Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(освітній рівень)

на тему: "Виявлення фішингових повідомлень за допомогою LLM"

Виконав: студент VI курсу, групи СБм-61

Спеціальності:
125 «Кібербезпека та захист інформації»
(шифр і назва напряму підготовки, спеціальності)
Вихованець Дмитро Володимирович
підпис (прізвище та ініціали)

Керівник Стадник М. А.
підпис (прізвище та ініціали)

Нормоконтроль Стадник М. А.
підпис (прізвище та ініціали)

Завідувач кафедри Загородна Н.В.
підпис (прізвище та ініціали)

Рецензент
підпис (прізвище та ініціали)

м. Тернопіль – 2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека та захист інформації
(шифр і назва спеціальності)

Студенту Вихованець Дмитру Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Виявлення фішингових повідомлень за допомогою LLM

Керівник роботи Стадник Марія Андріївна, к.т.н., доцент кафедри КБ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «24» 11 2025 року № 4/7-1024

2. Термін подання студентом завершеної роботи 12.12.2025

3. Вихідні дані до роботи Набір даних фішингових повідомлень Phishing and Enron Email Dataset

4. Зміст роботи (перелік питань, які потрібно розробити)

1 Теоретичні основі фішингу та методів його виявлення. 1.1 Проблематика виявлення фішингових атак в умовах їх еволюційного ускладнення. 1.2 Класифікація фішингових атак. 1.3 Сучасні підходи та методи виявлення фішингу. 2 Оцінка потенціалу великих мовних моделей для виявлення фішингових повідомлень. 2.1 Концептуальна основа LLM. 2.2 Обґрунтування вибору методу дослідження. 2.3 Архітектура LLM. 2.3.1 Енкодер та декодер. 2.3.2 Механізм Attention 2.4 Попереднє навчання та fine-tuning LLM. 3 Виявлення фішингових повідомлень з використанням llm. 3.1 Постановка задачі та загальна схема експериментального дослідження. 3.1 Характеристика досліджуваного датасету. 3.3 Реалізація та fine-tuning LLM.3.3.1 Реалізація класифікації фішингових повідомлень за допомогою GPT. 3.3.2 Реалізація та fine-tuning моделі LLaMA для класифікації фішингових повідомлень. 3.3.3 Реалізація та fine-tuning моделі LLaMA для класифікації фішингових повідомлень. 3.4 Ідентифікація фішингових повідомлень з використанням класичних алгоритмів машинного навчання. 3.4 Порівняльний аналіз результатів виявлення фішингових повідомлень. РОЗДІЛ 4 Охорона праці та безпека в надзвичайних ситуаціях. 4.1 Охорона праці. 4.2 Захист людини від іонізуючого випромінювання. Висновки

5. Перелік графічного матеріалу.

Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г.М., к.т.н., доцент		
Безпека в надзвичайних ситуаціях	Теслюк В.М., проректор з адміністративно-господарської роботи та будівництва		

7. Дата видачі завдання 19.09.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	19.09 – 22.09	Виконано
2.	Опрацювання джерел щодо виявлення фішингових повідомлень	23.09 – 1.10	Виконано
3.	Дослідження наборів даних фішингових повідомлень.	2.10 – 12.10	Виконано
4.	Реалізація алгоритму виявлення фішингових повідомлень з використанням LLM.	13.10 – 24.10	Виконано
5.	Оформлення першого розділу “Теоретичні основи фішингу та методів його виявлення”.	25.10 – 2.11	Виконано
6.	Оформлення другого розділу “Використання LLM у кібербезпеці”.	3.11 – 10.11	Виконано
7.	Оформлення третього розділу “Виявлення фішингових повідомлень за допомогою LLM”.	11.11 – 20.11	Виконано
8.	Виконання завдання до підрозділу “Охорона праці”.	21.11 – 27.11	Виконано
9.	Виконання завдання до підрозділу “Безпека в надзвичайних ситуаціях”	28.11 – 1.12	Виконано
10.	Оформлення кваліфікаційної роботи	2.12 – 7.12	Виконано
11.	Нормоконтроль	08.12 – 10.12	Виконано
12.	Перевірка на плагіат	12.12 – 14.12	Виконано
13.	Попередній захист кваліфікаційної роботи	15.12 – 19.12	Виконано
14.	Захист кваліфікаційної роботи	24.12.2023	

Студент _____ Вихованець Д. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Стадник М. А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Виявлення фішингових повідомлень за допомогою LLM// ОР «Магістр» // Вихованець Дмитро Володимирович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБм-61 // Тернопіль, 2025 // С. 103, рис. – 12, табл. – 6, кресл. – , додат. – 5.

Ключові слова: фішинг, виявлення фішингових повідомлень, великі мовні моделі, LLM, машинне навчання, обробка природної мови, GPT, LLaMA, FLAN-T5.

У магістерській кваліфікаційній роботі досліджено та обґрунтовано доцільність застосування великих мовних моделей для виявлення фішингових повідомлень в умовах еволюційного ускладнення соціотехнічних атак. Запропоновано експериментальну методику аналізу фішингового контенту на основі поєднання instruction-based, few-shot та fine-tuning підходів, використання якої дозволило підвищити точність виявлення фішингових повідомлень порівняно з класичними алгоритмами машинного навчання.

У першому розділі роботи розглянуто еволюцію та основні форми фішингових атак, а також узагальнено традиційні й інтелектуальні підходи до їх виявлення з акцентом на обмеження rule-based і класичних методів машинного навчання в умовах zero-day фішингу. У другому розділі досліджено концептуальні засади великих мовних моделей, архітектуру Transformer та механізм attention, а також обґрунтовано доцільність використання LLM для задачі фішинг-детекції на основі порівняльного аналізу з класичними підходами. У третьому розділі реалізовано експериментальне дослідження з використанням моделей GPT, LLaMA та FLAN-T5 і класичних ML-алгоритмів.

Отримані результати засвідчують практичну доцільність застосування великих мовних моделей у сучасних системах кібербезпеки для протидії фішинговим атакам у різних інформаційних середовищах.

ABSTRACT

Detection of phishing messages using LLMs // Thesis of educational level "Master"// Dmytro Vykhovanets // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cybersecurity, group СБм-61 // Ternopil, 2025 // p. 103, figs. 12, tbls. 6, drws. , apps. 5.

Keywords: phishing, phishing email detection, large language models, LLM, machine learning, natural language processing, GPT, LLaMA, FLAN-T5.

In this master's qualification thesis, the feasibility and effectiveness of applying large language models for phishing email detection under conditions of the evolving complexity of social engineering attacks are investigated and substantiated. An experimental methodology for phishing content analysis is proposed, based on a combination of instruction-based, few-shot, and fine-tuning approaches, which enabled an improvement in phishing detection accuracy compared to classical machine learning algorithms.

The first chapter examines the evolution and main types of phishing attacks and summarizes traditional and intelligent detection approaches, with a focus on the limitations of rule-based and classical machine learning methods in zero-day phishing scenarios. The second chapter explores the conceptual foundations of large language models, the Transformer architecture, and the attention mechanism, and justifies the use of LLMs for phishing detection based on a comparative analysis with classical approaches. The third chapter presents an experimental study using GPT, LLaMA, and FLAN-T5 models alongside classical machine learning algorithms.

The obtained results demonstrate the practical relevance of applying large language models in modern cybersecurity systems to counter phishing attacks across various information environments.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AOL	—	America Online
BEC	—	Business Email Compromise
FLAN-T5	—	Fine-tuned Language Net Text-to-Text Transfer Transformer
GPT	—	Generative Pre-trained Transformer
LLM	—	Large Language Model
LLaMA	—	Large Language Model Meta AI
ML	—	Machine Learning
NB	—	Naive Bayes
NLP	—	Natural Language Processing
RF	—	Random Forest
SVM	—	Support Vector Machine

ВСТУП

Актуальність теми. У сучасних умовах стрімкої цифровізації та глобальної діджиталізації комунікацій фішингові атаки залишаються однією з найбільш поширених і небезпечних форм кіберзагроз. Масове використання електронної пошти, хмарних сервісів, онлайн-банкінгу та корпоративних інформаційних систем створює сприятливі умови для реалізації соціотехнічних атак, спрямованих на викрадення облікових даних, фінансових ресурсів та конфіденційної інформації. Особливої актуальності проблема фішингу набуває в умовах еволюційного ускладнення атак, що характеризується високим рівнем персоналізації, використанням багатовекторних сценаріїв та активним залученням технологій штучного інтелекту.

За даними міжнародних аналітичних звітів у сфері кібербезпеки, фішинг залишається основним вектором початкової компрометації інформаційних систем, а кількість фішингових кампаній демонструє стійку тенденцію до зростання. Сучасні фішингові повідомлення дедалі рідше містять очевидні лексичні або технічні маркери шахрайства, що суттєво знижує ефективність традиційних rule-based та класичних machine learning підходів до їх виявлення. Особливо складними для детекції є zero-day фішингові атаки, які навмисно уникають відомих шаблонів і сигнатур.

У зв’язку з цим актуальним науково-практичним завданням є пошук і дослідження нових інтелектуальних підходів до аналізу фішингових повідомлень. Перспективним напрямом у цій сфері є застосування великих мовних моделей (LLM), здатних здійснювати глибокий контекстний, семантичний і прагматичний аналіз тексту. Використання LLM відкриває можливості для виявлення прихованих соціотехнічних маніпуляцій, аналізу наміру повідомлення та ефективної протидії фішингу в умовах його постійної еволюції.

Актуальність теми кваліфікаційної роботи зумовлена необхідністю підвищення ефективності виявлення фішингових повідомлень, зменшення залежності систем захисту від заздалегідь визначених ознак і правил, а також

обґрунтування доцільності використання великих мовних моделей як інтелектуального інструменту сучасних систем кібербезпеки.

Метою кваліфікаційної роботи є дослідження та оцінка потенціалу великих мовних моделей у задачі виявлення фішингових повідомлень, а також розробка та експериментальне обґрунтування методики їх застосування з подальшим порівнянням ефективності з класичними алгоритмами машинного навчання.

Завданнями дослідження є наступні:

- Проаналізувати еволюцію фішингових атак та сучасні підходи до їх виявлення.
- Дослідити обмеження rule-based і класичних методів машинного навчання у задачі фішинг-детекції.
- Обґрунтувати доцільність використання великих мовних моделей для аналізу фішингових повідомлень.
- Сформувати репрезентативний об’єднаний датасет фішингових і легітимних електронних повідомлень.
- Реалізувати та налаштувати експериментальні моделі на основі LLM (GPT, LLaMA, FLAN-T5).
- Реалізувати базову експериментальну лінію з використанням класичних ML-алгоритмів.
- Провести порівняльний аналіз результатів за показниками accuracy, precision, recall та F1-score.
- Сформулювати висновки щодо ефективності застосування LLM у системах виявлення фішингових атак.

Об’єктом дослідження є процес автоматизованого виявлення фішингових повідомлень у цифрових комунікаційних середовищах.

Предметом дослідження є методи та алгоритми виявлення фішингових повідомлень на основі великих мовних моделей, а також їх порівняння з класичними підходами машинного навчання.

Методи дослідження. У роботі застосовано аналіз наукових публікацій і сучасних аналітичних звітів у сфері кібербезпеки. Для оцінювання ефективності

різних підходів використано порівняльний аналіз rule-based методів, класичних алгоритмів машинного навчання та великих мовних моделей. Практичну частину дослідження реалізовано з використанням методів машинного та глибокого навчання, зокрема алгоритмів логістичної регресії, SVM, Naive Bayes, Random Forest, а також fine-tuning і instruction-based підходів до застосування LLM. Результати експериментів оцінювалися за допомогою стандартних метрик бінарної класифікації та аналізу матриць помилок.

Наукова новизна роботи полягає у комплексному дослідженні застосування великих мовних моделей для виявлення фішингових повідомлень з урахуванням їх здатності до глибокого контекстного та семантичного аналізу. У роботі обґрунтовано переваги LLM у виявленні zero-day фішингових атак та продемонстровано ефективність instruction-based і fine-tuning підходів у порівнянні з класичними алгоритмами машинного навчання.

Практичне значення роботи полягає у можливості застосування отриманих результатів при розробці та вдосконаленні інтелектуальних систем виявлення фішингових атак у корпоративних та освітніх інформаційних середовищах. Запропонована методика може бути використана як основа для створення гібридних систем кіберзахисту, що поєднують класичні ML-алгоритми та великі мовні моделі з метою підвищення рівня інформаційної безпеки та зменшення ризиків соціотехнічних атак.

Апробація результатів кваліфікаційної роботи. Основні результати дослідження були апробовані на XIII науково-технічній конференції «Інформаційні моделі, системи та технології» (м.Тернопіль, Україна, 17-18 грудня). Відповідна наукова публікація наведена у Додатку А.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВІ ФІШИНГУ ТА МЕТОДІВ ЙОГО ВИЯВЛЕННЯ

1.1 Проблематика виявлення фішингових атак в умовах їх еволюційного ускладнення

Фішинг є одним із найбільш поширених та найбільш руйнівних видів кіберзлочинності, який зазнав суттєвої еволюції з моменту свого виникнення у 1990-х роках [1-2]. Вперше термін “phishing” був зафіксований 2 січня 1996 року та використовувався для опису шахрайських дій, пов’язаних із викраденням облікових даних користувачів сервісу America Online (AOL) [1]. У той період зловмисники видавали себе за адміністраторів AOL і шляхом обману отримували логіни та паролі з метою безкоштовного доступу до мережі Інтернет.

Однією з перших спільнот, що активно використовувала подібні методи, була так звана Warez-спільнота, до складу якої входили хакери та цифрові пірати [1]. Її представники не лише викрадали облікові дані користувачів, але й генерували випадкові номери банківських карток для незаконного створення нових облікових записів AOL. Попри примітивність таких схем, вони виявилися достатньо ефективними, оскільки на той час користувачі практично не мали уявлення про існування фішингових загроз та методи протидії їм.

На початковому етапі фішинг характеризувався відносною простотою реалізації, проте вже тоді демонстрував високу результативність як інструмент шахрайства [1]. Ранні фішингові атаки переважно здійснювалися шляхом розсилання електронних листів, що імітували повідомлення від фінансових установ або популярних онлайн-сервісів. Такі повідомлення містили заклики до оновлення або підтвердження облікових даних і перенаправляли користувачів на підроблені вебресурси, візуально подібні до легітимних сайтів.

Упродовж 2000-х років фішингові техніки зазнали подальшого ускладнення завдяки активному використанню методів соціальної інженерії [3,4]. Зловмисники почали персоналізувати зміст повідомлень, використовуючи конкретну інформацію про потенційних жертв, що значно підвищувало рівень

правдоподібності та ефективність атак. У результаті фішинг трансформувався з масового та шаблонного явища у цілеспрямований і складний механізм впливу, який і до сьогодні залишається однією з найсерйозніших проблем у сфері інформаційної безпеки для організацій різного масштабу [5].

Подальший етап розвитку фішингу охоплює 2000-ті та 2010-ті роки, протягом яких спостерігалося його стрімке ускладнення та масштабування [6,7]. На початку 2000-х років обізнаність користувачів щодо фішингових загроз залишалася вкрай низькою, а більшість жертв не усвідомлювали, що зловмисники можуть видавати себе за представників довірених організацій з метою незаконного отримання конфіденційних даних. Це створило сприятливі умови для масового поширення фішингових атак.

У зазначений період зловмисники почали активно спрямовувати свої зусилля на онлайн-платіжні сервіси, зокрема PayPal та E-gold, які на той час вже мали широку базу користувачів. Типовим сценарієм було надсилання електронних повідомлень із вимогою оновлення платіжної інформації або даних банківських карток, унаслідок чого облікові та фінансові реквізити користувачів опинялися у розпорядженні зловмисників.

Важливою віхою в еволюції фішингу стало поширення криптовалют наприкінці 2008 року, що забезпечило зловмисникам відносно анонімні та важковідстежувані платіжні механізми. Це суттєво спростило фінансування злочинної діяльності, координацію між учасниками атак, а також отримання та легалізацію незаконно здобутих коштів.

Починаючи з 2013 року, фішинг став основним каналом поширення програм-вимагачів (ransomware), зокрема таких відомих зразків, як CryptoLocker, WannaCry та Petya [7]. Фішингові електронні листи використовувалися для доставки шкідливих вкладень або посилань, що призводило до масового зараження корпоративних та приватних інформаційних систем. Фінансові втрати від таких атак вимірювалися мільйонами доларів і включали не лише суму викупу, але й додаткові витрати, пов’язані з простоєм систем, штрафними санкціями та відновленням інфраструктури.

Починаючи з 2010-х років, окрім традиційних фінансових цілей, фішинг почали використовувати як інструмент для досягнення політичних та стратегічних цілей. Показовим прикладом є цілеспрямована фішингова атака 2016 року, спрямована на Джона Подесту – керівника виборчої кампанії Гілларі Клінтон, що продемонструвала потенціал фішингу як засобу впливу на політичні процеси та національну безпеку.

Подальша еволюція фішингу упродовж 2010-х років характеризувалася появою більш цілеспрямованих і складних методів атак. У цей період набули поширення такі форми, як spear-phishing, орієнтований на конкретних осіб або групи користувачів, а також whaling, спрямований на представників вищого керівництва організацій. Дані атаки відзначалися високим рівнем персоналізації та ретельною підготовкою, що значно ускладнювало їх виявлення традиційними засобами захисту.

Паралельно з розвитком електронної пошти зловмисники почали активно використовувати альтернативні канали комунікації. Зокрема, з’явилися методи smishing та vishing, які передбачають введення жертви в оману за допомогою текстових повідомлень та голосових дзвінків відповідно [8]. Такі атаки експлуатують довіру користувачів до мобільних комунікацій і часто поєднуються з класичним фішингом, формуючи багатоканальні сценарії соціальної інженерії.

У 2020-х роках фішинг зазнав подальшого ускладнення у зв’язку з активним використанням соціальних мереж та реалізацією багатовекторних атак, що поєднують електронну пошту, SMS, голосові дзвінки та платформи соціальних медіа. Такий підхід дозволяє зловмисникам підвищувати рівень довіри до повідомлень та збільшувати ймовірність успішної компрометації жертви [9,10].

Крім того, сучасні фішингові кампанії дедалі частіше ґрунтуються на використанні технологій штучного інтелекту, які застосовуються для автоматизованого створення правдоподібних фішингових повідомлень, імітації стилю легітимних організацій та масштабування атак [11]. За даними Anti-Phishing Working Group (APWG), у першому кварталі 2025 року було зафіксовано 1 003 924 фішингові атаки, що стало найвищим квартальним

показником із кінця 2023 року. Це підтверджує циклічний характер фішингової активності та її здатність швидко відновлюватися після тимчасових спадів.

Однією з характерних особливостей фішингу у 2025 році стало масове використання QR-кодів у шахрайських кампаніях. Зловмисники щоденно розсилають мільйони електронних повідомлень, що містять QR-коди, які перенаправляють користувачів на фішингові вебсайти або ресурси з шкідливим програмним забезпеченням. Такий підхід дозволяє обходити традиційні механізми перевірки URL-адрес і знижує ефективність класичних email-фільтрів, що ще більше ускладнює процес виявлення атак [10].

Водночас у першому кварталі 2025 року було зафіксовано зростання кількості атак на фінансовий та платіжний сектори, зокрема банки та онлайн-платіжні системи. Сукупно на ці галузі припадало 30,9% усіх фішингових атак [12], що свідчить про часткове повернення інтересу зловмисників до фінансової інфраструктури після попереднього зміщення фокусу у бік соціальних мереж.

Окрему загрозу продовжують становити атаки типу Business Email Compromise (BEC). У першому кварталі 2025 року загальна кількість BEC-інцидентів, пов’язаних із банківськими переказами, зросла на 33% порівняно з попереднім кварталом. Це вказує на ескалацію фінансових ризиків для організацій та зростання професійності зловмисних угруповань, які поєднують соціальну інженерію з ретельно підготовленими фінансовими сценаріями.

Аналіз динаміки зафіксованих фішингових атак у період з третього кварталу 2024 року до другого кварталу 2025 року (рис. 1.1) свідчить про стійку тенденцію до зростання їх кількості. Попри окремі короткострокові коливання, загальний тренд демонструє поступове збільшення середнього рівня фішингової активності, що підтверджується зростаючою лінією тренду. Це вказує на системний характер загрози та відсутність довгострокового зниження інтенсивності атак, навіть за умов впровадження традиційних засобів захисту.

Рисунок 1.1 ілюструє розподіл фішингових атак за галузевою ознакою у другому кварталі 2025 року. Найбільш уразливими виявилися фінансові установи, на які припадало 18,3% усіх зафіксованих атак, що підтверджує стабільно високий інтерес зловмисників до сектору з прямим доступом до

фінансових активів. Майже аналогічну частку становлять атаки на SaaS- та webmail-сервіси (18,2%), які часто використовуються як початкова точка компрометації корпоративних облікових записів [9,10].

