ВСТУП

У наш час світ переживає справжню інформаційну революцію: потік даних щодня зростає експоненціально, а разом із ним — і обсяги небажаних повідомлень, відомих під загальною назвою «спам». Спам проникає буквально в усі сфери цифрового спілкування: електронну пошту, SMS-повідомлення, месенджери, соціальні мережі, форуми та блоги. Масштаби цього явища вражають: за різними оцінками, більше половини всіх електронних листів у корпоративному секторі є спамом [1], а у мобільних мережах користувачі щодня отримують десятки небажаних SMS та чат-повідомлень [2].

Актуальність обраної теми дипломної роботи зумовлена стрімким зростанням обсягів спаму, який становить значну загрозу як для окремих користувачів, так і для організацій, призводячи до фінансових втрат, витоку конфіденційної інформації та зниження продуктивності. Існуючі методи фільтрації часто виявляються недостатньо ефективними проти сучасних, постійно еволюціонуючих тактик спамерів, що вимагає розробки та впровадження більш інтелектуальних та адаптивних систем захисту, особливо для українськомовного контенту, де специфічні мовні конструкції та культурний контекст можуть ускладнювати детекцію.

Небезпека спаму виходить далеко за межі простої «надокучливої реклами». Фішингові атаки, шахрайські схеми, розповсюдження шкідливого програмного забезпечення, обман довіри користувачів з фінансовою метою — усе це різновиди спаму, що завдають значних втрат як окремим людям, так і цілим організаціям. За оцінками аналітиків, через спам бізнес витрачає сотні мільйонів доларів щороку на інфраструктуру, підтримку та відновлення втраченої продуктивності працівників [3]. Кожен користувач електронної пошти може втратити до 15 хвилин щоденно лише на сортування корисних повідомлень від небажаних, а у випадку зараження шкідливим ПЗ — зазнати серйозних збитків і репутаційних втрат.

Раціональні правила фільтрації (ключові слова, чорні списки IP-адрес чи доменів) та статистичні методи (байєсівський спам-фільтр) довго й успішно використовувалися для боротьби з небажаними розсилками [4]. Проте сучасні спамерські тактики постійно еволюціонують: обфускація URL, «спінінг» тексту, мультимовність, гібридні атаки з елементами соціального інжинірингу вимагають від систем детекції гнучкіших, навчальних підходів. Саме тому останніми роками дедалі більше уваги приділяють методам машинного навчання та глибинного навчання, зокрема трансформерам на кшталт BERT, які довели свою ефективність у широкому спектрі задач обробки природної мови [5].

Об'єктом дослідження є процес автоматизованого виявлення спаму в текстових повідомленнях українською мовою.

Предметом дослідження є методи, моделі та алгоритми обробки природної мови на основі архітектури BERT, а також процеси збору, підготовки та розмітки даних для навчання системи детекції спаму, адаптованої до специфіки україномовного текстового контенту.

Метою дипломної роботи є розробка та дослідження інформаційної системи автоматизованого виявлення спаму в україномовних текстових повідомленнях на основі донавченої моделі BERT, що забезпечує високу точність класифікації та можливість інтеграції в різні комунікаційні платформи.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз предметної області: визначити класи спаму, основні канали поширення та загрози; оцінити існуючі методи фільтрації, їхні переваги та обмеження;

– здійснити збір та калібрування корпусу даних: створити репрезентативний корпус україномовних текстових повідомлень (близько 40 000), виконати первинну розмітку та ручну корекцію позначок;

– розробити методи препроцесингу: запровадити процедури очищення й нормалізації тексту, що включають обробку URL, електронних адрес, номерів телефонів, юзернеймів, хештегів і символів валют для підготовки даних до

токенізації;

– провести донавчання моделі BERT (bert-base-multilingual-cased) для задачі класифікації спаму: виконати експерименти з підбором гіперпараметрів та оцінити модель за ключовими метриками (accuracy, precision, recall, F1-score);

– виконати калібрування та валідацію моделі: провести оцінку на відкладеній тестовій вибірці, проаналізувати випадки неправильних класифікацій для виявлення слабких місць та шляхів подальшого поліпшення;

– розробити демонстраційний програмний інтерфейс (наприклад, Telegram-бот) для ілюстрації роботи системи класифікації повідомлень;

Для вирішення поставлених завдань у роботі використано комплекс наукових методів, зокрема: методи системного аналізу для дослідження предметної області та існуючих рішень; методи теорії ймовірностей та математичної статистики для обробки даних та оцінки результатів; методи машинного навчання, зокрема архітектури на основі трансформерів (BERT) для побудови класифікаційної моделі; методи обробки природної мови (NLP) для препроцесингу текстових даних; експериментальні методи для перевірки ефективності розробленої системи та підбору оптимальних параметрів моделі.

Практична значущість роботи полягає в тому, що створена система може бути розгорнута в реальних інформаційних сервісах для автоматичного фільтрування небажаного контенту в українськомовних чатах, системах обміну повідомленнями, коментарях на веб-ресурсах тощо. Розроблений підхід до формування спеціалізованого корпусу даних, що включає переклад та генерацію текстів, а також запропоновані методи препроцесингу, можуть бути адаптовані для вирішення аналогічних задач в інших доменах або для інших мов. Результати роботи можуть бути корисними для розробників програмного забезпечення, спеціалістів з інформаційної безпеки та дослідників у галузі обробки природної мови.

1 АНАЛІЗ ПРОБЛЕМИ ВИЯВЛЕННЯ СПАМУ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Поняття спаму та його вплив

У сучасному цифровому середовищі спам є однією з найпоширеніших загроз інформаційній безпеці та комфортному користуванню комунікаційними платформами. Термін «спам» походить від знаменитого скетчу британського комедійного гурту Monty Python, де слово «spam» повторюється настільки часто, що втрачає будь-який сенс. У контексті інформаційних технологій під спамом розуміють небажані масові повідомлення, які надсилаються без попередньої згоди отримувача та мають переважно рекламний, шахрайський або деструктивний характер.

Спам вирізняється кількома ключовими ознаками:

1. Неперсоналізованість – повідомлення формуються за однаковим шаблоном і розсилаються великій кількості адресатів без урахування їхніх інтересів чи потреб.

2. Нав’язливість – масове повторне надходження створює дискомфорт і відволікає користувачів від основних завдань.

3. Комерційно-шахрайська мета. Найчастіше спам містить рекламу товарів і послуг, фішингові посилання, запити на перерахування коштів або пропозиції преміальних послуг із прихованими платежами.

За змістом та цілями розрізняють кілька типів спаму. Рекламний спам орієнтований на просування товарів і послуг (знижки, курси, акції). Фішинговий спам має на меті викрадення персональних даних через підроблені повідомлення від імені банків чи популярних сервісів. Шкідливий спам розсилає вкладення з вірусами або троянами. Окремо виокремлюють соціальний інжиніринг, коли зловмисники маніпулюють емоціями, наприклад, видаючи себе за знайомих із проханням про термінову фінансову допомогу. Також у чатах і коментарях

зустрічається провокаційний контент – образливі чи сексуально забарвлені повідомлення, що створюють конфліктну або нездорову атмосферу й можуть розцінюватися як спам.

Наслідки поширення спаму виходять далеко за межі простого «загромадження» поштових скриньок чи чат-вікон. Соціально-економічні та безпекові наслідки включають:

– втрату продуктивності – працівники витрачають час на фільтрацію корисних повідомлень від небажаних – в середньому до 15 хвилин щодня;

– підвищене навантаження на інфраструктуру – масові розсилки спаму створюють додаткові витрати на зберігання та обробку даних, збільшення пропускної здатності мереж і підтримку серверів;

– ризики інформаційної безпеки – фішингові атаки та шкідливі вкладення можуть призвести до витоку паролів, втрати конфіденційної інформації, інфікування пристроїв вірусами та шкідливими програмами;

– репутаційні збитки – організації, чия контактна база була піддана розсилці спаму (наприклад, унаслідок витоку даних), втрачають довіру клієнтів і партнерів.

Розуміння природи спаму – від його етимології та ознак до різноманіття форм і наслідків – становить основу для створення ефективних систем захисту.

1.2 Нормативно-правова база (захист даних, фільтрація контенту)

При розробці системи виявлення спаму в текстових повідомленнях важливою складовою є дотримання чинної нормативно-правової бази, яка регулює обробку персональних даних, передачу електронних повідомлень, автоматичну фільтрацію контенту та використання інтелектуальних алгоритмів.

Перш за все, слід звернути увагу на Закон України "Про захист персональних даних" [6], який визначає правові засади обробки персональної інформації фізичних осіб. Згідно із законом, обробка персональних даних

повинна здійснюватися виключно на підставі згоди суб’єкта персональних даних або інших законних підстав. У системі виявлення спаму така згода може бути реалізована через політику конфіденційності [7], яку користувач приймає під час реєстрації.

Крім національного законодавства, варто враховувати норми міжнародного права, зокрема Загальний регламент захисту даних (GDPR), що діє на території Європейського Союзу. GDPR містить суворі вимоги до прозорості алгоритмів, права на видалення, обмеження обробки та передачі персональних даних. Застосування даної системи для користувачів із країн ЄС зобов’язує розробника дотримуватись зазначених положень.

Щодо фільтрації спаму, нормативна база включає також Закон України "Про електронні комунікації". Він встановлює, що будь-які масові розсилки (SMS, електронна пошта, повідомлення в месенджерах) повинні здійснюватися лише за згодою адресата. Таким чином, система фільтрації повинна точно розпізнавати небажані повідомлення, не допускаючи блокування легітимного контенту.

Також слід згадати рекомендації Міжнародного союзу електрозв’язку (ITU) стосовно боротьби зі спамом. ITU підкреслює необхідність впровадження інтелектуальних механізмів фільтрації з урахуванням мовних, культурних та етичних аспектів. У випадку мультимовної моделі, як-от Multilingual BERT, необхідно додатково тестувати точність виявлення спаму в різних мовах, щоб уникнути упередженості.

У контексті використання штучного інтелекту, дедалі більше значення мають етичні принципи: прозорість прийняття рішень, можливість аудиту алгоритмів, відповідальність за автоматизовані рішення. Розробники повинні забезпечити можливість пояснення результатів класифікації, особливо у випадках, коли повідомлення помилково визначене як спам.

Отже, нормативно-правове регулювання вимагає дотримання ряду принципів: збереження приватності, точності класифікації, прозорості роботи

алгоритму та відповідності законодавчим вимогам. Система виявлення спаму повинна бути не лише технічно ефективною, але й юридично безпечною у контексті обробки текстових повідомлень.

1.3 Канали поширення спаму

Спам може надходити різними шляхами, кожен із яких має свої особливості та вразливості. Вибір каналу визначається масштабом аудиторії, технічними можливостями спамера та бажаним ефектом. Нижче наведені чотири основні канали розповсюдження небажаних повідомлень:

1. Електронна пошта (e-mail). На сьогодні електронна пошта залишається найпоширенішим середовищем для спаму. Спамерські розсилки використовують величезні масиви адрес, інколи отримані зламаними базами даних або автоматичним скануванням веб-сторінок. Основні ризики:

– фішингові атаки, у яких листи маскуються під повідомлення від банків чи знайомих сервісів;

– розсилки з вкладеними файлами (документи з макросами, архіви), що містять шкідливий код;

– завантаження обсягу поштових скриньок й навантаження сервера, що підвищує витрати на інфраструктуру організації.

2. SMS/MMS та преміальні номери. Мобільні спам-повідомлення передаються безпосередньо на телефон користувача. Окрім рекламних SMS, поширені схеми з преміальними короткими номерами, за дзвінок або повідомлення на які списуються значні кошти. Типові прояви:

– рекламні розсилки з пропозиціями сервісів та підписок;

– шахрайські sms із проханням зателефонувати на платний номер або відправити код підтвердження;

– повідомлення із заманливими акціями, що супроводжуються прихованим стягненням грошей зі сторони оператора.

3. Месенджери та соціальні мережі. Платформи типу Telegram, WhatsApp, Viber, Facebook Messenger активно використовуються для бот-розсилок. У групах, каналах або особистих чатах спамери:

– надсилають рекламні чи фішингові посилання під виглядом «офіційного» повідомлення;

– створюють «роздавальні» акаунти, які запрошують приєднатися до закритих спільнот із інвестиційними схемами або купівлею курсів;

– поширюють провокаційний контент сексуального характеру, який поєднується з запитами про надсилання фотографій або оплати «секретних методик».

4. Коментарі на веб-ресурсах і форумах. Автоматизовані боти залишають непрошені повідомлення під статтями, оголошеннями та в розділах із відгуками:

– посилання на фішингові сайти або сторінки з шкідливим змістом;

– рекламні звернення в коментарях до блогів і новинних ресурсів, що заважає читачам отримувати релевантну інформацію;

– спам-форум, де розміщуються сотні «тем» зі схожими рекламними заголовками, що призводить до зниження якості обговорень і зменшення активності реальних користувачів.

Розуміння специфіки кожного каналу допомагає обирати адекватні методи фільтрації та боротьби зі спамом: від налаштувань поштових серверів і SMS-шлюзів до впровадження машинного навчання в месенджерах та системах модерації веб-коментарів.

1.4 Ключові інформативні ознаки системи детекції спаму

У контексті даної дипломної роботи досліджуваним інформаційним процесом є процес обміну текстовими повідомленнями в цифрових комунікаційних середовищах (месенджери, електронна пошта, веб-форми, коментарі на онлайн-платформах), який зазнає негативного впливу від

поширення спаму. Об'єктом дослідження виступає розроблювана інформаційна система – програмний модуль для автоматизованого виявлення спаму (spam_detector_module), призначений для інтеграції в різноманітні платформи з метою захисту користувачів та інформаційних ресурсів від небажаного контенту. Цей модуль, по суті, є спеціалізованим автоматизованим засобом обробки текстової інформації для прийняття рішення про її належність до категорії "спам" або "не-спам".

Для забезпечення ефективності такої системи та її переваг над існуючими аналогами, в рамках даної роботи акцент робиться на розробці та вдосконаленні наступних ключових інформативних ознак (параметрів):

1. Глибинне розуміння контексту українськомовного тексту. На відміну від традиційних фільтрів, що часто покладаються на наявність/відсутність ключових слів, розроблювана система використовує трансформерну модель BERT (bert-base-multilingual-cased). Це дозволяє аналізувати повідомлення не як набір окремих слів, а як цілісну семантичну структуру, враховуючи взаємозв'язки між словами та їхнє значення в конкретному контексті. Така ознака є критично важливою для виявлення складних форм спаму, де шкідливий намір може бути замаскований або виражений неявно, а також для коректної обробки специфіки української мови, її ідіом та сленгу.

2. Спеціалізований препроцесинг з урахуванням локальних особливостей. Розроблено кастомізований алгоритм попередньої обробки тексту, який включає заміну URL-адрес на токени виду [URL:домен], електронних листів на [EMAIL], телефонних номерів на [PHONE], специфічних для українського інтернет-простору юзернеймів та хештегів на [USERNAME] та [HASHTAG], а також символів валют (включно з гривнею ₴) на [CURRENCY]. Ці спеціальні токени стають частиною словника моделі, дозволяючи їй враховувати не лише факт наявності посилання чи контакту, але й потенційно аналізувати тип домену або структуру електронної адреси як ознаку. Цей підхід покращує точність ідентифікації потенційно небезпечних елементів повідомлення.

3. Адаптація до українськомовного корпусу шляхом цілеспрямованого збору та аугментації даних. Якість роботи моделі машинного навчання напряму залежить від навчальних даних. У даній роботі особлива увага приділяється формуванню збалансованого та репрезентативного корпусу українськомовних текстів, що включає не тільки збір реальних прикладів спаму та легітимних повідомлень, але й їх ручну верифікацію, а також цільовий переклад іншомовного спаму та генерацію додаткових прикладів за допомогою ChatGPT API. Це дозволяє "навчити" модель на специфічних патернах українського спаму та уникнути упередженості, властивої загальним багатомовним моделям, не адаптованим до локального контексту.

4. Мінімалістична та модульна архітектура кінцевого продукту. Кінцевим результатом є компактний програмний модуль (spam_detector_module), який інкапсулює всю логіку препроцесингу та інференсу моделі. Такий підхід забезпечує простоту інтеграції розробленого рішення в будь-які існуючі інформаційні системи (веб-сервіси, месенджери, поштові клієнти) без необхідності розгортання складної інфраструктури, що є важливою інформативною ознакою з точки зору практичного застосування.

Ці інформативні ознаки в комплексі спрямовані на створення системи детекції спаму, яка не лише досягає високої точності, але й адаптована до специфіки української мови та сучасних тактик розповсюдження небажаного контенту.

