Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Реалізація та порівняльний аналіз ефективності агентних інтелектуальних систем для автоматизованої обробки клієнтських звернень
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-22-2
Даніїл БЄЛЯЄВ
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)

Керівник ст. викл. Олександр СТЬОПІН
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Лариса ЧАЛА
(підпис) (власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Бєляєву Даніїлу Максимовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Реалізація та порівняльний аналіз ефективності агентнихінтелектуальних систем для автоматизованої обробки клієнтських звернень

затверджена наказом університету від 5 червня 2026 р. № 488Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 9 липня 2026 р.

3. Вихідні дані до роботи апаратна платформа – робоча станція з GPU AMD Radeon RX 9070 XT (16 ГБ VRAM); програмне середовище – n8n та Ollama для локального запуску моделей; набір з чотирьох відкритих великих мовних моделей (Gemma 3 12B, Qwen 3 8B, Llama 3.1 8B, Mistral 7B); тестова вибірка україномовних email-звернень та еталонна таблиця очікуваних значень.

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі та постановка задачі

2) Теоретичні дослідженя

3) Проєктування системи та її розробка

4) Експериментальне дослідження

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	05.06.2026	виконано
2	Аналіз предметної галузі	11.06.2026	виконано
3	Огляд існуючих метрик оцінювання якості	13.06.2026	виконано
4	Аналіз підходів вирішення задачі детекції	16.06.2026	виконано
5	Експериментальне моделювання та навчання	21.06.2026	виконано
	моделі		
6	Написання пояснювальної записки	23.06.2026	виконано
7	Перевірка на академічний плагіат	24.06.2026	виконано
8	Нормоконтроль	30.06.2026	виконано
9	Підготовка презентації та доповіді	01.07.2026	виконано
10	Попередній захист	03.07.2026	виконано
11	Рецензування	04.07.2026	виконано
12	Захист перед ЕК	09.07.2026	

Дата видачі завдання 5 червня 2026 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. Олександр СТЬОПІН
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 93 с., 13 рис., 14 табл., 4 дод., 42 джерела.

АГЕНТНА СИСТЕМА, ВЕЛИКА МОВНА МОДЕЛЬ, ЕКСТРАКЦІЯ ДАНИХ, ІНТЕРНЕТ-ПРОВАЙДЕР, КВАНТИЗАЦІЯ, КЛАСИФІКАЦІЯ ТЕКСТУ, ЛОКАЛЬНЕ РОЗГОРТАННЯ, ОБРОБКА ПРИРОДНОЇ МОВИ, N8N, OLLAMA.

Об’єкт дослідження – процес автоматизованої обробки клієнтських звернень інтернет-провайдера.

Предмет дослідження – агентні інтелектуальні системи на основі локальних великих мовних моделей для класифікації звернень та екстракції структурованих даних.

Мета роботи – реалізація та порівняльний аналіз ефективності чотирьох локальних LLM-агентів за критеріями точності класифікації, якості екстракції даних та швидкості обробки.

Методи дослідження – теоретичний, аналіз архітектури великих мовних моделей, токенізації та квантизації, та експериментальний, програмна реалізація агентного пайплайну та статистична обробка метрик. Розробку виконано на платформі n8n із середовищем Ollama.

У результаті роботи спроєктовано та реалізовано пайплайн обробки електронних звернень, який виконує класифікацію за шістьма категоріями, визначення тональності й пріоритету та екстракцію структурованих даних із записом у Google Sheets. Проведено серію експериментів із моделями Gemma 3 12B, Qwen 3 8B, Llama 3.1 8B та Mistral 7B на наборі україномовних звернень з еталонною таблицею. За результатами порівняння визначено модель з найкращим співвідношенням точності й продуктивності та сформульовано рекомендації щодо її вибору для практичного впровадження.

ABSTRACT

Bachelor’s thesis contains: 93 pp., 13 fig., 14 tabl., 4 ann., 42 references.

AGENT-BASED SYSTEM, DATA EXTRACTION, INTERNET SERVICE PROVIDER, LARGE LANGUAGE MODEL, LOCAL DEPLOYMENT, N8N, NATURAL LANGUAGE PROCESSING, OLLAMA, QUANTIZATION, TEXT CLASSIFICATION.

The object of research is the process of automated handling of internet service provider customer requests.

The subject of research is agent-based intelligent systems built on local large language models for the classification of requests and the extraction of structured data.

The purpose of the work is the implementation and comparative analysis of the efficiency of four local LLM agents in terms of classification accuracy, data extraction quality and processing speed.

Research methods – theoretical, analysis of large language model architecture, tokenization and quantization, and experimental, software implementation of the agent pipeline and statistical processing of metrics. The development was carried out on the n8n platform with the Ollama runtime.

As a result of the work, a pipeline for email request processing was designed and implemented, performing classification into six categories, sentiment and priority detection, and extraction of structured data written to Google Sheets. A series of experiments was conducted with the Gemma 3 12B, Qwen 3 8B, Llama 3.1 8B and Mistral 7B models on a set of Ukrainian-language requests with a reference table. Based on the comparison, the model with the best ratio of accuracy to performance was identified and recommendations on its selection for practical deployment were formulated.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 8
Вступ 9
1 Аналіз предметної галузі та постановка задачі 13
1.1 Характеристика ринку інтернет-провайдерів України та процесу обробки клієнтських звернень 13
1.2 Аналіз проблем та викликів автоматизації обробки звернень 17
1.3 Огляд існуючих рішень 20
1.4 Аналіз концепції агентних інтелектуальних систем на базі LLM 30
1.5 Обґрунтування інноваційності та переваг розроблюваної системи .. 33
1.6 Постановка задачі 34
2 Теоретичні дослідження 38
2.1 Архітектура Transformer як основа сучасних великих мовних моделей 38
2.2 Токенізація та її вплив на обробку українського тексту 41
2.3 Детальна характеристика обраних моделей 43
2.3.1 Gemma 3 12B (Google deepmind) 43
2.3.2 Qwen 3 8B (Alibaba Cloud) 44
2.3.3 Llama 3.1 8B (Meta) 45
2.3.4 Mistral 7B (Mistral AI) 46
2.4 Платформа n8n та її AI-можливості 47
2.5 Ollama як середовище локального запуску LLM 48
2.6 Квантизація моделей: формат GGUF та вплив на якість 49
2.7 Промпт-інжиніринг для структурованого виводу 51
2.8 Метрики оцінки якості класифікації та екстракції даних 52
2.9 Стандартні бенчмарки LLM та їх обмеження 54
3 Проєктування системи та її розробка 56
3.1 Проєктування загальної архітектури системи 56
3.2 Структура робочого процесу n8n 57

3.3 Розробка системного промпту для AI-агента 59
3.4 Конфігурація Ollama та GPU-прискорення 61
3.5 Обробка відповіді моделі – Code node 62
3.6 Інтеграція з Google Sheets 62
3.7 Розробка тестового набору даних 63
3.8 Обмеження та проєктні рішення 64
4 Експериментальні дослідження 64
4.1 Методика проведення експерименту 67
4.2 Результати порівняльного аналізу 67
4.3 Аналіз результатів класифікації 68
4.4 Аналіз результатів екстракції даних 68
4.5 Аналіз швидкості обробки 70
4.6 Аналіз типових помилок та обмежень 71
4.7 Інтерпретація виявлених кореляцій 74
4.8 Рекомендації щодо вибору моделі 76
4.9 Перспективи розвитку 77
Висновки 78
Перелік джерел посилання 78
Додаток А Приклади Тестових email-звернень 80
Додаток Б Системний промт AI-агента 88
Додаток В Ефективність токенізації великих мовних моделей для української мови 92
Додаток Г Відомість кваліфікаційної роботи 93

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AHT – Average Handle Time – середній час обробки звернення;

AI – Artificial Intelligence – штучний інтелект;

API – Application Programming Interface – програмний інтерфейс застосунку;

CRM – Customer Relationship Management – система управління взаємовідносинами з клієнтами;

CSAT – Customer Satisfaction Score – оцінка задоволеності клієнта;

GGUF – GPT-Generated Unified Format – формат зберігання квантизованих моделей;

GPU – Graphics Processing Unit – графічний процесор;

GQA – Grouped Query Attention – згрупований механізм уваги за запитами;

ISP – Internet Service Provider – інтернет-провайдер;

JSON – JavaScript Object Notation – текстовий формат обміну даними;

LLM – Large Language Model – велика мовна модель;

ML – Machine Learning – машинне навчання;

NER – Named Entity Recognition – розпізнавання іменованих сутностей;

NLP – Natural Language Processing – обробка природної мови;

RAG – Retrieval-Augmented Generation – генерація з доповненням пошуком;

REST – Representational State Transfer – архітектурний стиль веб-API;

SVM – Support Vector Machine – метод опорних векторів;

TF-IDF – Term Frequency – Inverse Document Frequency – статистична міра важливості слова;

VRAM – Video Random Access Memory – відеопам'ять графічного процесора.

ВСТУП

Сучасний ринок телекомунікаційних послуг в Україні характеризується високим рівнем конкуренції та значною кількістю учасників. За даними дашборду Офісу ефективного регулювання (BRDO), побудованого на основі реєстру Національної комісії, що здійснює державне регулювання у сфері електронних комунікацій (НКЕК), станом на кінець 2024 року в Україні зареєстровано понад 4100 провайдерів телекомунікаційних послуг, які обслуговують приблизно 8,5 мільйонів абонентів фіксованого широкосмугового доступу до мережі Інтернет [1]. При цьому 63% зареєстрованих провайдерів є фізичними особами-підприємцями (ФОП) [1], що свідчить про високу фрагментованість ринку та підкреслює потребу у доступних і масштабованих інструментах автоматизації, зокрема для малих та середніх операторів.