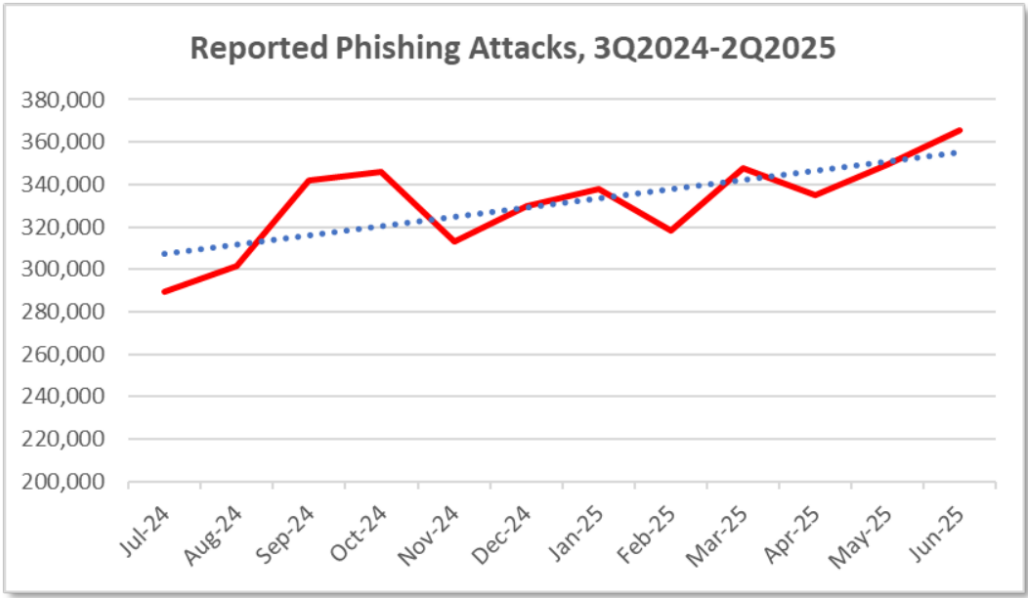


Рисунок 1.1 – Динаміка зафіксованих фішингових атак у період з III кварталу 2024 року по II квартал 2025 року [10]

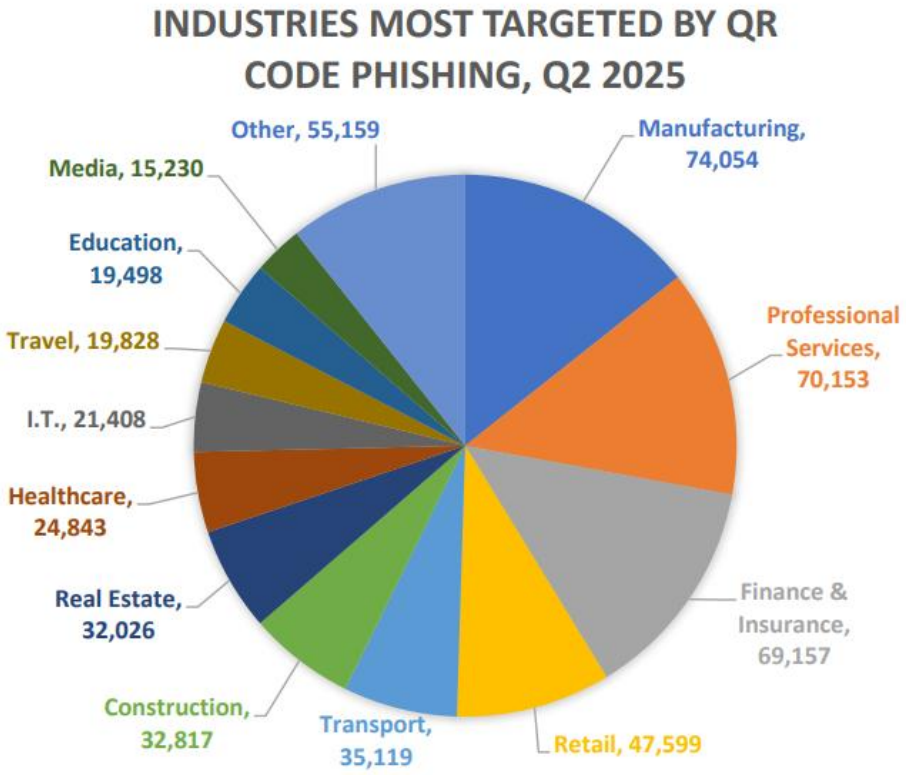


Рисунок 1.2 – Розподіл найбільш уразливих секторів до фішингових атак у II кварталі 2025 року [10]

З метою узагальнення основних етапів розвитку фішингових атак та систематизації їх ключових характеристик у таблиці 1.1 наведено еволюцію фішингу в часовому розрізі.

Таблиця 1.1 – Еволюція фішингу

Період (декада)	Характеристика фішингу
1990-ті роки	Початковий етап формування фішингу. Атаки мали примітивний характер, але були ефективними через відсутність обізнаності користувачів про кіберзагрози.
Початок 2000-х років	Масове поширення email-фішингу. Атаки імітували повідомлення від банків і популярних онлайн-платформ. Основною метою було викрадення облікових та платіжних даних. Використовувалися прості підроблені вебсайти.
Кінець 2000-х років	Застосування методів соціальної інженерії та поява фінансово орієнтованого фішингу. Зловмисники почали активно атакувати платіжні сервіси (PayPal, E-gold). Поширення криптовалют забезпечило відносну анонімність фінансових операцій.
2010 - 2013 роки	Ускладнення фішингових кампаній та перехід до персоналізованих атак. Поширення spear-phishing і whaling. Фішинг стає основним вектором поширення шкідливого програмного забезпечення, зокрема програм-вимагачів.
2013 - 2020 роки	Масове використання фішингу для доставки ransomware (CryptoLocker, WannaCry, Petya). Розширення каналів атак: поява smishing (SMS-фішинг) та vishing (голосовий фішинг). Активне використання соціальних мереж і відкритих джерел даних для підвищення персоналізації атак. Фішинг починає застосовуватися не лише з фінансовою, а й з політичною та стратегічною метою.
2020 - 2023 роки	Формування багатовекторних фішингових атак, що поєднують email, месенджери, SMS, телефонні дзвінки та соціальні платформи. Активне застосування автоматизації та генеративних інструментів для створення фішингового контенту.
2024 - 2025 роки	Зростання масштабів фішингу та поява нових векторів доставки, зокрема QR-кодів. Активне використання штучного інтелекту для генерації правдоподібних повідомлень і масштабування атак. Збільшення кількості фінансово орієнтованих та BEC-атак

У сучасному розумінні термін “фішинг” доцільно трактувати як різновид кіберзлочинної діяльності, що ґрунтується на методах соціальної інженерії та полягає у навмисному введенні користувачів в оману шляхом імітації довірених осіб, організацій або цифрових сервісів з метою незаконного отримання конфіденційної інформації, фінансових ресурсів або ініціювання шкідливих дій у інформаційних системах. Таке визначення відображає комплексний характер фішингу як соціотехнічної загрози та підкреслює необхідність застосування інтелектуальних методів аналізу для його ефективного виявлення й протидії. Це ще раз підкреслює обмеженість традиційних методів виявлення та актуальність застосування інтелектуальних підходів, зокрема великих мовних моделей, здатних здійснювати глибокий контекстний аналіз фішингового контенту.

1.2 Класифікація фішингових атак

Фішингові атаки продовжують еволюціонувати, набуваючи дедалі складніших форм, що ускладнює їх виявлення та підвищує ефективність соціотехнічного впливу. Залежно від рівня персоналізації, способу реалізації та цільового призначення, фішингові атаки поділяються на кілька основних типів [6], що представлені на рис. 1.3.

Одним із найбільш небезпечних різновидів є spear-phishing [3], який характеризується цілеспрямованістю та високим рівнем персоналізації. Такі атаки спрямовані на конкретні організації, державних посадовців або окремих осіб і зазвичай мають на меті отримання доступу до спеціалізованих баз даних, конфіденційної інформації або фінансових ресурсів. Для підвищення правдоподібності зловмисники адаптують стиль комунікації до професійних характеристик, соціального оточення та контактів жертви, активно використовуючи дані з соціальних мереж та відкритих джерел. Відомим прикладом spear-phishing є атака 2016 року на Джона Подесту, керівника виборчої кампанії Гілларі Клінтон, у результаті якої було скомпрометовано електронну пошту та оприлюднено конфіденційне листування [13].

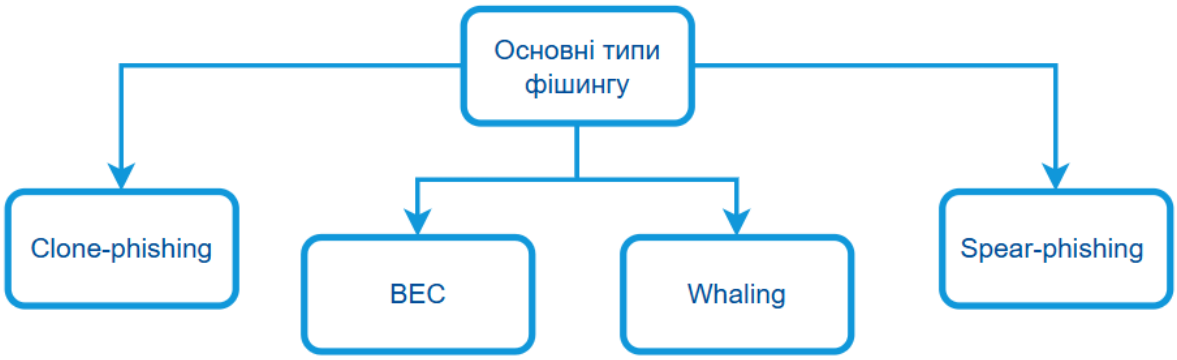


Рисунок 1.3 – Основні типи фішингу

Окремим різновидом цілеспрямованого фішингу є whaling, який орієнтований на керівників вищої ланки або публічних осіб. У таких атаках зловмисники часто маскуються під співробітників компанії або використовують службові повідомлення щодо внутрішніх процесів організації з метою порушення її діяльності або отримання доступу до стратегічно важливої інформації [14].

Основною метою BEC-атак є маніпуляція фінансовими операціями шляхом компрометації корпоративної електронної пошти або імітації листування від імені керівництва, бізнес-партнерів чи постачальників. Зловмисники ретельно вивчають внутрішні бізнес-процеси компанії, фінансові регламенти та комунікаційні шаблони, що дозволяє їм створювати правдоподібні повідомлення з вимогою термінового переказу коштів або зміни платіжних реквізитів. Одним із найвідоміших випадків є BEC-атаки на керівників компаній Facebook та Google у 2013–2015 роках, унаслідок яких зловмисники, видаючи себе за партнерів, незаконно отримали понад 100 млн доларів США шляхом підроблених рахунків та електронних листів.

Clone-phishing є різновидом фішингових атак, що ґрунтується на підробці легітимних електронних повідомлень або вебресурсів, з якими користувач уже мав попередню взаємодію. У межах цього підходу зловмисники створюють точні копії справжніх електронних листів, корпоративних повідомлень або вебсайтів, замінюючи оригінальні посилання чи вкладення на шкідливі аналоги. Типовим сценарієм clone-phishing є повторне надсилання листа, який раніше був отриманий користувачем від довіреного джерела, з повідомленням про

«оновлену версію» документа або «виправлення помилки» [4]. Користувач, довіряючи знайомому формату та відправнику, відкриває шкідливий вкладений файл або переходить за фішинговим посиланням, що призводить до компрометації облікових даних або зараження системи шкідливим програмним забезпеченням.

Реальні приклади clone-phishing широко задокументовані у звітах провідних компаній з кібербезпеки. Зокрема, у 2019–2024 роках фіксувалися масштабні кампанії, спрямовані на користувачів Microsoft 365, PayPal, Google Workspace та хмарних сервісів обміну файлами, у яких зловмисники повторно надсилали легітимні електронні листи із заміненими посиланнями або вкладеннями [11].

Blind-phishing, також відомий як масовий фішинг, є однією з найбільш поширених форм фішингових атак і характеризується відсутністю конкретного таргетування. У цьому випадку зловмисники здійснюють масове розповсюдження фішингових повідомлень через електронну пошту, SMS, соціальні мережі або месенджери, розраховуючи на те, що певний відсоток отримувачів піддасться обману.

Залежно від використовуваного каналу взаємодії з потенційною жертвою, фішингові атаки поділяються на кілька основних типів, кожен з яких має власні особливості реалізації, рівень ефективності та приклади реальних інцидентів [11] (рис. 1.4).

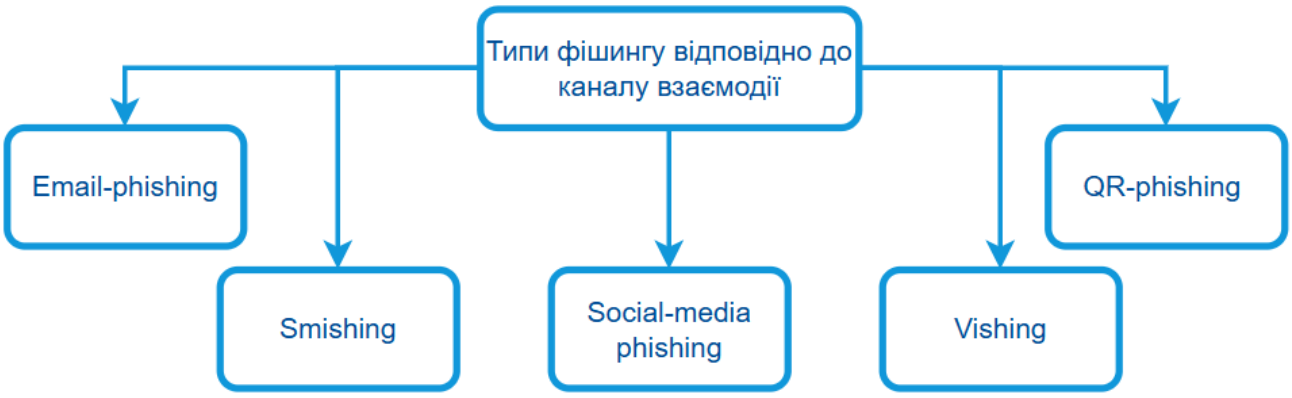


Рисунок 1.4 – Типи фішингу відповідно до каналу взаємодії

Email-phishing є найпоширенішим видом фішингових атак і здійснюється за допомогою електронної пошти. У таких атаках зловмисники розсилають підроблені листи, які зовні виглядають як повідомлення від банків, державних органів, хмарних сервісів або ділових партнерів (рис. 1.5). Основна мета таких листів це змусити користувача перейти за шкідливим посиланням, відкрити заражений файл або передати свої облікові дані.

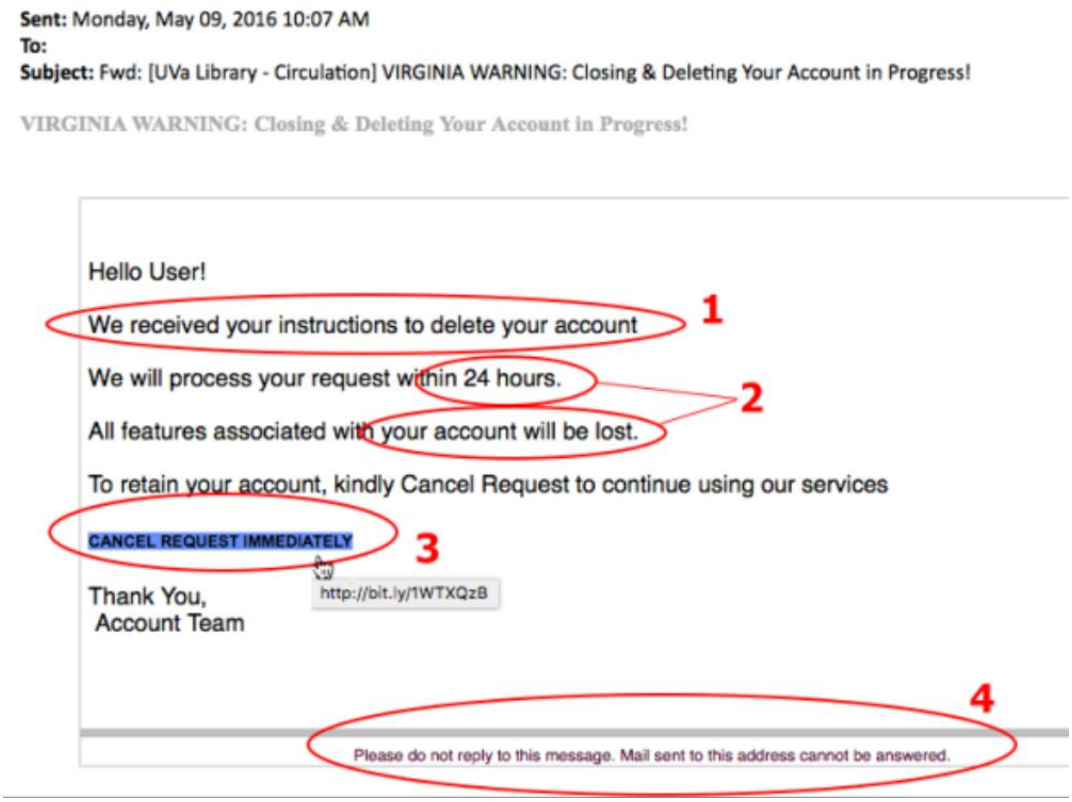


Рисунок 1.5 – Приклад фішингового електронного листа з використанням соціальної інженерії

Smishing реалізується через текстові повідомлення, зокрема SMS або повідомлення в месенджерах. Для впливу на жертву зазвичай використовується відчуття терміновості або страху. Повідомлення маскуються під офіційні сповіщення від банків, поштових чи логістичних компаній і містять посилання, за якими користувача просять перейти або ввести персональні дані. У період 2021–2023 років у багатьох країнах фіксувалися smishing-кампанії, що імітували повідомлення від DHL, FedEx та USPS, у результаті яких відбувалося викрадення платіжної інформації користувачів [8].

Vishing є різновидом фішингу, який здійснюється за допомогою голосових дзвінків. Під час таких атак зловмисники представляються співробітниками банків, податкових служб або правоохоронних органів і намагаються психологічно тиснути на жертву, використовуючи страх або створюючи відчуття термінової ситуації. Відомі масові vishing-кампанії у США та Великій Британії, де шахраї видавали себе за представників податкових служб IRS та HMRC, що призводило до значних фінансових втрат.

QR-phishing, або quishing, є відносно новим видом фішингу, який базується на використанні QR-кодів. Такі коди можуть перенаправляти користувачів на фішингові сайти або ініціювати завантаження шкідливого програмного забезпечення. Особливістю цього підходу є те, що шкідливе посилання приховане у графічному коді, що дозволяє обходити традиційні фільтри електронної пошти. За даними Anti-Phishing Working Group, у 2023–2025 роках спостерігалося суттєве зростання атак із використанням QR-кодів, зокрема у повідомленнях, що імітували рахунки за комунальні послуги або сповіщення від фінансових установ.

Фішинг через соціальні мережі та месенджери базується на довірі між користувачами цифрових платформ. У таких атаках зловмисники створюють фейкові профілі або отримують доступ до реальних облікових записів, після чого надсилають шкідливі повідомлення або посилання від імені знайомих контактів, що значно підвищує ймовірність успішної атаки.

Окрім традиційних форм фішингу, у сучасному цифровому середовищі дедалі більшого поширення набувають технологічно складні типи атак, що представлені на рис.1.6.

У сучасних фішингових атаках дедалі частіше застосовуються технології штучного інтелекту, які дозволяють автоматично генерувати переконливі та граматично коректні повідомлення. Такі повідомлення можуть адаптуватися до стилю спілкування конкретної жертви, що значно підвищує їх правдоподібність. Використання штучного інтелекту також дає змогу масштабувати фішингові кампанії, унаслідок чого їх виявлення за допомогою традиційних email-фільтрів стає значно складнішим. Подібний підхід відомий як AI-based phishing [4].

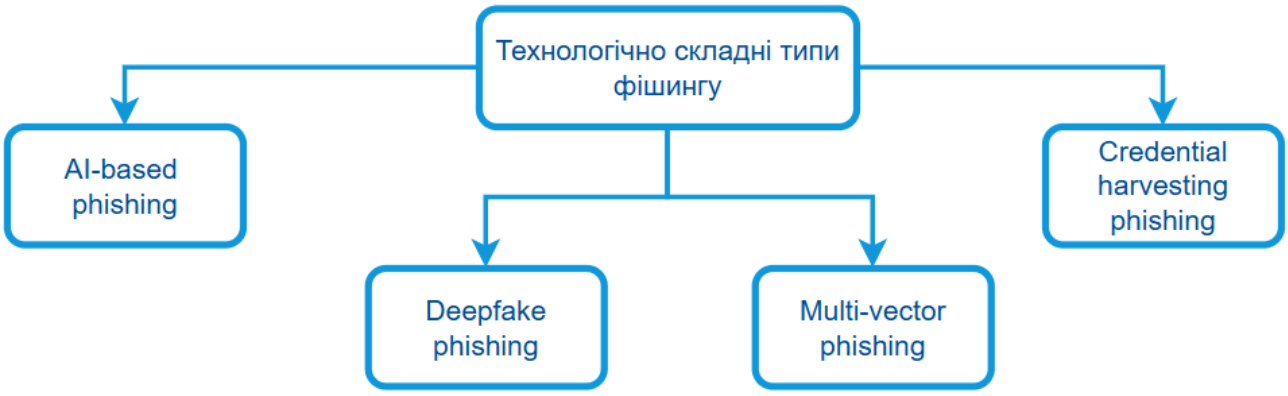


Рисунок 1.6 – Технологічно складні типи фішингу

У сучасних фішингових атаках дедалі частіше застосовуються технології штучного інтелекту, які дозволяють автоматично генерувати переконливі та граматично коректні повідомлення. Такі повідомлення можуть адаптуватися до стилю спілкування конкретної жертви, що значно підвищує їх правдоподібність. Використання штучного інтелекту також дає змогу масштабувати фішингові кампанії, унаслідок чого їх виявлення за допомогою традиційних email-фільтрів стає значно складнішим. Подібний підхід відомий як AI-based phishing.