1.5 Огляд традиційних методів фільтрації спаму

Незважаючи на стрімкий розвиток машинного навчання та глибинних нейронних мереж, класичні методи фільтрації спаму й досі залишаються основою багатьох систем захисту електронної пошти та повідомлень. У цьому підрозділі розглянемо основні підходи, їхні переваги й обмеження.

1. Правила на основі ключових слів і регулярних виразів. Цей метод

передбачає створення набору фільтрів, які шукають у тексті повідомлення певні слова, фрази або патерни. Прості шаблони (наприклад, “купіть”, “безкоштовно”, “гарантія”) пропускають чи блокують листи за ключовими словами. Регулярні вирази дають змогу відловлювати складніші структури, наприклад URL, номери телефонів або варіанти написання фішингових доменів. Переваги: легка реалізація, висока швидкість обробки, не потребує великих обчислювальних ресурсів. Обмеження: висока кількість хибнопозитивних або хибнонегативних спрацьовувань при мінімальних змінах у формулюваннях спаму; важкість підтримки та оновлення великого числа правил.

2. Чорні та білі списки доменів та IP-адрес. Фільтрація на рівні мережевої інфраструктури виконується за допомогою заборони або дозволу трафіку з певних IP-адрес або доменів. Чорний список містить адреси, з яких часто надходить спам, листи від них автоматично відхиляються. Білий список містить перевірені довірені джерела, листи від яких завжди проходять незалежно від іншого вмісту. Переваги: достатньо ефективний проти відомих спамерів; мінімізує навантаження на фільтри вищого рівня. Обмеження: спамери часто змінюють IP-адреси або використовують ботнети; ризик блокування легітимних серверів, якщо списки не оновлюються.

3. Байєсівська фільтрація. Метод статистичної класифікації, заснований на ймовірнісних моделях: для кожного слова розраховується ймовірність належності повідомлення до категорії “спам” або “не спам”. Етапи: збір корпусу позначених повідомлень; підрахунок частот появи токенів у спамі й у “хемі”; застосування формули Байєса для обчислення загальної ймовірності. Переваги: адаптивність – модель “вчиться” на нових даних; відносна простота імплементації. Обмеження: потребує достатньо великих і збалансованих навчальних вибірок; схильність до помилок у разі невдалого поповнення корпусу чи нерівномірності даних.

4. Проблема надчутливості (false-positive) та ухилення від фільтрів. Класичні системи часто жертвують часткою корисних повідомлень, щоб

зменшити кількість пропущеного спаму, або навпаки — пропускають частину спаму заради збереження легітимних листів. False-positive (помилкове блокування) призводить до втрати важливої інформації та критичного зниження довіри користувачів. False-negative (невиявлення спаму) допускає проникнення шкідливого чи шахрайського контенту. Спамери активно обходять фільтри, використовуючи обфускацію ("k@z1n0" замість "казино"), поділ ключових слів пробілами чи спеціальними символами, вставляння випадкових символів. Баланс між чутливістю та точністю фільтрації залишається ключовим викликом для розробників класичних рішень.

Усі перераховані методи мають свої сильні та слабкі сторони. Вони часто комбінуються в єдину систему багатоетапної фільтрації: спочатку блокуються очевидні відправники за чорними списками, потім застосовуються правила за ключовими словами і регулярними виразами, а на завершення коригується рішення статистичними або машинно-навчальними класифікаторами. Такий підхід покращує загальну точність, але зберігає обмеження кожного з окремих методів.

1.6 Сучасні підходи до виявлення спаму

Сучасні системи фільтрації спаму дедалі частіше спираються на методи машинного навчання, що дозволяють автоматично відокремлювати легітимні повідомлення від небажаних за допомогою статистичних залежностей і нелінійних моделей.

У класичних машинних алгоритмах, таких як логістична регресія або метод опорних векторів (SVM), текст переводиться у векторне представлення (наприклад, TF-IDF або n-грамні характеристики), після чого модель навчається відрізняти спам від «хему». Зазвичай цей підхід включає такі етапи:

– вибір ознак: частоти слів, позиційні індикатори, кількість посилань чи доменів;

– тренування моделі на великому наборі позначених прикладів;

– оптимізацію гіперпараметрів (регуляризація, вибір ядра для SVM, баланс класів).

Переваги цих методів – швидкість навчання і відносна прозорість рішення. Недоліки – потреба ручного підбору ознак та обмежена здатність моделі уловлювати контекстні й послідовні залежності в тексті.

Більш потужним кроком стали нейронні мережі на основі RNN (LSTM, GRU) або CNN, які працюють безпосередньо з послідовністю токенів. Вони здатні автоматично вивчати складні патерни:

– RNN-моделі обробляють текст як послідовність і запам’ятовують попередній контекст, що допомагає розрізняти багатозначні вирази;

– CNN-моделі застосовують згорткові фільтри до векторів слів, виявляючи локальні шаблони (наприклад, комбінації слів, що характерні для спаму).

Нейронні підходи часто досягають вищої точності, але потребують більшого обсягу даних і обчислювальних ресурсів, а також чутливі до вибору архітектури та довжини вхідних послідовностей

У трансформерних моделях (як-от BERT, RoBERTa, XLM-RoBERTa) використовується механізм самоуваги, що дає змогу моделі одночасно враховувати всі позиції в тексті. Переваги:

– утворення контекстуально залежних векторів токенів;

– можливість донавчання (fine-tuning) на невеликих корпусах спеціалізованих даних;

– висока стійкість до обфускації та спіну.

Типова схема включає попереднє донавчання на великому «масивному» корпусі, а потім — тонке настроювання на завданні класифікації спаму. Такі моделі демонструють найкращі результати, часто перевищуючи 98 % за метрикою F1, але за рахунок значних обчислювальних затрат і складності розгортання.

Таблиця 1.1 – Порівняльна таблиця методів виявлення спаму

Підхід	Представлення тексту	Облік контексту	Точність	Ресурси	Складність впровадження	Гнучкість до мов
Класичні ML-методи (LR, SVM)	TF-IDF, n-грам	Обмежений	Середня	Малі	Низька	Низька
Нейронні мережі (RNN, CNN)	Токени/вектори слів	Середній (через пам'ять RNN)	Висока	Середні	Середня	Обмежена
Трансформери (BERT, RoBERTa)	Контекстуальні вектори (self-attention)	Повний (вся послідовність одночасно)	Найвища (до 98% F1)	Високі	Висока	Висока (мультимовні моделі)

Порівняння продуктивності та точності в сучасних дослідженнях [8, с. 25-30; 9] показує, що:

– класичні ML-методи працюють швидше при наявних обмежених ресурсах, але поступаються в гнучкості та точності;

– нейронні мережі на RNN/CNN забезпечують кращу якість порівняно з LR/SVM за умови достатнього набору даних;

– трансформери виявляються найефективнішими в умовах змінного та багатомовного тексту, проте потребують оптимізації для продуктивного застосування в реальних системах.

З огляду на ці властивості, у нашому проекті було обрано донавчання моделі BERT (bert-base-multilingual-cased) як оптимальний баланс між точністю, можливістю роботи з українською мовою та доступністю для розгортання.

1.7 Постановка задачі дослідження

Аналіз сучасного стану проблеми виявлення спаму, особливостей його поширення та існуючих методів боротьби показує, що, незважаючи на значний прогрес, залишається актуальною потреба в розробці ефективних та адаптивних

систем, здатних протистояти новим тактикам спамерів, особливо в контексті українськомовного контенту. Традиційні методи, засновані на правилах та простих статистичних моделях, демонструють обмежену ефективність проти обфускованого та мінливого спаму. Сучасні підходи на основі машинного навчання, зокрема трансформерні архітектури типу BERT, показують високу точність, але їх застосування вимагає формування якісних навчальних корпусів та розробки спеціалізованих методів препроцесингу, адаптованих до мовних особливостей.

З огляду на вищезазначене, задачею даного дипломного дослідження є розробка та дослідження інформаційної системи для автоматизованого виявлення спаму в українськомовних текстових повідомленнях. Система повинна базуватися на донавченій моделі BERT (bert-base-multilingual-cased) і включати такі ключові аспекти:

1. Формування та підготовка спеціалізованого корпусу даних. Створення збалансованого корпусу текстових повідомлень (спам/не-спам) обсягом близько 40 000 прикладів, що включає збір даних з різних джерел, їх часткову ручну верифікацію, а також переклад та генерацію прикладів для підвищення якості та репрезентативності даних.

2. Розробка ефективного алгоритму препроцесингу тексту. Створення пайплайну попередньої обробки текстів, що враховує специфіку української мови та характерні ознаки спам-повідомлень, включаючи нормалізацію URL, електронних адрес, телефонних номерів, спеціальних символів та інших елементів, з метою їх перетворення на уніфіковані токени для моделі.

3. Донавчання та оптимізація моделі класифікації. Тонке налаштування попередньо навченої моделі bert-base-multilingual-cased на підготовленому українськомовному корпусі для задачі бінарної класифікації текстів (спам/не-спам). Дослідження впливу гіперпараметрів навчання на якість класифікації та досягнення високих показників точності (F1-score > 0.98).

4. Реалізація програмного модуля для детекції спаму. Створення

автономного програмного модуля, який інкапсулює розроблену логіку препроцесингу та навчену модель, і здатний приймати на вхід сирий текст та повертати мітку класифікації (спам/не-спам) з відповідною ймовірністю.

5. Експериментальна перевірка та оцінка ефективності розробленої системи. Проведення всебічного тестування розробленого модуля на відкладеній вибірці та, за можливості, на нових даних, аналіз результатів та порівняння з існуючими підходами.

При розробці системи виявлення спаму визначимо наступні вимоги та виклики проєкту:

– спам постійно еволюціонує: з’являються нові техніки обфускації URL, «спінінг» контенту, мультимовні повідомлення, що ускладнює його виявлення простими правилами;

– потрібен баланс між високою точністю (щоб мінімізувати false-positive) і продуктивністю (щоб система встигала обробляти великі обсяги повідомлень у реальному часі);

– необхідно врахувати різноманітність каналів поширення (e-mail, SMS, месенджери, веб-коментарі) та форматів контенту (текст, посилання, емодзі тощо);

– важливо забезпечити прозорість і можливість калібрування: корпус даних повинен регулярно уточнюватися і виправлятися для підтримки якості моделі;

– система має бути простою в інтеграції та розгортанні, щоб її можна було швидко підключити до існуючих сервісів (наприклад, через API або в контейнерах Docker).

Кінцевим результатом дослідження має стати програмний модуль, готовий до інтеграції в різні інформаційні системи для фільтрації небажаного контенту, та рекомендації щодо його подальшого вдосконалення.

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА КОМПОНЕНТІВ СИСТЕМИ АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ СПАМУ

У цьому розділі детально описано етапи проєктування та розробки ключових компонентів інформаційної системи для автоматизованого виявлення спаму в українськомовних текстових повідомленнях. Розглядається загальна архітектура системи, обґрунтовується вибір програмно-апаратних засобів, описуються процеси формування навчального корпусу, розробки структури бази даних, створення алгоритмів препроцесингу текстових даних, а також процес розробки та навчання класифікаційної моделі на основі архітектури BERT.

2.1 Загальна архітектура системи

Розроблювана система автоматизованого виявлення спаму складається з набору незалежних скриптів та модулів, які взаємодіють через єдину базу даних, що забезпечує гнучкість та масштабованість рішення. У фінальній базі даних, призначеній для навчання моделі, містяться тільки дві ключові колонки:

1. messages — текст повідомлення (сирий, частково перекладений українською або згенерований).

2. label — цільова мітка ("spam" (1) або "ham" (0)), визначена на основі джерела походження та результатів ручної верифікації.

Усі етапи життєвого циклу розробки системи, включаючи збір даних, їх переклад та аугментацію, очищення тексту, навчання й тестування моделі, реалізовані як окремі програмні скрипти, що взаємодіють із цією централізованою базою даних. Такий модульний підхід дозволяє гнучко модифікувати або доповнювати будь-який компонент системи без значного впливу на інші її частини.

На рисунку 2.1 графічно представлено структуру проєкту.

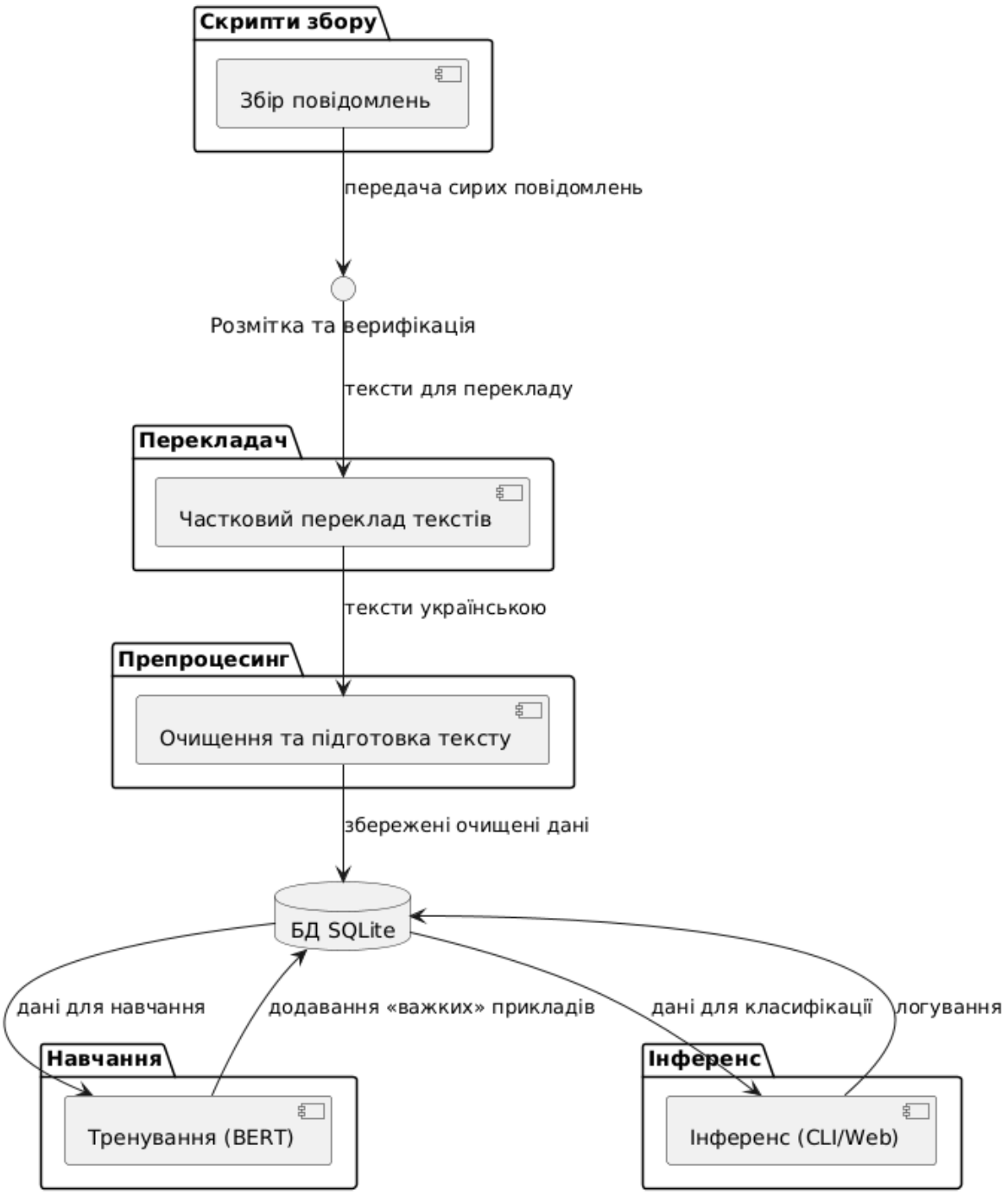


Рисунок 2.1 – Структурна схема проєкту

Основні компоненти архітектури, що детально описані у наступних підрозділах, включають:

– скрипти збору повідомлень: відповідають за автоматизований збір текстових даних з різноманітних джерел.

– модуль попередньої розмітки та верифікації: забезпечує початкове присвоєння міток та інструменти для їх ручної корекції оператором.

– модулі перекладу та генерації повідомлень: використовуються для збагачення та балансування навчального корпусу, зокрема шляхом перекладу іншомовного спаму та генерації синтетичних прикладів.

– модуль препроцесингу текстів: реалізує алгоритми очищення та нормалізації текстових даних перед їх подачею на вхід моделі.

– тренувальний модуль моделі BERT: включає скрипти для завантаження даних, їх токенізації, навчання класифікаційної моделі BERT та її валідації.

– модуль інференсу (кінцевий продукт): spam_detector_module, що інкапсулює навчену модель та логіку препроцесингу для класифікації нових повідомлень.