Така структура ринку, де переважають невеликі оператори з обмеженими кадровими та фінансовими ресурсами, додатково ускладнює організацію якісного клієнтського сервісу, адже далеко не кожен провайдер може дозволити собі утримувати велику цілодобову службу підтримки.

Щоденно служби підтримки цих провайдерів отримують значну кількість звернень від клієнтів, що надходять різними каналами комунікації: електронною поштою, телефоном, через чат-боти на веб-сайтах та в месенджерах. Електронна пошта залишається одним із трьох найбільш використовуваних каналів зв'язку зі службою підтримки (23% споживачів обирають саме email), поступаючись лише live-чату (41%) та телефону. Перевага email-каналу полягає у можливості детального опису проблеми, прикріплення документів та збереження документального сліду, що є критичним для формальних запитів та скарг [37].

Обробка клієнтських email-звернень є трудомістким процесом, що вимагає від оператора уважного прочитання листа, визначення категорії звернення, витягнення персональних даних клієнта (прізвище, номер

телефону, номер договору, адресу підключення), оцінки терміновості та формулювання відповіді. Середній час обробки одного звернення (Average Handle Time, AHT), для email-каналу за галузевими оцінками складає від 4 до 7 хвилин. За великих обсягів вхідної кореспонденції ручна обробка стає вузьким місцем у роботі служби підтримки: збільшується час реагування, зростає ймовірність помилкової класифікації, а рутинний характер роботи призводить до професійного вигорання операторів, що, за галузевими дослідженнями, охоплює більшість агентів служб підтримки [39].

Разом із тим стрімкий розвиток технологій штучного інтелекту, зокрема великих мовних моделей (Large Language Models, LLM), відкриває нові можливості для автоматизації обробки природномовних текстів. Моделі, такі як Llama 3.1 [15], Gemma 3 [12], Qwen 3 [14] та Mistral 7B [17], здатні розуміти контекст повідомлення, класифікувати його за тематикою, витягувати структуровані дані з неструктурованого тексту та оцінювати емоційну тональність.

Важливою перевагою сучасних відкритих LLM є можливість їх локального розгортання за допомогою інструментів, таких як Ollama [2], без передачі конфіденційних даних клієнтів на зовнішні сервери, що важливо для збереження конфіденційності клієнтської інформації.

Особливий інтерес становить агентний підхід до використання LLM, при якому модель функціонує не як простий генератор тексту, а як інтелектуальний агент, здатний виконувати складні комплексні задачі з використанням інструментів, дотримуючись заданих інструкцій та формуючи структурований вивід [19].

Поєднання агентного підходу з платформою автоматизації робочих процесів n8n [7] дозволяє побудувати повноцінний конвеєр обробки email-звернень без необхідності написання складного програмного коду, що робить рішення доступним для системних адміністраторів ISP.

Однак на сьогоднішній день бракує досліджень, які б системно порівнювали ефективність різних локальних LLM у ролі агентів для задачі

обробки клієнтських звернень українською мовою. Існуючі бенчмарки, такі як MMLU [28], HellaSwag [29] та ARC [30], оцінюють загальні знання та логічні здібності моделей переважно англійською мовою, однак не відображають їх здатність виконувати прикладні задачі комплексної обробки email-звернень у конкретному доменному контексті. Крім того, моделі з англоцентричними токенізаторами витрачають у 2–3 рази більше токенів на одне українське слово порівняно з англійським, що безпосередньо впливає на швидкість обробки та якість генерації [26].

Метою даної кваліфікаційної роботи є порівняльний аналіз ефективності використання агентних інтелектуальних систем на базі чотирьох локальних великих мовних моделей для автоматизованої обробки текстового вмісту клієнтських звернень, типових для сфери інтернет-провайдингу, що надходять електронною поштою.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

– провести аналіз предметної галузі обробки клієнтських звернень інтернет-провайдера та дослідити існуючі підходи до автоматизації цього процесу;

– дослідити теоретичні основи роботи великих мовних моделей, агентних систем та методів промпт-інжинірингу для отримання структурованого виводу;

– обрати та обґрунтувати набір моделей для порівняння, що представляють різні архітектурні підходи та організації-розробники;

– спроєктувати та реалізувати конвеєр автоматизованої обробки email-звернень із використанням платформи візуальної автоматизації робочих процесів та середовища локального запуску великих мовних моделей;

– розробити тестовий набір email-звернень та еталонну таблицю для об'єктивної верифікації результатів;

– провести серію експериментів та виконати порівняльний аналіз ефективності обраних моделей за метриками точності, повноти та швидкості обробки.

Як інструментальну основу реалізації обрано платформу візуальної автоматизації робочих процесів n8n та середовище локального запуску моделей Ollama, розгорнуті на робочій станції з відеокартою AMD Radeon RX 9070 XT (16 ГБ VRAM, Vulkan API).

Об'єктом дослідження є процес автоматизованої обробки текстового контенту, що імітує клієнтські звернення до інтернет-провайдера, які надходять засобами електронної пошти.

Предметом дослідження є агентні інтелектуальні системи на базі великих мовних моделей для класифікації, екстракції даних та аналізу тональності клієнтських звернень.

Практичне значення роботи полягає у створенні тестового прототипу системи автоматизованої первинної обробки email-звернень, який надалі може бути адаптований для будь-якого українського інтернет-провайдера. Результати порівняльного аналізу моделей дозволяють обґрунтовано обрати оптимальну модель залежно від пріоритетів організації: максимальна точність, мінімальний час обробки або баланс між цими показниками. Розроблений конвеєр у n8n може бути відтворений за мінімальних технічних зусиль, оскільки не потребує написання складного програмного коду.

Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, переліку джерел посилання та додатків. У першому розділі проаналізовано предметну галузь, систематизовано проблеми автоматизації обробки звернень та сформульовано постановку задачі. Другий розділ присвячено теоретичним основам великих мовних моделей і характеристиці чотирьох обраних моделей. У третьому розділі описано проєктування та реалізацію конвеєра обробки звернень у n8n з інтеграцією Ollama. Четвертий розділ містить методику, результати експериментального дослідження та порівняльний аналіз моделей. У висновках узагальнено основні результати роботи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Характеристика ринку інтернет-провайдерів України та процесу обробки клієнтських звернень

Інтернет-провайдер (Internet Service Provider, ISP) – це суб'єкт господарювання, що надає послуги доступу до мережі Інтернет, а також супутні телекомунікаційні послуги: IP-телебачення, VoIP-телефонію, хостинг, захист від DDoS-атак тощо. Правовою основою діяльності ISP в Україні є Закон «Про електронні комунікації» від 16.12.2020 № 1089-IX, що визначає порядок реєстрації операторів та провайдерів у реєстрі НКРЗІ (з 2022 року – НКЕК) [36].

Ринок фіксованого інтернет-доступу в Україні є одним із найбільш фрагментованих у Європі. За даними дашборду BRDO, побудованого на основі реєстру НКЕК, станом на листопад 2024 року на ринку діють понад 4100 провайдерів, з яких 63% – фізичні особи-підприємці (ФОП), товариства з обмеженою відповідальністю – 28%, решта – акціонерні товариства та державні підприємства [1] (рисунок 1.1).

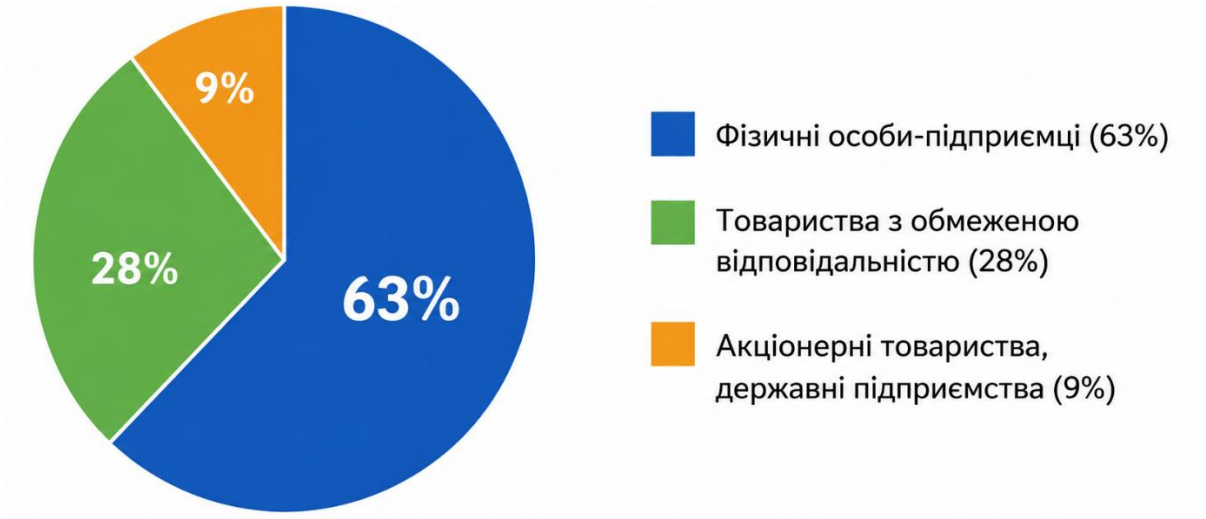


Рисунок 1.1 – Структура ринку ISP України за формою власності провайдерів

Загальна кількість абонентів фіксованого широкосмугового доступу оцінюється у 8,5 мільйонів, при цьому 10 найбільших провайдерів (Укртелеком, Київстар, Воля, Датагруп-Volia, Ланет, Тенет та ін.) сукупно генерують дохід 6,2 млрд грн із загального обсягу ринку 22,4 млрд грн, тобто лише 28% . Це означає, що 72% доходу розподіляється серед тисяч малих та середніх операторів, які, як правило, не мають ресурсів для впровадження дорогих корпоративних систем підтримки клієнтів.