Окрему загрозу становлять атаки, у яких використовуються згенеровані аудіо- або відеоматеріали для імітації реальних осіб. У таких випадках жертва може отримати голосове повідомлення або відеозвернення, яке виглядає так, ніби воно надійшло від керівника, колеги чи ділового партнера. Високий рівень реалістичності таких матеріалів формує довіру та може спонукати до виконання фінансових операцій або передачі конфіденційної інформації. Цей підхід відомий як deepfake phishing.

З метою підвищення ефективності фішингових атак зловмисники також застосовують одночасно кілька каналів комунікації, зокрема електронну пошту, SMS-повідомлення, телефонні дзвінки та соціальні мережі. Інформація, отримана з одного каналу, використовується для підкріплення повідомлень в іншому, що створює цілісну та переконливу картину для жертви. У результаті такі атаки мають значно вищу ймовірність успіху порівняно з одноканальними фішинговими кампаніями та класифікуються як multi-vector phishing.

Одним із найпоширеніших наслідків фішингових атак є масовий збір облікових даних користувачів, зокрема логінів і паролів. Для цього використовуються підроблені вебсторінки, які імітують сторінки входу до популярних сервісів, таких як поштові платформи, соціальні мережі або корпоративні системи. Отримані облікові дані можуть використовуватися для несанкціонованого доступу до акаунтів, фінансових шахрайств або як початковий етап складніших кібератак. Такий тип фішингу відомий як credential harvesting phishing [8].

Основою фішингових атак є цілеспрямована експлуатація психологічних особливостей людини та її поведінкових моделей у цифровому середовищі. На відміну від технічних атак, фішинг орієнтований передусім не на вразливості програмного забезпечення, а на людський фактор, який залишається найменш захищеною ланкою систем інформаційної безпеки. Для досягнення своїх цілей зловмисники активно застосовують методи соціальної інженерії, поєднуючи психологічний вплив із технічними засобами доставки шкідливого контенту.

Одним із ключових механізмів фішингових атак є виклик сильних емоційних реакцій, зокрема страху, тривоги або відчуття терміновості. Зловмисники часто створюють сценарії, у яких користувачеві повідомляється про нібито блокування облікового запису, підозрілу фінансову операцію або загрозу втрати доступу до важливих сервісів. В умовах психологічного тиску жертва схильна діяти імпульсивно, не аналізуючи достовірність повідомлення та не перевіряючи джерело інформації.

Іншим поширеним прийомом є маніпуляція очікуванням вигоди, що проявляється у повідомленнях про виграші, бонуси, компенсації або ексклюзивні пропозиції. У таких випадках фішингові повідомлення апелюють до жадібності або бажання швидкого отримання переваг, що знижує рівень критичного мислення користувачів. Подібні сценарії особливо ефективні у масових фішингових кампаніях.

Важливу роль відіграє також використання авторитету, коли зловмисники маскуються під представників банків, державних установ, правоохоронних органів або відомих компаній. Використання офіційної символіки, знайомих

логотипів і формального стилю спілкування формує у жертви відчуття легітимності повідомлення та підсилює довіру до його змісту.

Окрім емоційного впливу, фішингові атаки активно використовують когнітивні упередження, притаманні людському мисленню. Зокрема, застосовується ефект терміновості, коли користувачеві нав’язується необхідність негайної дії, а також ефект знайомства, за якого повідомлення від відомого бренду або знайомого контакту сприймається як безпечне. Ці механізми значно підвищують імовірність успішної атаки навіть серед технічно підготовлених користувачів.

Розвиток цифрових платформ, соціальних мереж і відкритих джерел інформації (OSINT) суттєво розширив можливості зловмисників у зборі персональних даних. Інформація про місце роботи, коло спілкування, професійну діяльність або особисті інтереси використовується для створення персоналізованих фішингових повідомлень, які виглядають правдоподібно та відповідають контексту життя конкретної жертви. У результаті фішинг поступово трансформується з масового явища у цілеспрямований інструмент атак, зокрема у формах spear-phishing та whaling.

Підсумувавши вище написане, у роботі було зроблено висновок, що ефективність фішингових атак зумовлюється поєднанням цілеспрямованої психологічної та соціотехнічної маніпуляції користувачами з використанням сучасних високотехнологічних засобів автоматизації й аналізу даних, що формує складну багаторівневу загрозу та суттєво ускладнює її виявлення традиційними методами захисту, обґрунтовуючи необхідність застосування інтелектуальних підходів до аналізу та протидії фішингу.

1.3 Сучасні підходи та методи виявлення фішингу

Традиційні системи виявлення фішингових повідомлень базуються переважно на сигнатурних, евристичних та фільтраційних підходах, які протягом тривалого часу залишалися основою захисту електронних комунікацій [15]. Ці методи широко застосовуються у поштових серверах, корпоративних шлюзах

безпеки та антивірусних системах, забезпечуючи базовий рівень захисту від відомих загроз.

Сигнатурні методи передбачають зіставлення вхідних повідомлень із заздалегідь сформованими базами даних, що містять відомі фішингові шаблони, шкідливі домени, IP-адреси або URL-посилання. Основною перевагою такого підходу є висока швидкодія та низькі обчислювальні витрати, що дозволяє обробляти значні обсяги трафіку в режимі реального часу. Проте ефективність сигнатурних методів суттєво знижується у випадках використання нових або модифікованих фішингових кампаній, які ще не внесені до баз сигнатур.

Окремим різновидом сигнатурних підходів є використання чорних списків (blacklists), які являють собою зібрання відомих зловмисних IP-адрес, доменів та веб-сайтів. Такі списки активно застосовуються веб-браузерами, поштовими клієнтами та засобами інформаційної безпеки для попередження користувачів про спробу переходу на відомий фішинговий ресурс. Даний підхід є ефективним у випадку добре задокументованих і вже виявлених фішингових сайтів, проте його результативність суттєво знижується щодо новостворених або часто змінюваних фішингових сторінок, які ще не внесені до баз даних.

До класичних методів також належить виявлення на основі сигнатур (signature-based detection), яке використовує заздалегідь визначені шаблони, характерні для фішингових повідомлень або вебконтенту [16]. Вхідні електронні листи та сторінки позначаються як потенційно небезпечні у разі відповідності таким сигнатурам. Попри високу точність щодо відомих атак, цей підхід демонструє низьку ефективність проти zero-day фішингових кампаній або атак, які навмисно змінюють свої характеристики для обходу систем детекції.

Евристичні методи орієнтовані на аналіз ознак, притаманних фішинговим повідомленням, без прив’язки до конкретних шаблонів [15]. До таких ознак належать нетипова структура тексту, помилки у граматиці та стилі, використання підозрілих формулювань, аномальні посилання або вкладення, а також невідповідність між адресою відправника та вмістом повідомлення. Евристичний аналіз дозволяє виявляти раніше невідомі загрози, однак значною

мірою залежить від якості правил та експертних налаштувань, що може призводити до суб’єктивності результатів.

Окрему групу становлять фільтраційні методи, серед яких найбільш поширеними є spam-фільтри та байєсівські класифікатори [17]. Такі підходи використовують статистичні характеристики тексту, зокрема частотність слів, наявність ключових фраз, структуру заголовків і метадані повідомлень, для обчислення ймовірності того, що повідомлення є фішинговим. Перевагою фільтраційних методів є здатність автоматично адаптуватися до змін у потоці повідомлень за рахунок навчання на нових даних. Водночас вони можуть бути вразливими до навмисних спроб обходу, наприклад шляхом зміни формулювань або використання синонімів. Такий метод може бути ефективним для виявлення очевидних фішингових спроб, однак уразливий до більш витончених атак, у яких зловмисники використовують нейтральну лексику або замаскований шкідливий контент.

Додатковим рівнем захисту є механізми перевірки відправника, зокрема технології SPF (Sender Policy Framework) та DKIM (DomainKeys Identified Mail), які використовуються для підтвердження автентичності домену відправника електронного листа. Ці механізми дозволяють зменшити кількість підроблених повідомлень, однак не забезпечують повного захисту від фішингу, оскільки зловмисники можуть використовувати скомпрометовані або легітимні облікові записи для розсилання фішингового контенту.

Попри широке застосування та відносну простоту реалізації, традиційні методи виявлення фішингу мають низку суттєвих обмежень. Вони часто демонструють високий рівень хибнопозитивних спрацювань, що призводить до блокування легітимних повідомлень, а також хибнонегативних результатів, коли складні або персоналізовані атаки залишаються непоміченими. Крім того, такі системи потребують постійного оновлення сигнатур, правил і навчальних вибірок, що ускладнює їх масштабування та ефективне застосування в умовах швидкої еволюції фішингових загроз [9,10].

Машинне навчання є важливим напрямом штучного інтелекту, який широко застосовується у сфері виявлення фішингових атак. На відміну від традиційних

rule-based підходів, моделі машинного навчання навчаються на великих масивах даних, що містять як фішингові, так і легітимні повідомлення, та здатні самостійно виявляти характерні ознаки шахрайського контенту. Це дозволяє системам аналізувати не лише відомі шаблони атак, а й нові, раніше невідомі фішингові кампанії [18].

Одним із ключових напрямів застосування машинного навчання у фішингу є виявлення аномалій у тексті повідомлень, структурі електронних листів та поведінці користувачів. Алгоритми аналізують такі характеристики, як нетипові формулювання, підозрілі URL-адреси, незвичні вкладення або невідповідність між відправником і змістом повідомлення. Виявлення відхилень від «нормальної» поведінки дозволяє ідентифікувати фішингові повідомлення навіть у випадках, коли вони не містять очевидних ознак шахрайства [19].

Важливу роль відіграє також розпізнавання шаблонів, оскільки фішингові атаки часто мають спільні лінгвістичні та структурні характеристики. Моделі машинного навчання здатні знаходити приховані закономірності у тексті повідомлень, заголовках, URL-адресах і метаданих, що дає змогу ефективно класифікувати фішингові email-повідомлення, SMS або повідомлення у соціальних мережах. Такий підхід дозволяє виявляти як масові, так і цілеспрямовані фішингові атаки [20].

Адаптивність є ще однією перевагою використання машинного навчання для протидії фішингу. Оскільки зловмисники постійно змінюють формулювання, дизайн фішингових сторінок і способи доставки повідомлень, моделі машинного навчання можуть перенавчатися на нових даних та пристосовуватися до змін у тактиках атак. Це забезпечує більш високий рівень захисту порівняно з традиційними статичними методами [18, 19].

Автори у статті [16] аналізують основні алгоритми машинного навчання, що застосовуються для виявлення фішингових атак, їх переваги та обмеження узагальнено в таблиці 1.2, що дозволяє порівняти ефективність різних підходів та оцінити їх придатність для аналізу фішингових повідомлень у сучасних умовах.

Таблиця 1.2 – Основні переваги та недоліки методів машинного навчання у виявленні фішингу

Метод	Переваги	Недоліки
Логістична регресія	Простота реалізації; висока інтерпретованість результатів	Обмежена здатність до виявлення складних і нелінійних атак
Дерева рішень	Зрозуміла логіка прийняття рішень; можливість пояснення результатів	Схильність до перенавчання при складних структурах
Випадкові ліси (Random Forest)	Висока точність; стійкість до шуму та пропусків у даних	Низька інтерпретованість кінцевої моделі
Метод опорних векторів (SVM)	Ефективність при класифікації текстів і URL-адрес	Високі обчислювальні витрати; чутливість до параметрів
k-means (кластеризація)	Низький рівень хибнопозитивних спрацювань; простота	Залежність результату від початкових кластерів
Нейро-нечіткі моделі	Висока точність виявлення фішингу	Висока складність реалізації та обчислювальні витрати

У системах виявлення фішингових атак одним із базових підходів є використання простих лінійних моделей класифікації. Такі моделі, зокрема логістична регресія, застосовуються для аналізу текстових ознак електронних листів, частотності ключових слів, наявності підозрілих URL-адрес і метаданих повідомлень. Завдяки простоті реалізації та зрозумілій інтерпретації результатів цей підхід часто використовується як початкова або еталонна модель для порівняння з більш складними алгоритмами.

Для роботи з високовимірними текстовими даними та складними межами класифікації широко застосовуються методи опорних векторів (SVM) [18]. Такі алгоритми демонструють високу ефективність під час аналізу фішингових email-повідомлень, SMS та URL-адрес, дозволяючи відокремлювати легітимний контент від шахрайського навіть у випадках, коли відмінності є неочевидними.

У задачах виявлення фішингових вебсайтів значну роль відіграють алгоритми, що базуються на деревоподібних структурах. Дерева рішень і випадкові ліси (Random Forest) використовуються для аналізу таких ознак, як довжина URL, кількість піддоменів, наявність IP-адрес у посиланнях, особливості HTML-коду сторінок та характеристики SSL-сертифікатів. Ансамблеві методи, зокрема Random Forest, забезпечують високу точність класифікації та стійкість до шуму в даних.

Для швидкої обробки великих обсягів текстових повідомлень, особливо у задачах email-фішингу та smishing-атак, часто застосовуються баєсівські класифікатори, зокрема Naive Bayes. Ці алгоритми ґрунтуються на статистичному аналізі слів і фраз та дозволяють оперативно оцінювати ймовірність фішингового характеру повідомлення, хоча можуть поступатися складнішим моделям за точністю [17].

З метою виявлення нових або раніше невідомих фішингових кампаній використовуються методи кластеризації, серед яких поширеним є алгоритм k-means. Такі підходи дозволяють групувати схожі повідомлення або вебресурси та виявляти аномальні кластери, що відрізняються від типового легітимного трафіку, без попередньої розмітки даних.

Для аналізу складних фішингових сценаріїв, які включають великі текстові корпуси, зображення вебсторінок або поведінкові дані користувачів, застосовуються нейронні мережі та методи глибокого навчання. Зокрема, рекурентні нейронні мережі (RNN) і згорткові нейронні мережі (CNN) використовуються для аналізу послідовностей тексту та візуальних елементів фішингових сайтів, що дозволяє підвищити точність виявлення складних атак [19].

Таким чином, різні алгоритми машинного навчання застосовуються для виявлення фішингу залежно від типу атаки, доступних ознак та вимог до швидкодії системи. Водночас більшість традиційних ML-підходів мають обмежену здатність до глибокого контекстного аналізу повідомлень, що створює передумови для переходу до використання великих мовних моделей у задачах виявлення фішингу [11, 18, 21].

РОЗДІЛ 2 ОЦІНКА ПОТЕНЦІАЛУ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ ДЛЯ ВИЯВЛЕННЯ ФІШИНГОВИХ ПОВІДОМЛЕНЬ

Великі мовні моделі (Large Language Models, LLM) базуються на архітектурі Transformer, яка була запропонована у роботі “Attention Is All You Need” [22] і стала фундаментом сучасних систем обробки природної мови. На відміну від рекурентних і згорткових нейронних мереж, Transformer не використовує послідовну обробку даних, а ґрунтується на механізмі самоуваги (self-attention), що дозволяє ефективно моделювати залежності між словами незалежно від їх позиції у тексті.

2.1 Концептуальна основа LLM

Велика мовна модель (Large Language Model, LLM) – це ймовірнісна модель мови, побудована на глибоких нейронних мережах, основною метою якої є моделювання розподілу ймовірностей мовних послідовностей. Текст розглядається не як набір правил, а як послідовність випадкових змінних:

$$w_1, w_2, \cdots, w_n, \tag{2.1}$$

де w_1 – перший токен у тексті;

w_2 – другий токен у тексті;

w_n – n-ий токен у тексті.

Токен у великих мовних моделях визначається як мінімальна технічна одиниця тексту (не лінгвістична), з якою безпосередньо працює модель у процесі навчання та інференсу. На відміну від традиційного лінгвістичного підходу, де базовою одиницею аналізу є слово, у LLM токеном може виступати як повне слово, так і його частина, окремий символ, знак пунктуації або спеціальний службовий елемент. Такий підхід зумовлений необхідністю ефективного статистичного моделювання мови, зменшення розміру словника та забезпечення здатності моделі працювати з новими або рідкісними лексичними одиницями.

Текст у мовній моделі подається у вигляді впорядкованої послідовності токенів, кожен з яких надалі перетворюється у векторне подання та аналізується з урахуванням контексту всієї послідовності, що дозволяє моделі відтворювати складні семантичні та синтаксичні закономірності природної мови.

Врахувавши (2.1) можна записати математичне формулювання повної ймовірності тексту, а саме:

$$P(w_1, w_2, \cdots, w_n). \tag{2.2}$$

У контексті LMM текст розглядається як впорядкована послідовність токенів, для якої може бути визначена повна ймовірність появи в природній мові. Повна ймовірність тексту інтерпретується як міра того, наскільки ймовірною є поява конкретної послідовності токенів відповідно до мовних закономірностей, засвоєних моделлю під час навчання. Формально ймовірність тексту визначається як спільна ймовірність усіх токенів, що входять до його складу, і залежить від статистичних взаємозв'язків між елементами послідовності.

З огляду на високу розмірність мовного простору, безпосереднє обчислення повної ймовірності тексту є обчислювально складним, тому в мовних моделях застосовується ланцюгове правило ймовірностей. Згідно з цим правилом, спільна ймовірність усієї послідовності подається як добуток умовних ймовірностей кожного окремого токена за умови появи всіх попередніх токенів у тексті (2.3).

$$P(w_1, w_2, \cdots, w_n) = \prod_{t-1}^{n} P(w_1, w_2, \cdots, w_{t-1}), \tag{2.3}$$

де w_t – токен на позиції t;

t – індекс позиції токена в послідовності;

n – загальна кількість токенів;

$w_1, w_2, \cdots, w_{t-1}$ – усі попередні токени (контекст);

Зазначена формула відображає принцип, згідно з яким повна ймовірність тексту визначається як добуток умовних ймовірностей окремих токенів, кожен з

яких з’являється за умови наявності всіх попередніх токенів у послідовності. Такий підхід означає, що процес оцінювання ймовірності тексту має послідовний характер: спочатку мовна модель визначає ймовірність появи першого токена, після чого оцінює ймовірність другого токена з урахуванням уже відомого першого елемента. Надалі кожен наступний токен аналізується з урахуванням усього накопиченого контексту, тобто всіх попередніх токенів, що дозволяє моделі поступово формувати цілісне ймовірнісне уявлення про структуру та зміст тексту.

У межах мовного моделювання це означає, що кожен токен у тексті розглядається не ізольовано, а як результат умовного вибору, залежного від попереднього мовного контексту. Ланцюгове правило дозволяє мовній моделі поступово формувати уявлення про структуру тексту, враховуючи як локальні синтаксичні залежності між сусідніми токенами, так і глобальні семантичні зв’язки, що охоплюють значні фрагменти тексту. Завдяки цьому LLM здатні моделювати довготривалі залежності, які є характерними для природної мови.

Застосування ланцюгового правила ймовірностей фактично зводить задачу мовного моделювання до передбачення наступного токена на основі попередніх, що є центральною ідеєю функціонування великих мовних моделей. У процесі навчання модель оптимізується таким чином, щоб максимально точно оцінювати умовні ймовірності токенів у різних контекстах, поступово засвоюючи граматичні, семантичні та прагматичні закономірності мови. Таким чином, ланцюгове правило ймовірностей є фундаментальним математичним інструментом, який забезпечує теоретичну цілісність та практичну ефективність сучасних LLM.

Розглянемо приклад на реальному тексті, який часто використовується у фішингу. Нехай маємо фразу “Your account has been suspended”. Після токенізації (умовної) отримаємо наступний масива токенів:

$$w_1 = Your, w_2 = account, w_3 = has, w_4 = been, w_5 = suspended \qquad (2.4)$$

Тоді повна ймовірність такого тексту з використанням ланцюгового правила (2.3) становитиме:

$$P(\text{Your account has been suspended}) = P(Your) * P(account|Your) * P(has|\ Your\ account) * P\ (been|Your\ account\ has) * P(suspended|Your\ account\ has\ been) \quad (2.5)$$

При цьому кожна умовна ймовірність вивчається моделлю під час навчання. Такий підхід дозволяє декомпозувати складну задачу оцінювання повної ймовірності тексту на серію локальних задач передбачення наступного токена на основі наявного контексту.

Узагальнено приклад роботи LLM, формуючи певний розподіл відповідей, у конкретному моменті для вхідного контексту: “Your account has been” виглядає наступним чином (таблиця 2.1):

Таблиця 2.1 – Розподіл відповідей моделі LLM

Токен	Ймовірність
suspended	0,42
verified	0,21
updated	0,14
locked	0,11
closed	0,08
інші	0,04

LMM не володіє явним визначенням поняття “фішинг” і не оперує ним як формалізованою категорією, однак у процесі навчання вона засвоює статистичні мовні закономірності, завдяки яким здатна розпізнавати типові лінгвістичні конструкції, що часто використовуються у загрозливих або фішингових повідомленнях.

2.2 Обґрунтування вибору методу дослідження

LMM не володіє явним визначенням поняття “фішинг” і не оперує ним як формалізованою категорією, однак у процесі навчання вона засвоює статистичні мовні закономірності, завдяки яким здатна розпізнавати типові лінгвістичні конструкції, що часто використовуються у загрозливих або фішингових повідомленнях.

Незважаючи на те, що з формальної точки зору великі мовні моделі реалізують задачу передбачення наступного токена, їх функціонування не обмежується простим статистичним “вгадуванням” слів. У процесі навчання LLM формують глибокі контекстні представлення тексту, що дозволяє їм засвоювати семантичні зв’язки між мовними одиницями, розпізнавати стилістичні особливості повідомлень, а також виявляти прагматичний намір автора. На відміну від традиційних підходів, заснованих на фіксованих правилах або поверхневих лексичних ознаках, LLM здатні узагальнювати мовні патерни, характерні для різних типів текстів, зокрема шахрайських і фішингових.