– інструменти тестування та демонстрації: включають CLI-утиліти та демонстраційний Telegram-бот для перевірки роботи системи.

2.2 Обґрунтування вибору програмних та апаратних засобів

Вибір програмних та апаратних засобів є важливим етапом проєктування, що визначає ефективність розробки, продуктивність та можливості подальшого масштабування системи.

Основною мовою програмування для всіх компонентів системи обрано Python (версія 3.12.3). Цей вибір зумовлений її перевагами: висока читабельність коду, велика кількість готових бібліотек для наукових обчислень, обробки природної мови (NLP) та машинного навчання, а також активна спільнота розробників.

Для роботи з трансформерними моделями, зокрема BERT, використано бібліотеку Hugging Face Transformers. Вона надає зручний API для завантаження попередньо навчених моделей, їх тонкого налаштування (fine-tuning) та інференсу.

Як фреймворк для глибинного навчання було обрано PyTorch, який є однією з провідних платформ для побудови та тренування нейронних мереж і

тісно інтегрований з Hugging Face Transformers.

Для маніпуляцій з даними, їх аналізу та підготовки до подачі в модель використовується бібліотека Pandas, що дозволяє ефективно працювати з табличними даними.

Обчислення метрик якості класифікації (accuracy, precision, recall, F1-score) здійснюється за допомогою функцій з бібліотеки Scikit-learn.

Для зберігання навчального корпусу повідомлень та їх міток обрано SQLite. Це легка, файлова, вбудована реляційна СУБД, яка не потребує окремого серверного процесу. Такий вибір є оптимальним для даного проєкту, оскільки забезпечує простоту розгортання, легкість доступу до даних для різних скриптів та можливість ручної верифікації даних за допомогою стандартних інструментів (наприклад, DB Browser for SQLite) без необхідності налаштування складних серверних рішень.

Розробка велася з використанням інтегрованого середовища розробки (IDE), такого як JetBrains PyCharm, що забезпечує зручні інструменти для написання коду, дебагінгу та управління проєктом.

Усі етапи розробки, включаючи збір та обробку даних, а також навчання та тестування моделі BERT, виконувалися на персональному ноутбуці з наступними характеристиками: процесор Intel Core Ultra 7 155H, дискретна відеокарта NVIDIA GeForce RTX 4050 Laptop GPU з 6 ГБ відеопам'яті GDDR6 та 16 ГБ оперативної пам'яті DDR5 з частотою 5600 МГц. Наявність графічного процесора (GPU) з підтримкою технології CUDA дозволила суттєво прискорити процес навчання моделі, який є ресурсоємним. Для інших завдань, таких як розробка скриптів, препроцесинг даних та робота з базою даних, вказаної конфігурації було більш ніж достатньо для комфортної та ефективної роботи.

Для демонстрації роботи кінцевого модуля spam_detector_module було вирішено розробити клієнтський застосунок у вигляді Telegram-бота, використовуючи Telegram Bot API. Це дозволяє наочно продемонструвати функціональність системи в інтерактивному режимі.

Обраний стек технологій забезпечує необхідну гнучкість для дослідження, розробки та тестування системи автоматизованого виявлення спаму, а також створює передумови для її потенційного впровадження в реальні інформаційні системи.

2.3 Збір і калібрування корпусу повідомлень

У цьому підрозділі описується формування навчального корпусу текстових повідомлень для моделі. Розглядаються джерела даних, автоматична початкова розмітка, ручна верифікація, а також процес генерації та перекладу додаткових прикладів. В результаті формується збалансований корпус близько 40 000 записів із чіткими мітками “spam” (1) та “ham” (0).

Джерела даних:

1. Повідомлення зі «спам-джерела». У якості умовного спам-джерела використовувалися канали та сервіси, призначені для зберігання підозрілих рекламних і шахрайських повідомлень іншими сервісами для виявлення спаму. Усі записи із цього джерела вважалися спамом за замовчуванням. З кожного опублікованого повідомлення виділялися тільки ті частини тексту, які містили рекламно-шахрайський контент; технічні метадані, коментарі та додаткові елементи виключалися. Автоматично присвоєна мітка “1” (spam) базувалася виключно на походженні тексту із зазначеного ресурсу. Багато записів спочатку були англомовними або іншою іноземною мовою. Для адаптації корпусу до українського контексту частину цих текстів перекладали на українську за допомогою ChatGPT API (моделлю "gpt-4.1-mini"). Переклад виконувався зі збереженням структури повідомлення, стилю оригінального тексту, сленгових чи галузевих виразів [10].

2. Повідомлення з “громадських” (неспамових) джерел. Для корпусу ham використовувалися публічні тексти із університетських груп, тематичних форумів, політичних спільнот і різних інформаційних порталів. На етапі

початкового збору всі такі повідомлення автоматично позначалися як “0” (ham), оскільки походили з ресурсів, орієнтованих на обмін думками або інформування, а не на рекламу. Після імпорту до бази оператор вручну переглядав кожен запис і, у разі виявлення прихованих рекламних фрагментів, фішингових посилань чи іншого небажаного контенту, змінював мітку на “1”. Для забезпечення достатньої кількості прикладів ham деяку частину дружніх текстів формували вручну або отримували за допомогою ChatGPT API із завданням, наприклад “Напиши дружнє повідомлення без жодної комерційної згадки”.

Процес автоматичної початкової розмітки. Усі зібрані повідомлення зберігались (або переносились) в єдину таблицю бази даних SQLite із двома стовпцями:

1. messages (TEXT) – текст повідомлення, ще не очищений, оскільки очищення відбуватиметься на етапі передачі навчальних даних тренувальній моделі.

2. label (INTEGER) – початкова мітка: 1 для повідомлень зі спам-джерела, 0 для повідомлень із громадських ресурсів.

Такий підхід суттєво прискорював збір даних, оскільки не вимагав негайної ручної перевірки кожного запису: оператор міг сконцентруватися на верифікації потенційно “сумнівних” рядків, а не на первинному розподілі класів.

Наступним етапом є ручна перевірка й корекція міток. Після автоматичної розмітки оператор відкривав базу даних у будь-якому SQLite-редакторі (наприклад, DB Browser for SQLite). Кожен запис у таблиці подавався у вигляді двох колонок: messages (текст) та label (поточна мітка). Оператор переглядав кожне повідомлення без зайвих метаданих, і, якщо в тексті із громадського джерела виявлявся рекламний фрагмент або фішингове посилання, змінював label із 0 на 1. Аналогічно, якщо зі спам-джерела випадково потрапляв не рекламний, а звичайний текст (наприклад, політична новина без жодних ознак спаму), оператор призначав мітку 0. Ніяких спеціальних скриптів для перевірки не використовувалося: достатньо було відкрити таблицю в БД та змінити

значення label безпосередньо у відповідному полі. На рисунку 2.2 показано процес перевірки повідомлень на коректність попередньої розмітки, робота ведеться в програмі DB Browser for SQLite.

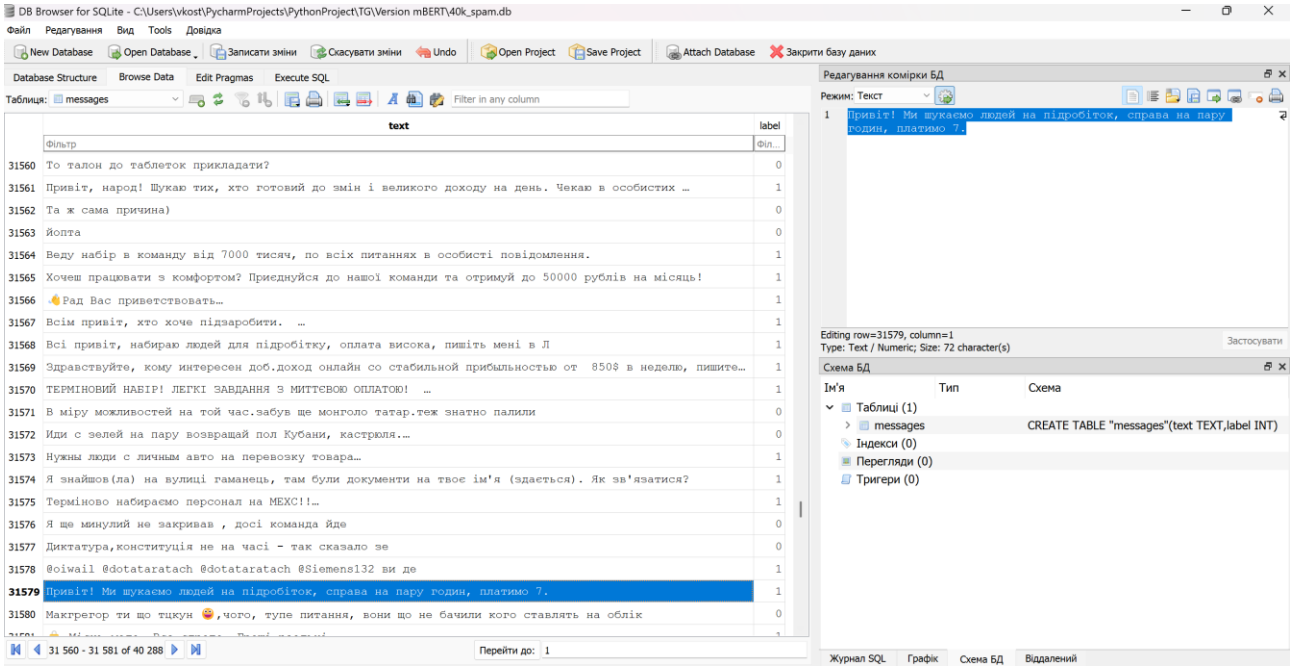


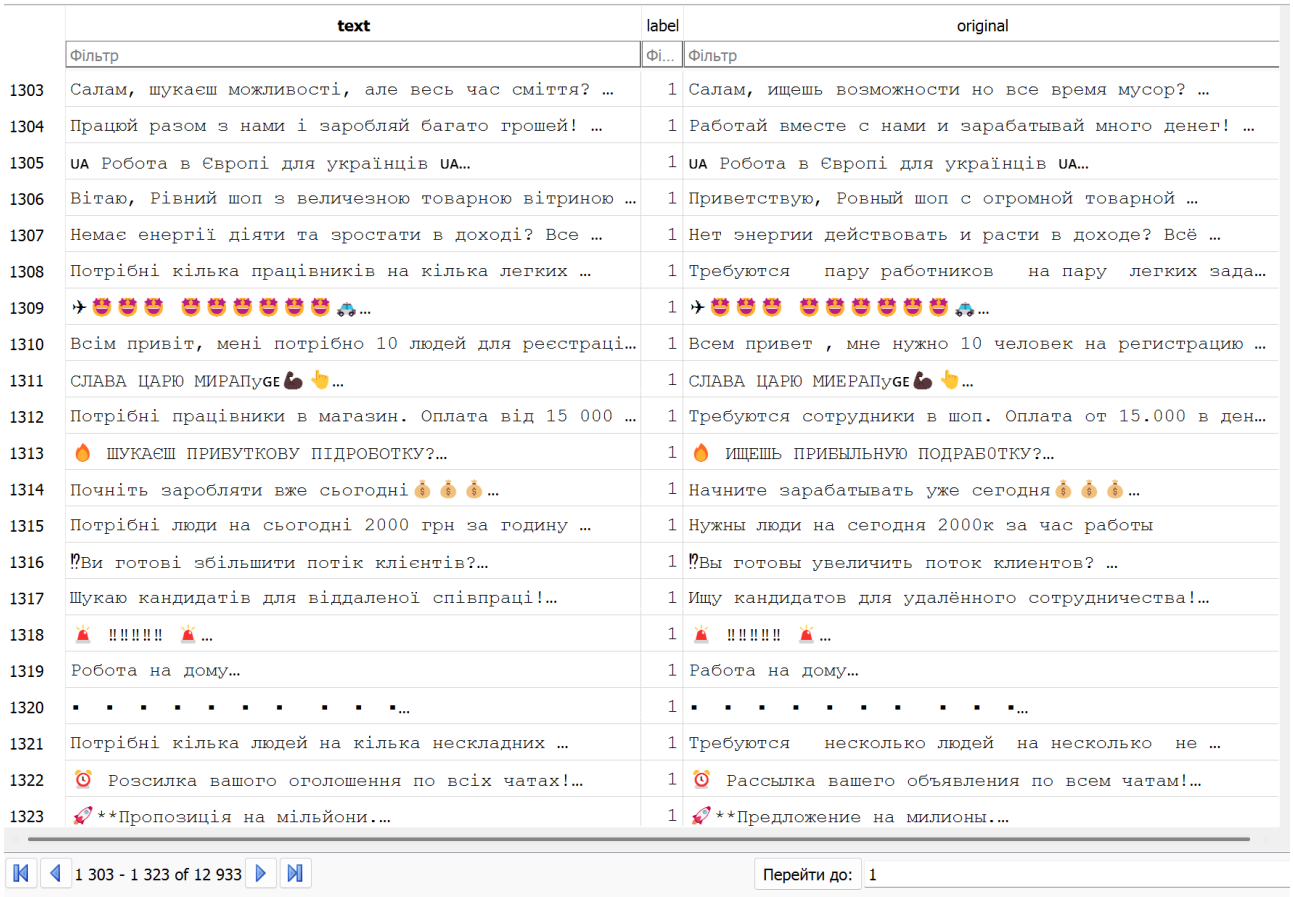
Рисунок 2.2 – Приклад вікна SQLite-редактора для верифікації даних

Наступний крок покращення якості БД, та разом з нею вихідної моделі, є генерація та переклад додаткових прикладів через ChatGPT API. Після попередньої ручної верифікації корпусу інколи виявлялися «прогалини» в навчальних даних, які проявлялися під час тестування моделі. Для «заточування» моделі на ці випадки використовувалися два окремих підходи:

1. Генерація нових прикладів (спам/ham) для виправлення виявлених слабких місць. Після аналізу помилок моделі формувалися конкретні запити до ChatGPT API для генерації текстів, що імітують проблемні для моделі категорії (наприклад, "Напиши рекламне повідомлення з натяком на безкоштовний приз..."). Кожен згенерований або перекладений зразок одразу додавався до БД з відповідною міткою і перевірявся оператором на реалістичність.

2. Переклад іноземномовних фрагментів спаму на українську. З джерела спам-повідомлень бралася лише частина тексту, що містила рекламно-шахрайський контент. Ці фрагменти передавалися на переклад через ChatGPT

API із запитом: «Переклади цей рекламно-шахрайський текст на українську, зберігаючи структуру, тональність і сленгові вирази». Отриманий український фрагмент після стандартного препроцесингу зберігався в таблиці БД з міткою label = 1.



	text	label	original
	Фільтр	Фі...	Фільтр
1303	Салам, шукаєш можливості, але весь час сміття? …	1	Салам, ищешь возможности но все время мусор? …
1304	Працюй разом з нами і заробляй багато грошей! …	1	Работай вместе с нами и зарабатывай много денег! …
1305	UA Робота в Європі для українців UA…	1	UA Робота в Європі для українців UA…
1306	Вітаю, Рівний шоп з величезною товарною вітриною …	1	Приветствую, Ровный шоп с огромной товарной …
1307	Немає енергії діяти та зростати в доході? Все …	1	Нет энергии действовать и расти в доходе? Всё …
1308	Потрібні кілька працівників на кілька легких …	1	Требуются пару работников на пару легких зада…
1309	✈️😍😍😍 😍😍😍😍😍😍😍🚜…	1	✈️😍😍😍 😍😍😍😍😍😍🚜…
1310	Всім привіт, мені потрібно 10 людей для реєстраці…	1	Всем привет , мне нужно 10 человек на регистрацию …
1311	СЛАВА ЦАРЮ МИРАПуGE🫰👆…	1	СЛАВА ЦАРЮ МИЕРАПуGE🫰👆…
1312	Потрібні працівники в магазин. Оплата від 15 000 …	1	Требуются сотрудники в шоп. Оплата от 15.000 в ден…
1313	🔥 ШУКАЄШ ПРИБУТКОВУ ПІДРОБОТКУ?…	1	🔥 ИЩЕШЬ ПРИБЫЛЬНУЮ ПОДРАБ0ТКУ?…
1314	Почніть заробляти вже сьогодні💰💰💰…	1	Начните зарабатывать уже сегодня💰💰💰…
1315	Потрібні люди на сьогодні 2000 грн за годину …	1	Нужны люди на сегодня 2000к за час работы
1316	⁉️Ви готові збільшити потік клієнтів?…	1	⁉️Вы готовы увеличить поток клиентов? …
1317	Шукаю кандидатів для віддаленої співпраці!…	1	Ищу кандидатов для удалённого сотрудничества!…
1318	🚨 ‼️‼️‼️‼️ 🚨…	1	🚨 ‼️‼️‼️‼️ 🚨…
1319	Робота на дому…	1	Работа на дому…
1320	▪ ▪ ▪ ▪ ▪ ▪ ▪ ▪ ▪ ▪ ▪…	1	▪ ▪ ▪ ▪ ▪ ▪ ▪ ▪ ▪ ▪ ▪…
1321	Потрібні кілька людей на кілька нескладних …	1	Требуются несколько людей на несколько не …
1322	⏰ Розсилка вашого оголошення по всіх чатах!…	1	⏰ Рассылка вашего объявления по всем чатам!…
1323	🚀**Пропозиція на мільйони.…	1	🚀**Предложение на милионы.…

1 303 - 1 323 of 12 933 Перейти до: 1

Рисунок 2.3 – Приклад повідомлень, які були перекладені та використані в роботі

Таким чином, додаткові приклади з’являлися лише тоді, коли було виявлено конкретні «слабкі місця» моделі, що дозволило цілеспрямовано покращувати якість навчальних даних.