Саме така фрагментованість ринку формує специфічний запит на технології автоматизації. Якщо великі оператори можуть дозволити собі дорогі корпоративні CRM-системи з вбудованими модулями штучного інтелекту, то для тисяч малих і середніх провайдерів подібні рішення залишаються фінансово недоступними.

Швидкість фіксованого інтернет-доступу в Україні у 2024 році становила в середньому 80,2 Мбіт/с, що відповідає 70-му місцю у світовому рейтингу Speedtest Global Index [38]. Попри воєнний стан, ринок демонструє стійкість: кількість абонентів залишається стабільною, а конкуренція між провайдерами зберігається на високому рівні.

Діяльність кожного ISP нерозривно пов'язана із взаємодією з клієнтами протягом усього життєвого циклу надання послуги: від моменту підключення до можливого розірвання договору. Клієнти звертаються до провайдера з різноманітних причин. На основі аналізу типових категорій звернень у системах підтримки телекомунікаційних компаній можна виділити наступні основні групи:

– технічна підтримка – повідомлення про відсутність доступу до Інтернету, низьку швидкість з'єднання, нестабільну роботу обладнання (роутер, ONT, медіаконвертер), проблеми з Wi-Fi, необхідність перезавантаження порту на комутаторі, налаштування мережевого обладнання. За індустріальними оцінками, технічні звернення складають від 40% до 55% усього потоку;

– фінансові питання (бухгалтерія) – уточнення стану рахунку, запити щодо нарахувань та заборгованості, зміни способу оплати, повернення коштів, пояснення деталізації. Частка – 15–20%;

– зміна тарифного плану – запити на перехід на інший тариф, підключення додаткових послуг (IP-TV, статична IP-адреса, антивірус), збільшення або зменшення швидкості доступу;

– підключення та відключення – заявки на підключення нових абонентів, перенесення точки підключення на іншу адресу, тимчасове призупинення або повне відключення послуги, відновлення після призупинення;

– скарги – незадоволення якістю обслуговування, скарги на роботу технічних спеціалістів, претензії щодо невиконання умов договору, вимоги компенсації за тривалу відсутність послуги;

– спам та нерелевантні листи – рекламні розсилки, автоматичні сповіщення платіжних систем, випадкові листи, що не потребують обробки оператором.

Орієнтовний розподіл цих категорій за часткою у загальному потоці звернень наведено на рисунку 1.2.

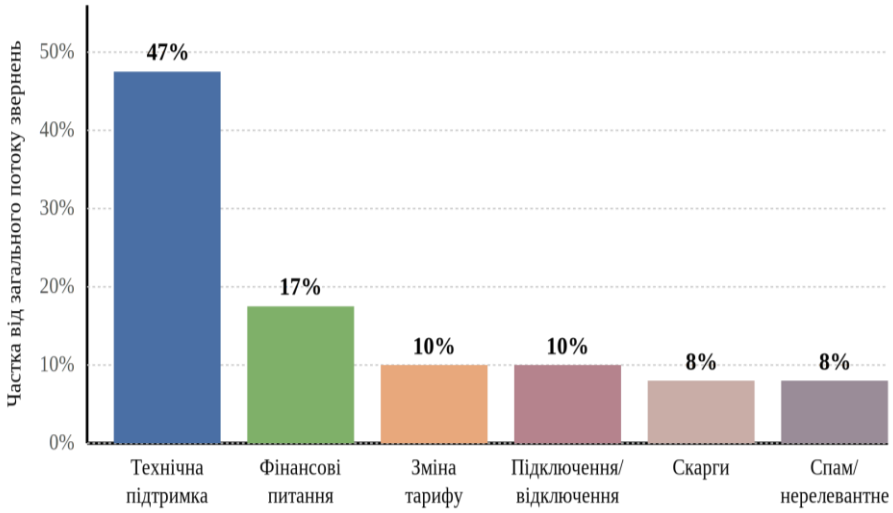


Рисунок 1.2 – Орієнтовний розподіл категорій клієнтських звернень у службі підтримки інтернет-провайдера (створено за допомогою ШІ)

Типовий процес ручної обробки email-звернення оператором служби підтримки включає наступні послідовні етапи: прочитання листа та ідентифікація клієнта за наявними даними (ім'я, email, номер телефону, номер договору); визначення категорії звернення відповідно до внутрішніх регламентів; оцінка пріоритетності (наприклад, масова аварія vs загальне питання); формування короткого резюме проблеми для внесення у CRM-систему; прийняття рішення про необхідність відповіді та, за потреби, підготовка відповіді або перенаправлення звернення до відповідного підрозділу (технічний відділ, фінансовий відділ, відділ продажів).

Кожен із цих етапів вимагає когнітивних зусиль та часу, особливо коли клієнт описує проблему нечітко, використовує неформальну лексику або поєднує декілька питань в одному листі. Метрика AHT (Average Handle Time) для email-каналу, розраховується як сума часу читання, класифікації, внесення даних у CRM та підготовки відповіді, поділена на кількість оброблених звернень. Індустріальний орієнтир «гарного» AHT для email складає близько 6 хвилин, хоча для складних технічних звернень цей показник може сягати 10–15 хвилин.

Для середнього провайдера з абонентською базою 50–100 тисяч клієнтів щоденний потік email-звернень може становити 100–300 листів, а у періоди масових аварій або змін тарифів – зростати у 3–5 разів. За AHT = 6 хвилин оператор здатний обробити ~80 звернень за 8-годинну зміну, що при обсязі 300 листів вимагає залучення мінімум 4 операторів лише на первинну обробку (класифікація та маршрутизація), без урахування підготовки відповідей.

Проблема ускладнюється високим рівнем вигорання операторів підтримки. 77% агентів підтримки повідомляють про зростання та ускладнення робочого навантаження, а 56% – про досвід емоційного вигорання у поточній ролі [39]. Окремі дослідження сегменту контактних центрів фіксують, що 63% агентів зазнають високого рівня вигорання, а 74% переживають вигорання протягом кар'єри [40]. Автоматизація рутинних

етапів обробки звернень (класифікація, екстракція даних, визначення пріоритетності) дозволяє зменшити когнітивне навантаження на операторів та вивільнити їх для виконання завдань, що потребують емпатії та нестандартного мислення.

1.2 Аналіз проблем та викликів автоматизації обробки звернень

При розробці системи автоматизованої обробки клієнтських email-звернень інтернет-провайдера виникає ряд суттєвих проблем та викликів, що ускладнюють побудову точного та надійного рішення. Далі наведено систематичний аналіз кожної з цих проблем.

Неструктурованість вхідних даних. На відміну від веб-форм із фіксованими полями, email-звернення являє собою довільний текст, у якому клієнт описує проблему у вільній формі. Персональні дані – прізвище, номер телефону, адреса підключення, номер договору – можуть бути розкидані по всьому тілу листа без будь-яких міток чи заголовків. Особливо складним є витягнення номера договору: в українських ISP він, як правило, являє собою просте 5–8-значне число, що стоїть у тексті між прізвищем та номером телефону без будь-якого пояснювального підпису на кшталт «Номер договору:» або «Контракт №». Система повинна «здогадатися» про семантичне значення числа виключно з контексту, що є нетривіальною задачею навіть для сучасних LLM.

Мовні особливості української мови. Клієнти українських ISP пишуть звернення українською мовою у вільній формі, часто з використанням розмовної лексики, скорочень, сленгу та граматичних помилок. Моделі з англоцентричними токенізаторами (наприклад, Llama) витрачають у 2–3 рази більше токенів на одне українське слово порівняно з англійським, що сповільнює обробку та звужує ефективне контекстне вікно [26]. Кількісний аналіз цього ефекту наведено у підрозділі 2.2. Додаткову складність створюють специфічні формати: українські адреси (вул. Сумська, 25, кв. 14),

прізвища з відмінюванням (від Петренка, Петренку), номери телефонів у різних форматах (+380, 0, 8-050, (050)).

Різноманітність тематик та тональностей. Звернення клієнтів охоплюють широкий спектр тем – від простих технічних питань до емоційних скарг та загроз розірвання договору. Один і той самий лист може містити ознаки декількох категорій одночасно: наприклад, клієнт скаржиться на якість Інтернету (технічна підтримка), вимагає перерахунок за дні без зв'язку (фінансове питання) та погрожує розірвати договір (скарга). Визначення основної категорії та пріоритетності в таких випадках вимагає глибокого розуміння контексту та семантичних зв'язків між частинами тексту.

Оманливість тем листів. Тема email-листа не завжди відповідає його справжньому змісту. Клієнт може вказати тему «Питання по тарифу», а в тілі листа описати технічну проблему з роутером. Або ж у відповідь на маркетинговий лист провайдера (з темою «Знижка 50% на новий тариф!») написати скаргу на якість обслуговування. Система повинна аналізувати саме тіло листа, а не покладатися на тему, що вимагає обробки всього тексту моделлю.

Конфіденційність персональних даних. Email-звернення містять персональні дані клієнтів: прізвища, адреси, номери телефонів, номери договорів. Передача таких даних на зовнішні хмарні сервіси для обробки (наприклад, через API OpenAI або Google) створює ризики їх витоку та втрати контролю над конфіденційною інформацією клієнтів. Тому збереження повного контролю над цими даними робить локальне розгортання моделей важливою вимогою для провайдерів.

