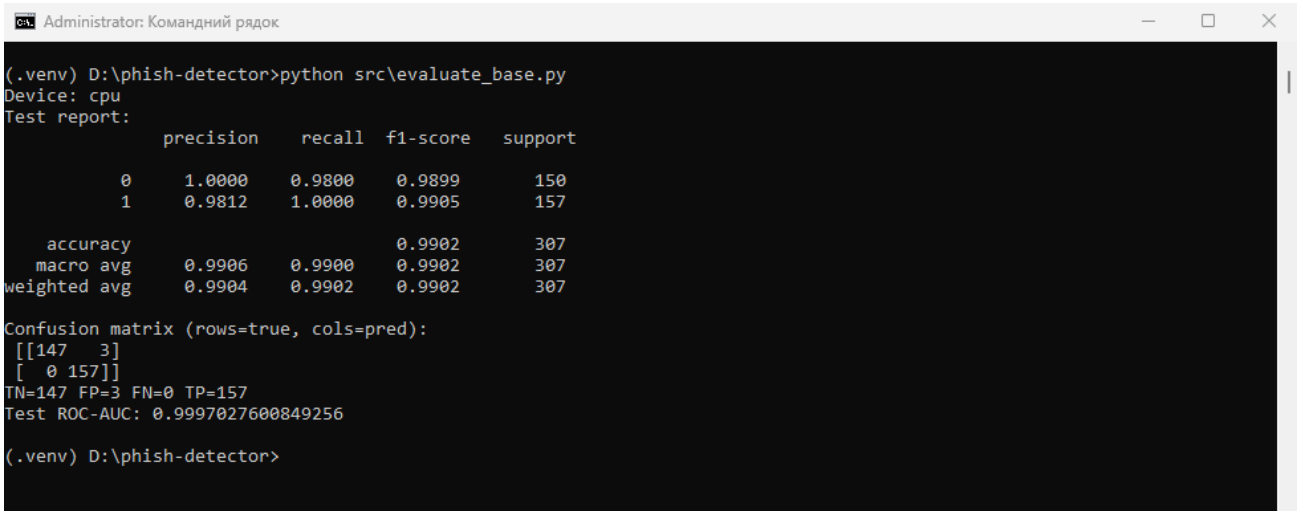

Рисунок Б.2 – Оцінювання базової моделі на тестовій вибірці (evaluate_base.py)

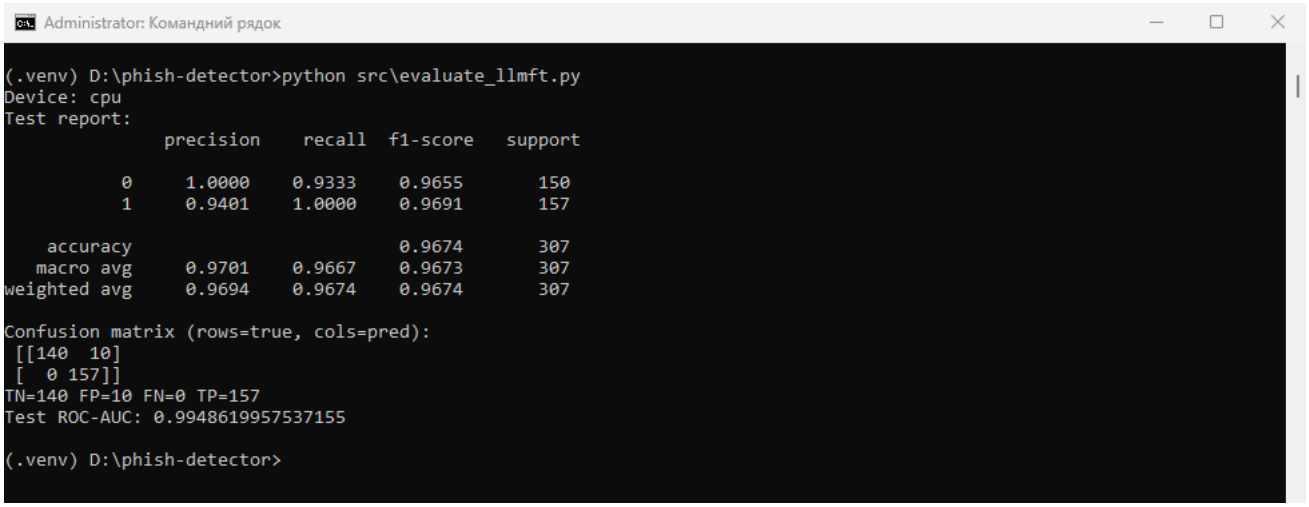

Рисунок Б.3 – Оцінювання моделі після донавчання на LLM-наборі (evaluate_llmft.py)