Підсумкова структура корпусу. Після завершення етапів збору, автоматичної розмітки, ручної перевірки та додаткової генерації/перекладу загальний корпус містить приблизно 40 000 записів:

– кількість спам-повідомлень (label = 1): близько 20 000;

– кількість ham-повідомлень (label = 0): близько 20 000.

На фінальному етапі (передачі тексту для тренування моделі) всі тексти з поля messages подавалися очищеними, з заміною специфічних елементів на відповідні токени.

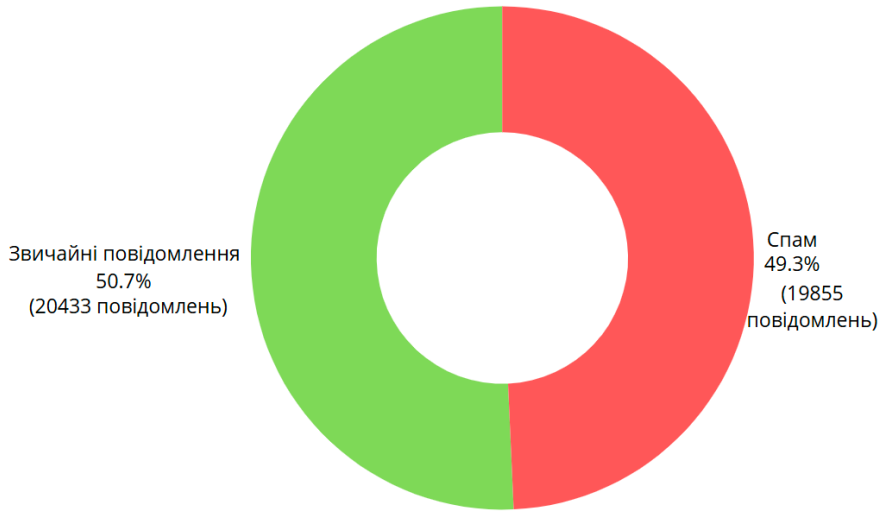


Рисунок 2.4 – Діаграма розподілу кількості спам і ham-повідомлень у підсумковому корпусі

Отже, для побудови навчального корпусу було реалізовано таку послідовність дій: збір даних, автоматичне присвоєння початкових міток, ручна верифікація та корекція, генерація та переклад додаткових прикладів. Цей багатоетапний підхід забезпечує високу якість розмітки корпусу та адаптацію до українських текстів.

Для глибшого розуміння структури та змісту зібраного навчального корпусу було проведено детальний якісний аналіз репрезентативної вибірки даних. З загального масиву у 40 288 повідомлень було випадковим чином відібрано 2000 записів (що становить приблизно 5% від усього корпусу), які було класифіковано за більш вузькими тематичними категоріями.

Для спам-повідомлень було визначено 10 поширених категорій, таких як "Вакансії/заробіток", "Реклама каналів", "Фінансовий спам" тощо. Аналогічно,

для легітимних повідомлень було виділено 10 категорій, що відображають типові сценарії спілкування, наприклад, "Особисте спілкування", "Технічні питання", "Новини/політика". Результати цього підрахунку представлені у таблиці 2.1 (для спам-повідомлень) та таблиці 2.2 (для звичайних повідомлень).

Таблиця 2.1 – Підрахунок класифікацій спам повідомлень

Категорія	Кількість	Відсоткове співвідношення
Вакансії/заробіток	655	32.75%
Фінансовий спам	47	2.35%
Продаж наркотиків	10	0.5%
Азартні ігри	15	0.75%
Еротичний спам	53	2.65%
Реклама каналів	122	6.1%
Послуги хакерів	20	1%
Фальшиві документи	21	1.05%
Реклама товарів	39	1.95%
Посилання-фішинг	9	0.45%
Загалом	991	49.55%

Таблиця 2.2 – Підрахунок класифікацій звичайних повідомлень

Категорія	Кількість	Відсоткове співвідношення
Особисте спілкування	339	16.95%
Технічні питання	123	6.15%
Навчання/робота	21	1.05%
Новини/політика	128	6.4%
Розваги/ігри	81	4.05%
Запрошення на заходи	19	0.95%
Коди підтвердження	16	0.8%
Прохання про допомогу	130	6.5%
Громадські обговорення	124	6.2%
Привітання/подяки	28	1.4%
Загалом	1009	50.45%

На основі наданих таблиць можна зробити наступні висновки:

1. Збалансованість вибірки:

– вибірка з 2000 повідомлень є добре збалансованою: 991 спам-повідомлення (49.55%) та 1009 звичайних повідомлень (50.45%). Це дуже добре для об'єктивної оцінки та свідчить про якість усього корпусу, що є важливим для навчання неупередженої моделі.

2. Аналіз категорій спаму (табл. 2.1):

– домінуюча категорія: абсолютним лідером є категорія "Вакансії/заробіток" (655 повідомлень), що становить 32.75% від усієї вибірки та близько 66% від усього спаму (655 / 991). Це ключовий висновок, який показує, що значна частина спаму в українському сегменті месенджерів пов'язана саме з пропозиціями роботи, легкого заробітку та фінансовими схемами;

– друга за популярністю категорія: "Реклама каналів" (122 повідомлення, 6.1%) також є дуже поширеним видом спаму, особливо в Telegram, де відбувається переливання аудиторії;

– інші значущі категорії: "Еротичний спам" (53), "Фінансовий спам" (47), та "Реклама товарів" (39) також становлять помітну частину спаму;

– критично небезпечний спам: Категорії, як-от "Посилання-фішинг" (9), "Продаж наркотиків" (10) та "Послуги хакерів" (20), хоч і не є найчисельнішими, є надзвичайно важливими для виявлення через свою високу небезпеку для користувачів. Їх наявність у корпусі є обов'язковою для навчання надійної моделі.

3. Аналіз категорій звичайних повідомлень (табл. 2.2):

– найбільша категорія: "Особисте спілкування" (339 повідомлень, 16.95%) є основою "не-спам" корпусу, що цілком очікувано і правильно, оскільки модель має навчитися відрізняти саме таку типову розмовну мову;

– різноманітність тем: Корпус "не-спам" повідомлень дуже різноманітний. Присутні такі великі категорії як "Прохання про допомогу" (130), "Новини/політика" (128), "Громадські обговорення" (124) та "Технічні питання"

(123). Це гарантує, що модель не буде вважати спамом будь-яке повідомлення, що не є простим "привіт/як справи";

– складні для розрізнення випадки: Наявність категорії "Навчання/робота" (21 повідомлення) є дуже важливою. Ці легітимні повідомлення за лексикою можуть бути схожими на спам з категорії "Вакансії/заробіток", що створює гарний виклик для моделі і перевіряє її здатність розуміти контекст, а не лише ключові слова.

2.4 Функціональне моделювання

Для моделювання функціональної структури системи виявлення спаму було побудовано діаграму прецедентів (рис. 2.5), яка відображає взаємодію користувача із системою, та діаграму послідовностей (рис. 2.6), що демонструє обробку повідомлення у середовищі Python із використанням моделі BERT.

Діаграма прецедентів (рис. 2.5) ілюструє основні сценарії взаємодії акторів із розробленою системою. Вона дозволяє візуалізувати функціональні вимоги та визначити межі системи. У даному випадку виділено два ключових актори: «Користувач застосунку» та «Розробник».

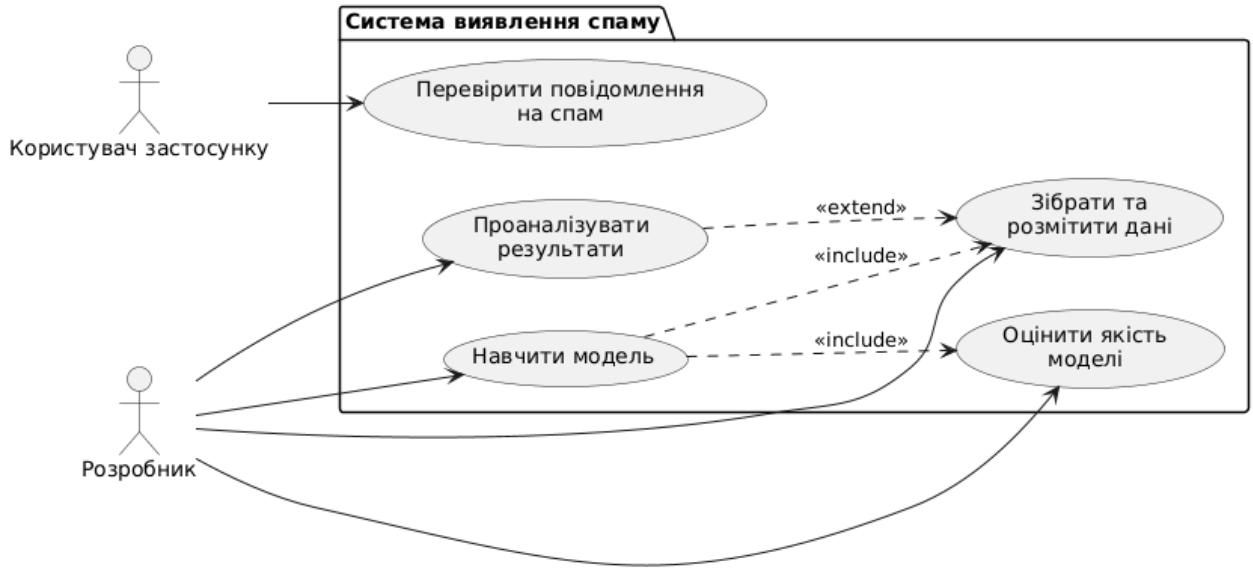


Рисунок 2.5 – Діаграма прецедентів системи

«Користувач застосунку» є основним актором, який ініціює прецедент «Перевірити повідомлення на спам». «Розробник» виконує адміністративні та дослідницькі функції: «Зібрати та розмітити дані», «Навчити модель», «Оцінити якість моделі» та «Проаналізувати результати». Прецедент «Навчити модель» включає (відношення <<include>>) етап збору та розмітки даних, оскільки навчання неможливе без підготовленого корпусу. Аналогічно, оцінка якості моделі є невід'ємною частиною її навчання. У свою чергу, аналіз результатів може розширювати (відношення <<extend>>) процес збору даних, якщо виявляється необхідність доповнення навчального корпусу для покращення якості, як це було описано в попередніх підрозділах.

Діаграма послідовностей (рис. 2.6) деталізує взаємодію між об’єктами системи під час виконання прецеденту «Перевірити повідомлення на спам». Вона показує хронологічний порядок обміну повідомленнями між користувачем, демонстраційним застосунком (ботом), модулем детекції та моделлю BERT.

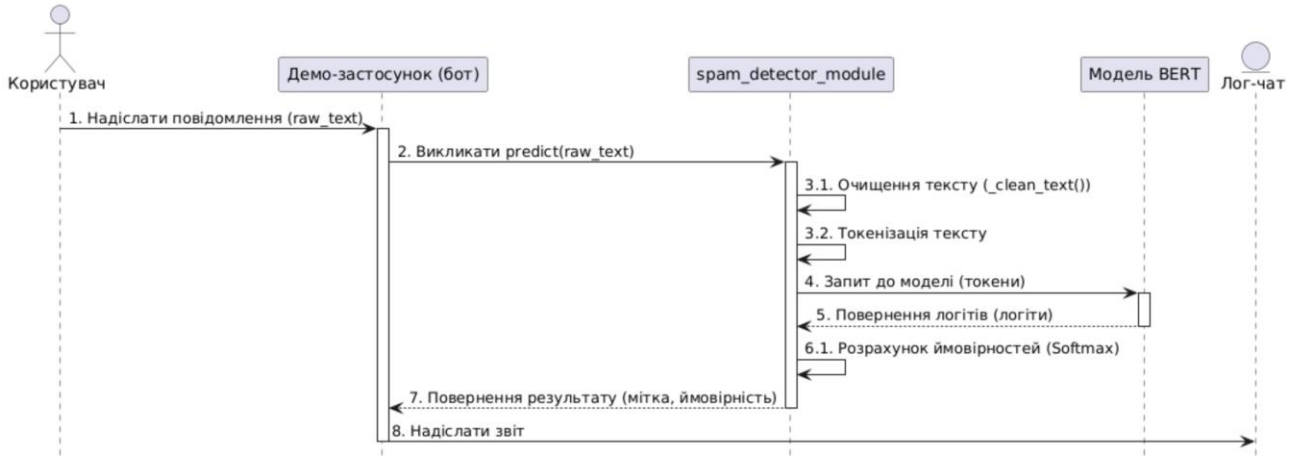


Рисунок 2.6 – Діаграма послідовностей для процесу розпізнавання спаму

Як видно з діаграми, процес починається з надсилання повідомлення користувачем. Демонстраційний застосунок викликає функцію predict() модуля spam_detector_module. Модуль послідовно виконує внутрішню логіку: очищує текст, токенізує його та передає на обробку моделі BERT. Після отримання прогнозу (логітів) від моделі, модуль розраховує ймовірності та повертає фінальний результат застосунку, який, у свою чергу, надсилає звіт у лог-чат.

Активаційні прямокутники на лініях життя об'єктів показують час, протягом якого об'єкт був активним (виконував операцію).

2.5 Структура бази даних (SQLite)

Для зберігання всього навчального корпусу (зібраних, верифікованих, перекладених та згенерованих повідомлень) використовується реляційна система управління базами даних SQLite. Обрано мінімалістичну, але ефективну логічну модель даних, що складається з єдиної таблиці. Така спрощена, денормалізована структура була обрана для максимального спрощення ETL-процесів (Extract, Transform, Load) та уніфікації взаємодії всіх програмних компонентів системи з даними.

Опис таблиці messages: база даних містить єдину ключову таблицю messages, що призначена для зберігання текстових повідомлень та відповідних їм міток.

Опис полів таблиці messages: message (тип даних: TEXT, обмеження: NOT NULL) – основне поле, що зберігає текстовий контент повідомлення. На різних етапах підготовки даних це може бути сирий текст, отриманий з джерела, текст після перекладу або згенерований за допомогою ChatGPT. Поле не може бути порожнім. label (тип даних: INTEGER, обмеження: NOT NULL) – поле, що зберігає цілочислову мітку класу, до якого належить повідомлення. Використовується бінарна система розмітки: 0 – для легітимних повідомлень ("не-спам", ham). 1 – для спам-повідомлень (spam). Поле не може бути порожнім.

Оскільки база даних складається з однієї таблиці, зв'язки між таблицями (foreign key constraints) відсутні.

Фізична реалізація структури таблиці messages в SQLite визначається наступним SQL-запитом:

```
CREATE TABLE IF NOT EXISTS messages (
    message TEXT NOT NULL,
    label   INTEGER NOT NULL);
```

Усі незалежні скрипти, що є частиною системи (збір даних, препроцесинг,

верифікація, навчання моделі), використовують саме цю таблицю для отримання даних для обробки або для додавання нових, верифікованих записів до навчального корпусу. Табличне представлення структури БД наведено в таблиці 2.3.

Таблиця 2.3 – Опис полів таблиці messages

Назва поля	Тип поля	Розмірність	Поясненя
message	TEXT	Динамічний	Повідолмення в сирому вигляді
label	INTEGER	1 байт (boolean)	1 (спам) або 0 (не спам)

2.6 Процес очищення й підготовки текстів (препроцесинг)

На етапі підготовки навчальних даних кожне повідомлення, що зберігається у базі даних, проходить низку операцій з очищення та нормалізації. Це необхідно для забезпечення консистентності вхідних даних для моделі BERT та видалення "шуму", який може негативно вплинути на якість навчання. Детальна реалізація функції препроцесингу Clean_Text наведена у Додатку А. Нижче описано ключові кроки цього процесу (рис. 2.7).