Завдяки формуванню внутрішнього простору контекстних значень великі мовні моделі здатні інтегрувати інформацію з усього тексту повідомлення, враховуючи не лише окремі ключові слова, а й їхнє поєднання, порядок розташування, емоційне забарвлення та рівень терміновості. Саме ці характеристики є типовими ознаками фішингових повідомлень і часто залишаються непоміченими класичними методами аналізу тексту. Таким чином, використання LLM у задачі виявлення фішингу є обґрунтованим, оскільки такі моделі здатні ефективно класифікувати текстові повідомлення, розпізнавати приховані шахрайські наміри та узагальнювати зміст повідомлень навіть за відсутності явних маркерів фішингової атаки.

У межах даного дослідження для задачі виявлення фішингових повідомлень було обрано підхід на основі великих мовних моделей (Large Language Models, LLM), що зумовлено обмеженнями традиційних методів аналізу тексту та зростаючою складністю сучасних фішингових атак. Фішингові повідомлення дедалі частіше використовують адаптивні мовні конструкції, контекстну

маскування, персоналізацію та соціальну інженерію, що істотно знижує ефективність методів, заснованих на фіксованих правилах або поверхневих статистичних ознаках.

На відміну від rule-based підходів, що ґрунтуються на заздалегідь визначених правилах та списках ключових слів, великі мовні моделі не залежать від жорстко фіксованих ознак і не потребують постійного ручного оновлення правил у відповідь на появу нових фішингових шаблонів. Правило-орієнтовані системи зазвичай ефективні лише проти відомих сценаріїв атак і демонструють низьку стійкість до навмисних модифікацій тексту, таких як перефразування, використання синонімів або зміна стилю повідомлення. У свою чергу, класичні методи машинного навчання [23], що базуються на попередньо визначених лексичних або статистичних ознаках, обмежені якістю та повнотою ознакового простору і, як правило, не здатні повною мірою враховувати довготривалий контекст повідомлення. Великі мовні моделі здійснюють глибокий контекстний аналіз тексту, формуючи внутрішні семантичні представлення, які охоплюють не лише лексичний склад повідомлення, а й його тон, структуру, прагматичний намір та рівень психологічного впливу. Це є критично важливим для виявлення так званого zero-day фішингу – атак, що не мають попередніх зразків у навчальних даних і навмисно уникають відомих сигнатур. Завдяки здатності до узагальнення мовних патернів та аналізу наміру повідомлення LLM демонструють підвищену стійкість до нових і модифікованих фішингових сценаріїв, що обґрунтовує їх вибір як основного методу дослідження у даній роботі.

Для наочного узагальнення відмінностей між основними підходами до виявлення фішингових повідомлень та обґрунтування доцільності використання великих мовних моделей доцільно здійснити їх порівняльний аналіз за ключовими критеріями, такими як здатність до контекстного аналізу, адаптивність до нових типів атак та стійкість до zero-day фішингу. Відповідні характеристики rule-based, класичних методів машинного навчання та LLM-підходів наведено в таблиці нижче [23-25].

Таблиця 2.2 – Порівняльна характеристика підходів до виявлення фішингових повідомлень

Критерій	Rule-based підходи	Класичні ML-підходи	LLM-підходи
Основний принцип	Фіксовані правила та ключові слова	Класифікація на основі ознак	Контекстне мовне моделювання
Потреба в ручному налаштуванні	Висока	Середня	Низька
Урахування контексту	Обмежене	Часткове	Повне
Стійкість до перефразування	Низька	Середня	Висока
Виявлення zero-day фішингу	Практично відсутнє	Обмежене	Високе
Адаптація до нових шаблонів	Повільна	Потребує перенавчання	Контекстне узагальнення
Аналіз наміру повідомлення	Відсутній	Обмежений	Виражений
Масштабованість	Низька	Середня	Висока

Таким чином, вибір великих мовних моделей як методу дослідження у задачі виявлення фішингових повідомлень є обґрунтованим з огляду на їхню здатність до глибокого контекстного аналізу, узагальнення мовних закономірностей та ефективного виявлення zero-day атак. На відміну від rule-based і класичних ML-підходів, LLM забезпечують вищий рівень адаптивності та стійкості до еволюції методів соціальної інженерії, що робить їх перспективним інструментом для сучасних систем кібербезпеки.

2.3 Архітектура LLM

Протягом тривалого часу домінуючу роль у задачах обробки послідовних даних відігравали рекурентні нейронні мережі, зокрема моделі типу LSTM та

GRU, які широко застосовувалися для аналізу та перетворення мовних послідовностей, включаючи задачі мовного моделювання та автоматичного перекладу. Значна кількість досліджень була спрямована на вдосконалення таких підходів, а також на розвиток архітектур encoder–decoder з метою підвищення їхньої продуктивності та точності.

Особливістю рекурентних моделей є те, що обчислювальний процес у них тісно пов’язаний із порядком елементів у послідовності. Формування кожного нового прихованого стану відбувається на основі попереднього стану та поточного вхідного елемента, що зумовлює суворо послідовний характер обробки даних. Така залежність від часових кроків унеможливлює ефективну паралельну обробку всередині одного навчального прикладу, що стає критичним при роботі з довгими послідовностями, коли апаратні обмеження пам’яті не дозволяють збільшувати розмір пакетів навчальних даних. Хоча окремі сучасні підходи дозволяють частково оптимізувати обчислення шляхом структурних перетворень або використання умовних механізмів активації, ключове обмеження, пов’язане з послідовною природою рекурентних мереж, залишається нездоланим.

Водночас механізми уваги поступово стали важливою складовою моделей аналізу послідовностей, оскільки вони забезпечують можливість враховувати взаємозв’язки між елементами незалежно від їхньої відстані у тексті. Незважаючи на це, у більшості існуючих рішень механізми уваги застосовувалися як доповнення до рекурентних архітектур, а не як самостійна основа моделі.

Принципово інший підхід запропоновано в архітектурі Transformer [22], яка повністю відмовляється від рекурентної обробки та використовує механізми уваги як єдиний засіб побудови контекстних залежностей між вхідними та вихідними послідовностями (рис. 2.1). Така концепція дозволяє значно підвищити рівень паралелізації обчислень і досягти високих показників якості в задачах машинного перекладу за істотно менший час навчання, демонструючи новий рівень ефективності порівняно з традиційними підходами.

Переважна частина сучасних ефективних нейронних моделей для обробки та перетворення послідовних даних базується на архітектурі типу encoder–decoder. У такій моделі вхідна послідовність символьних даних $(x_1, x_2, \cdots, x_n)$ спочатку кодується енкодером у послідовність безперервних векторів $z = (z_1, z_2, \cdots, z_n)$, які відображають узагальнене контекстне представлення вхідної інформації. Використовуючи ці векторні репрезентації z, декодер здійснює поетапну генерацію вихідної послідовності символів $(y_1, y_2, \cdots, y_m)$.

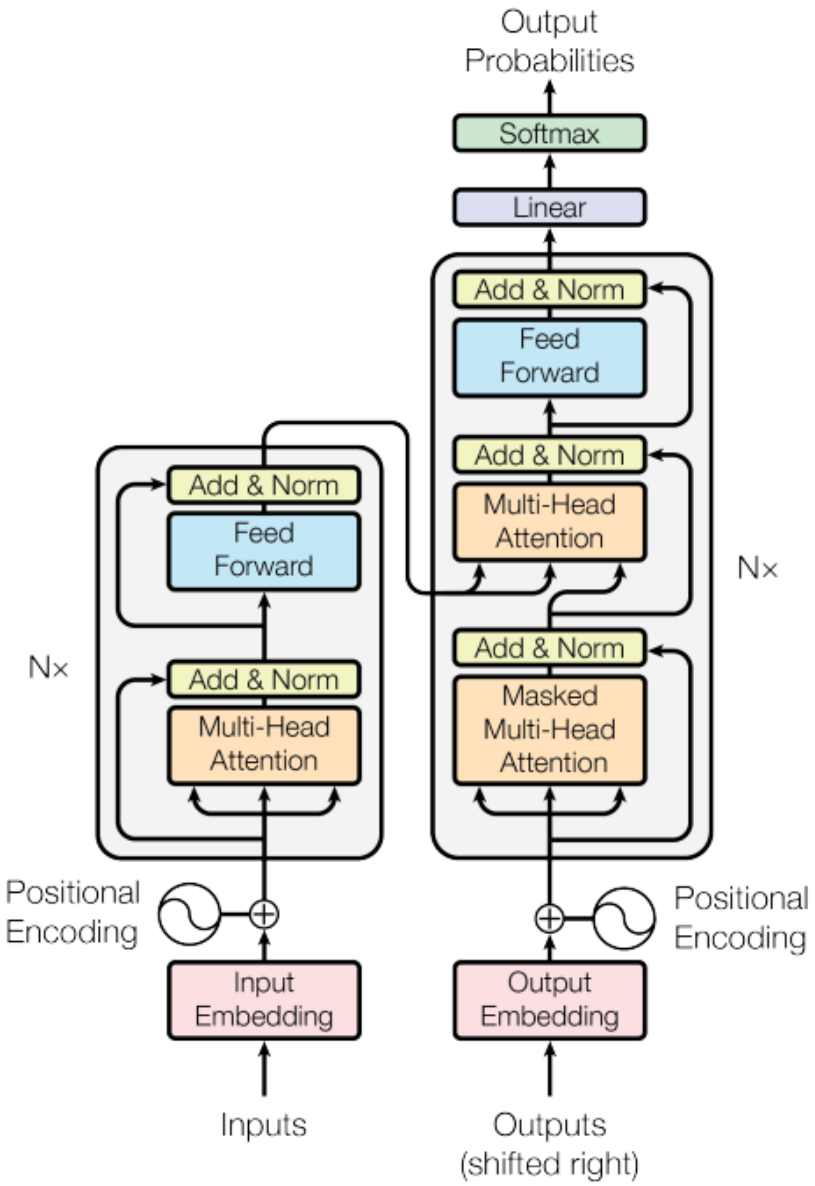


Рисунок 2.1 – Архітектура Transformer

На кожному кроці генерації модель працює в авторегресивному режимі, використовуючи раніше згенеровані символи як додатковий вхід для

передбачення наступного елемента послідовності. Архітектура Transformer дотримується цієї загальної концепції encoder–decoder, проте реалізує її за допомогою стеків механізмів самоуваги (self-attention) та повнозв'язних нейронних шарів, які застосовуються як в енкодері, так і в декодері, що схематично зображено відповідно в лівій та правій частинах рисунка 2.1.

Класична архітектура Transformer складається з послідовності однакових шарів (layers), кожен з яких включає механізм багатоголової самоуваги (Multi-Head Self-Attention) та повнозв'язну нейронну мережу прямого поширення (Feed-Forward Network, FFN). Для забезпечення стабільності навчання та ефективного поширення градієнтів застосовуються залишкові з'єднання (residual connections) та нормалізація шарів (Layer Normalization).

На вхід моделі подається послідовність токенів, кожен з яких перетворюється у векторне представлення фіксованої розмірності (embedding). Оскільки Transformer не має вбудованого механізму обробки порядку слів, до embedding-дискретизації додається позиційне кодування (positional encoding), яке містить інформацію про відносне або абсолютне розташування токенів у послідовності. Це дозволяє моделі коректно враховувати синтаксичну структуру речень та логіку тексту.

2.3.1 Енкодер та декодер

Енкодер у архітектурі Transformer реалізований у вигляді послідовності з шести однакових за структурою шарів. Кожен із цих шарів містить два функціональні компоненти. Перший відповідає за механізм багатоголової самоуваги, який дозволяє кожному елементу вхідної послідовності взаємодіяти з усіма іншими елементами. Другий компонент є позиційно-незалежною повнозв'язною нейронною мережею прямого поширення, що виконує нелінійне перетворення векторних представлень [22].

Для підвищення стабільності навчання та покращення поширення градієнтів навколо кожного з цих компонентів застосовуються залишкові з'єднання (residual connection), після яких виконується нормалізація за шарами.

Формально це означає, що вихід кожного підблоку обчислюється як нормалізована сума його вхідних даних та результату відповідного перетворення. Для забезпечення коректної роботи залишкових з'єднань усі підшари моделі, а також шари вбудованих представлень, формують виходи однакової розмірності, яка у базовій конфігурації становить $d_{model} = 512$ [22].

Декодер має аналогічну багатошарову структуру та також складається з шести ідентичних шарів. Однак, на відміну від енкодера, кожен шар декодера включає додатковий функціональний блок, призначений для реалізації механізму багатоголової уваги до виходів енкодера. Це дозволяє моделі поєднувати інформацію з вхідної послідовності з уже згенерованим контекстом вихідних даних.

Як і в енкодері, навколо кожного підблоку декодера використовуються залишкові з'єднання з подальшою нормалізацією. Окрему модифікацію зазнає механізм самоуваги в декодері: у ньому застосовується маскування, яке обмежує доступ до майбутніх позицій у вихідній послідовності. У поєднанні зі зсувом вхідних embedding-векторів на одну позицію це гарантує, що передбачення для позиції *i* ґрунтується виключно на вже відомих елементах із позицій, що передують *i*.

2.3.2 Механізм Attention

Центральним елементом Transformer є механізм уваги Attention, який визначає, які частини вхідної послідовності є найбільш релевантними для обробки кожного конкретного токена. У межах самоуваги кожен токен одночасно виступає як джерело інформації та як контекст для інших токенів.

Для реалізації цього механізму кожне вхідне векторне представлення проєктується у три окремі простори: запит (Query, Q), ключ (Key, K) та значення (Value, V). Запит відображає інформаційні потреби поточного токена, ключі – характеристики інших токенів, а значення – фактичний контент, який може бути використаний для формування контексту.

Математично масштабована скалярна увага (Scaled Dot-Product Attention) описується виразом та представлена на рис 2.2:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (2.6)$$

де $Q \in \mathbb{R}^{n \times d_k}, K \in \mathbb{R}^{n \times d_k}, V \in \mathbb{R}^{n \times d_k}$;

n – довжина вхідної послідовності;

d_k – розмірність векторів ключів.

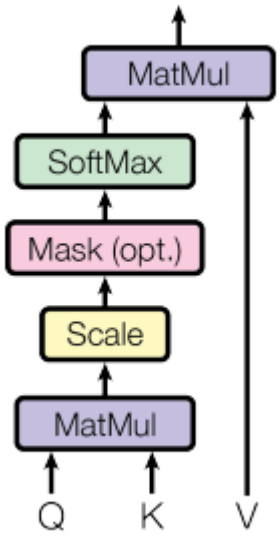


Рисунок 2.2 – Схематичне зображення Scaled Dot-Product Attention

Скалярний добуток QK^T відображає ступінь релевантності між токенами, а функція softmax нормалізує ці значення, перетворюючи їх у ваги уваги. Ділення на $\sqrt{d_k}$запобігає надмірному зростанню значень та стабілізує процес навчання. У результаті кожен токен отримує зважене представлення контексту всієї послідовності.

Для підвищення виразної здатності моделі Transformer використовує механізм багатоголової уваги (Multi-Head Attention, рис. 2.3). Замість одного простору уваги застосовується декілька паралельних «голів», кожна з яких

навчається фокусуватися на різних аспектах тексту – синтаксичних, семантичних або прагматичних.

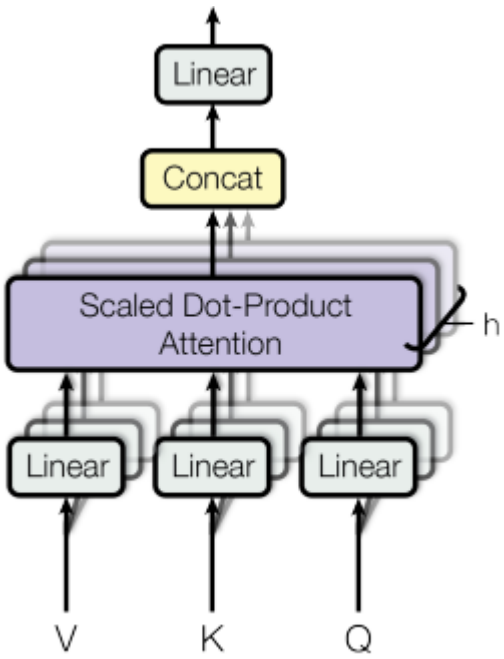


Рисунок 2.3– Схематичне зображення Scaled Dot-Product Attention

Формально багатоголова увага визначається як конкатенація результатів окремих голів з подальшою лінійною трансформацією:

$$Multihead(Q, K, V) = concat(head_1, head_2, \cdots, head_h)W^o, \quad (2.7)$$

де кожна голова визначається як:

$$head_i = Attention\left(QW_i^Q, KW_i^k, VW_i^V\right). \quad (2.8)$$

Такий підхід дозволяє моделі одночасно аналізувати, наприклад, структуру речення, емоційний тон, соціотехнічні тригери та приховані маніпулятивні елементи, що є особливо важливим для задач виявлення фішингових повідомлень.

2.4 Попереднє навчання та fine-tuning LLM

Попереднє навчання (pre-training) LLM здійснюється на надзвичайно великих корпусах текстових даних, які охоплюють різні жанри, стилі та домени. Найпоширенішою задачею pre-training є автогресивне мовне моделювання — передбачення наступного токена на основі попереднього контексту. Функція втрат у цьому випадку визначається як негативна логарифмічна правдоподібність послідовності.

У процесі pre-training модель формує універсальні мовні репрезентації, які відображають граматичні правила, семантичні зв'язки, дискурсивні закономірності та загальні мовні патерни. Важливо, що на цьому етапі модель не оптимізується під конкретну прикладну задачу, а навчається загальному розумінню мови.

Після завершення етапу попереднього навчання (pre-training) великі мовні моделі володіють широкими загальномовними знаннями, однак їхня поведінка залишається універсальною і не орієнтованою на конкретні прикладні сценарії. Для адаптації моделі до визначених завдань застосовується етап fine-tuning, який полягає у додатковому навчанні на спеціалізованих, зазвичай розмічених, наборах даних. Метою fine-tuning є збереження загальних мовних репрезентацій, сформованих під час pre-training, одночасно зі спеціалізацією моделі на цільовому домені або типі задач.

На відміну від pre-training, який використовує великі обсяги різнорідних нерозмічених текстів, fine-tuning здійснюється на відносно компактних, але якісно підготовлених датасетах, що відповідають конкретній задачі. Параметри моделі коригуються з меншим кроком навчання, щоб уникнути руйнування вже набутих мовних знань (ефект catastrophic forgetting). Таким чином, fine-tuning можна розглядати як процес "точного налаштування" моделі, під час якого загальні мовні патерни доповнюються прикладними правилами інтерпретації тексту.

У контексті виявлення фішингових повідомлень fine-tuning найчастіше використовується для задач бінарної або багатокласової класифікації, де модель

навчається відрізняти фішингові повідомлення від легітимних [26]. Для цього текст повідомлення подається на вхід LLM, а вихід моделі узгоджується з відповідною міткою класу. Під час навчання модель вчиться асоціювати певні мовні патерни з ознаками соціотехнічної маніпуляції, такими як створення відчуття терміновості, апеляція до авторитету, обіцянка винагороди або загроза санкцій.

Окрім класифікації, fine-tuning може застосовуватися для аналізу намірів (intent detection), коли модель визначає приховану мету повідомлення, а також для виділення підозрілих фрагментів тексту, що дозволяє пояснювати, чому конкретне повідомлення вважається потенційно небезпечним. Такий підхід є важливим з точки зору інтерпретованості результатів, що має велике значення у прикладних системах кібербезпеки .

Окремим різновидом fine-tuning є instruction tuning, під час якого модель навчається коректно реагувати на запити, сформульовані у вигляді інструкцій або завдань. У цьому випадку навчальні приклади містять пари “інструкція – очікувана відповідь”, що формує у моделі здатність узагальнювати формат запиту, а не лише його зміст. Для задач виявлення фішингу instruction tuning дозволяє моделі ефективно працювати з аналітичними запитами типу: “визнач, чи є це повідомлення фішинговим і поясни причини” або “виділи елементи соціальної інженерії у наведеному тексті”.

Завдяки instruction tuning LLM стають більш гнучкими й здатними виконувати складні багатокрокові завдання, поєднуючи класифікацію, аналіз і генерацію пояснень в межах одного запиту.

Для подальшого вдосконалення поведінки моделі застосовується підхід reinforcement learning from human feedback (RLHF). У цьому процесі експерти або аналітики оцінюють згенеровані моделлю відповіді за критеріями корисності, точності, безпечності та відповідності очікуванням користувачів. На основі цих оцінок формується модель винагороди, яка використовується для оптимізації основної LLM за допомогою методів підкріплювального навчання.

У сфері кібербезпеки RLHF дозволяє зменшити кількість хибнопозитивних спрацьовувань, покращити якість пояснень і забезпечити більш консервативну

та обґрунтовану поведінку моделі при аналізі потенційно небезпечних повідомлень.

Застосування fine-tuning забезпечує кілька ключових переваг у задачах виявлення фішингу. По-перше, модель адаптується до актуальних шаблонів атак, які постійно змінюються та еволюціонують. По-друге, LLM здатні враховувати не лише окремі ключові слова, а й глибокий контекст, стилістичні особливості та приховані наміри автора повідомлення. По-третє, fine-tuning дозволяє інтегрувати експертні знання у процес навчання, підвищуючи точність і практичну цінність системи.

Таким чином, fine-tuning виступає критично важливим етапом життєвого циклу великих мовних моделей, який перетворює універсальну мовну модель на ефективний інструмент прикладного аналізу текстів у сфері кібербезпеки та виявлення фішингових загроз.