Обмежені обчислювальні ресурси. Локальний запуск великих мовних моделей вимагає значних обчислювальних ресурсів, зокрема потужних GPU. Моделі з кількістю параметрів 7–8 мільярдів потребують щонайменше 8 ГБ VRAM, а моделі з 12–13 мільярдами параметрів – 10–16 ГБ VRAM [5]. Застосування методів квантизації, зокрема 4-бітної Q4_K_M

та 8-бітної Q8_0, дозволяє зменшити вимоги до пам'яті на 50–75% [32], однак може впливати на якість генерації: перехід від повної точності (FP16) до Q4_K_M призводить до збільшення perplexity на ~0,054, що є прийнятним для більшості прикладних задач [33].

Нестабільність формату виводу LLM. Великі мовні моделі за своєю природою є генеративними та імовірнісними системами [11]. При необхідності отримання структурованого JSON-виводу модель може додавати пояснювальний текст перед або після JSON-об'єкта наприклад, «Ось результат обробки:», пропускати обов'язкові поля, використовувати невірні типи даних або формувати синтаксично некоректний JSON. Без спеціальних технік промпт-інжинірингу до 15–20% відповідей моделей містять помилки у JSON-структурі [23].

Галюцинації та вигадування даних. Однією з фундаментальних проблем великих мовних моделей є схильність до генерації правдоподібної, але фактично невірної інформації – так званих «галюцинацій». Галюцинації у LLM виникають через невідповідність навчальних даних, обмеженість контексту та імовірнісну природу генерації [24]. У контексті обробки email-звернень це може проявлятися у вигадуванні номера договору, якого немає в тексті листа, або приписуванні клієнту адреси, яку він не вказував. Агентні системи на базі LLM додатково піддаються ефекту «інкрементальних галюцинацій», коли помилки накопичуються в ході багатокрокової взаємодії [25].

Зведений перелік виявлених проблем та їх потенційний вплив на систему автоматизації наведено в таблиці 1.1.

Таблиця 1.1 – Проблеми автоматизації обробки email-звернень та їх вплив

Проблема	Вплив на систему	Можливе рішення
Неструктурованість даних	Складність NER, пропуск полів	Детальний системний промпт з прикладами

Продовження таблиці 1.1

Проблема	Вплив на систему	Можливе рішення
Мовні особливості української	Зниження якості на 15–25%	Мультимовні моделі (Gemma 3, Qwen 3)
Різноманітність тематик	Неоднозначна класифікація	Enum-обмеження, few-shot приклади
Оманливість тем листів	Невірна класифікація за subject	Аналіз body, ігнорування subject
Конфіденційність ПД	Ризик витоку даних	Локальне розгортання (Ollama)
Обмежені ресурси GPU	Неможливість запуску великих моделей	Квантизація (Q4_K_M, Q8_0)
Нестабільність JSON-виводу	Помилки парсингу 15–20%	Post-processing: indexOf/lastIndexOf
Галюцинації LLM	Вигадані дані клієнтів	Prompt engineering, валідація

Як видно з таблиці 1.1, проблеми автоматизації обробки email-звернень охоплюють якість розпізнавання даних, коректність класифікації та технічні обмеження (конфіденційність, ресурси GPU, нестабільність виводу). Запропоновані рішення – детальний системний промпт, мультимовні моделі, enum-обмеження, локальне розгортання та квантизація – покладено в основу архітектури розроблюваної системи.

1.3 Огляд існуючих рішень

На сьогоднішній день існує ряд підходів та комерційних рішень для автоматизації обробки клієнтських звернень. Далі наведено огляд найбільш релевантних з детальним зазначенням функціональних можливостей, переваг та недоліків кожного підходу. Розглянуті рішення умовно можна поділити на дві групи: готові хмарні платформи з вбудованим ШІ та гнучкі інструменти автоматизації робочих процесів.

1.3.1 Zendesk AI

Zendesk є однією з найпоширеніших хмарних платформ для управління клієнтською підтримкою. Платформа пропонує вбудовані засоби штучного інтелекту – Zendesk AI для автоматизації обробки звернень клієнтів. Zendesk AI використовує власні моделі, навчені на мільярдах реальних тикетів зі своєї платформи.

З точки зору агентного AI-функціоналу Zendesk пропонує: Intelligent Triage – автоматична класифікація тикетів за наміром (intent), мовою та тональністю; Generative Reply – генерація відповідей на основі бази знань з використанням GPT-4; Agent Copilot – підказки оператору в реальному часі щодо можливих рішень; Automated Resolutions – повністю автономне вирішення типових запитів без участі оператора; Macro Suggestions – AI-рекомендації макросів на основі аналізу контексту розмови. Однак усі AI-компоненти Zendesk працюють переважно з англійською мовою, а для української мови функціонал Intelligent Triage та Generative Reply недоступний. Інтерфейс хмарної платформи Zendesk з модулем Zendesk AI наведено на рисунку 1.3.

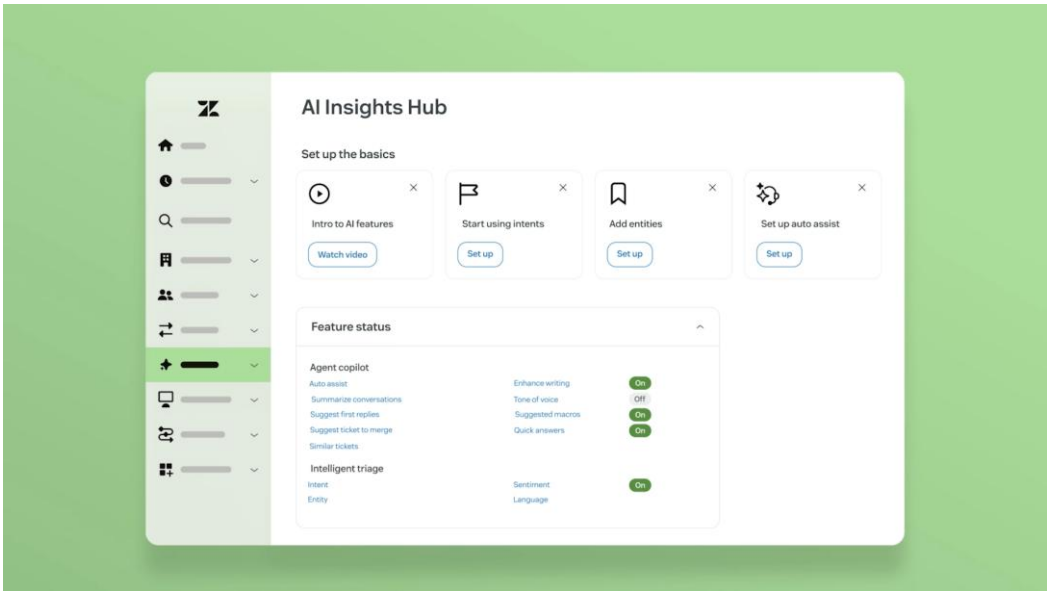


Рисунок 1.3 – Інтерфейс хмарної платформи Zendesk з модулем Zendesk AI

Тарифна модель Zendesk передбачає базову підписку від $19 до $169 за агента на місяць, плюс додатковий модуль Advanced AI за $50/агент/міс. Автоматизовані резолюції (Automated Resolutions) тарифікуються окремо: $1,50 за резолюцію при щорічному зобов'язанні або $2,00 при оплаті за фактом. Для команди з 5 операторів із 500 автоматизованих резолюцій на місяць загальна вартість може складати $1325–$1575 щомісяця [42].

Перевагами є зрілість платформи, широка екосистема інтеграцій та відсутність потреби у технічних знаннях для базового налаштування. Ключовими недоліками для українських ISP є: висока вартість, обмежена підтримка української мови в AI-компонентах, передача конфіденційних даних клієнтів на зовнішні сервери та неможливість адаптації AI-моделей під специфічну термінологію конкретного провайдера.

1.3.2 Freshdesk Freddy AI

Freshdesk – конкурент Zendesk від компанії Freshworks, що пропонує AI-помічника Freddy. Freddy здатний автоматично категоризувати тикети, пропонувати відповіді агентам на основі бази знань (KB Suggestions). Freddy AI Copilot коштує $29/агент/міс. при річній оплаті; AI Agent (автоматичні бот-сесії) тарифікується окремо за кількість сесій [41].

Агентний функціонал Freshdesk Freddy AI включає: Freddy AI Agent – автономний бот, здатний вести діалог з клієнтом, витягувати інформацію та виконувати дії (створення тикету, пошук у базі знань) без участі оператора; Freddy Copilot – AI-помічник для операторів, що аналізує контекст розмови, пропонує відповіді та автоматично заповнює поля тикету; Auto-triage – автоматичне визначення пріоритету, категорії та відповідального агента; Sentiment Analysis – визначення тональності звернення в реальному часі для ескалації негативних випадків. Freddy AI Agent тарифікується за кількість бот-сесій, Copilot – $29/агент/міс. при річній оплаті.

Інтерфейс хмарної платформи Freshdesk з помічником Freddy AI наведено на рисунку 1.4.