1. Завантаження сирого тексту: Початковий текст береться з поля message таблиці messages.
2. Обробка URL: Усі URL-адреси (з протоколами http://, https:// або без них) замінюються на токен [URL:домен]. Домен витягується, приводиться до нижнього регістру, з нього видаляються префікс www. та номер порту. Якщо домен невалідний або його не вдалося виділити, використовуються токени [URL_UNKNOWN] або [URL_INVALID].
3. Заміна email-адрес, телефонів, юзернеймів і хештегів:
4. Email-адреси (something@domain.tld) замінюються на [EMAIL].
5. Телефонні номери різних форматів замінюються на [PHONE].
6. Юзернейми (@username) замінюються на [USERNAME].
7. Хештеги (#topic) замінюються на [HASHTAG].
8. Заміна символів валют: Поширені символи валют ($, €, £, ₴, ₿ тощо) замінюються на токен [CURRENCY].

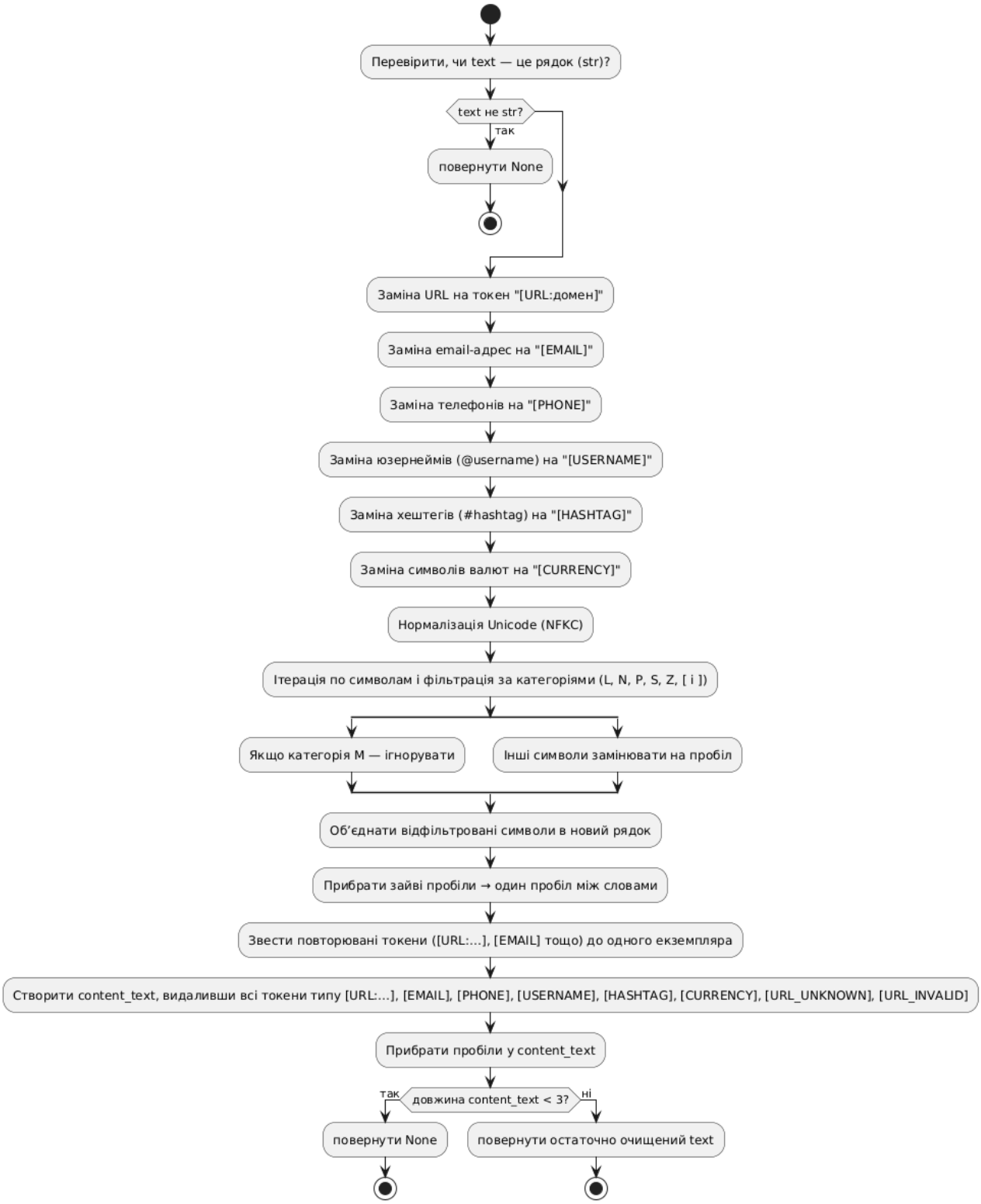


Рисунок 2.7 – Діаграма робочого процесу попередньої обробки тексту

9. Нормалізація Unicode (NFKC): Застосовується для уніфікації представлення символів.

10. Фільтрація непотрібних символів: Залишаються лише літери (категорія

'L'), цифри ('N'), пунктуація ('P'), символи, включно з емодзі ('S'), пробіли ('Z') та квадратні дужки [ ]. Інші символи (наприклад, діакритики категорії 'M' або контрольні символи) видаляються або замінюються на пробіл.

11. Зведення зайвих пробілів і дублюючих токенів: Послідовності пробілів замінюються одним пробілом. Повторювані спеціальні токени (наприклад, [URL:site.com] [URL:site.com]) об'єднуються в один екземпляр.

12. Перевірка на довжину корисного тексту: Після тимчасового видалення всіх спеціальних токенів перевіряється, чи залишається в тексті щонайменше 3 значущих символи (літери, цифри, пунктуація). Якщо ні, текст вважається нерелевантним, і функція повертає None. В іншому випадку повертається остаточно очищений рядок.

Цей алгоритм препроцесингу гарантує, що на вхід моделі BERT подаються уніфіковані та очищені дані, що сприяє підвищенню якості класифікації.

2.7 Розробка та навчання класифікаційної моделі BERT

У цьому підрозділі описується процес розробки та навчання класифікатора на основі донавченої моделі BERT для задачі розпізнавання спаму. Детальна реалізація тренувального скрипта train_spam_classifier.py наведена у Додатку А.

Скрипт виконує послідовність операцій: завантаження корпусу з SQLite, очищення текстів за допомогою функції clean_text, розбиття даних на тренувальний та тестовий набори, ініціалізація токенізатора і моделі BERT, налаштування гіперпараметрів та об’єкта Trainer з бібліотеки Transformers, запуск процесу навчання, оцінка моделі на тестовому наборі та збереження найкращої версії моделі.

Загальна архітектура тренувального скрипта:

1. Зчитування та очищення даних: Дані (поля message та label) зчитуються з бази даних 40k_spam.db у pandas DataFrame. До кожного повідомлення застосовується функція clean_text. Повідомлення, що після очищення виявилися

занадто короткими (повернули None), відкидаються.

2. Розбивка на тренувальний і тестовий набори: Очищені дані діляться у співвідношенні 80% (тренувальні) на 20% (тестові) із збереженням балансу класів (стратифікація за полем label). Використовується фіксоване значення RANDOM_STATE для відтворюваності результатів.

3. Ініціалізація токенізатора та моделі: Використовується попередньо навчена модель bert-base-multilingual-cased. Токенізатор завантажується за допомогою AutoTokenizer.from_pretrained('bert-base-multilingual-cased'). Модель для класифікації послідовностей ініціалізується через AutoModelForSequenceClassification.from_pretrained('bert-base-multilingual-cased', num_labels=2).

4. Формування PyTorch Dataset: Створено кастомний клас SpamDataset, що наслідує torch.utils.data.Dataset. Він приймає тексти, мітки, токенізатор та максимальну довжину послідовності (max_length=128). Тексти токенізуються, і для кожного прикладу зберігаються input_ids та attention_mask. Паддінг виконується динамічно за допомогою DataCollatorWithPadding під час формування батчів.

5. Налаштування гіперпараметрів навчання: Основні гіперпараметри, визначені для навчання:

- NUM_EPOCHS = 3;
- TRAIN_BATCH_SIZE = 22;
- EVAL_BATCH_SIZE = 16;
- LEARNING_RATE = 2e-5;
- TOKENIZER_MAX_LENGTH = 128;
- WEIGHT_DECAY = 0.01.

Навчання конфігурується через об'єкт TrainingArguments, який також керує логуванням, стратегією оцінки (eval_strategy="epoch") та збереженням моделі (save_strategy="epoch", load_best_model_at_end=True за метрикою "f1"). Нижче наведено опис ключових параметрів та обґрунтування їх вибору:

– NUM_EPOCHS = 3 (Кількість епох навчання): Епоха – це один повний прохід моделі через увесь навчальний набір даних. Значення 3 є стандартним та рекомендованим для донавчання (fine-tuning) моделей типу BERT. Менша кількість епох може призвести до недонавчання, тоді як більша – до перенавчання (overfitting), коли модель надто добре "запам'ятовує" навчальні дані, але втрачає здатність узагальнювати на нових, небачених прикладах;

– TRAIN_BATCH_SIZE = 22 (Розмір батчу для навчання): Це кількість навчальних прикладів (повідомлень), які одночасно подаються на вхід моделі за одну ітерацію для оновлення її ваг. Розмір батчу впливає на стабільність навчання та використання обчислювальних ресурсів. Значення 22 було підібрано експериментально як максимально можливе, що вміщується у 6 ГБ відеопам'яті наявної відеокарти NVIDIA RTX 4050, забезпечуючи при цьому стабільне навчання моделі;

– EVAL_BATCH_SIZE = 16 (Розмір батчу для оцінки): Аналогічно до TRAIN_BATCH_SIZE, але використовується на етапі валідації. Цей параметр не впливає на оновлення ваг моделі, тому його вибір менш критичний, але також залежить від обсягу відеопам'яті;

– LEARNING_RATE = 2e-5 (Швидкість навчання): Це коефіцієнт, який визначає, наскільки сильно змінюються ваги моделі на кожному кроці навчання. Для донавчання трансформерних моделей використовуються малі значення швидкості навчання, щоб не "зруйнувати" знання, отримані моделлю на етапі попереднього навчання на величезних масивах даних. Значення 2e-5 (0.00002) є типовим і рекомендованим для таких задач;

– TOKENIZER_MAX_LENGTH = 128 (Максимальна довжина послідовності): Це максимальна кількість токенів, до якої буде обрізано довге повідомлення або доповнено коротке. Вибір 128 токенів є компромісом між здатністю охопити більшість текстових повідомлень у месенджерах та ефективним використанням пам'яті і швидкодією моделі;

– WEIGHT_DECAY = 0.01 (Коефіцієнт згасання ваг): Це техніка

регуляризації (L2-регуляризація), що допомагає запобігти перенавчанню. Вона додає до функції втрат невеликий "штраф" за надто великі значення ваг моделі, спонукаючи її до пошуку більш простих та узагальнюючих рішень;

– стратегії оцінки та збереження: eval_strategy="epoch" та save_strategy="epoch": Ці параметри вказують, що оцінку якості моделі на валідаційному наборі та збереження її контрольної точки потрібно виконувати після завершення кожної епохи навчання. load_best_model_at_end=True та metric_for_best_model="f1": Це важливе налаштування, яке вказує Trainer-у відстежувати метрику F1-міра на валідаційному наборі після кожної епохи. Після завершення всіх епох навчання, буде автоматично завантажено ту версію моделі, яка показала найкраще значення цієї метрики. Це гарантує, що фінально збережена модель буде найефективнішою з точки зору балансу чіткості та повноти, а не просто моделлю з останньої епохи, яка могла почати перенавчатися.

6. Обчислення метрик: Для оцінки якості моделі під час навчання та валідації використовується функція compute_metrics, яка обчислює accuracy, precision, recall та f1-score (для бінарної класифікації).

7. Створення об’єкта Trainer та запуск тренування: Об'єкт Trainer ініціалізується з моделлю, аргументами навчання, навчальним та валідаційним датасетами, токенізатором, колатором даних та функцією обчислення метрик. Процес навчання запускається викликом trainer.train(). На рисунках 2.8-2.9 показано скріншоти виводу в консоль повідомлень та даних при навчанні моделі.

```
2025-06-15 13:39:58,380 - INFO - Розподіл класів у тестовому наборі:
label
0    0.507385
1    0.492615
Name: proportion, dtype: float64
2025-06-15 13:39:58,380 - INFO - Ініціалізація моделі 'bert-base-multilingual-cased' та токенізатора...
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-multilingual-cased and are newly initialized: ['
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
/mnt/c/Users/vkost/PycharmProjects/PythonProject/TG/Version mBERT/train_spam_classifier_v2Gem.py:237: FutureWarning: `tokenizer` is deprecated and will be
  trainer = Trainer(
2025-06-15 13:40:02,631 - INFO - Початок тренування моделі...
  2%|         | 100/4320 [00:22<15:07,  4.65it/s]{'loss': 0.281, 'grad_norm': 14.023798942565918, 'learning_rate': 1.9546296296296298e-05, 'epoch': 0.07
```

Рисунок 2.8 – Вивід в консоль повідомлень про початок навчання моделі

```
100%|██████████| 4320/4320 [16:05<00:00,  4.78it/s]
100%|██████████| 496/496 [00:12<00:00, 38.39it/s]
                                                 {'eval_loss': 0.10196505486965l8, 'eval_accuracy': 0.9805580103522282, 'eval_precision': 0.9795291709314228, 'ev
{'train_runtime': 995.5251, 'train_samples_per_second': 95.467, 'train_steps_per_second': 4.339, 'train_loss': 0.07039134732826992, 'epoch': 3.0}
100%|██████████| 4320/4320 [16:35<00:00,  4.34it/s]
2025-06-15 13:56:38,415 - INFO - Тренування завершено.
2025-06-15 13:56:38,417 - INFO - Оцінка моделі...
100%|██████████| 496/496 [00:12<00:00, 38.92it/s]
2025-06-15 13:56:51,383 - INFO - Результати оцінки: {'eval_loss': 0.10196505486965l8, 'eval_accuracy': 0.9805580103522282, 'eval_precision': 0.9795291709314228,
2025-06-15 13:56:51,384 - INFO - Збереження фінальної навченої моделі в ./results_v_cased_improved/spam40k_v(example)...
2025-06-15 13:56:56,026 - INFO - Навчена модель та токенізатор збережені в ./results_v_cased_improved/spam40k_v(example)

Process finished with exit code 0
```

Рисунок 2.9 – Вивід в консоль повідомлень про завершення тренування

8. Збереження моделі та токенізатора: Після завершення навчання найкраща модель та токенізатор зберігаються у вказану директорію (./models/spam_bert) за допомогою методів model.save_pretrained() та tokenizer.save_pretrained().

На рисунку 2.10 графічно показана схема роботи скрипта.