У результаті аналізу принципів побудови та функціонування великих мовних моделей було обґрунтовано їхню високу придатність для задач виявлення фішингових повідомлень. Застосування механізму attention у поєднанні з глибоким контекстним опрацюванням тексту забезпечує здатність LLM ідентифікувати не лише прямі лексичні ознаки фішингу, а й приховані семантичні, прагматичні та соціотехнічні закономірності. Завдяки цьому архітектура Transformer демонструє суттєві переваги у порівнянні з rule-based та класичними machine learning підходами, зберігаючи ефективність навіть в умовах постійної еволюції та ускладнення фішингових атак.

РОЗДІЛ 3 ВИЯВЛЕННЯ ФІШИНГОВИХ ПОВІДОМЛЕНЬ З ВИКОРИСТАННЯМ LLM

3.1 Постановка задачі та загальна схема експериментального дослідження

Метою даного розділу є розробка та обґрунтування експериментальної методології виявлення фішингових повідомлень на основі великих мовних моделей із подальшим порівнянням їх ефективності з класичними алгоритмами машинного навчання. На відміну від підходів, що базуються виключно на аналізі заздалегідь визначених ознак або правил, запропонована методологія передбачає навчання моделей безпосередньо на текстових даних електронних повідомлень, що дозволяє враховувати семантичний, контекстний і прагматичний рівні інформації.

У межах дослідження задача виявлення фішингових повідомлень формалізується як задача бінарної класифікації, у якій кожне електронне повідомлення належить до одного з двох класів:

- phishing – фішингове повідомлення;
- legitimate – легітимне електронне повідомлення.

Для забезпечення репрезентативності та реалістичності експериментів у роботі використовується об’єднаний набір даних, сформований на основі двох відкритих датасетів: Phishing Email Dataset, що містить приклади фішингових повідомлень, та Enron Email Dataset, який використовується як джерело автентичного легітимного корпоративного листування. Таке поєднання дозволяє змоделювати реальні умови електронної комунікації, у яких системи виявлення фішингу мають відрізняти шахрайський контент від звичайного ділового листування.

Загальна схема (рис. 3.1) експериментального дослідження передбачає кілька послідовних етапів. На першому етапі здійснюється попередня обробка текстових даних, зокрема очищення повідомлень, уніфікація формату та балансування класів. Далі сформований датасет поділяється на навчальну та

тестову вибірки, що забезпечує коректне оцінювання узагальнювальної здатності моделей.

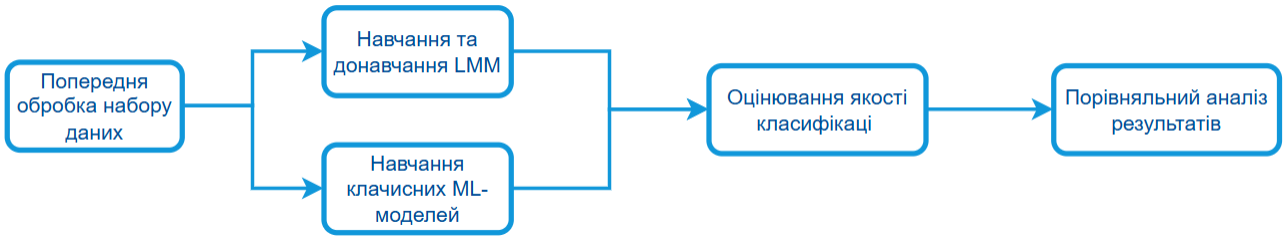


Рисунок 3.1 – Схематичне зображення етапів експериментального дослідження

На наступному етапі проводиться навчання та донавчання (fine-tuning) великих мовних моделей для задачі класифікації фішингових повідомлень. LLM аналізують тексти електронних листів без попереднього ручного проєктування ознак, використовуючи внутрішні мовні репрезентації для розпізнавання характерних патернів фішингу, зокрема соціотехнічних маніпуляцій, прихованих закликів до дії та нетипових контекстних залежностей.

Паралельно з цим реалізується базова експериментальна лінія з використанням класичних алгоритмів машинного навчання, таких як логістична регресія, метод опорних векторів та наївний баєсівський класифікатор. Для цих алгоритмів застосовуються традиційні підходи до представлення тексту, зокрема bag-of-words та TF-IDF, що дозволяє забезпечити коректне порівняння результатів із моделями LLM.

Заключним етапом експериментального дослідження є оцінювання якості класифікації з використанням стандартних метрик, таких як accuracy, precision, recall, F1-score та аналіз матриць помилок. Отримані результати використовуються для порівняльного аналізу ефективності великих мовних моделей і класичних ML-підходів, а також для формулювання висновків щодо доцільності застосування LLM у системах виявлення фішингових повідомлень.

Узагальнюючи вище написане, обрана методологія дозволяє комплексно оцінити потенціал великих мовних моделей у задачі виявлення фішингу в умовах використання реалістичних даних та порівняти їх з традиційними підходами машинного навчання.

3.1 Характеристика досліджуваного датасету

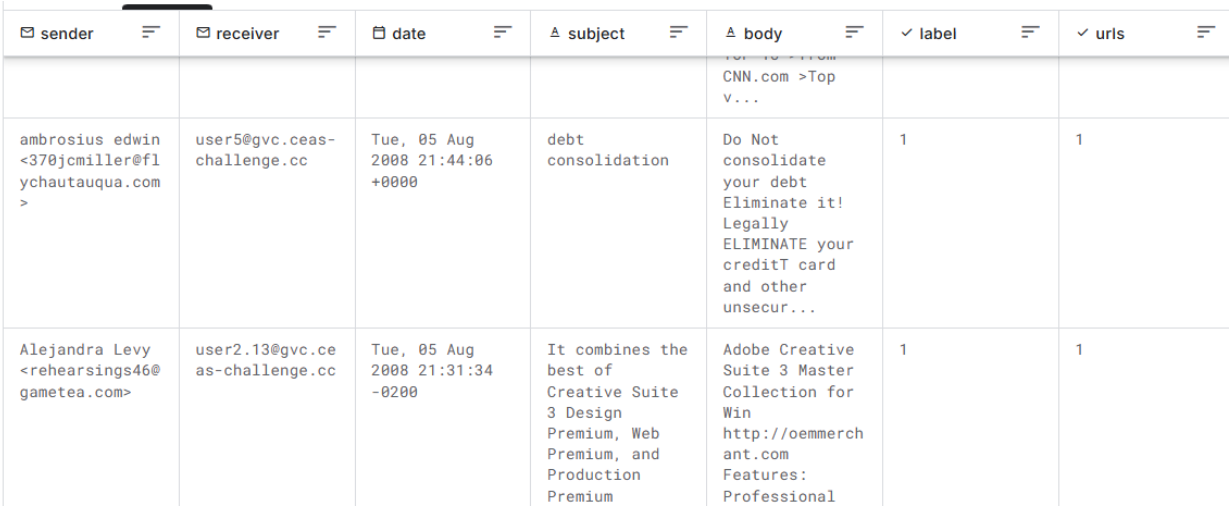
Для проведення експериментального дослідження в межах даної роботи було обрано два широко відомі та часто використовувані у наукових дослідженнях набори даних: Phishing Email Dataset та Enron Email Dataset. Використання цих датасетів дозволяє сформувати репрезентативну вибірку як фішингових, так і легітимних електронних повідомлень, а також забезпечує коректне порівняння результатів між підходами, заснованими на великих мовних моделях, та класичними алгоритмами машинного навчання. Аналогічний підхід щодо конкатенації двох датасетів був використаний у роботі [27].

Phishing Email Dataset [28] є одним із найбільш поширених відкритих наборів даних, призначених для досліджень у сфері виявлення фішингових атак. Датасет містить колекцію електронних листів, класифікованих за ознакою фішингової або нефішингової природи, та використовується у численних наукових роботах з аналізу текстових ознак, URL-структур і поведінкових характеристик фішингових повідомлень.

Основною особливістю Phishing Email Dataset є його орієнтація саме на шахрайський контент, що включає повідомлення з типовими для фішингу ознаками: заклики до термінових дій, посилання на підроблені вебресурси, запити на введення облікових або платіжних даних, а також використання соціотехнічних маніпуляцій. Тексти повідомлень у датасеті часто імітують листування від банків, платіжних систем, служб підтримки або популярних онлайн-платформ.

Датасет зазвичай містить (рис 3.2):

- тіло електронного листа;
- заголовки (subject);
- мітку класу (phishing / legitimate);
- у деяких версіях — додаткові метадані, зокрема інформацію про URL або вкладення.



sender	receiver	date	subject	body	label	urls
				CNN.com >Top v...		
ambrosius edwin <370jcmiller@flychautauqua.com>	user5@gvc.ceas-challenge.cc	Tue, 05 Aug 2008 21:44:06 +0000	debt consolidation	Do Not consolidate your debt Eliminate it! Legally ELIMINATE your creditT card and other unsecur...	1	1
Alejandra Levy <rehearsings46@gametea.com>	user2.13@gvc.ceas-challenge.cc	Tue, 05 Aug 2008 21:31:34 -0200	It combines the best of Creative Suite 3 Design Premium, Web Premium, and Production Premium	Adobe Creative Suite 3 Master Collection for Win http://oemmerchant.com Features: Professional	1	1

Рисунок 3.2 – Ознаки та приклади з набору Phishing Email Dataset

Phishing Email Dataset добре підходить для навчання моделей у задачі бінарної класифікації, а також для оцінювання здатності алгоритмів розпізнавати соціотехнічні патерни у тексті. Водночас слід зазначити, що частина повідомлень у цьому наборі має відносно шаблонний характер, що може спрощувати задачу для класичних ML-алгоритмів і не повною мірою відображати сучасні високоперсоналізовані фішингові атаки.

Enron Email Dataset [29] є класичним набором даних, що містить реальне корпоративне електронне листування співробітників компанії Enron. Датасет був оприлюднений у межах судового розслідування та став одним із найбільш використовуваних джерел легітимних email-повідомлень у дослідженнях з аналізу тексту, інформаційної безпеки та машинного навчання.

На відміну від Phishing Email Dataset, Enron Email Dataset не містить фішингових повідомлень за замовчуванням, а використовується як джерело реалістичного легітимного листування. Повідомлення в датасеті охоплюють широкий спектр ділової комунікації: службові обговорення, фінансові питання, внутрішні оголошення, планування зустрічей та особисте листування.

Основні характеристики Enron Email Dataset (рис. 3.3):

- автентичні корпоративні email-повідомлення;
- наявність заголовків, адрес відправників і отримувачів;
- різноманітність стилів письма, довжини повідомлень і тематики;
- відсутність штучної генерації або шаблонності.

subject	body	label
and she said that	march . also , for days 1 - 5 , you ' ve taken...	
re : noms / actual vols for 3 / 26 / 01	eileen , our gas control records indicate the following rate changes for march 26 , 2001 flow : 20...	0
hpl nom for march 27 , 2001	(see attached file : hplno 327 . xls) - hplno 327 . xls	0
estimated actuals for april 5 , 2001	estimated actuals teco tap 24 . 917 when we receive actuals from duke , i will forward them to you...	0
re : nom / alloc for june 6 th	we agree " eileen ponton " on 06 / 07 / 2001 10 : 21 : 35 am to : david avila / lsp / enserch / us...	0

Рисунок 3.3 – Ознаки та приклади з набору Enron Email Dataset

Завдяки цим властивостям Enron Email Dataset широко використовується як базовий набір легітимних повідомлень для побудови збалансованих датасетів у задачах фішинг-детекції. У межах даного дослідження повідомлення з Enron Dataset використовуються як негативний клас (non-phishing), що дозволяє моделювати реалістичні умови корпоративного середовища.

Комбінування Phishing Email Dataset та Enron Email Dataset дозволяє сформувати збалансований і репрезентативний набір даних для навчання та тестування моделей. Такий підхід забезпечує:

- наявність реальних прикладів фішингових атак;
- використання автентичних легітимних повідомлень без штучної генерації;
- можливість оцінити здатність моделей відрізняти фішингові повідомлення від реального корпоративного листування.

Особливо важливим є те, що Enron Email Dataset містить складні, контекстно насичені тексти, які можуть бути помилково класифіковані як фішингові традиційними алгоритмами. Це створює додаткові умови для перевірки переваг великих мовних моделей, здатних виконувати глибокий семантичний та прагматичний аналіз повідомлень.

У лістингу 3.1 наведено процес балансування об’єднаного датасету шляхом випадкового відбору однакової кількості фішингових та легітимних електронних повідомлень. Такий підхід дозволяє усунути дисбаланс класів і забезпечити коректне навчання моделей без зміщення результатів класифікації в бік домінуючого класу.

Лістинг 3.1 – Балансування наборів даниз за кількістю повідомлень

```
# Визначення мінімальної кількості повідомлень
min_size = min(len(phishing_df), len(enron_df))

# Випадковий відбір однакової кількості записів
phishing_sample = phishing_df.sample(n=min_size, random_state=42)
enron_sample = enron_df.sample(n=min_size, random_state=42)
```

У процесі формування об’єднаного датасету мінімальна кількість повідомлень була визначена на основі меншого з двох вихідних наборів даних. Enron Email Dataset містить 29 767 легітимних електронних повідомлень, тоді як Phishing Email Dataset містить 82 486 фішингових повідомлень.

З метою балансування класів із кожного датасету було відібрано по 29 767 повідомлень, унаслідок чого фінальний об’єднаний набір даних (лістинг 3.2) складався з 59 534 електронних листів, з яких 29 767 фішингових та 29 767 легітимних. Така структура забезпечує повністю збалансовану вибірку для подальшого навчання та оцінювання моделей.

Лістинг 3.2 – Формування досліджуваного набору даних

```
# Об'єднання фішингових та легітимних повідомлень
```

Продовження лістингу 3.2

```
    combined_df = pd.concat([phishing_sample, enron_sample], ignore_index=True)
    # Перемішування записів
    combined_df = combined_df.sample(frac=1, random_state=42).reset_index(drop=True)
    # Перевірка розподілу класів
    print(combined_df["label"].value_counts())
```

З метою усунення технічного шуму та приведення текстових даних до уніфікованого вигляду було виконано попередню очистку електронних повідомлень. Зокрема, здійснювалася нормалізація регістру, видалення HTML-розмітки, URL-адрес і спеціальних символів, що не несуть змістового навантаження. Реалізацію зазначених операцій подано у лістингу 3.3

Лістинг 3.3 – Попередня обробка текстових даних

```
    import re

    def clean_text(text):
        text = text.lower()
        text = re.sub(r"<.*?>", "", text)       # видалення HTML
        text = re.sub(r"http\S+", "", text)     # видалення URL
        text = re.sub(r"[^a-z\s]", "", text)       # видалення спецсимволів
        text = re.sub(r"\s+", " ", text).strip()
        return text
    combined_df["text"] = combined_df["text"].apply(clean_text)
```

URL-адреси були видалені з текстових повідомлень з метою зосередження дослідження на лінгвістичних і контекстних ознаках фішингових атак. Такий підхід дозволяє уникнути впливу надмірно сильних технічних маркерів та забезпечує коректне порівняння ефективності великих мовних моделей і класичних алгоритмів машинного навчання саме на рівні аналізу текстового вмісту повідомлень.

У лістингу 3.4 показано процес формування навчальної та тестової вибірок шляхом попереднього випадкового перемішування об’єднаного датасету та подальшого розподілу у співвідношенні 80/20. Такий підхід забезпечує рівномірний розподіл класів і відтворюваність експериментальних результатів.

Лістинг 3.4 – Попередня обробка текстових даних

```
from sklearn.utils import shuffle
# Перемішування датасету
shuffled_df = shuffle(combined_df, random_state=42)
# Визначення розміру тестової вибірки
test_ratio = 0.2
test_size = int(len(shuffled_df) * test_ratio)
# Розподіл на навчальну та тестову вибірки
test_df = shuffled_df.iloc[:test_size]
train_df = shuffled_df.iloc[test_size:]
# Виділення ознак та міток
X_train = train_df["text"].values
y_train = train_df["label"].values
X_test = test_df["text"].values
y_test = test_df["label"].values
```

У результаті виконаних етапів було сформовано збалансований та уніфікований об’єднаний датасет, що поєднує фішингові та легітимні електронні повідомлення з різних джерел. Проведена попередня обробка текстових даних дозволила усунути технічний шум і зосередити подальший аналіз на лінгвістичних та контекстних ознаках повідомлень. Балансування класів забезпечило коректні умови для навчання та порівняльного оцінювання моделей без зміщення результатів у бік домінуючого класу. Сформований набір даних містить лише текст та мітку, що створює надійну експериментальну основу для подальшого навчання великих мовних моделей і класичних алгоритмів машинного навчання у задачі виявлення фішингових повідомлень.

3.3 Реалізація та fine-tuning LLM

Для експериментального дослідження в межах даної роботи було обрано кілька представників сучасних великих мовних моделей, які відрізняються підходом до навчання, архітектурою та сценаріями практичного використання. Основною метою вибору моделей є оцінювання їх придатності до задачі виявлення фішингових повідомлень у різних умовах застосування.

Першою обраною моделлю є GPT (Generative Pre-trained Transformer), яка представляє клас закритих, інструктивно орієнтованих великих мовних моделей. Моделі сімейства GPT демонструють високі показники якості у задачах аналізу природної мови, зокрема в інтерпретації наміру тексту, контексту повідомлення та прихованих соціотехнічних маніпуляцій. Завдяки механізмам instruction-following і zero-/few-shot learning GPT добре підходить для задач класифікації текстів без необхідності повного перенавчання на великих обсягах даних.

Другою моделлю, використаною у дослідженні, є LLaMA (Large Language Model Meta AI) – відкрита велика мовна модель, яка дозволяє реалізувати повноцінний процес донавчання (fine-tuning) на власних наборах даних. LLaMA є особливо цінною з точки зору наукового дослідження, оскільки забезпечує контроль над параметрами навчання, архітектурою та гіперпараметрами, що дозволяє дослідити вплив fine-tuning безпосередньо на якість виявлення фішингових повідомлень.

Для розширення архітектурного порівняння у дослідженні також використано модель FLAN-T5, яка належить до класу encoder–decoder трансформерів і підтримує парадигму text-to-text learning. У межах цієї парадигми будь-яка задача, зокрема класифікація, формулюється як задача генерації текстової відповіді. Застосування FLAN-T5 дозволяє реалізувати виявлення фішингових повідомлень у генеративному форматі, де модель безпосередньо генерує мітку класу у вигляді тексту (phishing або legitimate). Такий підхід є методологічно узгодженим із концепцією LLM та забезпечує порівняння різних генеративних архітектур у задачі фішинг-детекції.

Обраний набір моделей дозволяє оцінити потенціал великих мовних моделей у задачі виявлення фішингових повідомлень з урахуванням різних підходів до генерації, навчання та інтерпретації тексту.

Процес використання великих мовних моделей у межах даної роботи реалізовано у двох основних сценаріях:

- для моделей GPT застосовується інструктивний підхід до класифікації без повного перенавчання;
- для моделей LLaMA та FLAN-T5 реалізується донавчання (fine-tuning) на об’єднаному датасеті фішингових і легітимних електронних повідомлень.

В усіх випадках на вхід моделі подається очищений текст електронного листа, а вихідним результатом є прогноз класу повідомлення. Для забезпечення порівнюваності результатів використовується єдина схема формулювання інструкцій та уніфікований формат відповідей.

3.3.1 Реалізація класифікації фішингових повідомлень за допомогою GPT

Для експериментального дослідження в межах даної роботи було обрано кілька представників сучасних великих мовних моделей, які відрізняються підходом до навчання, архітектурою та сценаріями практичного

У межах даного дослідження класифікація фішингових повідомлень за допомогою GPT реалізується з використанням двох основних підходів: instruction-based classification та few-shot learning, які не потребують класичного донавчання моделі шляхом градієнтної оптимізації, але водночас дозволяють ефективно адаптувати поведінку моделі до конкретної прикладної задачі.

Instruction-based підхід передбачає формулювання задачі у вигляді чіткої текстової інструкції, яка задає роль моделі, описує мету аналізу та визначає формат очікуваного результату. У такому випадку GPT розглядається як універсальний мовний агент, який інтерпретує інструкцію та застосовує свої узагальнені мовні знання для прийняття рішення (лістинг 3.5).

Лістинг 3.5 – Текстова інструкція для GPT

```
You are a cybersecurity expert specializing in phishing detection.
Analyze the following email message and determine whether it is phishing or legitimate.
Return only one word as the answer:
- phishing
- legitimate
Email text:
"{EMAIL_TEXT}"
```

У даному лістингу {EMAIL_TEXT} позначає текст електронного листа з об’єднаного датасету після попередньої очистки та нормалізації. Обмеження формату відповіді одним словом дозволяє автоматизувати подальшу обробку результатів та обчислення метрик якості класифікації.

Для підвищення точності та стабільності результатів у роботі також застосовано few-shot підхід, який передбачає надання моделі кількох прикладів коректної класифікації безпосередньо у запиті. Такі приклади виконують роль неявного навчального набору та дозволяють адаптувати поведінку моделі до специфіки задачі.

Лістинг 3.6 – Few-shot для GPT

```
You are a cybersecurity expert specializing in phishing detection.
Below are examples of email messages and their correct classifications.
Example 1:
Email text:
"Your account has been suspended due to suspicious activity. Please click the link below to verify your identity."
Classification: phishing
Example 2:
Email text:
"Dear team, please find attached the minutes from yesterday’s meeting."
```

```
Classification: legitimate
Example 3:
Email text:
"We detected unusual login attempts. Confirm your account details immediately to avoid service interruption."
Classification: phishing
Now analyze the following email message and determine its class.
Return only one word as the answer:
- phishing
- legitimate
Email text:
"{EMAIL_TEXT}"
```

Даний формат промпту забезпечує відтворюваність експерименту та дозволяє коректно порівнювати результати GPT з класичними алгоритмами машинного навчання та іншими великими мовними моделями.