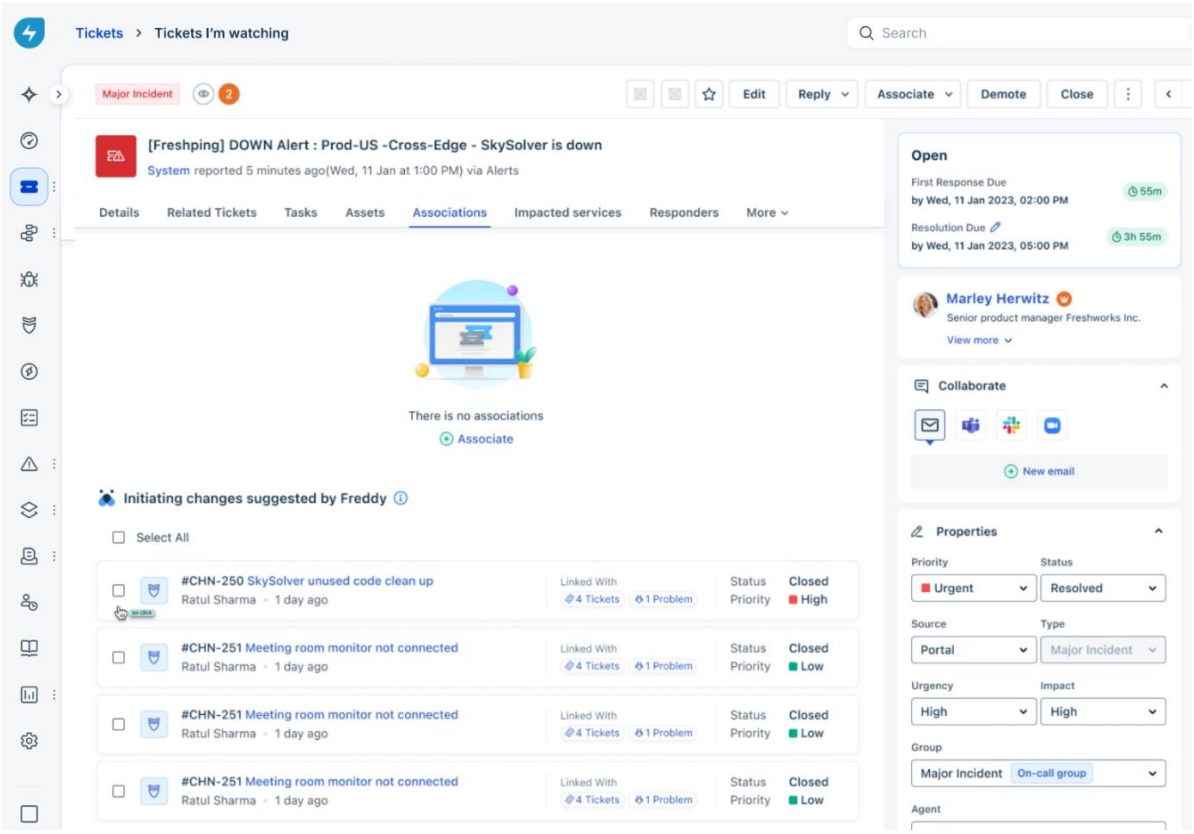


Рисунок 1.4 – Інтерфейс хмарної платформи Freshdesk з помічником Freddy AI

Переваги: нижча вартість порівняно з Zendesk, інтегрований AI без зовнішніх налаштувань, автоматичне формування відповідей на типові запити. Недоліки: обмежена точність при роботі з нестандартними зверненнями та мовами, відмінними від англійської; залежність від хмарної інфраструктури; відсутність підтримки української мови в AI-компонентах; обмежені можливості кастомізації моделей. Як і Zendesk, платформа орієнтована на англомовний ринок, що обмежує її застосовність для українських інтернет-провайдерів.

1.3.3 Комерційні LLM API (OpenAI, Anthropic)

Ряд компаній впроваджують обробку клієнтських звернень через API великих комерційних мовних моделей. GPT-4o від OpenAI та Claude від Anthropic є найбільш потужними загальнодоступними моделями. Обидві підтримують українську мову, здатні виконувати складні задачі класифікації, екстракції та аналізу тональності. OpenAI надає механізм Structured Outputs [4], [5], для гарантованого отримання валідного JSON, а також Function Calling для виклику зовнішніх інструментів.

Агентні можливості комерційних LLM API значно перевищують просту генерацію тексту. OpenAI пропонує Function Calling – механізм, що дозволяє моделі самостійно обирати та викликати зовнішні функції (API-запити, запис у БД, відправка повідомлень) на основі аналізу контексту. Structured Outputs гарантують 100% відповідність JSON-виводу заданій схемі на рівні декодера. Anthropic Claude надає аналогічний механізм Tool Use з підтримкою паралельного виклику декількох інструментів. Gemini від Google підтримує Grounding – підключення зовнішніх джерел знань для зменшення галюцинацій. Mistral AI пропонує Agents API з вбудованою підтримкою Code Interpreter, Web Search та підключення довільних функцій. Таким чином, усі провідні LLM API підтримують повноцінний агентний сценарій з автономним прийняттям рішень та взаємодією з зовнішніми системами. Порівняння тарифів найпоширеніших комерційних LLM API наведено в таблиці 1.2 (орієнтовні дані).

Таблиця 1.2 – Порівняння тарифів комерційних LLM API

Модель	Вхідні токени ($/1M)	Вихідні токени ($/1M)	Контекст
GPT-4o (OpenAI)	2.50	10.00	128K
Gemini 1.5 Pro (Google)	1.25	5.00	2M
Claude 3.5 Sonnet (Anthropic)	3.00	15.00	200K
Mistral Large (Mistral AI)	2.00	6.00	128K

Переваги: найвища якість розуміння контексту серед доступних LLM, підтримка української мови, гарантований структурований вивід. Недоліки: висока вартість при великих обсягах (при обробці 300 листів/день, ~2000 токенів/лист, витрати на GPT-4o складають ~$45–90 на місяць); передача персональних даних; залежність від доступності зовнішнього API; неможливість роботи офлайн.

1.3.4 Класичні NLP-підходи

До появи великих мовних моделей основними методами автоматичної класифікації текстів були алгоритми машинного навчання: Naive Bayes, Support Vector Machine (SVM), Random Forest у поєднанні з методами представлення тексту TF-IDF або Bag of Words. Також широко використовувались системи на основі правил (rule-based) та ключових слів. Комбінація TF-IDF + SVM забезпечує високу точність на задачі мультикласової класифікації email-повідомлень, однак ці результати отримані на англомовних даних із відносно чіткими категоріями.

Переваги: висока швидкість (тисячі листів за секунду), мінімальні вимоги до обчислювальних ресурсів, прозорість рішень. Недоліки: необхідність великих розмічених наборів даних (тисячі прикладів на клас), нездатність одночасно класифікувати та витягувати структуровані дані, відсутність розуміння контексту та семантичних зв'язків, складність підтримки при зміні категорій.

1.3.5 Локальні LLM через Ollama / LM Studio

Відносно новий підхід, що базується на запуску відкритих великих мовних моделей на локальному обладнанні. Ollama [2], [3], [4] – інструмент з відкритим кодом для запуску, управління та обслуговування LLM на локальному комп'ютері, що надає OpenAI-сумісний REST API. LM

Studio [17] – аналогічний інструмент з графічним інтерфейсом, що підтримує формати GGUF та MLX. Обидві платформи дозволяють запускати моделі з різними рівнями квантизації (від Q2_K до Q8_0), адаптуючи вимоги до VRAM під доступне обладнання [5], [6].

Інтерфейс інструмента локального розгортання великих мовних моделей Ollama наведено на рисунку 1.5.

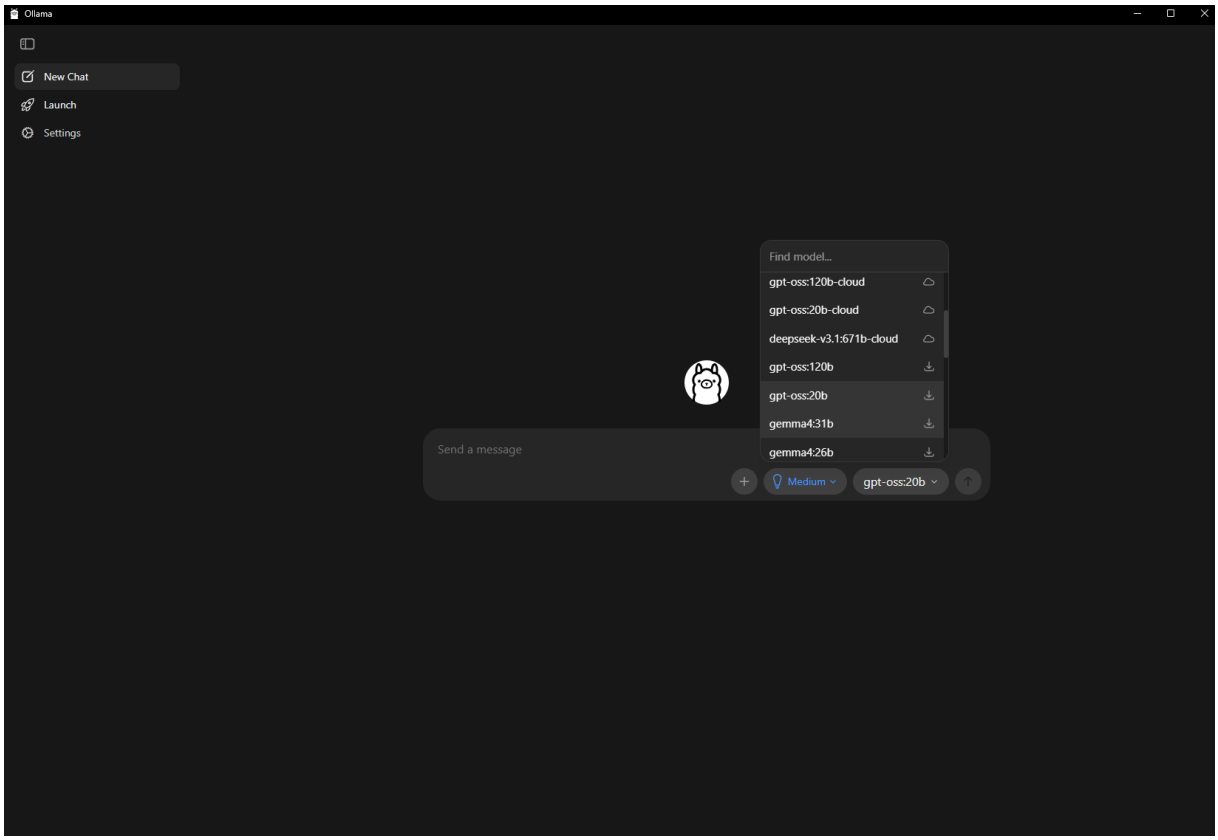


Рисунок 1.5 – Інтерфейс інструмента локального розгортання великих мовних моделей Ollama

Переваги: повна конфіденційність (дані не покидають локальну мережу), відсутність щомісячних платежів, можливість вибору оптимальної моделі, незалежність від Інтернету. Недоліки: вимоги до GPU (мінімум 8 ГБ VRAM), нижча якість порівняно з GPT-4o/Claude, необхідність самостійного налаштування та моніторингу.