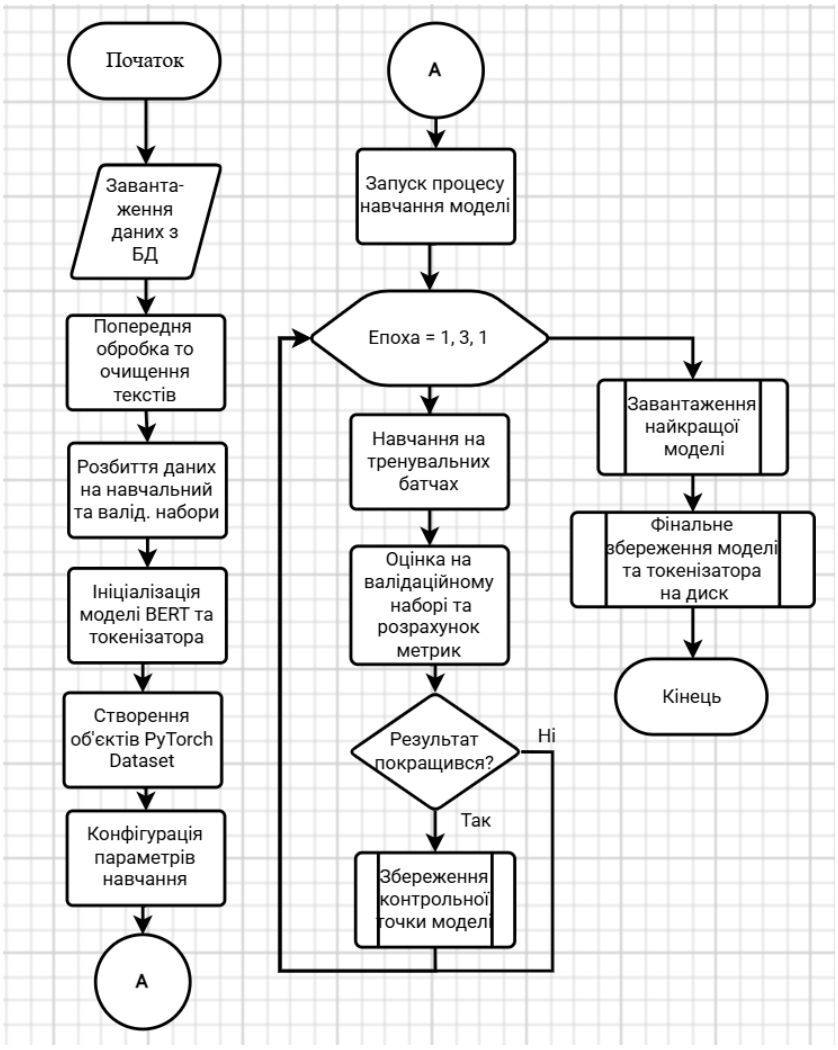


Рисунок 2.10 – Схема роботи скрипта навчання моделі

Після трьох епох навчання на валідаційній вибірці (близько 7900 прикладів), яка не використовувалася безпосередньо для оновлення ваг моделі, було отримано наступні показники ефективності класифікатора:

– eval_loss: 0.1099 – Значення функції втрат на валідаційному наборі. Чим нижче це значення, тим краще модель генералізує дані, які не бачила під час навчання. Втрати показують, наскільки прогнози моделі відхиляються від реальних міток;

– eval_accuracy: 0.9791 (або 97.91%) – Точність (Accuracy). Частка правильно класифікованих прикладів (як спаму, так і не-спаму) від загальної кількості прикладів у валідаційному наборі. Показує загальну коректність роботи моделі;

– eval_precision: 0.9784 (або 97.84%) – Чіткість (Precision) для класу "спам". Частка повідомлень, які модель правильно ідентифікувала як спам, серед усіх повідомлень, які модель позначила як спам. Висока чіткість означає, що якщо модель каже, що це спам, то це, ймовірно, дійсно спам (мало хибнопозитивних спрацювань);

– eval_recall: 0.9792 (або 97.92%) – Повнота (Recall) для класу "спам". Частка повідомлень, які модель правильно ідентифікувала як спам, серед усіх реально спамових повідомлень у валідаційному наборі. Висока повнота означає, що модель добре виявляє більшість спаму (мало хибнонегативних спрацювань);

– eval_f1: 0.9788 (або 97.88%) – F1-міра (F1-score). Гармонійне середнє між чіткістю (precision) та повнотою (recall). Ця метрика є особливо корисною, коли важливий баланс між precision та recall, або коли класи незбалансовані (хоча у даному випадку корпус збалансований).

Ці результати свідчать про високу ефективність розробленого класифікатора, де показники за основними метриками якості наближаються до 98%. Це підтверджує успішність обраного підходу до донавчання моделі BERT та підготовки даних.

Висновок до другого розділу

У другому розділі було докладно описано етапи проєктування та розробки ключових компонентів системи автоматизованого виявлення спаму в українськомовних текстових повідомленнях.

Обґрунтовано загальну архітектуру системи та вибір програмно-апаратних засобів. Система спроєктована як набір незалежних модулів, що взаємодіють через єдину базу даних SQLite, забезпечуючи гнучкість та легкість модифікації. Вибір Python, Hugging Face Transformers, PyTorch, SQLite та використання персонального ноутбука на базі NVIDIA GeForce RTX 4050 дозволив ефективно реалізувати всі етапи розробки та навчання моделі.

Розроблено методику збору та калібрування навчального корпусу, що включає автоматичну первинну розмітку, ретельну ручну верифікацію та цільову генерацію/переклад додаткових прикладів. Сформовано збалансований корпус з близько 40 000 українськомовних повідомлень, що забезпечує високу якість та репрезентативність навчальних даних.

Спроєктовано та реалізовано ефективний алгоритм препроцесингу текстів. Алгоритм включає нормалізацію URL, контактних даних, символів валют та інших елементів з їх перетворенням на уніфіковані токени, що покращує якість вхідних даних для моделі.

Розроблено та навчено класифікаційну модель на основі архітектури BERT (bert-base-multilingual-cased). Завдяки підбору оптимальних гіперпараметрів та використанню якісного навчального корпусу, навчена модель продемонструвала високі показники ефективності, досягнувши точності, чіткості, повноти та F1-міри близько 98% на валідаційній вибірці.

Реалізовані на цьому етапі компоненти та моделі є основою для подальшої розробки програмного модуля для практичного застосування та його всебічного тестування, що буде описано у наступному розділі.

3 РОЗРОБКА ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМНОГО МОДУЛЯ ДЕТЕКЦІЇ СПАМУ

У цьому розділі представлено детальний опис розробки кінцевого програмного модуля для виявлення спаму, призначеного для практичного застосування. Також буде наведено методику та результати його експериментального дослідження й тестування на різноманітних повідомленнях для оцінки ефективності та надійності в умовах, наближених до реальних. Основна увага приділяється створенню автономного модуля, його функціональним можливостям та перевірці якості його роботи.

3.1 Технології та інструменти, що використовуються в роботі

Для реалізації системи автоматизованого виявлення спаму були задіяні сучасна мова програмування Python, її бібліотеки та сервіси, що забезпечують гнучкість розробки, високу продуктивність та простоту розгортання.

Використано віртуальні середовища (venv) для ізоляції залежностей, а також менеджер пакетів pip для встановлення та оновлення бібліотек.

Для NLP та машинного навчання було використано ряд бібліотек.

Hugging Face Transformers: основна бібліотека для завантаження, донавчання та інференсу трансформерних моделей (зокрема BERT). Забезпечує доступ до попередньо навчених моделей та інструментів для їх тонкого налаштування.

PyTorch: відкрита бібліотека машинного навчання, що використовується як бекенд для моделей з Hugging Face Transformers, забезпечуючи ефективні обчислення на CPU та GPU.

Scikit-learn: використовується для реалізації класичних алгоритмів класифікації (якщо потрібне порівняння), оцінки метрик (accuracy, precision, recall, F1-score), розбиття даних на навчальну та тестову вибірки, а також для

підбору гіперпараметрів.

Pandas: бібліотека для маніпулювання даними, зокрема для завантаження, обробки та аналізу датасетів у табличному вигляді (наприклад, при читанні даних з SQLite).

Для зберігання та калібрування корпусу повідомлень було обрано СУБД SQLite, як легку вбудовану реляційну базу даних без необхідності окремого сервера. Використовується для зберігання зібраних повідомлень та їх міток, а також для ручної верифікації та корекції даних оператором.

Інструментом для розгортання та демонстрації роботи розроблюваного модуля послужила бібліотека Telegram Bot API, для створення клієнтської частини у вигляді Telegram-бота, що приймає повідомлення, передає їх на класифікацію та надсилає результати користувачу.

Вибір цих технологій та інструментів дозволив створити гнучкий та ефективний пайплайн для розробки, навчання та тестування системи детекції спаму.

3.2 Розробка програмного модуля spam_detector_module для виявлення спаму

Ключовим результатом дипломної роботи є створення автономного програмного модуля spam_detector_module, який інкапсулює всю необхідну логіку для класифікації текстових повідомлень на "спам" та "не-спам" (ham). Цей модуль спроєктовано з акцентом на простоту використання та легкість інтеграції в будь-які інші програмні системи, що є основною вимогою до кінцевого продукту даної роботи.

Програмний модуль spam_detector_module розроблений як самодостатній інструмент, що виконує повний цикл аналізу текстового повідомлення для визначення його належності до спаму. Основними функціями модуля є:

1. Прийом вхідного тексту: модуль розрахований на прийом будь-якого

текстового повідомлення у вигляді рядка (raw text). Відсутні спеціальні вимоги до попереднього форматування чи підготовки вхідних даних з боку користувача або інтегруючої системи.

2. Автоматична попередня обробка (препроцесинг) тексту: всередині модуля реалізовано виклик розробленого та описаного у підрозділі 2.6 алгоритму очищення та нормалізації тексту. Цей етап є критично важливим і включає обробку URL-адрес, електронних поштових адрес, номерів телефонів, специфічних для інтернет-комунікації юзернеймів та хештегів, символів валют, а також нормалізацію Unicode та видалення нерелевантних або "шумових" символів.

3. Токенізація тексту: після очищення текст передається на токенізацію. Для цього використовується токенізатор bert-base-multilingual-cased, який застосовувався і під час навчання основної класифікаційної моделі, що забезпечує консистентність представлення даних.

4. Класифікація тексту за допомогою моделі BERT: отримані токени подаються на вхід донавченій моделі bert-base-multilingual-cased (деталі навчання якої описані в підрозділі 2.5). Модель генерує прогноз, визначаючи, до якого класу ("спам" чи "не-спам") належить вхідне повідомлення.

5. Повернення результату класифікації: кінцевим результатом роботи модуля є бінарна мітка: 1, що відповідає класу "спам", або 0, що відповідає класу "не-спам" (ham). Передбачено також можливість (хоча в базовому найпростішому інтерфейсі не акцентовано) отримання ймовірнісної оцінки належності до класу "спам", що може бути використано для налаштування порогу спрацьовування в інтегруючих системах.

Основна ідея модуля полягає в тому, щоб надати розробникам максимально простий та уніфікований інтерфейс ("подай текст – отримай результат"), приховавши всю складність внутрішніх процесів обробки природної мови та роботи нейронної мережі.

Програмний модуль spam_detector_module розроблено як Python-пакет.

Основна функціональність зосереджена у файлі detector.py. Ключовим елементом інтерфейсу є функція predict(), яка має наступну сигнатуру predict(raw_text: str, threshold: float = 0.5) -> int, float .

Вхідні параметри:

– raw_text: str – обов'язковий параметр, що приймає сирий текстовий рядок для аналізу;

– threshold: float – опціональний параметр, який визначає поріг спрацьовування для класифікації повідомлення як спам. За замовчуванням встановлено значення 0.5. Якщо розрахована моделлю ймовірність належності повідомлення до класу "спам" перевищує це значення, повідомлення класифікується як спам.

Вихідне значення:

– int – функція повертає ціле число: 1, якщо повідомлення визнано спамом, або 0, якщо воно класифіковане як "не-спам" (ham);

– float – повертає число з плаваючою комою, що відповідає ймовірності повідомлення на відповідність до спаму.

Важливою особливістю реалізації є механізм "лінивого завантаження" (lazy loading) моделі BERT та її токенізатора. Ці компоненти завантажуються в пам'ять лише під час першого виклику функції predict(). При наступних викликах використовуються вже завантажені екземпляри, що значно підвищує продуктивність при обробці серії повідомлень, оскільки усувається необхідність повторного завантаження важких файлів моделі.

Директорія з файлами навченої моделі (що включає pytorch_model.bin – ваги моделі, та config.json – конфігурацію моделі) та файлами токенізатора (vocab.txt, tokenizer_config.json, special_tokens_map.json) є невід'ємною частиною модуля. Згідно з описом у Розділі 2, ця директорія повинна розташовуватися відносно файлу detector.py так, щоб модуль міг її знайти та завантажити ресурси.

Структура файлів модуля має наступний вигляд (рис. 3.1):

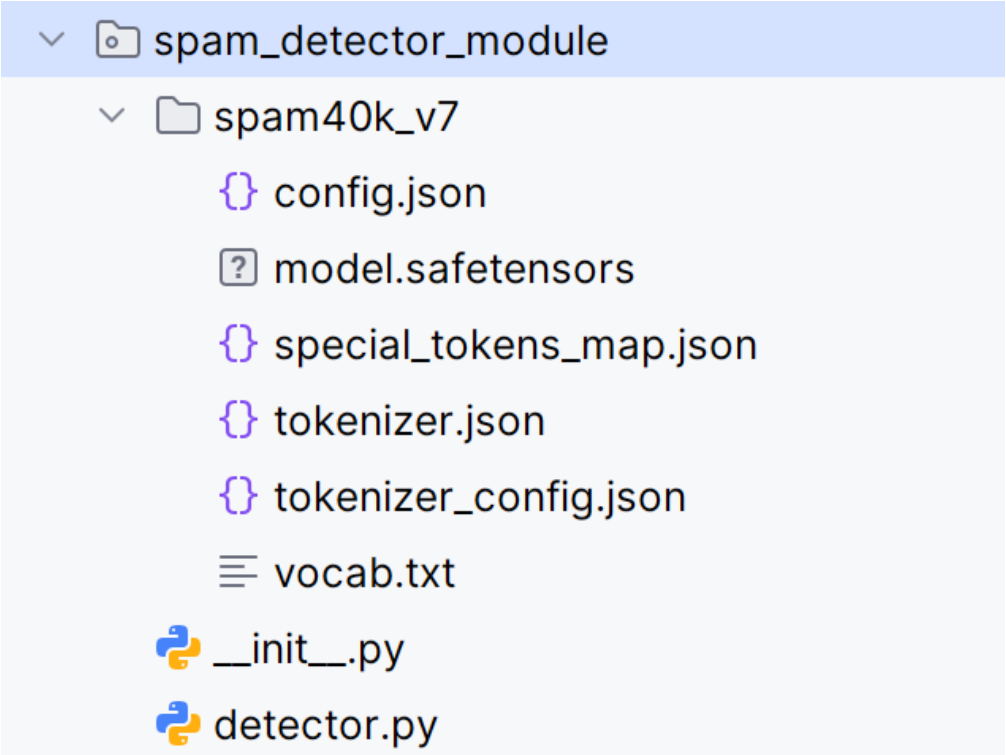


Рисунок 3.1 – Файлова структура модуля

Робота функції predict(raw_text) всередині модуля відбувається за наступним алгоритмом:

1. Ініціалізація ресурсів (при першому виклику):

– перевіряється, чи вже завантажені в пам'ять об'єкти моделі BERT та токенізатора;

– якщо ні (це перший виклик функції з моменту запуску програми), виконується завантаження токенізатора AutoTokenizer.from_pretrained (MODEL_DIR) та моделі Auto Model For Sequence Classification.from_pretrained (MODEL_DIR). MODEL_DIR вказує на локальний шлях до збережених файлів навченої моделі;

– модель переміщується на відповідний обчислювальний пристрій (CPU або GPU, якщо torch.cuda.is_available() повертає True);

– модель переводиться в режим оцінки за допомогою model.eval(), що вимикає механізми, специфічні для навчання (наприклад, Dropout).

2. Перевірка валідності вхідних даних:

– здійснюється перевірка, чи є raw_text рядком (isinstance(raw_text, str));

– перевіряється, чи raw_text не є порожнім після видалення пробільних символів з початку та кінця (raw_text.strip());

– якщо будь-яка з цих перевірок не проходить, функція повертає 0 (не-спам), вважаючи такі вхідні дані нерелевантними для аналізу.

3. Очищення тексту:

– до вхідного рядка raw_text застосовується функція clean_text(), описана в підпункті 2.6 та Додатках А та Б. Ця функція виконує всі необхідні заміни (URL, email тощо) та нормалізацію.

4. Перевірка результату очищення:

– якщо функція clean_text() повернула None (що означає, що після очищення текст став порожнім або містить менше трьох значущих символів), функція predict() повертає 0.

5. Токенізація підготовленого тексту:

– очищений текст передається на вхід завантаженому токенізатору;

– токенізація виконується з параметрами: truncation=True (обрізання тексту, якщо він довший за максимальну довжину), max_length = TOKENIZER_MAX_LENGTH (константа, що відповідає 128 токенам), padding = "max_length" (доповнення послідовності спеціальними токенами до максимальної довжини, якщо вона коротша), та return_tensors="pt" (повернення результату у вигляді тензорів PyTorch);

– отримані тензори (input_ids, attention_mask) переміщуються на той самий обчислювальний пристрій, де знаходиться модель.

6. Отримання прогнозу від моделі (інференс):

– виконується в контексті with torch.no_grad():, щоб вимкнути обчислення градієнтів, які не потрібні на етапі інференсу та лише споживають ресурси;

– токенізовані дані подаються на вхід моделі: logits = model (**inputs).logits. Модель повертає "сирі" значення (логіти) для кожного класу.

7. Розрахунок ймовірностей класів:

– до отриманих логітів застосовується функція Softmax по останній осі (F.softmax(logits, dim=-1)), щоб перетворити їх на розподіл ймовірностей. Результатом є тензор, де probs[0][0] – ймовірність класу "не-спам", а probs[0][1] – ймовірність класу "спам";

– витягується ймовірність того, що повідомлення є спамом: spam_prob = probs[0][1].item().