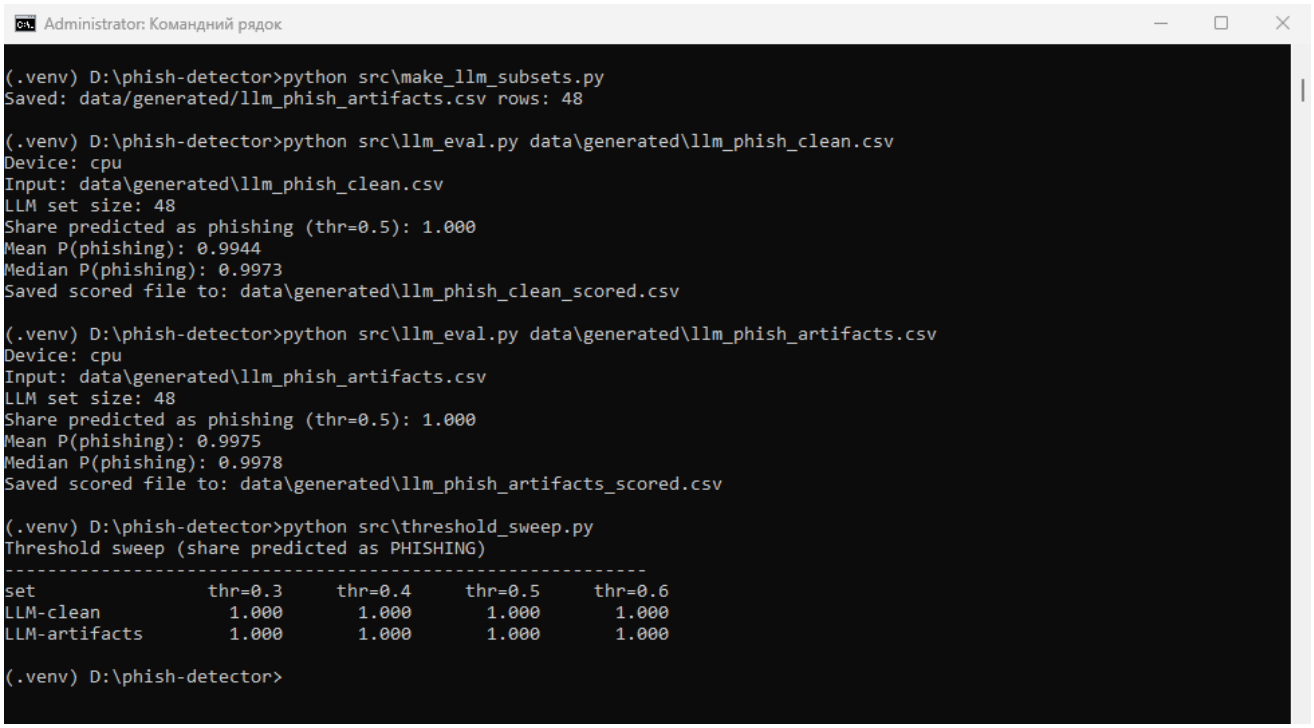

Рисунок Б.4 – Оцінювання на LLM-згенерованих повідомленнях та аналіз порогів (llm_eval.py, threshold_sweep.py)

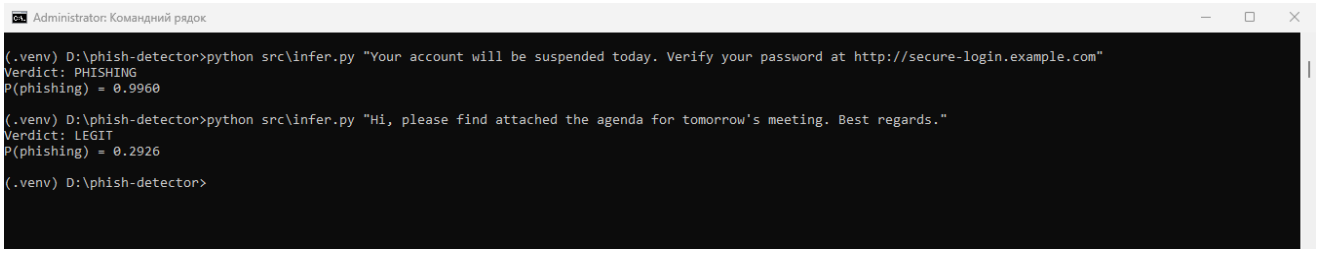

Рисунок Б.5 – Приклад інференсу для окремих повідомлень (infer.py)