1.3.6 Low-code платформи автоматизації з AI-інтеграціями

Окрему категорію рішень для автоматизації обробки звернень становлять low-code платформи оркестрації робочих процесів. На відміну від готових систем підтримки (підрозділи 1.3.1–1.3.2) чи прямого звернення до моделі через API (підрозділ 1.3.3), такі платформи не виконують обробку самостійно, а виступають середовищем, у якому з окремих вузлів збирається конвеєр: отримання листа, його передавання до мовної моделі, постобробка відповіді та запис результату. Найпоширеніші представники цього класу – n8n [7], [8], Make [9] та Zapier [10] – дозволяють будувати такі конвеєри у візуальному редакторі, без написання значного обсягу коду.

Серед перелічених платформ для задачі цієї роботи найбільший інтерес становить n8n, оскільки вона має вбудований вузол AI Agent і, на відміну від конкурентів, підтримує повністю локальне розгортання (self-hosted) через Docker [19]. У поєднанні з локальним запуском моделей через Ollama це забезпечує повну конфіденційність даних, що є критичним для обробки персональних даних клієнтів.

Вузол AI Agent реалізує повноцінний агентний цикл на базі фреймворку LangChain. Його ключові можливості такі: підтримка парадигми ReAct (Thought – Action – Observation) з автоматичним плануванням послідовності дій; підключення довільних LLM-провайдерів таких як: OpenAI, Ollama, Anthropic, Google, Groq, Mistral, HuggingFace через уніфікований інтерфейс; набір інструментів (HTTP Request, Code, Calculator, Google Sheets, SQL та інші), які агент обирає самостійно; підтримка пам’яті для збереження контексту між зверненнями; вбудований парсер структурованого виводу для витягнення JSON з відповіді моделі. На відміну від Make та Zapier, які надають лише модулі для виклику API комерційних моделей без агентного циклу, n8n дозволяє побудувати повноцінного агента з автономним вибором інструментів та ітеративною обробкою.

Поряд із технічними можливостями істотним критерієм вибору категорії рішення є вартість обробки. Розглянуті у підрозділах 1.3.1 – 1.3.5 підходи суттєво різняться за моделлю тарифікації: хмарні платформи стягують щомісячну плату за кожного агента, комерційні API – оплату за обсяг оброблених токенів, тоді як локальні моделі після одноразового придбання обладнання не мають змінних витрат на кожне звернення. Щоб зробити ці підходи зіставними, орієнтовну вартість обробки одного email-звернення зведено до спільної одиниці – центів за звернення – і подано на рисунку 1.6.

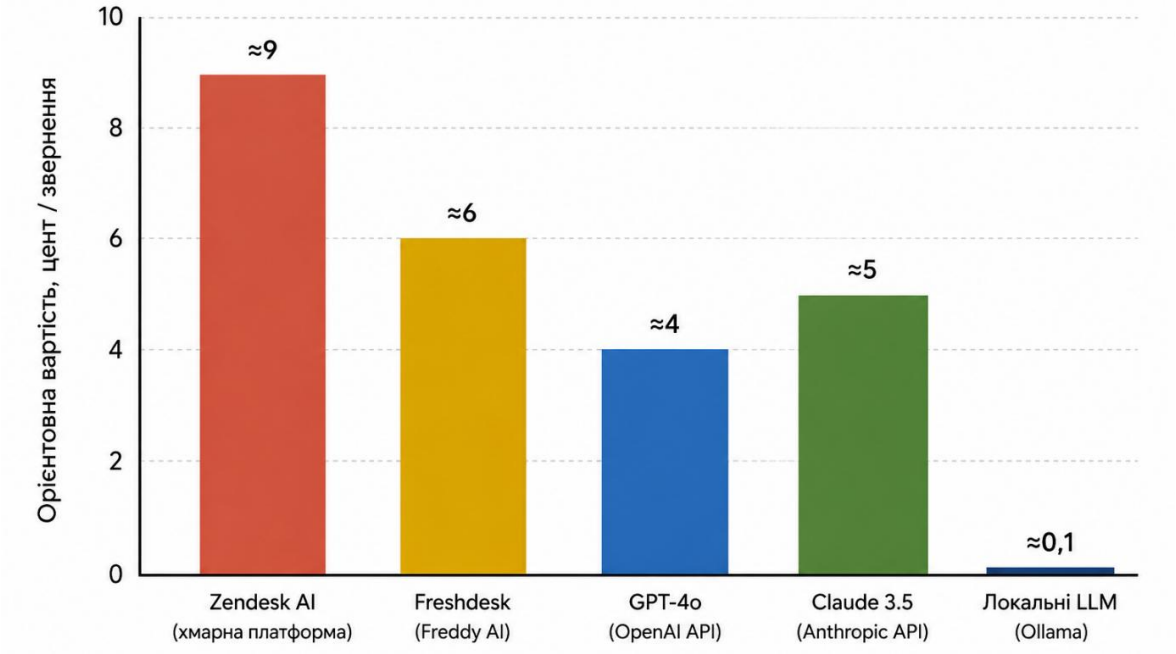


Рисунок 1.6 – Орієнтовна порівняльна вартість обробки одного email-звернення для різних категорій рішень, (створено за допомогою ШІ)

Як видно з рисунка 1.6, різниця між підходами сягає двох порядків. Обробка звернення на хмарній платформі Zendesk AI коштує близько 9 центів, на Freshdesk – близько 6 центів; використання комерційних API GPT-4o та Claude 3.5 обходиться приблизно у 4-5 центів за звернення. Натомість запуск локальної моделі через Ollama знижує вартість до приблизно 0,1 цента – вона визначається лише витратами на

електроенергію, тоді як вартість самого обладнання є одноразовою. Для провайдера з відчутним потоком звернень така різниця стає вирішальним аргументом на користь локального розгортання.

Серед трьох платформ Make і Zapier пропонують ширші бібліотеки готових інтеграцій, однак обидві є хмарними, не допускають локального розгортання та обмежують кількість виконань у безкоштовному плані.

Натомість n8n поширюється за моделлю fair-code, підтримує повністю локальне (self-hosted) розгортання через Docker, має вбудований агентний вузол на базі LangChain з нативною інтеграцією Ollama та не обмежує кількість виконань при self-hosted розгортанні. Для задачі цієї роботи більша бібліотека готових інтеграцій не є визначальною, тоді як можливість локального розгортання і зняття обмежень на кількість оброблених звернень – принципові.

Обґрунтування вибору. Для побудови конвеєра обробки звернень у цій роботі обрано платформу n8n. Вона надає все необхідне для поставленої задачі: повністю локальне (self-hosted) розгортання, готовий вузол AI Agent з нативною інтеграцією Ollama та відсутність обмежень на кількість оброблених звернень. Платформа також проста у встановленні й налаштуванні – конвеєр збирається візуально, без написання значного обсягу коду. Завдяки цьому рішення повністю відповідає вимогам конфіденційності та не створює змінних витрат на кожне звернення.

Розглянуті у підрозділах 1.3.1–1.3.6 категорії рішень зручно порівняти напряму – за спільними критеріями: конфіденційністю, якістю обробки природної мови, підтримкою української мови, можливостями екстракції структурованих даних, вартістю та автономністю.

Таке порівняння показує, що жоден підхід не є найкращим за всіма критеріями одночасно. Готові хмарні платформи прості у впровадженні, але не забезпечують конфіденційності та підтримки української мови. Комерційні API дають найвищу якість обробки, проте передбачають передавання даних до хмари та помітні змінні витрати. Класичні NLP-

методи працюють локально, але поступаються за якістю й потребують окремого навчання під кожну задачу. Натомість локальні великі мовні моделі у поєднанні з n8n дають високу конфіденційність, прийнятну якість обробки, повноцінну екстракцію структурованих даних і мінімальні змінні витрати – тому саме цей підхід обрано основою практичної частини роботи. Водночас якість обробки тут істотно залежить від конкретної моделі, що й зумовлює потребу в порівняльному дослідженні, якому присвячено наступні розділи.

1.4 Аналіз концепції агентних інтелектуальних систем на базі LLM

Для розуміння підходу, обраного в даній роботі, необхідно розглянути концепцію агентних систем на базі великих мовних моделей, яка суттєво відрізняється від простого використання LLM як генератора тексту.

Велика мовна модель у базовому режимі функціонує як функція, що приймає на вхід текстовий запит (промпт) та генерує текстову відповідь на основі авторегресивного передбачення наступного токена [11]. Така модель не має стану між запитами, не може взаємодіяти із зовнішніми системами та обмежена лише генерацією тексту. AI-агент, у свою чергу, являє собою більш складну систему, в якій LLM виступає ядром прийняття рішень, але доповнюється компонентами, що суттєво розширюють його можливості.