8. Прийняття фінального рішення:

– розрахована spam_prob порівнюється із вхідним параметром threshold;

– якщо spam_prob > threshold, функція повертає 1 (повідомлення класифіковано як спам);

– в іншому випадку функція повертає 0 (повідомлення класифіковано як не-спам).

У випадку виникнення непередбачених помилок під час завантаження моделі з файлів або під час її виконання, модуль spam_detector_module може генерувати виняток типу RuntimeError, надаючи інформацію про причину збою. Це дозволяє системі, яка використовує модуль, коректно обробляти такі ситуації.

Розроблений таким чином модуль spam_detector_module являє собою готове до використання рішення, що забезпечує високу точність виявлення спаму в українськомовних текстах завдяки поєднанню ефективного препроцесингу та потужності донавченої моделі BERT.

Повний код модуля знаходится в додатку Б.

На рисунку 3.2 показано схему роботи модуля.

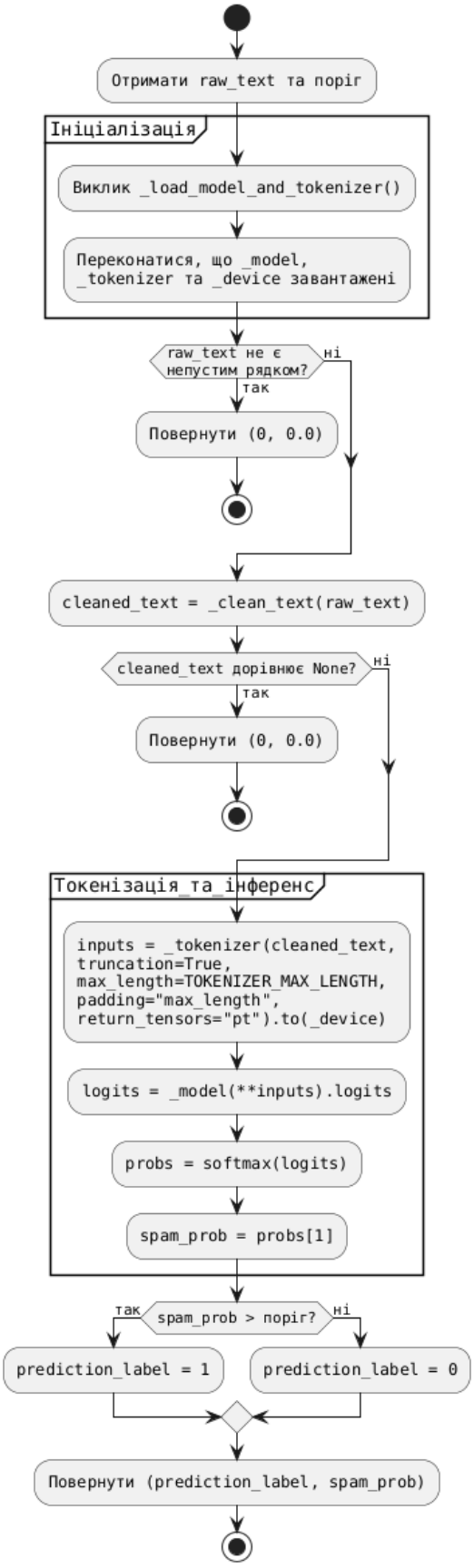


Рисунок 3.2 – Діаграма активностей, яка ілюструє алгоритм обробки тексту для виявлення спаму за допомогою машинного навчання

3.3 Розробка демонстраційного застосунку на основі платформи обміну миттєвими повідомленнями

Для наочної демонстрації роботи розробленого модуля spam_detector_module та перевірки його ефективності в умовах, наближених до реального використання, було створено клієнтський застосунок. Цей застосунок реалізовано у вигляді автоматизованого агента (бота) для популярної платформи обміну миттєвими повідомленнями. Такий вибір дозволяє легко тестувати класифікатор на різноманітних текстових даних, що надсилаються користувачами в інтерактивному режимі, та оперативно отримувати результати аналізу.

Основна мета розробки демонстраційного застосунку – надати простий та інтуїтивно зрозумілий спосіб взаємодії з системою виявлення спаму. Платформи обміну миттєвими повідомленнями широко використовуються, що робить такий інтерфейс доступним для широкого кола користувачів без необхідності встановлення спеціалізованого програмного забезпечення.

Демонстраційний застосунок виконує наступні завдання:

1. Прийом повідомлень: Автоматично отримує текстові повідомлення, надіслані в групові чати, де він присутній, або в особисті повідомлення (якщо це передбачено логікою платформи).

2. Інтеграція з модулем детекції: Передає отриманий текст на аналіз до розробленого програмного модуля spam_detector_module.

3. Логування результатів: Надсилає детальний звіт про результат класифікації кожного повідомлення (мітка "спам"/"не-спам" та ймовірність спаму) у спеціально призначений для цього приватний чат або канал (далі – лог-чат). Це дозволяє розробнику або адміністратору системи відстежувати роботу класифікатора та збирати дані для подальшого аналізу чи донавчання моделі.

4. Мінімальний вплив на користувацький досвід: У публічних чатах бот працює у "тихому" режимі, не відповідаючи безпосередньо на повідомлення

користувачів, а лише аналізуючи їх та надсилаючи звіти в закритий лог-чат. Це зроблено для того, щоб не заважати звичайному спілкуванню учасників.

Використання такого застосунку дозволяє не тільки продемонструвати функціональність модуля spam_detector_module, але й провести його тестування на реальних, "живих" даних, що надходять від користувачів.

Демонстраційний застосунок (бот) розроблено на мові Python з використанням бібліотеки python-telegram-bot (версія 22.0), яка надає зручний інтерфейс для взаємодії з API відповідної платформи обміну повідомленнями [11]. Повний код застосунку наведено у Додатку В (файл bot.py).

Архітектура застосунку включає наступні ключові елементи:

1. Ініціалізація та конфігурація. При запуску бот зчитує конфігураційні параметри: унікальний токен доступу до API платформи (TELEGRAM_BOT_TOKEN) та ідентифікатор лог-чату (LOG_CHAT_ID). Налаштовується система логування для фіксації подій роботи самого бота (старт, зупинка, помилки).

2. Обробник повідомлень (check_message). Це асинхронна функція, яка активується для кожного нового текстового повідомлення, що надходить у чати, де бот є учасником (і має відповідні дозволи на читання повідомлень). Функція ігнорує повідомлення без текстового вмісту та повідомлення, що надходять із самого лог-чату (щоб уникнути зациклювання). Отримується текст повідомлення, інформація про відправника (ім'я, юзернейм, ID) та чат (назва, ID).

3. Взаємодія з модулем spam_detector_module. Текст отриманого повідомлення передається на вхід функції predict() модуля spam_detector_module. Модуль повертає кортеж, що містить бінарну мітку (0 – не спам, 1 – спам) та числове значення ймовірності належності повідомлення до класу "спам".

4. Формування та відправка звіту. На основі результату від модуля spam_detector_module формується текстовий звіт. Звіт містить:

– емодзі-індикатор та текстовий результат ("СПАМ" / "Не спам");

– розраховану ймовірність спаму у відсотковому форматі;

– інформацію про користувача-відправника;

– інформацію про чат, звідки надійшло повідомлення;

– сам текст повідомлення (обрізаний до 1500 символів для уникнення надто довгих логів).

Сформований звіт надсилається як звичайне текстове повідомлення у лог-чат за допомогою методу context.bot.send_message(). Використання спеціальних режимів форматування (наприклад, Markdown) було виключено для спрощення та уникнення потенційних помилок з екрануванням символів.

5. Обробка помилок. Передбачено обробку можливого RuntimeError від модуля spam_detector_module (наприклад, якщо виникла проблема під час аналізу). У такому випадку в лог-чат надсилається повідомлення про помилку аналізу. Також реалізовано загальний блок try-except для відлову інших непередбачених помилок під час обробки повідомлення або надсилання звіту, з відповідним логуванням у консоль.

На рисунку 3.3 графічно представлено пайплайн роботи бота і обробки повідомлень від користувачів.

Рисунок 3.3 – Діаграма активностей демонстраційного застосунку

3.4 Опис інтерфейсу користувача та перевірка працездатності системи

З точки зору кінцевого користувача в чаті, де відбувається моніторинг, бот є "невидимим". Він не реагує на команди та не надсилає повідомлень у цей чат. Його єдина функція – аналіз вхідних текстових повідомлень.

Інтерфейс взаємодії існує лише для адміністратора/розробника системи через лог-чат:

– вхідні дані для бота: будь-яке текстове повідомлення в моніторинговому чаті;

– вихідні дані (у лог-чаті): повідомлення-звіт, що містить результат класифікації та метадані оригінального повідомлення. Приклад формату звіту наведено на рисунку 3.4:

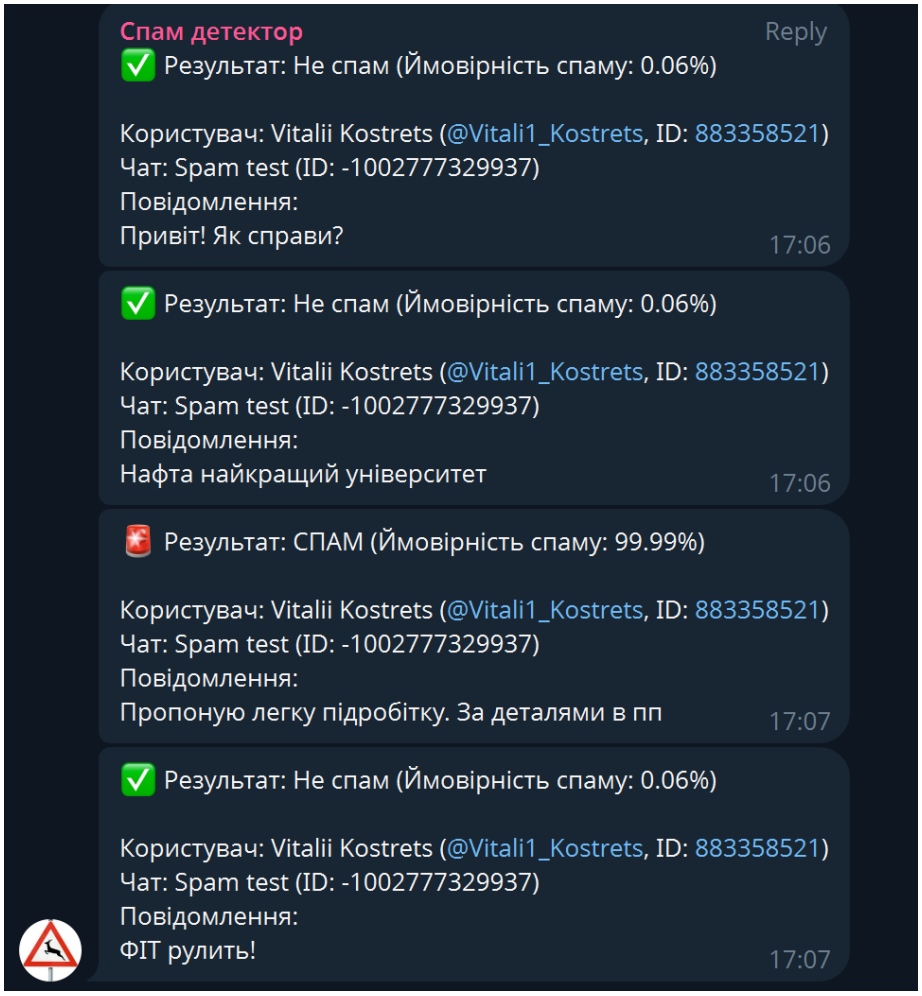


Рисунок 3.4 – Скріншот звітів з лог-чату

Запуск бота здійснюється шляхом виконання скрипта bot.py (повний код наведено в додатку В). Бот працює в режимі постійного опитування (polling) API платформи обміну повідомленнями, очікуючи на нові повідомлення. Для зупинки роботи бота необхідно перервати виконання скрипта (наприклад, відправити Enter у консолі).

Реалізація такого демонстраційного застосунку дозволяє ефективно перевіряти роботу розробленого модуля spam_detector_module на різноманітних реальних даних та слугує наочним прикладом його інтеграції в системи обміну повідомленнями.

Для об'єктивної оцінки ефективності розробленого програмного модуля spam_detector_module було проведено його всебічне тестування на незалежному наборі даних. Метою тестування було визначення якості класифікації текстових повідомлень на "спам" та "не-спам" в умовах, наближених до реального використання, а також аналіз типових помилок моделі.

Основною метою тестування було кількісне та якісне оцінювання здатності розробленого модуля spam_detector_module коректно ідентифікувати спам в українськомовних текстових повідомленнях.

Для досягнення поставленої мети було визначено наступні завдання:

1. Сформувати тестовий набір даних, що складається з реальних текстових повідомлень, зібраних з відкритих джерел (месенджерів), та провести їх ручну експертну розмітку на класи "спам" та "не-спам". Цей набір даних не повинен перетинатися з даними, використаними для навчання та валідації моделі на етапі розробки.

2. Розрахувати ключові метрики якості бінарної класифікації: точність (Accuracy), чіткість (Precision), повноту (Recall) та F1-міру (F1-score) для класу "спам".

3. Побудувати та проаналізувати матрицю помилок для візуалізації результатів класифікації та виявлення типів помилок (хибнопозитивні та хибнонегативні спрацювання).

4. Провести якісний аналіз випадків помилкової класифікації для виявлення потенційних слабких місць моделі та можливих шляхів її подальшого вдосконалення.

3.4.1 Розробка тестового набору даних

Для проведення тестування було зібрано та вручну розмічено набір зі 100 текстових повідомлень (повний перелік тестових повідомлень з еталонними мітками наведено у Додатку Г). Джерелом даних слугували публічні групи та канали в месенджері. Повідомлення відбиралися таким чином, щоб представити різноманітні типи контенту:

– спам-повідомлення (50 прикладів): Включали типову рекламу товарів та послуг (наприклад, повідомлення №1 "Робота для студентів – 7500 грн?/вечір", №26 "🔷 Шукаєш підробіток з нуля?"), пропозиції заробітку в інтернеті (№2 "Термінове завдання – 600$ /результат одразу", №7 "Працюй 3–5 годин на день та отримуй від 8 000 до 15 000 гривень..."), фінансові схеми та пропозиції, пов'язані з криптовалютою (№8 "ПУМБ/Райф Нові реєстрації/перекази", №30 "Купівля USDT з безпечною угодою..."), повідомлення з підозрілими посиланнями та закликами до дії (№53 "🎁 Виграй айфон! Просто натисни на це посилання..."), а також інші види небажаного контенту, характерні для українського сегменту (№28 "Запрошуємо до роботи owner-операторів...", №60 "💊 Купи ліки без рецепта...");

– не-спам повідомлення (50 прикладів): Включали звичайні розмовні повідомлення (№11 "От і договорились", №81 "Привіт! Як справи?", №100 "Гарного вечора!"), питання та відповіді в тематичних групах (№31 "На якій вулиці ТЦК знаходиться...", №40 "До якого числа перездачі?", №46 "Де в ІФ можна зшити бакалаврську роботу?"), оголошення нерекламного характеру (№38 "Чи є хтось зараз на калиновій слободі?"), а також інші приклади легітимного спілкування без комерційного чи шахрайського підтексту.

Кожне повідомлення було ретельно проаналізовано та однозначно

віднесено до одного з двох класів: "спам" (мітка 1) або "не-спам" (мітка 0), згідно з наданим переліком. Розмітка здійснювалася вручну. Зібрані дані (текст повідомлення та еталонна мітка) були збережені у таблиці test_messages бази даних SQLite для подальшої автоматизованої обробки. Така кількість та структура тестового набору (50% спаму та 50% не-спаму) дозволяє отримати статистично значущі результати та оцінити узагальнюючу здатність моделі на нових, "небачених" нею даних.

3.4.2 Методика проведення тестування та критерії оцінки

Тестування модуля spam_detector_module було автоматизовано за допомогою спеціально розробленого Python-скрипта evaluate_model.py (код наведено у Додатку Д). Скрипт виконує наступні кроки:

– завантажує тестові дані (текст повідомлення та еталонну мітку) з бази даних test_data.db з таблиці test_messages.

– послідовно передає кожне текстове повідомлення на вхід функції predict() модуля spam_detector_module. При цьому використовувався поріг (threshold) для прийняття рішення, встановлений за замовчуванням у модулі (0.5).

– зберігає отриману від модуля прогнозну мітку (0 або 1).

Після обробки всіх тестових повідомлень, скрипт порівнює список еталонних міток зі списком прогнозних міток та обчислює метрики якості класифікації за допомогою функцій з бібліотеки scikit-learn.