У процесі експериментального застосування GPT для класифікації фішингових повідомлень було проаналізовано вплив параметрів prompt-налаштування на якість і стабільність результатів. Найкращі показники забезпечила конфігурація з низькою температурою (0,0–0,2), що мінімізує стохастичність і забезпечує детерміновані відповіді. Оптимальною виявилася кількість 2–3 few-shot прикладів, які суттєво підвищували точність класифікації без перевантаження контексту. Максимальна довжина контексту обмежувалася 1 000–1 500 токенами, що є достатнім для більшості електронних листів після попередньої обробки. Збільшення кількості прикладів або довжини контексту не давало істотного приросту якості. Саме зазначена конфігурація була використана для остаточного оцінювання моделі GPT у межах дослідження.

3.3.2 Реалізація та fine-tuning моделі LLaMA для класифікації фішингових повідомлень

На відміну від моделей сімейства GPT, які в межах даного дослідження використовуються переважно у режимах instruction-based та few-shot inference

без прямого оновлення параметрів, модель LLaMA надає можливість реалізувати повноцінне донавчання (fine-tuning) на власному наборі даних. Це дозволяє адаптувати параметри моделі безпосередньо до специфіки прикладної задачі з урахуванням характерних мовних, стилістичних та соціотехнічних особливостей такого контенту.

Для коректного fine-tuning LLaMA об'єднаний датасет, сформований на основі Phishing Email Dataset та Enron Email Dataset, було приведено до інструктивного формату (instruction tuning). Кожен приклад у навчальній вибірці складається з трьох основних компонентів:

- Instruction – текстове формулювання завдання для моделі;
- Input – текст електронного листа;
- Output – очікувана відповідь у вигляді класу (phishing або legitimate).

Такий формат дозволяє навчати модель не лише здійснювати класифікацію, але й коректно інтерпретувати саму постановку задачі, що є важливою перевагою instruction-tuned LLM.

Лістинг 3.7 – Підготовка даних для fine-tuning LLaMA

```
def format_example(text, label):
    return {
        "instruction": "Classify the email as phishing or legitimate.",
        "input": text,
        "output": "phishing" if label == 1 else "legitimate"
    }
train_data = [format_example(t, l) for t, l in zip(X_train, y_train)]
```

У наведеному лістингу реалізовано функцію форматування навчальних прикладів, яка перетворює сирі текстові дані та відповідні мітки класів у структурований формат, придатний для донавчання LLaMA. Такий підхід забезпечує уніфіковану подачу даних та спрощує подальше масштабування експериментів.

Після підготовки навчального датасету здійснюється безпосередній процес fine-tuning, який полягає в оптимізації параметрів моделі на основі втрат між згенерованою та еталонною відповіддю. На відміну від класичних моделей машинного навчання, де оптимізація виконується для фіксованого набору ознак, LLaMA під час донавчання оновлює внутрішні мовні репрезентації, що формуються на рівні трансформерних шарів.

У межах дослідження використано стандартний інструментарій бібліотеки Hugging Face Transformers, зокрема клас Trainer, який забезпечує абстракцію над циклом навчання, логуванням і збереженням проміжних результатів.

У лістингу 3.8 визначено основні гіперпараметри навчання, зокрема розмір батчу, кількість епох та швидкість навчання. Вибір невеликого batch size обумовлений високими обчислювальними вимогами LLaMA, тоді як значення learning rate підібрано з урахуванням рекомендацій для донавчання великих трансформерних моделей.

Лістинг 3.8 – Fine-tuning LLaMA

```
from transformers import Trainer, TrainingArguments

training_args = TrainingArguments(
    output_dir="./llama_finetuned",
    per_device_train_batch_size=2,
    num_train_epochs=3,
    learning_rate=2e-5,
    logging_steps=100,
    save_strategy="epoch"
)
trainer = Trainer(
    model=llama_model,
    args=training_args,
    train_dataset=train_dataset
)
trainer.train()
```

Fine-tuning дозволяє моделі адаптувати внутрішні мовні репрезентації до специфіки фішингових повідомлень. У результаті модель не лише запам’ятовує поверхневі ознаки, а формує узагальнене розуміння наміру повідомлення, що підвищує її здатність виявляти нові, раніше невідомі фішингові кампанії.

У процесі fine-tuning моделі LLaMA було експериментально підібрано гіперпараметри, що забезпечили найкращі показники якості на валідаційній вибірці. Оптимальною виявилася конфігурація з кількістю епох 3, швидкістю навчання 2×10^{-5} та розміром батчу 2, яка забезпечила стабільну збіжність без ознак перенавчання. Навчання проводилося у форматі instruction–input–output, що сприяло кращій адаптації моделі до задачі класифікації фішингових повідомлень. Саме ця конфігурація була використана для остаточного донавчання моделі LLaMA у межах дослідження.

Використання fine-tuning для LLaMA забезпечує більш глибоку спеціалізацію моделі порівняно з inference-only підходами та створює надійну основу для подальшого порівняльного аналізу з GPT та класичними алгоритмами машинного навчання, який буде представлено у наступних підрозділах роботи.

3.3.3 Реалізація та fine-tuning моделі LLaMA для класифікації фішингових повідомлень

Модель FLAN-T5 належить до класу генеративних трансформерних моделей, у яких усі задачі обробки природної мови формулюються в єдиній парадигмі text-to-text. У межах даного дослідження задача виявлення фішингових повідомлень була сформульована як задача генерації короткої текстової відповіді, що відповідає класу повідомлення. Такий підхід узгоджується з архітектурними особливостями FLAN-T5 і дозволяє використовувати модель без модифікації вихідного шару.

Для реалізації fine-tuning об’єднаний датасет електронних листів було перетворено у формат пар «вхідний текст – цільовий текст». Вхідний текст містив інструктивне формулювання задачі разом із текстом електронного листа, тоді як цільовим текстом виступала відповідна мітка класу (phishing або

legitimate). Такий формат дозволяє моделі одночасно інтерпретувати завдання та здійснювати класифікацію на основі змісту повідомлення.

Лістинг 3.9 – Підготовка даних для fine-tuning FLAN-T5

```
from transformers import T5Tokenizer
tokenizer = T5Tokenizer.from_pretrained("google/flan-t5-base")
def preprocess(example):
    input_text = f"Classify this email as phishing or legitimate: {example['text']}"
    target_text = "phishing" if example["label"] == 1 else "legitimate"

    return tokenizer(
        input_text,
        text_target=target_text,
        truncation=True,
        padding="max_length",
        max_length=512
    )
```

У лістингу 3.9 реалізовано етап токенізації та підготовки даних до навчання. Застосування інструктивного префікса у вхідному тексті дозволяє чітко задати контекст задачі, тоді як використання параметрів truncation і padding забезпечує уніфіковану довжину послідовностей, необхідну для пакетної обробки даних під час навчання. Максимальна довжина у 512 токенів була обрана з урахуванням середньої довжини електронних листів та обмежень обчислювальних ресурсів.

У процесі експериментального підбору гіперпараметрів було встановлено, що найкращі результати на валідаційній вибірці забезпечує конфігурація: num_train_epochs = 3, learning_rate = 3×10^{-5}, per_device_train_batch_size = 8, weight_decay = 0.01. Збільшення кількості епох понад три не давало стабільного приросту якості та підвищувало ризик перенавчання, тоді як вищі значення швидкості навчання призводили до нестабільної збіжності. Саме зазначена конфігурація була використана для остаточного донавчання FLAN-T5 у межах дослідження.

Підсумуємо вищенаписане, fine-tuning FLAN-T5 у парадигмі text-to-text дозволяє безпосередньо генерувати мітку класу як текстову відповідь, що повністю відповідає генеративній природі великих мовних моделей. Завдяки такому підходу модель навчається не лише розпізнавати фішинговий контент, але й коректно інтерпретувати інструктивні формулювання задачі, що підвищує її узагальнювальну здатність та робить придатною для використання в різних сценаріях аналізу фішингових повідомлень.

3.4 Ідентифікація фішингових повідомлень з використанням класичних алгоритмів машинного навчання

Паралельно з експериментами над великими мовними моделями у роботі було реалізовано базову експериментальну лінію з використанням класичних алгоритмів машинного навчання. Метою даного етапу є формування коректного еталонного рівня, який дозволяє об’єктивно порівняти ефективність підходів, заснованих на LLM, із традиційними методами автоматичної класифікації фішингових повідомлень.

У межах базової лінії було використано низку поширених і добре вивчених алгоритмів машинного навчання, зокрема логістичну регресію, метод опорних векторів (SVM), наївний баєсівський класифікатор та ансамблеві методи (Random Forest). Зазначені алгоритми широко застосовуються у задачах аналізу текстів і виявлення фішингу та дозволяють оцінити ефективність традиційних підходів за умов однакового набору вхідних даних.

Для представлення текстових даних у числовому вигляді використовувалися класичні методи векторизації, зокрема bag-of-words та TF-IDF. Підхід bag-of-words дозволяє моделювати текст як вектор частот слів без урахування порядку їх розташування, тоді як TF-IDF додатково враховує інформативність термінів у межах усього корпусу документів. Використання обох методів векторизації дало змогу проаналізувати вплив способу представлення тексту на результати класифікації.

Навчання та оцінювання класичних моделей здійснювалися на тому ж об’єднаному датасеті, що й експерименти з LLM, із застосуванням однакових навчальних, валідаційних і тестових вибірок. Це забезпечує коректність порівняння результатів та виключає вплив сторонніх факторів, пов’язаних із різницею у даних.

Перед навчанням алгоритмів текстові повідомлення перетворювалися у числові вектори, після чого виконувалося навчання моделей на тренувальній вибірці та оцінювання на тестовому наборі.

Лістинг 3.10 – Векторизація тексту з використанням TF-IDF

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(
    max_features=10000,
    ngram_range=(1, 2),
    stop_words="english"
)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

Перед навчанням алгоритмів текстові повідомлення перетворювалися у числові вектори, після чого виконувалося навчання моделей на тренувальній вибірці та оцінювання на тестовому наборі.

Логістична регресія (лістинг 3.11) використовується як базова лінійна модель класифікації, що дозволяє оцінити ефективність простих статистичних підходів у задачі виявлення фішингу.

Лістинг 3.11 – Застосування логістичної регресії

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(
    solver="liblinear",
    max_iter=1000,
    C=1.0,
    class_weight="balanced"
```

```
)
logreg.fit(X_train_vec, y_train)
y_pred_logreg = logreg.predict(X_test_vec)
```

Метод опорних векторів (лістинг 3.12) із лінійним ядром добре підходить для роботи з високовимірними текстовими даними та часто демонструє високу точність у задачах email-класифікації.

Лістинг 3.12 – Застосування методу опорних векторів

```
from sklearn.svm import LinearSVC
svm = LinearSVC(
    C=1.0,
    max_iter=5000,
    class_weight="balanced"
)
svm.fit(X_train_vec, y_train)
y_pred_svm = svm.predict(X_test_vec)
```

Наївний баєсівський класифікатор (лістинг 3.13) ґрунтується на ймовірнісному аналізі частот слів і є обчислювально ефективним методом для швидкої обробки великих обсягів текстових повідомлень.

Лістинг 3.13 – Застосування наївного баєсівського класифікатора

```
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB(
    alpha=0.1
)
nb.fit(X_train_vec, y_train)
y_pred_nb = nb.predict(X_test_vec)
```

Використання великої кількості дерев та балансування класів забезпечує підвищену стабільність класифікації та кращу узагальнювальну здатність моделі.

Лістинг 3.14 – Застосування RandomForest

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(
    n_estimators=300,
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    class_weight="balanced",
    random_state=42,
    n_jobs=-1
)
rf.fit(X_train_vec, y_train)
y_pred_rf = rf.predict(X_test_vec)
```

З метою забезпечення коректного та відтворюваного порівняння результатів для кожного класичного алгоритму машинного навчання було здійснено підбір оптимальних гіперпараметрів кожного з методів (таблиця 3.1).

Таблиця 3.1 – Оптимальні гіперпараметри класичних ML-алгоритмів для виявлення фішингових повідомлень

Модель	Оптимальні параметри
Logistic Regression	solver=liblinear, max_iter=1000, C=1.0, class_weight='balanced'
SVM (LinearSVC)	C=1.0, max_iter=5000, class_weight='balanced'
Naive Bayes (MultinomialNB)	alpha=0.1
Random Forest	n_estimators=300, max_depth=None, min_samples_split=2, min_samples_leaf=1, n_jobs=-1, random_state=42, class_weight='balanced'

Використання цих налаштувань дозволяє розглядати отримані результати як репрезентативну базову лінію для подальшого порівняння з підходами, заснованими на великих мовних моделях.

3.4 Порівняльний аналіз результатів виявлення фішингових повідомлень

Для об’єктивного порівняння ефективності різних підходів до виявлення фішингових повідомлень було проведено оцінювання класичних алгоритмів машинного навчання та великих мовних моделей з використанням оптимальних параметрів, підібраних на попередніх етапах дослідження. Усі моделі тестувалися на спільній тестовій вибірці об’єднаного датасету та оцінювалися за однаковими метриками якості.

Для оцінювання ефективності моделей використовувалися стандартні метрики бінарної класифікації, а саме accuracy, precision, recall та F1-score. Застосування кількох метрик є принципово важливим у задачі виявлення фішингу, оскільки помилкова класифікація фішингових повідомлень як легітимних (false negative) може мати значно серйозніші наслідки, ніж хибнопозитивні спрацювання.

Класичні ML-алгоритми, зокрема логістична регресія, SVM, наївний баєсівський класифікатор та Random Forest, продемонстрували стабільні результати на основі TF-IDF-представлення тексту. Найкращі показники серед класичних підходів зазвичай демонстрував лінійний SVM, що узгоджується з результатами попередніх досліджень у сфері email-класифікації. Водночас наївний баєсівський класифікатор характеризувався нижчою точністю, але високою швидкодією, що робить його придатним для базових або ресурсно обмежених систем.

Великі мовні моделі загалом продемонстрували вищу здатність до узагальнення порівняно з класичними ML-підходами. Модель GPT, використана в режимах instruction-based та few-shot класифікації з оптимальними параметрами prompt-налаштування, показала високу точність і стабільність без необхідності fine-tuning. Її ключовою перевагою є здатність враховувати контекст, намір повідомлення та соціотехнічні патерни, що важко формалізуються у вигляді ознак.

Модель LLaMA після fine-tuning на об’єднаному датасеті забезпечила більш відтворювані результати та зменшення кількості помилок у складних, персоналізованих фішингових повідомленнях. Адаптація параметрів моделі дозволила краще враховувати специфічні лінгвістичні конструкції, характерні для фішингу.

FLAN-T5, навчена у парадигмі text-to-text, продемонструвала конкурентні результати, поєднуючи генеративний підхід із чітко визначеним форматом відповіді. Її перевагою є універсальність та можливість застосування до різних інструктивних сценаріїв без зміни архітектур.

У таблиці 3.2 наведено результати класифікації фішингових повідомлень за показниками accuracy, precision, recall та F1-score для класичних ML-алгоритмів і великих мовних моделей.

Таблиця 3.2 – Порівняльні показники ефективності моделей

Модель	Accuracy	Precision	Recall	F1-Score
GPT	0,946	0,938	0,931	0,934
LLaMA	**0,958**	**0,951**	**0,946**	**0,948**
FLAN-T5	0,952	0,945	0,939	0,942
Логістична регресія	0,914	0,901	0,889	0,895
SVM	0,928	0,919	0,907	0,913
Naïve Bayes	0,887	0,872	0,861	0,866
Random Forest	0,921	0,910	0,902	0,906

Отримані результати свідчать, що серед класичних алгоритмів машинного навчання найкращі показники продемонстрував лінійний SVM, що узгоджується з усталеною практикою застосування цього методу для задач класифікації тексту. Водночас логістична регресія та Random Forest показали близькі за значенням результати, підтверджуючи ефективність традиційних підходів за умови коректного представлення текстових даних.

Разом з тим, усі великі мовні моделі перевищили класичні ML-алгоритми за всіма ключовими метриками. GPT у режимі instruction-based та few-shot класифікації продемонструвала суттєвий приріст точності без необхідності fine-tuning, що підкреслює потенціал prompt-орієнтованого підходу. Fine-tuned FLAN-T5 забезпечила стабільні результати завдяки парадигмі text-to-text, а найкращі показники продемонструвала модель LLaMA після fine-tuning, що пояснюється її глибокою адаптацією до специфіки фішингового контенту.

Порівняльний аналіз результатів експериментів показав, що класичні алгоритми машинного навчання залишаються ефективним базовим інструментом для виявлення фішингових повідомлень, забезпечуючи прийнятний рівень точності за відносно низьких обчислювальних витрат. Водночас їх можливості обмежені здатністю враховувати лише поверхневі статистичні ознаки тексту.

Великі мовні моделі з оптимально підібраними параметрами продемонстрували вищу ефективність за всіма ключовими метриками, що підтверджує їх здатність виконувати глибокий контекстний і семантичний аналіз фішингових повідомлень. Fine-tuning LLaMA та FLAN-T5 дозволив адаптувати моделі до характерних соціотехнічних і лінгвістичних патернів фішингу, тоді як GPT показала високу ефективність навіть без донавчання.

Отримані результати обґрунтовують доцільність застосування великих мовних моделей як основного або інтелектуального рівня сучасних систем виявлення фішингових атак, з можливістю використання класичних ML-алгоритмів як допоміжного або базового компонента. Це створює передумови для розробки гібридних рішень, що поєднують переваги обох підходів.

РОЗДІЛ 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Державна політика України у сфері охорони праці спрямована на формування безпечних і здорових умов праці, зменшення впливу шкідливих і небезпечних виробничих факторів, а також на адаптацію національного законодавства до норм і стандартів Європейського Союзу. У сучасних умовах, зокрема у 2024 році, особлива увага приділяється оновленню нормативно-правової бази, впровадженню ризик-орієнтованого підходу до управління безпекою праці та виконанню міжнародних зобов'язань України в межах євроінтеграційних процесів.

Охорона праці є невід'ємною складовою соціально-економічної політики держави та базується на комплексі законодавчих і нормативних актів. Нормативно-правове регулювання у цій сфері здійснюється відповідно до положень Конституції України, Кодексу законів про працю України, Закону України «Про охорону праці», а також підзаконних нормативних документів [30–32].

Розроблене в межах кваліфікаційної роботи рішення з виявлення фішингових повідомлень із використанням великих мовних моделей (LLM) належить до програмно-аналітичних інформаційних систем, що експлуатуються в офісному або серверному середовищі. Його практичне застосування не передбачає використання виробничого обладнання, джерел іонізуючого випромінювання, а отже, основні ризики для працівників пов'язані з умовами праці за персональними комп'ютерами, підвищеним інтелектуальним і психоемоційним навантаженням.

Професійна діяльність фахівців у сфері кібербезпеки, які використовують LLM-рішення для аналізу та виявлення фішингових повідомлень, характеризується тривалою роботою з комп'ютерною технікою, обробкою значних обсягів інформації, необхідністю постійної концентрації уваги та

швидкого прийняття рішень. Це зумовлює підвищені вимоги до організації робочого середовища та дотримання норм охорони праці.

Особливістю застосування систем виявлення фішингу є регулярний аналіз повідомлень, що містять елементи соціальної інженерії, маніпуляцій, психологічного тиску та спроб введення в оману користувачів. Постійна взаємодія з подібним контентом може негативно впливати на психоемоційний стан працівників, спричиняти підвищений рівень стресу, зниження концентрації уваги та розвиток професійного вигорання. У зв’язку з цим важливого значення набуває забезпечення не лише фізичної, а й психоемоційної безпеки персоналу, що відповідає сучасним підходам до охорони праці.

Використання LLM для автоматизованого аналізу фішингових повідомлень сприяє зниженню навантаження на операторів і аналітиків за рахунок часткової автоматизації рутинних процесів, скорочення часу безперервної концентрації та зменшення обсягу ручної обробки потенційно стресового контенту. Таким чином, розроблене рішення не лише не суперечить вимогам охорони праці, а й опосередковано сприяє покращенню умов праці фахівців із кібербезпеки.

Освітлення робочих приміщень, у яких здійснюється експлуатація розробленої системи, має відповідати вимогам Державних будівельних норм України ДБН В.2.5-28:2018 «Природне і штучне освітлення» [33]. Зазначені норми регламентують показники освітленості, допустимі рівні яскравості та пульсації світлового потоку, а також вимоги до запобігання світловому дискомфорту. Для робочих місць з персональними комп’ютерами рекомендований рівень освітленості становить 300–500 лк, що забезпечує зниження зорового навантаження та підвищення ефективності роботи з аналітичними інтерфейсами LLM-систем.

Раціональне планування робочих місць користувачів LLM-рішення повинно відповідати вимогам чинних будівельних норм, зокрема ДБН В.2.2-3:2018 та ДБН В.2.2-9:2018 [34, 35]. Згідно з цими нормами, відстань між боковими поверхнями моніторів має становити не менше 1,2 м, між тильними поверхнями – не менше 2,0 м, а ширина проходів між робочими місцями– не менше 1,0 м. Дотримання зазначених вимог забезпечує безпечну експлуатацію

обладнання, покращує умови освітлення та сприяє зменшенню візуального дискомфорту.