У сучасній науковій літературі LLM-агент визначається як система, що використовує велику мовну модель як центральний контролер для планування, прийняття рішень та виконання дій у навколишньому середовищі [19]. На відміну від класичного використання мовної моделі як простого генератора тексту, агентний підхід передбачає, що модель самостійно декомпозує поставлену задачу на послідовність кроків та визначає, які саме дії необхідно виконати для досягнення кінцевої мети. Цю класифікацію розширюють на одно- та багатоагентні сценарії, що передбачають кооперацію агентів між собою та з людиною [20]. Така

архітектура відкриває широкі можливості для автоматизації складних інтелектуальних процесів, які раніше потребували безпосередньої участі людини. Окремо досліджуються методи оцінювання LLM-агентів, зокрема бенчмарки для tool use, планування та багатокрокової взаємодії [21].

Ключовим елементом архітектури агента є парадигма ReAct (Reasoning + Acting) (рисунок 1.7) [22], яка поєднує міркування та дії в ітеративному циклі:

– thought (думка) – модель аналізує поточну задачу, визначає, яку інформацію потрібно отримати або яку дію виконати;

– action (дія) – модель викликає зовнішній інструмент (tool) або формує запит до API;

– observation (спостереження) – модель аналізує результат дії та використовує його для подальшого міркування або формування фінальної відповіді.

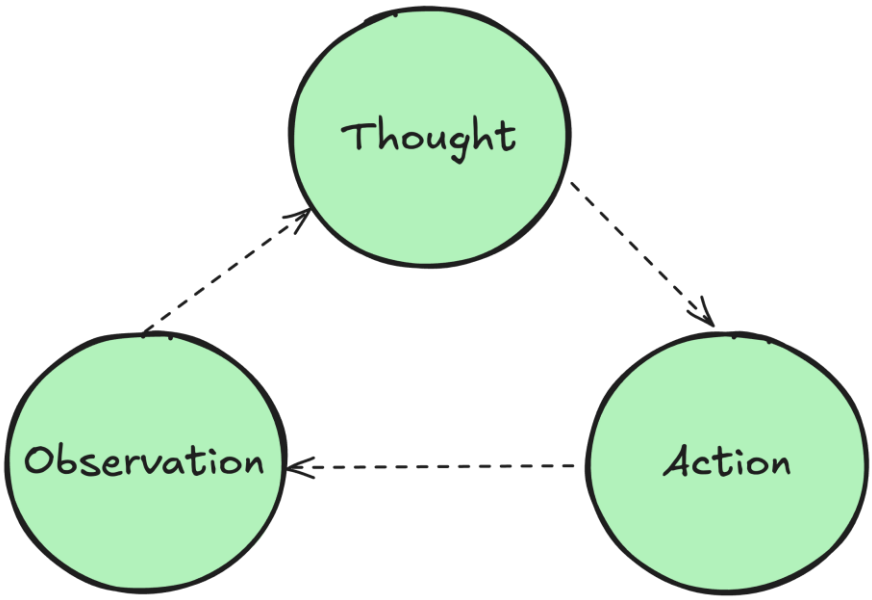


Рисунок 1.7 – Цикл ReAct

Типова архітектура LLM-агента наведена в таблиці 1.3, включає наступні компоненти.

Таблиця 1.3 – Компоненти архітектури LLM-агента

Компонент	Опис	Реалізація в даній роботі
LLM-ядро (Brain)	Модель, що відповідає за розуміння та генерацію	Gemma 3 / Qwen 3 / Llama 3.1 / Mistral 7B через Ollama
Системний промпт	Інструкції: роль, задачі, формат виводу	Єдиний промпт для всіх моделей (14 полів JSON)
Інструменти (Tools)	Зовнішні функції для виконання дій	Не використовуються (single-turn agent)
Пам'ять (Memory)	Збереження контексту між запитами	Контекстне вікно поточного листа
Цикл виконання	Ітеративний процес ReAct	Один крок: вхід → генерація → JSON

У контексті задачі обробки клієнтських email-звернень агентний підхід має суттєві переваги перед простим запитом до LLM. По-перше, агент може бути налаштований на виконання комплексної задачі: одночасну класифікацію звернення за 6 категоріями, визначення пріоритетності за 3-бальною шкалою, екстракцію 6 полів персональних даних, аналіз тональності та формування резюме – все це в рамках одного запиту. По-друге, системний промпт дозволяє задати чіткі правила класифікації, перелік допустимих значень (enum-обмеження), приклади коректної обробки (few-shot examples) та інструкції щодо поведінки у граничних випадках [23], що значно підвищує стабільність та якість виводу.

У даній роботі агентний підхід реалізується засобами вузла AI Agent платформи n8n [7], який забезпечує інтеграцію з Ollama для виклику локальних LLM, підтримку системного промпту та структурованого виводу. Такий підхід дозволяє побудувати єдиний конвеєр обробки, у якому змінюється лише мовна модель. Кожна з чотирьох досліджуваних моделей працює як ядро окремого агента з ідентичним системним промптом, що дозволяє коректно порівнювати саме здатності моделей, а не відмінності в налаштуваннях. Завдяки цьому забезпечується однаковість умов.

1.5 Обґрунтування інноваційності та переваг розроблюваної системи

Наведений огляд існуючих рішень показав, що кожен із розглянутих підходів має суттєві обмеження. Комерційні платформи (Zendesk, Freshdesk) є дорогими та не забезпечують підтримки української мови в AI-компонентах. Комерційні LLM API (OpenAI , Anthropic) створюють ризики конфіденційності та генерують постійні операційні витрати. Класичні NLP-методи не здатні виконувати комплексну обробку неструктурованого тексту (класифікація + NER + sentiment) в рамках однієї моделі.

Система, що розробляється в рамках даної кваліфікаційної роботи, спрямована на подолання зазначених обмежень та пропонує наступні рішення:

– повністю локальне розгортання – всі компоненти системи (n8n, Ollama, моделі) працюють на локальному обладнанні. Жодні дані клієнтів не передаються на зовнішні сервери, що дозволяє зберегти повний контроль над конфіденційними даними клієнтів;

– порівняльний аналіз чотирьох моделей від чотирьох різних розробників – Gemma 3 12B (Google DeepMind, США) [12], Qwen 3 8B (Alibaba Cloud, Китай) [14], Llama 3.1 8B (Meta, США) [15] та Mistral 7B (Mistral AI, Франція) [17]. Різноманіття організацій забезпечує репрезентативність порівняння;

– агентний підхід замість простого запиту – використання AI Agent вузла n8n з ReAct-парадигмою [22] забезпечує комплексну обробку: класифікація за 6 категоріями, пріоритетність, екстракція 6 полів персональних даних, аналіз тональності та формування резюме – все в рамках одного виклику;

– low-code реалізація з мінімальним порогом входу – конвеєр обробки побудований візуально в n8n; єдиний фрагмент коду – нода парсингу JSON-відповіді моделі;

– структурований JSON-вивід із 14 полями – модель формує повний набір даних для подальшої маршрутизації, інтеграції з CRM та побудови аналітики;

– GPU-прискорення на AMD через Vulkan API – система працює на продуктивному GPU AMD RX 9070 XT (16 ГБ VRAM) з використанням експериментального Vulkan-бекенду Ollama [5, 6], що розширює доступність рішення за межі екосистеми NVIDIA;

– адаптація до українського мовного контексту – системний промпт враховує специфіку: формат адрес (вул./пр./пров.), номерів телефонів (+380), особливості нумерації договорів, мовні особливості української (розмовна лексика, скорочення, відмінювання).

1.6 Постановка задачі

На основі проведеного аналізу предметної галузі, існуючих рішень та визначених проблем формулюється наступна задача дослідження.

Необхідно спроєктувати, реалізувати та дослідити прототип системи автоматизованої обробки текстового контенту клієнтських звернень, що надходять засобами електронної пошти, на базі агентних інтелектуальних систем із використанням локальних великих мовних моделей. Прототип повинен приймати на вхід текстовий контент та формувати на виході структурований JSON-об'єкт, що містить результати класифікації, а саме кластеризації звернень за категоріями відповідно до бізнес-процесів служби підтримки, та екстракції даних і аналізу тональності. Сформований у такому вигляді результат має бути придатним для подальшої автоматичної маршрутизації звернень та інтеграції з інформаційними системами провайдера без додаткової ручної обробки.

Вихідний JSON-об'єкт повинен містити 14 полів, специфікація яких наведена у таблиці 1.4.

Таблиця 1.4 – Структура вихідного JSON-об'єкта (14 полів)

Поле	Тип	Опис	Джерело
message_id	string	Ідентифікатор повідомлення Gmail	Gmail API
gmail_link	string	Пряме посилання на лист	Формується в n8n
confidence_score	number (0–1)	Впевненість моделі у результаті	Генерується LLM
summary	string	Резюме звернення (1–2 речення)	Генерується LLM
processing_time	number (с)	Час обробки у секундах	Обчислюється в Code node
sender_name	string	Ім'я та прізвище відправника	Витягується LLM з тексту
sender_email	string	Email відправника	Gmail API
phone_number	string	Номер телефону клієнта	Витягується LLM з тексту
address	string	Адреса підключення	Витягується LLM з тексту
contract_id	string	Номер договору	Витягується LLM з тексту
category	enum (6 значень)	Категорія звернення	Класифікується LLM
priority	enum (1, 2, 3)	Пріоритетність обробки	Класифікується LLM
requires_response	boolean	Потребує відповіді?	Класифікується LLM
sentiment	enum (3 значення)	Тональність звернення	Класифікується LLM

Поле category може приймати одне з 6 значень: техпідтримка, бухгалтерія, підключення, відключення, спам_реклама, інше. Поле priority використовує 3-бальну шкалу: 1 – відповідь не потрібна (спам, авто-сповіщення), 2 – не терміново (загальні питання), 3 – терміново (аварії, скарги). Поле sentiment – positive, neutral або negative.