Для оцінки якості роботи класифікатора було використано наступні стандартні метрики для задач бінарної класифікації:

– матриця помилок (Confusion Matrix): Таблиця, що відображає кількість правильно та неправильно класифікованих прикладів для кожного класу. Вона містить наступні значення:

– True Positives (TP): Кількість спам-повідомлень, правильно класифікованих як спам;

– True Negatives (TN): Кількість не-спам повідомлень, правильно класифікованих як не-спам;

– False Positives (FP): Кількість не-спам повідомлень, помилково класифікованих як спам (помилка I роду);

– False Negatives (FN): Кількість спам-повідомлень, помилково класифікованих як не-спам (помилка II роду);

– точність (Accuracy): Загальна частка правильно класифікованих повідомлень. Розраховується як (TP+TN)/(TP+TN+FP+FN);

– чіткість (Precision) для класу "Спам": Частка повідомлень, які дійсно є спамом, серед усіх, що система позначила як спам. Розраховується як TP/(TP+FP). Високе значення цієї метрики важливе для мінімізації блокування легітимних повідомлень;

– повнота (Recall) для класу "Спам": Частка виявленого спаму серед усіх реальних спам-повідомлень. Розраховується як TP/(TP+FN). Високе значення цієї метрики важливе для максимального охоплення та виявлення спаму;

– F1-міра (F1-score) для класу "Спам": Гармонійне середнє між чіткістю та повнотою, що дозволяє збалансовано оцінити якість роботи класифікатора, особливо якщо важливі обидві метрики. Розраховується як

$$F1 = 2 \cdot (Precision \cdot Recall)/(Precision + Recall).$$

3.4.3 Результати тестування та їх аналіз

Після запуску скрипта evaluate_model.py на підготовленому тестовому наборі зі 100 повідомлень було отримано наступні результати ефективності роботи програмного модуля spam_detector_module.

Матриця помилок (Confusion Matrix), що деталізує результати класифікації на тестовому наборі, представлена у таблиці 3.1.

Таблиця 3.1 – Матриця помилок на тестовому наборі даних

Реальний клас	Прогноз не-спам	Прогноз спам
Не-спам (0)	49 (TN)	1 (FP)
Спам(1)	0 (FN)	50 (TP)

Згідно з даними матриці помилок:

– True Negatives (TN) = 49: 49 легітимних повідомлень ("не-спам") були правильно ідентифіковані системою;

– False Positives (FP) = 1: 1 легітимне повідомлення було помилково класифіковане як "спам";

– False Negatives (FN) = 0: Жодне спам-повідомлення не було пропущено системою; всі спам-повідомлення були коректно виявлені;

– True Positives (TP) = 50: Всі 50 спам-повідомлень були правильно ідентифіковані як "спам".

На основі цих значень були розраховані ключові метрики якості класифікації, які наведено у Таблиці 3.2.

Таблиця 3.2 – Розраховані метрики якості класифікації на тестовому наборі

Метрика	Значення	Опис
Точність (Accuracy)	0.9900	Загальна частка правильно класифікованих повідомлень
Чіткість (Precision) для класу "Спам"	0.9804	Частка істинно спамових серед усіх, що визначені як спам
Повнота (Recall) для класу "Спам"	1.0000	Частка виявленого спаму серед усіх фактично спамових повідомлень
F1-міра (F1-score) для класу "Спам"	0.9901	Збалансована оцінка, що враховує чіткість та повноту для класу "Спам"

Проведене тестування продемонструвало високу ефективність розробленого модуля spam_detector_module для виявлення спаму в українськомовних текстових повідомленнях на незалежному тестовому наборі.

Загальна Точність (Accuracy) класифікації склала 0.9900 (99%), що означає, що 99 зі 100 тестових повідомлень були класифіковані правильно. Це дуже високий показник, що свідчить про надійність системи в цілому.

Для класу "Спам" Чіткість (Precision) досягла значення 0.9804 (98.04%). Це означає, що серед 51 повідомлення, які система позначила як спам (50 TP + 1 FP), 50 дійсно були спамом. Це свідчить про низьку кількість хибнопозитивних спрацювань, тобто модель рідко помилково блокує легітимні повідомлення. У даному тестуванні було лише одне таке хибнопозитивне спрацювання. Повідомленням, що спричинило FP, було: "Хто сьогодні на зустрічі?". Дане повідомлення є коротким, розмовним і не містить жодних очевидних ознак спаму, таких як посилання, заклики до дії чи специфічні ключові слова. Ймовірно, модель помилково віднесла його до спаму через специфічну комбінацію токенів, яка могла рідко зустрічатися в навчальному корпусі у контексті "не-спам" повідомлень, або ж модель могла вловити неочевидні для людини патерни, які асоціюються зі спамом у її внутрішніх представленнях. Такі випадки підкреслюють складність розпізнавання коротких, контекстуально бідних повідомлень та вказують на потенційну область для подальшого донавчання моделі на більшій кількості подібних прикладів "не-спаму".

Повнота (Recall) для класу "Спам" склала 1.0000 (100%). Це ідеальний результат, який свідчить про те, що розроблений модуль успішно ідентифікував абсолютно всі 50 спам-повідомлень, які були присутні у тестовому наборі, і не пропустив жодного (FN=0). Висока повнота є критично важливою характеристикою для спам-фільтрів, оскільки головне завдання – максимально ефективно виявляти та блокувати небажаний контент.

F1-міра (F1-score) для класу "Спам", що є гармонійним середнім між чіткістю та повнотою, склала 0.9901 (99.01%). Таке високе значення підтверджує відмінний баланс між здатністю системи точно ідентифікувати спам та не пропускати його, що робить її ефективним інструментом для фільтрації.

Результати тестування на незалежному наборі даних підтверджують висновки, зроблені на етапі валідації моделі під час навчання (див. підрозділ 2.5), і свідчать про високу узагальнюючу здатність розробленої моделі BERT та ефективність запропонованого підходу до препроцесингу даних. Єдине хибнопозитивне

спрацювання на 100 тестових прикладах вказує на високу надійність системи при роботі з даними джерелом яких є українські чати та канали в популярних месенджерах.

Висновок до третього розділу

У третьому розділі бакалаврської роботи було виконано практичну реалізацію та експериментальне дослідження розробленої системи автоматизованого виявлення спаму. Було досягнуто наступних ключових результатів:

Розроблено кінцевий програмний продукт у вигляді автономного Python-модуля spam_detector_module. Цей модуль інкапсулює всю необхідну логіку для класифікації текстових повідомлень, включаючи попередню обробку тексту та інференс донавченої моделі BERT. Спроєктований інтерфейс модуля, що складається з єдиної функції predict(), забезпечує простоту його інтеграції в будь-які сторонні програмні системи.

Створено демонстраційний застосунок на основі платформи обміну миттєвими повідомленнями для наочної ілюстрації роботи розробленого модуля. Реалізований автоматизований агент (бот) здатен у фоновому режимі аналізувати повідомлення в чатах та надсилати детальні звіти про результати класифікації у спеціально відведений лог-чат, що підтверджує практичну застосовність та гнучкість розробленого рішення.

Проведено всебічне тестування ефективності системи на незалежному, вручну розміченому тестовому наборі даних зі 100 повідомлень (50 спам, 50 не-спам). Результати експериментального дослідження продемонстрували високу якість роботи класифікатора:

– загальна точність (Accuracy) склала 99%, що свідчить про високу надійність системи;

– повнота (Recall) для класу "спам" досягла ідеального показника 100%, що

вказує на здатність системи виявляти всі спам-повідомлення у тестовому наборі без пропусків;

– чіткість (Precision) для класу "спам" склала 98.04%, що свідчить про низький ризик хибного блокування легітимних повідомлень (лише одне хибнопозитивне спрацювання на 100 прикладів);

– збалансована F1-міра досягла 99.01%, що підтверджує високу ефективність класифікатора в цілому.

Таким чином, завдання, поставлені для третього розділу, були повністю виконані. Практична реалізація та експериментальне дослідження підтвердили, що розроблений програмний модуль є ефективним та надійним інструментом для автоматизованого виявлення спаму в українськомовних текстових повідомленнях, готовим до практичного застосування.

Якісний аналіз результатів тестування показав, що єдиною помилкою системи було хибнопозитивне спрацювання на короткому розмовному повідомленні ("Хто сьогодні на зустрічі?"). Це свідчить про високу чутливість моделі, але водночас вказує на потенційну вразливість при обробці контекстуально бідних текстів, де відсутні явні ознаки спаму чи "не-спаму". Ідеальний показник повноти (100%) є надзвичайно важливим практичним досягненням, оскільки підтверджує, що система не пропускає спам, що є пріоритетним завданням для будь-якого спам-фільтра.

З практичної точки зору, розроблений модуль продемонстрував високу швидкодію, що дозволяє його інтеграцію в системи, які працюють в режимі реального часу, наприклад, для модерації чатів. Водночас, варто відзначити, що ефективність системи напряму залежить від репрезентативності навчального корпусу. Нові, не бачені раніше тактики спамерів можуть потенційно знизити точність виявлення, що підкреслює важливість механізмів періодичного донавчання та оновлення моделі на актуальних даних для підтримки її ефективності в довгостроковій перспективі.

ВИСНОВКИ

У даній бакалаврській роботі було запропоновано вирішення актуальної науково-практичної задачі розробки та дослідження інформаційної системи автоматизованого виявлення спаму в українськомовних текстових повідомленнях. На основі проведених досліджень та виконаної роботи можна зробити наступні висновки:

1. Проведено детальний аналіз проблеми поширення спаму в сучасних цифрових комунікаційних середовищах, визначено його основні типи, канали розповсюдження та негативний вплив. Розглянуто існуючі традиційні та сучасні методи фільтрації спаму, виявлено їх переваги та недоліки. Обґрунтовано доцільність використання трансформерної архітектури BERT (bert-base-multilingual-cased) для ефективної класифікації українськомовного текстового контенту, враховуючи її здатність до глибокого розуміння контексту та адаптації до специфіки мови. Сформульовано постановку задачі дослідження, що включала розробку спеціалізованого корпусу даних, алгоритму препроцесингу, навчання моделі та створення програмного модуля.

2. Розроблено методику формування та підготовки навчального корпусу даних, що складається з близько 40 000 текстових повідомлень. Ця методика включала збір даних з відкритих джерел, їх автоматичну первинну розмітку, ретельну ручну верифікацію, а також інноваційні підходи до збагачення корпусу шляхом перекладу іншомовних прикладів спаму та генерації синтетичних даних за допомогою ChatGPT API. Створено збалансований корпус, що забезпечило високу якість та репрезентативність даних для навчання моделі.

3. Спроєктовано та реалізовано ефективний алгоритм препроцесингу текстових повідомлень, адаптований до особливостей української мови та характерних ознак спаму. Алгоритм включає заміну URL-адрес, електронних поштових скриньок, телефонних номерів, юзернеймів, хештегів та символів валют на спеціалізовані токени, а також нормалізацію Unicode та видалення

нерелевантних символів. Це дозволило уніфікувати вхідні дані для моделі BERT та підвищити якість її роботи.

4. Здійснено донавчання моделі bert-base-multilingual-cased на підготовленому корпусі для задачі бінарної класифікації спаму. Підібрано оптимальні гіперпараметри навчання (кількість епох, розмір батчу, швидкість навчання). Навчена модель продемонструвала високі показники ефективності на валідаційній вибірці, досягнувши точності та F1-міри близько 98%, що підтвердило успішність обраного підходу.

5. Розроблено кінцевий програмний продукт – автономний Python-модуль spam_detector_module. Модуль інкапсулює логіку препроцесингу та навчену модель BERT, надаючи простий інтерфейс (функцію predict()) для класифікації вхідного тексту на "спам" або "не-спам" та отримання ймовірнісної оцінки. Реалізовано механізм "лінивого завантаження" моделі для оптимізації продуктивності.

6. Для демонстрації практичної застосовності розробленого модуля було створено програмний застосунок у вигляді автоматизованого агента (бота) для платформи обміну миттєвими повідомленнями. Застосунок успішно інтегрується з модулем spam_detector_module, аналізує повідомлення в реальному часі та надсилає звіти про класифікацію у спеціальний лог-чат.

7. Проведено експериментальне тестування розробленого програмного модуля spam_detector_module на незалежному, вручну розміченому наборі зі 100 українськомовних текстових повідомлень (50 спам, 50 не-спам). Результати тестування підтвердили високу ефективність системи: загальна точність (Accuracy) склала 99%, повнота (Recall) для класу "спам" – 100%, чіткість (Precision) – 98.04%, та F1-міра – 99.01%. Виявлено лише одне хибнопозитивне спрацювання, що свідчить про високу надійність системи.

Отримані в ході експериментального тестування метрики є показовими. Ідеальний показник повноти (Recall = 100%) свідчить, що система продемонструвала здатність ідентифікувати всі спам-повідомлення у тестовому

наборі, що є критично важливим для практичного застосування, оскільки мінімізує ризик пропуску небажаного або шкідливого контенту. Водночас висока чіткість (Precision = 98.04%) при єдиному хибнопозитивному спрацюванні підкреслює надійність системи та низьку ймовірність помилкового блокування легітимних повідомлень. Аналіз цього єдиного випадку помилки на короткому розмовному повідомленні вказує на складність класифікації контекстуально бідних текстів і визначає напрямок для подальшого покращення шляхом збагачення навчального корпусу подібними прикладами.

Ключовим внеском даної роботи є не лише досягнення високих метрик якості, а й розробка комплексного підходу, адаптованого до специфіки українськомовного інформаційного простору.

Практична значущість отриманих результатів полягає у створенні готового до використання програмного модуля, який може бути легко інтегрований в різноманітні інформаційні системи (месенджери, веб-сайти, поштові клієнти) для ефективної фільтрації спаму в українськомовному сегменті Інтернету. Розроблений підхід до формування корпусу та препроцесингу даних може бути використаний для вирішення аналогічних задач обробки природної мови.

Напрямки подальших досліджень можуть включати: розширення навчального корпусу для охоплення нових видів спаму та складних випадків; експерименти з іншими трансформерними архітектурами або методами оптимізації моделі (наприклад, квантизація, дистиляція) для підвищення швидкодії без суттєвої втрати точності; розробку механізмів адаптивного донавчання моделі на основі зворотного зв'язку від користувачів; дослідження можливості багатокласової класифікації спаму за його типами.

Таким чином, дана бакалаврська робота поєднує фундаментальні дослідження в області обробки природної мови з практичною реалізацією сучасних методів машинного навчання, що забезпечують надійну та ефективну детекцію спаму в різноманітних каналах цифрового спілкування українською мовою.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Share of spam messages in total e-mail traffic worldwide from 2014 to 2023 / Statista, 2024. URL: https://www.statista.com/statistics/420391/spam-email-traffic-share/ (дата звернення: 6.03.2025)

2. Al-momani A., Al-Khasawneh A., Al-kasasbeh M. A Comprehensive Survey on SMS Spam Filtering Techniques. *Journal of Computer Science,* 2016. Vol. 12. no. 1. P. 14–22.

3. Moore T., Clayton R., Anderson R. The Economics of Online Crime. *Journal of Economic Perspectives*. 2009. Vol. 23, no. 3. P. 3–20.

4. Graham P. A Plan for Spam. August 2002. URL: http://www.paulgraham.com/spam.html (дата звернення: 8.03.2025).

5. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Minneapolis, Minnesota, USA, June 2-7, 2019. Stroudsburg, PA : Association for Computational Linguistics, 2019. P. 4171–4186.

6. Про захист персональних даних : Закон України від 01.06.2010 № 2297-VI. URL: https://zakon.rada.gov.ua/laws/show/2297-17 (дата звернення: 13.04.2025).

7. Політика конфіденційності Telegram / Telegram, 2025. URL: https://telegram.org/privacy/ua (дата звернення: 14.04.2025).

8. Seth A., Dahiya N., Sangwan S. Comparative Analysis of Machine Learning and Deep Learning Models for Spam Detection. *2022 6th International Conference on Computing Methodologies and Communication* (ICCMC) : Proceedings. Erode, India, March 29-31, 2022. Piscataway, NJ : IEEE, 2022. P. 1369–1375.

9. Jain G., Sharma M., Agrawal B. Spam Detection on Social Media using BERT and Deep Learning Models. *2021 11th International Conference on Cloud Computing,*

Data Science & Engineering (Confluence) : Proceedings. Noida, India, January 28-29, 2021. Piscataway, NJ : IEEE, 2021. P. 783–787.

10. OpenAI. GPT-4 Technical Report / OpenAI. March 2023. (arXiv:2303.08774). URL: https://arxiv.org/abs/2303.08774 (дата звернення: 18.04.2025).

11. Python-telegram-bot: We have a bot for that / The python-telegram-bot Team, 2024. URL: https://python-telegram-bot.org/ (дата звернення: 17.05.2025).