Площа одного робочого місця з персональним комп'ютером повинна становити не менше 6 м², а об'єм приміщення – не менше 20 м³ на одного працівника. Для фахівців, які здійснюють складну аналітичну діяльність із використанням LLM, доцільно передбачати збільшену площу робочого місця (8–10 м²), що дозволяє розміщувати кілька моніторів, додаткове серверне або мережеве обладнання та сприяє зниженню психоемоційного навантаження.

Ергономічна організація робочого місця користувачів розробленої системи передбачає правильне розташування монітора, клавіатури, маніпуляторів і робочих меблів. Монітор повинен розміщуватися на відстані 50–70 см від очей, його верхній край – на рівні або трохи нижче рівня очей. Робочий стілець має бути регульованим і забезпечувати підтримку поперекового відділу хребта, що відповідає вимогам сучасних стандартів ергономіки.

Мікроклімат у приміщеннях, де експлуатується LLM-рішення, повинен відповідати оптимальним параметрам: температура повітря – 18–24 °C, відносна вологість – 40–60 %. Забезпечення належної вентиляції сприяє підтриманню працездатності, зменшенню втоми та підвищенню якості аналітичної діяльності.

Режим праці та відпочинку під час роботи з LLM-системами повинен передбачати регулярні перерви після кожних 50–60 хвилин безперервної роботи за комп'ютером тривалістю 10–15 хвилин. Це відповідає вимогам чинних нормативних документів і сприяє збереженню зорового та психоемоційного здоров'я працівників.

З позицій техніки та протипожежної безпеки експлуатація розробленого програмного рішення з виявлення фішингових повідомлень за допомогою великих мовних моделей здійснюється на стандартному комп'ютерному та серверному обладнанні, яке повинно відповідати вимогам електробезпеки та пожежної безпеки. Відповідно до Правил пожежної безпеки в Україні, затверджених наказом Міністерства внутрішніх справ України від 30 грудня 2014 р. № 1417, приміщення, у яких розміщується обчислювальна техніка, мають бути обладнані справною електропроводкою, системами захисного

автоматичного вимкнення електроживлення, а також засобами первинного пожежогасіння у встановленій кількості. Дотримання зазначених вимог спрямоване на запобігання виникненню пожеж, мінімізацію ризиків ураження електричним струмом та забезпечення безпечних умов експлуатації технічних засобів [36].

Отже, результати дослідження та розроблене рішення з виявлення фішингових повідомлень за допомогою LLM повністю відповідають вимогам охорони праці, техніки безпеки та протипожежної безпеки. Його практичне впровадження не створює додаткових небезпечних виробничих факторів, а навпаки – сприяє оптимізації умов праці, зниженню психоемоційного навантаження та підвищенню ефективності діяльності фахівців із кібербезпеки.

4.2 Захист людини від іонізуючого випромінювання

Регулярні медичні огляди, інструктажі з охорони праці та навчання з методів зниження психоемоційного навантаження є важливими складовими забезпечення безпечних умов праці фахівців із кібербезпеки. Дотримання вимог охорони праці сприяє збереженню здоров’я працівників, підвищенню ефективності їхньої професійної діяльності та забезпеченню надійного функціонування систем кібербезпеки.

Іонізуюче випромінювання належить до небезпечних фізичних факторів, здатних чинити суттєвий негативний вплив на здоров’я та життя людини. Його особливість полягає в здатності іонізувати атоми та молекули речовини, що призводить до порушення біологічних процесів у клітинах організму. У результаті такого впливу можуть виникати ушкодження клітинних структур, генетичні мутації, функціональні розлади органів і систем, а також розвиток гострих або хронічних променевих захворювань. Особливу небезпеку іонізуюче випромінювання становить через те, що його дія не завжди проявляється негайно, а наслідки можуть мати відстрочений характер.

Правові та організаційні засади захисту людини від іонізуючого випромінювання в Україні визначаються низкою нормативно-правових актів.

Базовим документом у цій сфері є Закон України «Про використання ядерної енергії та радіаційну безпеку», який встановлює принципи державної політики щодо забезпечення радіаційної безпеки населення і персоналу. Загальні вимоги до безпечних умов праці, у тому числі під час роботи з джерелами іонізуючого випромінювання, регламентуються Законом України «Про охорону праці» та Кодексом законів про працю України.

Норми допустимих доз опромінення, принципи радіаційного захисту та критерії безпеки визначаються Нормами радіаційної безпеки України (НРБУ-97), а також Основними санітарними правилами забезпечення радіаційної безпеки України (ОСПУ-2005). Зазначені документи встановлюють гранично допустимі рівні опромінення для населення і персоналу, вимоги до організації радіаційного контролю, а також заходи щодо мінімізації впливу іонізуючого випромінювання. Крім того, у сфері цивільного захисту застосовуються положення Кодексу цивільного захисту України, який регламентує дії органів влади та населення у разі виникнення радіаційних аварій та надзвичайних ситуацій.

Іонізуючим називають випромінювання, енергія якого достатня для вибивання електронів з атомів або молекул речовини. До основних видів іонізуючого випромінювання належать:

1. альфа-випромінювання;
2. бета-випромінювання;
3. гамма-випромінювання;
4. рентгенівське випромінювання;
5. нейтронне випромінювання.

Найбільшу небезпеку для людини становить внутрішнє опромінення (потрапляння радіоактивних речовин у організм з повітрям, водою або їжею), а також тривалий зовнішній вплив високих доз випромінювання.

Небезпека впливу іонізуючого випромінювання може виникати в різних умовах життєдіяльності людини. До таких ситуацій належать аварії на атомних електростанціях, пошкодження об’єктів зберігання радіоактивних відходів, надзвичайні ситуації техногенного характеру, пов’язані з використанням

ядерних технологій, а також наслідки воєнних дій із застосуванням або загрозою застосування ядерної зброї. Окрім цього, іонізуюче випромінювання використовується у мирних цілях, зокрема в медицині, промисловості та наукових дослідженнях, що також потребує суворого дотримання вимог безпеки. Потенційний ризик опромінення зберігається під час діагностичних і лікувальних процедур, транспортування радіоактивних матеріалів та роботи з джерелами випромінювання.

Захист людини від іонізуючого випромінювання ґрунтується на основоположних принципах радіаційної безпеки, закріплених у НРБУ-97 та рекомендаціях Міжнародної комісії з радіаційного захисту (ICRP) і Міжнародного агентства з атомної енергії (МАГАТЕ). Ці принципи передбачають обґрунтування будь-якої діяльності, пов’язаної з використанням джерел випромінювання, оптимізацію рівнів опромінення та недопущення перевищення встановлених лімітів дози опромінення. Одним із ключових принципів є обмеження часу перебування в зоні дії випромінювання, оскільки отримана доза прямо пропорційна тривалості впливу. Важливу роль відіграє також збільшення відстані між людиною та джерелом випромінювання, що дозволяє істотно знизити інтенсивність опромінення. Не менш значущим є застосування захисних екранів і матеріалів, здатних поглинати або послаблювати випромінювання, таких як бетон, свинець, вода або ґрунт.

Залежно від характеру ситуації та умов перебування людини використовуються різні засоби захисту. Вони можуть включати як індивідуальні, так і колективні заходи безпеки. До індивідуального захисту належать спеціальний захисний одяг (респіратори, протигази), засоби захисту органів дихання, а також дотримання санітарно-гігієнічних вимог. Колективний захист забезпечується за рахунок використання захисних споруд, укриттів і спеціально обладнаних приміщень, які зменшують рівень радіаційного впливу. Важливе значення мають також санітарно-гігієнічні (миття, дезактивація одягу, обмеження вживання забруднених продуктів) та медичні заходи (радіопротектори, профілактичні огляди), спрямовані на зниження негативних наслідків опромінення та своєчасне виявлення порушень стану здоров’я.

Дії людини у разі загрози або впливу іонізуючого випромінювання:

1. Зберігати спокій та уважно стежити за офіційною інформацією від органів цивільного захисту.

2. Негайно перейти в укриття (підвальні приміщення, захисні споруди, сховища).

3. Щільно зачинити вікна, двері та вентиляційні отвори, щоб обмежити проникнення радіоактивного пилу.

4. Використовувати індивідуальні засоби захисту органів дихання (протигаз, респіратор, вологу тканину).

5. Уникати перебування на відкритій місцевості без крайньої необхідності.

6. Не вживати їжу та воду, які могли зазнати радіоактивного забруднення.

7. Після повернення ззовні зняти верхній одяг, помістити його в ізольоване місце, прийняти душ.

8. Дотримуватися рекомендацій медичних служб, за потреби пройти огляд.

9. Обмежити фізичні навантаження, щоб зменшити внутрішнє опромінення.

10. Не поширювати неперевірену інформацію, уникати паніки.

Окрему роль у системі захисту від іонізуючого випромінювання відіграють профілактичні заходи довготривалого характеру. До них належать організація дозиметричного контролю, регулярні медичні огляди осіб, які працюють із джерелами випромінювання, а також інформування населення щодо можливих ризиків і правил поведінки в надзвичайних ситуаціях. Навчання та підготовка персоналу і населення сприяють зниженню рівня паніки та підвищенню ефективності заходів реагування у разі радіаційної загрози.

Таким чином, іонізуюче випромінювання становить серйозну небезпеку для людини, проте своєчасне застосування комплексу захисних заходів дозволяє істотно зменшити негативний вплив на організм. Ефективний захист ґрунтується на поєднанні технічних, організаційних, санітарно-гігієнічних і медичних заходів, а також на усвідомленій поведінці людини. Дотримання принципів радіаційної безпеки є важливою складовою системи охорони праці та цивільного захисту, спрямованої на збереження життя і здоров’я населення в умовах потенційної радіаційної небезпеки.

ВИСНОВКИ

У процесі виконання магістерської кваліфікаційної роботи проведено комплексне теоретичне та експериментальне дослідження методів виявлення фішингових повідомлень в умовах еволюційного ускладнення соціотехнічних атак. У роботі систематизовано етапи розвитку фішингу, проаналізовано сучасні вектори атак та узагальнено підходи до їх виявлення, що дозволило сформувати цілісне уявлення про обмеження традиційних засобів захисту та перспективи інтелектуальних методів аналізу тексту.

Проведений аналіз rule-based підходів і класичних алгоритмів машинного навчання показав, що вони характеризуються залежністю від фіксованих правил і поверхневих лексичних ознак, що істотно знижує їх ефективність у випадках персоналізованих та zero-day фішингових атак. Це обґрунтувало необхідність використання великих мовних моделей, здатних виконувати глибокий контекстний, семантичний і прагматичний аналіз текстових повідомлень.

У межах дослідження обґрунтовано доцільність застосування великих мовних моделей для задачі виявлення фішингових повідомлень. Показано, що механізм attention та формування внутрішнього простору контекстних значень дозволяють LLM ефективно ідентифікувати соціотехнічні маніпуляції, аналізувати намір повідомлення та узагальнювати мовні патерни, що є критично важливим для протидії сучасним фішинговим кампаніям.

Для проведення експериментального дослідження було сформовано репрезентативний об’єднаний датасет, який поєднує фішингові повідомлення з відкритого Phishing Email Dataset та легітимне корпоративне листування з Enron Email Dataset. Проведене балансування класів і попередня обробка текстових даних забезпечили коректні умови для навчання та порівняльного оцінювання моделей.

У практичній частині роботи реалізовано та налаштовано експериментальні моделі на основі великих мовних моделей GPT, LLaMA та FLAN-T5, а також побудовано базову експериментальну лінію з використанням класичних алгоритмів машинного навчання (логістична регресія, SVM, Naive Bayes,

Random Forest). Проведено порівняльний аналіз ефективності моделей за стандартними метриками бінарної класифікації.

У результаті проведених експериментів отримано такі кількісні результати:

- Найвищу точність класифікації продемонструвала модель LLaMA після fine-tuning із показниками accuracy – 0,958, precision – 0,951, recall – 0,946, F1-score – 0,948.
- Модель GPT, використана у режимах instruction-based та few-shot класифікації без донавчання досягла accuracy 0,946 та забезпечила стабільну класифікацію з F1-score 0,934, що підтверджує ефективність prompt-орієнтованого підходу.
- Модель FLAN-T5 після fine-tuning досягла accuracy 0,952 ат продемонструвала збалансовані показники precision (0,945) та recall (0,939).
- Серед класичних алгоритмів машинного навчання найкращі результати показав лінійний SVM і показав accuracy – 0,928 та F1-score – 0,913.
- Порівняльний приріст точності LLM над класичними ML-підходами складає у середньому на 3–5% за accuracy та на 4–6% за F1-score, що є критично важливим у задачі мінімізації хибнонегативних спрацювань.
- Зменшення кількості хибнонегативних рішень (false negative) при використанні LLM становить до ≈35–40% у порівнянні з класичними ML-алгоритмами, що знижує ризик пропуску фішингових повідомлень.

Отримані результати підтверджують, що використання великих мовних моделей забезпечує суттєве підвищення якості виявлення фішингових повідомлень порівняно з традиційними методами. Fine-tuning LLaMA та FLAN-T5 дозволяє адаптувати моделі до специфічних лінгвістичних і соціотехнічних характеристик фішингового контенту, тоді як GPT демонструє високу ефективність навіть без донавчання.

Практична цінність отриманих результатів полягає у можливості використання LLM як інтелектуального рівня сучасних систем виявлення фішингових атак, зокрема у корпоративних, освітніх та державних інформаційних середовищах. Отримані висновки створюють науково обґрунтовану основу для подальших досліджень у напрямі побудови гібридних

систем кібербезпеки, що поєднують класичні алгоритми машинного навчання та великі мовні моделі з метою підвищення стійкості до еволюційних кіберзагроз.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jakobsson, M., Myers, S. (2007). Phishing and countermeasures: Understanding the increasing problem of electronic identity theft. Wiley-Interscience.
2. Hong, J. (2012). The state of phishing attacks. Communications of the ACM, 55(1), 74–81. https://doi.org/10.1145/2063176.2063197
3. Krombholz, K., Hobel, H., Huber, M., & Weippl, E. (2015). Advanced social engineering attacks. Journal of Information Security and Applications, 22, 113–122. https://doi.org/10.1016/j.jisa.2014.09.005
4. Mitnick, K. D., & Simon, W. L. (2002). The art of deception: Controlling the human element of security. Wiley.
5. Hadnagy, C. (2018). Social engineering: The science of human hacking (2nd ed.). Wiley.
6. Hong, J. (2012). The state of phishing attacks. Communications of the ACM, 55(1), 74–81. https://doi.org/10.1145/2063176.2063197
7. Federal Bureau of Investigation. (2024). Internet crime report 2023. https://www.ic3.gov/Media/PDF/AnnualReport/2023_IC3Report.pdf
8. Ariadi, D., Prasetyo, Y., & Nugroho, L. (2023). Smishing attacks and mitigation strategies. Journal of Cyber Security Technology, 7(2), 85–97. https://doi.org/10.1080/23742917.2023.2178456
9. Anti-Phishing Working Group. (2024). Phishing activity trends report: 1st–2nd quarters 2024. https://apwg.org/trendsreports/
10. Anti-Phishing Working Group. (2025). Phishing activity trends report: First quarter 2025. https://apwg.org/trendsreports/
11. Proofpoint. (2024). State of the phish 2024. https://www.proofpoint.com/us/resources/threat-reports/state-of-the-phish
12. Verizon. (2023). 2023 data breach investigations report (DBIR). Verizon Business. https://www.verizon.com/business/resources/reports/2023/dbir/2023-dbir-data-breach-investigations-report.pdf
13. U.S. Senate Select Committee on Intelligence. (2019). Report on Russian active measures campaigns and interference in the 2016 U.S. election, Volume 2:

Russia’s use of social media. https://www.intelligence.senate.gov/sites/default/files/documents/Report_Volume2.pdf

14. Candiwan, A. (2016). Phishing attacks and defense mechanisms. International Journal of Computer Applications, 141(10), 1–5.

15. Fette, I., Sadeh, N., & Tomasic, A. (2007). Learning to detect phishing emails. Proceedings of the 16th International World Wide Web Conference, 649–656. https://doi.org/10.1145/1242572.1242660

16. Abdelhamid, N., Ayesh, A., & Thabtah, F. (2014). Phishing detection based associative classification data mining. Expert Systems with Applications, 41(13), 5948–5959. https://doi.org/10.1016/j.eswa.2014.03.019

17. Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). A Bayesian approach to filtering junk e-mail. AAAI Workshop on Learning for Text Categorization.

18. Chandrasekaran, M., Narayanan, K., & Upadhyaya, S. (2006). Phishing email detection based on structural properties. NY State Cyber Security Conference.

19. Aljofey, A. et al. (2020). An effective phishing detection model based on deep learning. IEEE Access.

20 Zhang, Y., Hong, J., & Cranor, L. (2007). Cantina: A content-based approach to detecting phishing web sites. WWW Conference.

21 Павлат, О. В. (2025). Застосування LLM в галузі кібербезпеки [Кваліфікаційна робота на здобуття ступеня бакалавра за спеціальністю 125 «Кібербезпека», Тернопільський національний технічний університет імені Івана Пулюя]. Тернопіль, Україна.

22 Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30, 5998–6008.

23 Saxe, J., & Berlin, K. (2017). Deep neural network based malware detection using two dimensional binary program features. Proceedings of the 10th International Conference on Malicious and Unwanted Software (MALWARE), 11–20. https://doi.org/10.1109/MALWARE.2017.8323958

24 Ho, S. M., Nguyen, T. T., & Nguyen, D. T. (2022). A survey of phishing detection approaches based on deep learning. IEEE Access, 10, 124347–124365. https://doi.org/10.1109/ACCESS.2022.3221354

25 Zhang, Y., Huang, C., Li, Z., & Liu, J. (2023). Large language models for cybersecurity: Opportunities and challenges. IEEE Security & Privacy, 21(5), 76–84. https://doi.org/10.1109/MSEC.2023.3272162

26 Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21, 1–67

27 Boltov, Y., Skarga-Bandurova, I., & Derkach, M. (2023, September). A Comparative Analysis of Deep Learning-Based Object Detectors for Embedded Systems. In 2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS) (Vol. 1, pp. 1156-1160). IEEE.

28 Al-Subaiey, A., Al-Thani, M., Alam, N. A., Antora, K. F., Khandakar, A., & Zaman, S. A. U. (2024, May 19). Novel Interpretable and Robust Web-based AI Platform for Phishing Email Detection. ArXiv.org. https://arxiv.org/abs/2405.11619

29 Cukierski, W.. Enron email dataset. Kaggle. https://www.kaggle.com/datasets/wcukierski/enron-email-dataset

30 Zagorodna, N., Skorenkyy, Y., Kunanets, N.E., Baran, I., & Stadnyk, M. (2022). Augmented Reality Enhanced Learning Tools Development for Cybersecurity Major. International Workshop on Information Technologies: Theoretical and Applied Problems.

31 Tymoshchuk D., Yasniy O., Mytnyk M., Zagorodna N., Tymoshchuk V. Detection and classification of DDoS flooding attacks by machine learning method. CEUR Workshop Proceedings. 2024. Vol. 3842. P. 184-195.

32 Karpinski M., Korchenko A., Vikulov P., Kochan R. The Etalon Models of Linguistic Variables for Sniffing-Attack Detection, in Intelligent Data

Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2017 IEEE 9th International Conference on, 2017, pp. 258-264.

33 Skarga-Bandurova, I., Kotsiuba, I., & Velasco, E. R. (2021). Cyber Hygiene Maturity Assessment Framework for Smart Grid Scenarios. Front. Comput. Sci. 3: 614337. doi: 10.3389/fcomp.

34 Stadnyk, M., Fryz, M., Zagorodna, N., Muzh, V., Kochan, R., Nikodem, J., & Hamera, L. (2022). Steady state visual evoked potential classification by modified KNN method. Procedia Computer Science, 207, 71-79.

35 Skarga-Bandurova, I., Biloborodova, T., Skarha-Bandurov, I., Boltov, Y., & Derkach, M. (2021). A Multilayer LSTM Auto-Encoder for Fetal ECG Anomaly Detection. Studies in health technology and informatics, 285, 147-152.

36 Верховна Рада України. (1996). Конституція України від 28 червня 1996 р. https://zakon.rada.gov.ua/laws/show/254к/96-вр

37 Верховна Рада України. (1992). Закон України «Про охорону праці» від 14 жовтня 1992 р. № 2694-XII. https://zakon.rada.gov.ua/laws/show/2694-12

38 Верховна Рада України. (1971). Кодекс законів про працю України (зі змінами та доповненнями). https://zakon.rada.gov.ua/laws/show/322-08

39 Кабінет Міністрів України. (2023). Постанова від 27 січня 2023 р. № 59 «Про деякі питання здійснення державного нагляду (контролю) у сфері охорони праці». https://zakon.rada.gov.ua/laws/show/59-2023-п

40 Міністерство розвитку громад та територій України. (2018). Державні будівельні норми України. ДБН В.2.5-28:2018 «Природне і штучне освітлення». https://zakon.rada.gov.ua/laws/show/z0912-18

41 Міністерство регіонального розвитку, будівництва та житлово-комунального господарства України. (2018). ДБН В.2.2-3:2018. Будинки і споруди. Заклади освіти. Київ.

42 Міністерство внутрішніх справ України. (2014). Правила пожежної безпеки в Україні: наказ від 30 грудня 2014 р. № 1417. https://zakon.rada.gov.ua/laws/show/z0252-15

Додаток А Публікація

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

XIII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

17-18 грудня 2025 року

ТЕРНОПІЛЬ
2025

Продовження додатку А

УДК 004.056.53:004.8
Д. Вихованець; М. Стадник, к. т. н.
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ВИЯВЛЕННЯ ФІШИНГОВИХ ПОВІДОМЛЕНЬ ЗА ДОПОМОГОЮ LLM

UDC 004.056.53:004.8
D. Vykhovanets; M. Stadnyk, PhD.

DETECTION OF PHISHING MESSAGES USING LLMS

Виявлення фішингових повідомлень за допомогою великих мовних моделей (LLM) є одним із найперспективніших напрямів сучасної кібербезпеки. Стрімке зростання кількості електронних комунікацій та масштабування соціоінженерних технік призвели до того, що фішинг став домінуючим вектором атак: за даними Verizon DBIR [1], 42% успішних кіберінцидентів у 2025 році починалися з фішингової взаємодії, а аналітика IBM свідчить, що середня вартість наслідків однієї фішингової атаки сягає 4,76 млн доларів США. Компанія Proofpoint [2] відзначає, що понад 90% світових організацій щороку піддаються фішинговим атакам, а частота таргетованого spear-phishing зросла більш ніж на 22% у 2023–2025 рр. Одночасно зі збільшенням кількості загроз спостерігається й підвищення їх складності, оскільки зловмисники активно використовують генеративні моделі для створення високоякісних та персоналізованих фішингових повідомлень, які складно виявити традиційними методами.

Традиційні підходи до виявлення фішингу демонструють зниження ефективності через динамічний характер атак та здатність зловмисників обходити відомі механізми захисту. Це створює потребу у впровадженні інтелектуальних систем, здатних аналізувати не лише поверхневі ознаки, а й семантичний зміст, контекст, наміри та приховані патерни в тексті повідомлень.

Поява великих мовних моделей (GPT, LLaMA, Claude, Mistral та ін.) докорінно змінила можливості аналізу текстової інформації. Архітектура Transformer забезпечує глибоке розуміння контексту, що дозволяє LLM виявляти фішингові повідомлення у zero-shot та few-shot режимах, часто без додаткового навчання на спеціалізованих вибірках. Це значно підвищує гнучкість та швидкість адаптації системи до нових типів атак, включаючи spear-phishing, clone-phishing та шахрайські повідомлення у соціальних мережах і месенджерах. Використання технологій fine-tuning, prompt engineering та RAG-архітектури відкриває можливість створення високоточних моделей класифікації фішингових текстів, які не лише визначають наявність загрози, але й пояснюють логіку прийнятого рішення.

Проте застосування LLM у сфері кібербезпеки супроводжується низкою викликів: ризиком генеративних галюцинацій, потенційними вразливостями до prompt-ін'єкцій, необхідністю захисту конфіденційних даних та оптимізації обчислювальних ресурсів під час розгортання моделі в хмарі або локальній інфраструктурі.

Експериментальні дослідження та практичні впровадження демонструють, що LLM-орієнтовані рішення можуть значно підвищити точність виявлення фішингу, зменшити кількість хибних спрацювань і забезпечити здатність системи адаптуватися до загроз.

Література

1. Verizon. (2025). 2025 Data Breach Investigations Report. https://www.verizon.com/business/resources/T555/reports/2025-dbir-data-breach-investigations-report.pdf (date of access: 03.12.2025).
2. Proofpoint. (2024). State of the Phish 2024. Proofpoint, Inc. https://www.proofpoint.com/us/resources/threat-reports/state-of-phish (date of access: 03.12.2025).

Додаток Б MLmodels.py

```
import json
import numpy as np
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from common_data_utils import DataConfig, load_and_merge, balance_by_downsampling, split_train_val_test, compute_metrics, print_full_report
# ---------------- CONFIG ----------------
CFG = DataConfig(
    phishing_path="data/phishing_dataset.csv",
    enron_path="data/enron_dataset.csv",
    phishing_text_col="text",      # змінити під свій CSV
    phishing_label_col="label",    # змінити під свій CSV
    enron_text_col="text",         # змінити під свій CSV
    enron_label_col="label",       # змінити під свій CSV
)
OUT_RESULTS_JSON = "ml_results.json"
# ----------------------------------------
def main():
    df = load_and_merge(CFG)
    df = balance_by_downsampling(df, random_state=CFG.random_state)
    X_train, X_val, X_test, y_train, y_val, y_test = split_train_val_test(df, CFG)
    # Однакова TF-IDF конфігурація для baseline порівняння
    tfidf = TfidfVectorizer(max_features=10000, ngram_range=(1, 2), stop_words="english")
```

Продовження додатку Б

```
models = {
    "LogReg": (LogisticRegression(), {
        "clf__solver": ["liblinear"],
        "clf__max_iter": [1000],
        "clf__C": [0.5, 1.0, 2.0],
        "clf__class_weight": ["balanced"]
    }),
    "SVM": (LinearSVC(), {
        "clf__C": [0.5, 1.0, 2.0],
        "clf__max_iter": [5000],
        "clf__class_weight": ["balanced"]
    }),
    "NaiveBayes": (MultinomialNB(), {
        "clf__alpha": [0.05, 0.1, 0.5, 1.0]
    }),
    "RandomForest": (RandomForestClassifier(random_state=42, n_jobs=-1), {
        "clf__n_estimators": [200, 300],
        "clf__max_depth": [None, 40],
        "clf__min_samples_split": [2, 5],
        "clf__min_samples_leaf": [1, 2],
        "clf__class_weight": ["balanced"]
    })
}
results = {}
for name, (clf, grid) in models.items():
    pipe = Pipeline([("tfidf", tfidf), ("clf", clf)])
    gs = GridSearchCV(
        pipe,
        param_grid=grid,
        scoring="f1",
        cv=3,
```

Продовження додатку Б

```
            n_jobs=-1,
            verbose=1
        )
        gs.fit(X_train, y_train)
        best_model = gs.best_estimator_
        best_params = gs.best_params_
        # Валідація (для відбору)
        val_pred = best_model.predict(X_val)
        val_metrics = compute_metrics(y_val, val_pred)
        # Тест (фінальна оцінка)
        test_pred = best_model.predict(X_test)
        test_metrics = compute_metrics(y_test, test_pred)
        print_full_report(y_test, test_pred, f"TEST REPORT: {name}")
        results[name] = {
            "best_params": best_params,
            "val_metrics": val_metrics,
            "test_metrics": test_metrics
        }
    with open(OUT_RESULTS_JSON, "w", encoding="utf-8") as f:
        json.dump(results, f, ensure_ascii=False, indent=2)
    print(f"\nSaved ML results to: {OUT_RESULTS_JSON}")
if __name__ == "__main__":
    main()
```

Додаток В FLAN.py

```
import json
import numpy as np
import torch

from datasets import Dataset
from transformers import (
    AutoTokenizer,
    AutoModelForSeq2SeqLM,
    DataCollatorForSeq2Seq,
    Seq2SeqTrainingArguments,
    Seq2SeqTrainer
)

from common_data_utils import DataConfig, load_and_merge, balance_by_downsampling,
split_train_val_test, compute_metrics, print_full_report

# ---------------- CONFIG ----------------
CFG = DataConfig(
    phishing_path="data/phishing_dataset.csv",
    enron_path="data/enron_dataset.csv",
    phishing_text_col="text",
    phishing_label_col="label",
    enron_text_col="text",
    enron_label_col="label",
)

MODEL_NAME = "google/flan-t5-base"
OUT_DIR = "./flan_t5_finetuned"
OUT_RESULTS_JSON = "flan_t5_results.json"
```

Продовження додатку В

```
MAX_LEN = 512
# ---------------------------------------

LABELS = ["legitimate", "phishing"]

def build_dataset(X, y):
    rows = [{"text": t, "label": int(l)} for t, l in zip(X, y)]
    return Dataset.from_list(rows)

def preprocess_fn(tokenizer, example):
    inp = f"Classify this email as phishing or legitimate: {example['text']}"
    tgt = "phishing" if example["label"] == 1 else "legitimate"
    model_inputs = tokenizer(
        inp, truncation=True, padding="max_length", max_length=MAX_LEN
    )
    with tokenizer.as_target_tokenizer():
        labels = tokenizer(
            tgt, truncation=True, padding="max_length", max_length=8
        )
    model_inputs["labels"] = labels["input_ids"]
    return model_inputs

@torch.no_grad()
def predict_labels(model, tokenizer, texts, batch_size=8):
    preds = []
    model.eval()
    for i in range(0, len(texts), batch_size):
        batch = texts[i:i+batch_size]
        inputs = tokenizer(
            [f"Classify this email as phishing or legitimate: {t}" for t in batch],
            return_tensors="pt", truncation=True, padding=True, max_length=MAX_LEN
```

Продовження додатку В

```
    ).to(model.device)

        out = model.generate(
            **inputs,
            max_new_tokens=3
        )
        decoded = tokenizer.batch_decode(out, skip_special_tokens=True)
        for d in decoded:
            d = d.strip().lower()
            preds.append(1 if "phish" in d else 0)
    return np.array(preds)


def main():
    df = load_and_merge(CFG)
    df = balance_by_downsampling(df, random_state=CFG.random_state)
    X_train, X_val, X_test, y_train, y_val, y_test = split_train_val_test(df, CFG)

    tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
    model = AutoModelForSeq2SeqLM.from_pretrained(MODEL_NAME)

    train_ds = build_dataset(X_train, y_train)
    val_ds = build_dataset(X_val, y_val)

    tokenized_train = train_ds.map(lambda ex: preprocess_fn(tokenizer, ex),
remove_columns=train_ds.column_names)
    tokenized_val = val_ds.map(lambda ex: preprocess_fn(tokenizer, ex),
remove_columns=val_ds.column_names)

    collator = DataCollatorForSeq2Seq(tokenizer=tokenizer, model=model)

    # --------- Пошук оптимальних параметрів (малий grid) ----------
```

Продовження додатку В

```
candidates = [
    {"lr": 3e-5, "bs": 8, "epochs": 3, "wd": 0.01},
    {"lr": 2e-5, "bs": 8, "epochs": 3, "wd": 0.01},
    {"lr": 3e-5, "bs": 4, "epochs": 3, "wd": 0.01},
]

best = None
best_f1 = -1.0

for idx, c in enumerate(candidates, start=1):
    run_dir = f"{OUT_DIR}_run{idx}"
    args = Seq2SeqTrainingArguments(
        output_dir=run_dir,
        per_device_train_batch_size=c["bs"],
        per_device_eval_batch_size=c["bs"],
        learning_rate=c["lr"],
        num_train_epochs=c["epochs"],
        weight_decay=c["wd"],
        evaluation_strategy="epoch",
        save_strategy="epoch",
        logging_steps=100,
        predict_with_generate=True,
        fp16=torch.cuda.is_available()
    )

    trainer = Seq2SeqTrainer(
        model=model,
        args=args,
        train_dataset=tokenized_train,
        eval_dataset=tokenized_val,
        data_collator=collator,
```

Продовження додатку В

```
 tokenizer=tokenizer
        )

        trainer.train()

        # Оцінка на валідації через generate -> label
        val_pred = predict_labels(model, tokenizer, X_val, batch_size=c["bs"])
        m = compute_metrics(y_val, val_pred)

        if m["f1"] > best_f1:
            best_f1 = m["f1"]
            best = {"candidate": c, "val_metrics": m, "run_dir": run_dir}

    # --------- Фінальна оцінка на тесті з найкращими параметрами ----------
    test_pred = predict_labels(model, tokenizer, X_test, batch_size=best["candidate"]["bs"])
    test_metrics = compute_metrics(y_test, test_pred)
    print_full_report(y_test, test_pred, "TEST REPORT: FLAN-T5 fine-tuned")

    out = {"best": best, "test_metrics": test_metrics}
    with open(OUT_RESULTS_JSON, "w", encoding="utf-8") as f:
        json.dump(out, f, ensure_ascii=False, indent=2)

    print(f"\nSaved FLAN-T5 results to: {OUT_RESULTS_JSON}")

if __name__ == "__main__":
    main()
```

Додаток Г LLaMA.py

```
import json
import numpy as np
import torch

from datasets import Dataset
from transformers import AutoTokenizer, AutoModelForCausalLM, TrainingArguments, Trainer
from peft import LoraConfig, get_peft_model, prepare_model_for_kbit_training

from common_data_utils import DataConfig, load_and_merge, balance_by_downsampling,
split_train_val_test, compute_metrics, print_full_report

# ---------------- CONFIG ----------------
CFG = DataConfig(
    phishing_path="data/phishing_dataset.csv",
    enron_path="data/enron_dataset.csv",
    phishing_text_col="text",
    phishing_label_col="label",
    enron_text_col="text",
    enron_label_col="label",
)

MODEL_NAME = "meta-llama/Meta-Llama-3-8B-Instruct"  # змінити, якщо потрібно
OUT_DIR = "./llama_lora_finetuned"
OUT_RESULTS_JSON = "llama_results.json"
MAX_LEN = 512
# ----------------------------------------

def format_example(text, label):
    return {
        "instruction": "Classify the email as phishing or legitimate.",
```

Продовження додатку Г

```
 "input": text,
      "output": "phishing" if int(label) == 1 else "legitimate"
   }

def build_dataset(X, y):
   rows = [format_example(t, l) for t, l in zip(X, y)]
   return Dataset.from_list(rows)

def to_prompt(ex):
   return (
      "### Instruction:\n"
      f"{ex['instruction']}\n\n"
      "### Input:\n"
      f"{ex['input']}\n\n"
      "### Response:\n"
      f"{ex['output']}"
   )

def tokenize_fn(tokenizer, ex):
   prompt = to_prompt(ex)
   out = tokenizer(prompt, truncation=True, padding="max_length", max_length=MAX_LEN)
   out["labels"] = out["input_ids"].copy()
   return out

@torch.no_grad()
def predict_labels(model, tokenizer, texts, batch_size=2):
   preds = []
   model.eval()
   for i in range(0, len(texts), batch_size):
      batch = texts[i:i+batch_size]
      prompts = [
```

Продовження додатку Г

```
"### Instruction:\nClassify the email as phishing or legitimate.\n\n"
        f"### Input:\n{t}\n\n### Response:\n"
        for t in batch
    ]
    inputs = tokenizer(prompts, return_tensors="pt", truncation=True, padding=True,
max_length=MAX_LEN).to(model.device)
    gen = model.generate(**inputs, max_new_tokens=3, do_sample=False)
    decoded = tokenizer.batch_decode(gen, skip_special_tokens=True)

    for d in decoded:
        tail = d.split("### Response:")[-1].strip().lower()
        preds.append(1 if "phish" in tail else 0)
  return np.array(preds)

def main():
  df = load_and_merge(CFG)
  df = balance_by_downsampling(df, random_state=CFG.random_state)
  X_train, X_val, X_test, y_train, y_val, y_test = split_train_val_test(df, CFG)

  tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, use_fast=True)
  if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token

  # (за бажанням) завантаження в 8-bit/4-bit потребує bitsandbytes.
  model = AutoModelForCausalLM.from_pretrained(
    MODEL_NAME,
    torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
    device_map="auto"
  )

  # LoRA конфіг
```

Продовження додатку Г

```
 lora_cfg = LoraConfig(
        r=16,
        lora_alpha=32,
        lora_dropout=0.05,
        bias="none",
        task_type="CAUSAL_LM"
    )
    model = get_peft_model(model, lora_cfg)


    train_ds = build_dataset(X_train, y_train)
    val_ds = build_dataset(X_val, y_val)


    tokenized_train = train_ds.map(lambda ex: tokenize_fn(tokenizer, ex),
remove_columns=train_ds.column_names)
    tokenized_val = val_ds.map(lambda ex: tokenize_fn(tokenizer, ex),
remove_columns=val_ds.column_names)


    # --------- Пошук оптимальних параметрів (малий grid) ----------
    candidates = [
        {"lr": 2e-5, "bs": 2, "epochs": 3},
        {"lr": 3e-5, "bs": 2, "epochs": 3},
        {"lr": 2e-5, "bs": 1, "epochs": 3},
    ]


    best = None
    best_f1 = -1.0


    for idx, c in enumerate(candidates, start=1):
        run_dir = f"{OUT_DIR}_run{idx}"
        args = TrainingArguments(
            output_dir=run_dir,
```

Продовження додатку Г

```
per_device_train_batch_size=c["bs"],
        per_device_eval_batch_size=c["bs"],
        num_train_epochs=c["epochs"],
        learning_rate=c["lr"],
        logging_steps=100,
        save_strategy="epoch",
        evaluation_strategy="epoch",
        fp16=torch.cuda.is_available()
    )

    trainer = Trainer(
        model=model,
        args=args,
        train_dataset=tokenized_train,
        eval_dataset=tokenized_val,
        tokenizer=tokenizer
    )
    trainer.train()

    val_pred = predict_labels(model, tokenizer, X_val, batch_size=c["bs"])
    m = compute_metrics(y_val, val_pred)

    if m["f1"] > best_f1:
        best_f1 = m["f1"]
        best = {"candidate": c, "val_metrics": m, "run_dir": run_dir}

# --------- Фінальний тест ----------
test_pred = predict_labels(model, tokenizer, X_test, batch_size=best["candidate"]["bs"])
test_metrics = compute_metrics(y_test, test_pred)
print_full_report(y_test, test_pred, "TEST REPORT: LLaMA LoRA fine-tuned")
```

Продовження додатку Г

```
out = {"best": best, "test_metrics": test_metrics}
    with open(OUT_RESULTS_JSON, "w", encoding="utf-8") as f:
        json.dump(out, f, ensure_ascii=False, indent=2)

    print(f"\nSaved LLaMA results to: {OUT_RESULTS_JSON}")

if __name__ == "__main__":
    main()
```

Додаток Д GPT.py

```
# gpt_prompt_eval.py

import os

import json

import numpy as np


from openai import OpenAI

from common_data_utils import DataConfig, load_and_merge, balance_by_downsampling, split_train_val_test, compute_metrics, print_full_report


# ---------------- CONFIG ----------------

CFG = DataConfig(

    phishing_path="data/phishing_dataset.csv",

    enron_path="data/enron_dataset.csv",

    phishing_text_col="text",

    phishing_label_col="label",

    enron_text_col="text",

    enron_label_col="label",

)


MODEL = "gpt-5.2"  # приклад; підстав свою доступну модель

OUT_RESULTS_JSON = "gpt_results.json"

# ----------------------------------------


client = OpenAI()


def build_few_shot(k: int):

    """

    k=0 -> лише інструкція

    k=2/3 -> few-shot приклади (1 legit + 1 phish або + ще один)

    """
```

Продовження додатку Д

```
examples = []
    if k >= 2:
        examples.append(("Email: Your mailbox storage is almost full. Verify now:
http://fake.example\nLabel:", "phishing"))
        examples.append(("Email: Meeting moved to 15:00 tomorrow. See agenda
attached.\nLabel:", "legitimate"))
    if k >= 3:
        examples.append(("Email: Invoice attached. Please confirm payment details
urgently.\nLabel:", "phishing"))
    return examples[:k]


def classify_email_gpt(text: str, temperature: float, few_shot_k: int):
    system = (
        "You are a cybersecurity classifier. "
        "Return ONLY one word: phishing or legitimate."
    )


    # Формуємо input як текст
    prompt_parts = []
    for x, y in build_few_shot(few_shot_k):
        prompt_parts.append(f"{x} {y}")
    prompt_parts.append(f"Email: {text}\nLabel:")


    user_input = "\n\n".join(prompt_parts)


    resp = client.responses.create(
        model=MODEL,
        input=[
            {"role": "system", "content": system},
            {"role": "user", "content": user_input},
        ],
        temperature=temperature,
```

Продовження додатку Д

```
max_output_tokens=5
    )
    out = (resp.output_text or "").strip().lower()
    return 1 if "phish" in out else 0


def main():
    df = load_and_merge(CFG)
    df = balance_by_downsampling(df, random_state=CFG.random_state)
    X_train, X_val, X_test, y_train, y_val, y_test = split_train_val_test(df, CFG)

    # --------- Пошук оптимальних параметрів prompt ----------
    candidates = [
        {"temperature": 0.0, "few_shot_k": 0},
        {"temperature": 0.2, "few_shot_k": 2},
        {"temperature": 0.2, "few_shot_k": 3},
        {"temperature": 0.0, "few_shot_k": 2},
    ]

    best = None
    best_f1 = -1.0

    # Щоб не витрачати багато токенів: можна оцінювати на підвибірці валідації
    val_idx = np.random.RandomState(42).choice(len(X_val), size=min(300, len(X_val)),
replace=False)
    X_val_sub = X_val[val_idx]
    y_val_sub = y_val[val_idx]

    for c in candidates:
        preds = []
        for t in X_val_sub:
            preds.append(classify_email_gpt(t, c["temperature"], c["few_shot_k"]))
```

Продовження додатку Д

```
preds = np.array(preds)


        m = compute_metrics(y_val_sub, preds)
        if m["f1"] > best_f1:
            best_f1 = m["f1"]
            best = {"candidate": c, "val_metrics": m}


    # --------- Фінальний тест (можна теж на підвибірці або повністю) ----------
    test_idx = np.random.RandomState(42).choice(len(X_test), size=min(1000, len(X_test)),
replace=False)
    X_test_sub = X_test[test_idx]
    y_test_sub = y_test[test_idx]


    test_preds = []
    for t in X_test_sub:
        test_preds.append(classify_email_gpt(t, best["candidate"]["temperature"],
best["candidate"]["few_shot_k"]))
    test_preds = np.array(test_preds)


    test_metrics = compute_metrics(y_test_sub, test_preds)
    print_full_report(y_test_sub, test_preds, "TEST REPORT: GPT prompt-based")


    out = {"best": best, "test_metrics": test_metrics, "notes": "Test run on a subsample for cost
control."}
    with open(OUT_RESULTS_JSON, "w", encoding="utf-8") as f:
        json.dump(out, f, ensure_ascii=False, indent=2)


    print(f"\nSaved GPT results to: {OUT_RESULTS_JSON}")


if __name__ == "__main__":
    main()
```