Для порівняльного аналізу обрано чотири моделі, характеристики яких наведені у таблиці 1.5.

Таблиця 1.5 – Обрані моделі

Модель	Розробник	Країна	Параметри	Контекст
Gemma 3 12B	Google DeepMind	США	12 млрд	128K
Qwen 3 8B	Alibaba Cloud	Китай	8 млрд	128K
Llama 3.1 8B	Meta	США	8 млрд	128K
Mistral 7B	Mistral AI	Франція	7.3 млрд	32K

Вибір саме цих моделей обумовлений декількома факторами. По-перше, моделі представляють чотири різні організації-розробники, що мають різні підходи до навчання, різні навчальні дані та різну ступінь підтримки мультимовності. По-друге, обрані моделі (7-12 мільярдів параметрів) повністю розміщуються у 16 ГБ відеопам'яті використаної відеокарти AMD Radeon RX 9070 XT, що дозволяє запускати їх локально навіть без агресивної квантизації; ця відеокарта належить до верхнього цінового сегмента і забезпечує достатній запас продуктивності для моделей такого класу. По-третє, усі обрані моделі доступні через Ollama [2] та мають відкриті ваги, що забезпечує відтворюваність експерименту. По-четверте, вибір обґрунтований рейтингами LMArena [18] та даними lmmarketcap.com, де ці моделі стабільно входять до числа лідерів серед моделей відповідного розміру.

Оцінка ефективності кожної моделі виконується шляхом обробки тестового набору email-звернень та порівняння результатів з еталонною таблицею. Порівняння виконується за п'ятьма групами критеріїв:

– точність класифікації (Accuracy) – відсоток збігів з еталоном для полів category, priority, sentiment, requires_response;

– якість екстракції даних – відсоток коректно витягнутих значень (exact match) для полів sender_name, phone_number, address, contract_id;

– якість резюме – експертна оцінка адекватності поля summary;

– надійність JSON-виводу – відсоток відповідей з валідним JSON, що містить усі 14 полів;

– швидкість обробки – середній час обробки одного звернення у секундах.

Очікуваним результатом вирішення поставленої задачі є працездатний прототип, здатний перетворювати неструктурований текст email-звернення на структурований JSON-об'єкт із прийнятною для практичного застосування точністю, а також обґрунтований висновок щодо того, яка з чотирьох моделей найкраще підходить для цієї задачі за кожним із пріоритетів – максимальною точністю, мінімальним часом обробки або балансом між ними. Окрім кількісних показників, важливою характеристикою прототипу є його відтворюваність, тобто можливість розгортання та повторного запуску системи без залучення вузькопрофільних фахівців. Це робить отримані результати застосовними не лише в межах даного дослідження, а й у реальних умовах експлуатації.

Саме сукупність наведених критеріїв, а не окрема метрика, є підставою для підсумкових рекомендацій: модель, що лідирує за точністю класифікації, може поступатися за швидкістю обробки, тому остаточний вибір залежить від вимог конкретного провайдера. Слід зауважити, що жодна з моделей апріорі не може вважатися найкращою для всіх сценаріїв одночасно, оскільки кожна з них має власні сильні та слабкі сторони. Тому ключову цінність становить не вибір єдиної універсальної моделі, а саме порівняльна оцінка, яка дозволяє співвіднести характеристики кожної моделі з конкретними потребами служби підтримки.

Такий підхід забезпечує гнучкість прийняття рішень та можливість адаптації системи під різні умови експлуатації. Вирішення сформульованої задачі потребує попереднього розгляду теоретичних основ роботи великих мовних моделей – їхньої архітектури, принципів токенізації україномовного тексту та методів отримання структурованого виводу, – яким присвячено наступний розділ.

2 ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ

2.1 Архітектура Transformer як основа сучасних великих мовних моделей

Всі чотири моделі, що досліджуються в даній роботі (Gemma 3, Qwen 3, Llama 3.1, Mistral 7B), побудовані на основі архітектури Transformer, запропонованої у 2017 році [11]. Ця архітектура стала фундаментом революції у обробці природної мови (NLP) та замінила попередні підходи на основі рекурентних нейронних мереж (RNN) та мереж довгої короткочасної пам'яті (LSTM).

У початковій редакції [11] Transformer розв’язував задачу машинного перекладу й тому поєднував дві частини: кодувальник (encoder), який згортає вхідні токени у векторне представлення, та декодувальник (decoder), який на основі цього представлення породжує вихідну послідовність. Для задачі породження тексту окремий кодувальник не потрібен, тому всі чотири досліджувані моделі належать до сім’ї суто декодувальних (decoder-only) архітектур. У такій конфігурації класифікація звернення та екстракція полів зводяться до того самого механізму послідовного породження токенів, що й вільна генерація тексту [11].

Принциповою відмінністю Transformer від рекурентних мереж є відмова від послідовного проходження тексту на користь механізму самоуваги (self-attention). У межах цього механізму для кожного токена обчислюється, наскільки сильно з ним пов’язаний кожен інший токен послідовності, після чого його представлення оновлюється як зважена сума представлень усіх токенів. Завдяки цьому модель напряму вловлює залежності між віддаленими словами незалежно від відстані між ними. Формально операція самоуваги записується як:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (2.1)$$

де Q (Query), K (Key) та V (Value) – матриці, отримані лінійним перетворенням вхідних ембедингів;

d_k – розмірність ключів [11].

Масштабування на √d_k запобігає надмірному зростанню значень скалярного добутку при великих розмірностях, що могло б призвести до насичення функції softmax та нульових градієнтів.

Один набір проєкцій Q, K, V здатний відстежувати лише один тип зв'язків між токенами. Щоб модель могла одночасно враховувати різні аспекти контексту – синтаксичні, семантичні, позиційні, – застосовують багатоголову увагу (Multi-Head Attention, MHA). У цьому випадку обчислення уваги повторюється h разів незалежними «головами» зі своїми лінійними проєкціями, а отримані результати об'єднуються в єдине представлення:

$$\text{MultiHead}(\text{Q}, \text{K}, \text{V}) = \text{Concat}(\text{head}_1, \dots, \text{head}_\text{h})\,\text{W}^\text{O},\ \text{head}_\text{i} = \text{Attention}\left(\text{QW}_\text{i}^\text{Q}, \text{KW}_\text{i}^\text{K}, \text{VW}_\text{i}^\text{V}\right), \quad (2.2)$$

де head_i – вихід і-ї голови уваги;

h – кількість голів;

W^O – матриця вихідної лінійної проєкції;

W_i^Q, W_i^K, W_i^V – матриці лінійних проєкцій запитів, ключів і значень для і-ї голови;

Concat – операція конкатенації виходів усіх голів [11].

Узагальнену архітектуру моделі Transformer, що лежить в основі всіх досліджуваних у роботі моделей, наведено на рисунку 2.1. Вона побудована на механізмі уваги та складається з блоків, які повторюються кілька разів, що дозволяє ефективно опрацьовувати послідовності тексту.

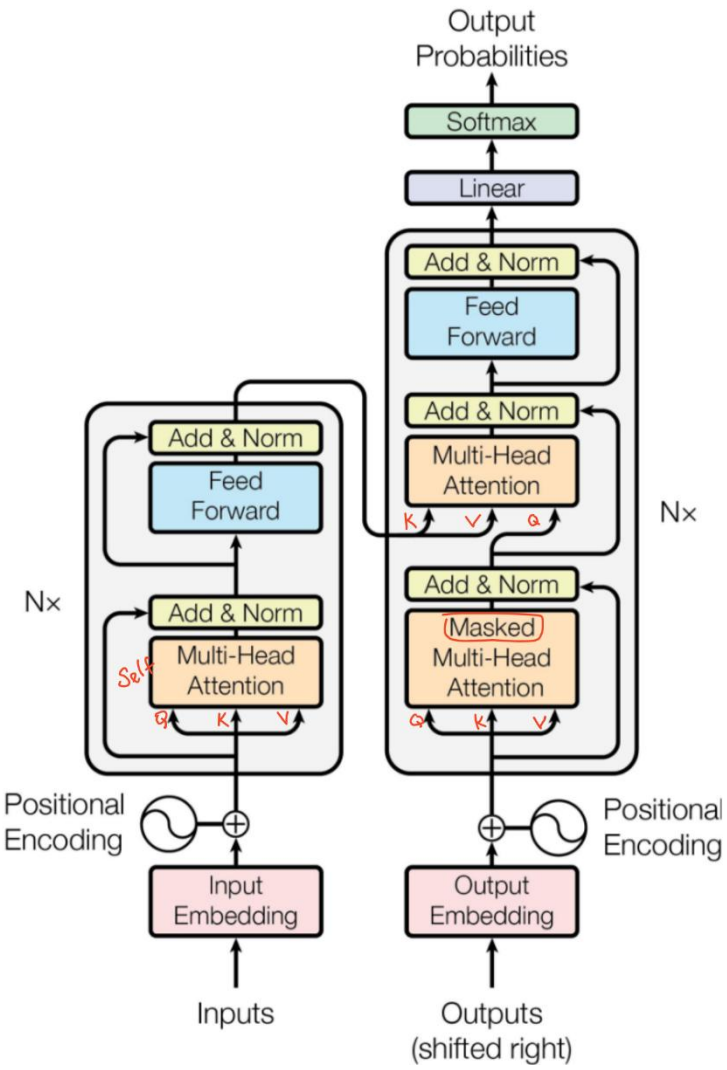


Рисунок 2.1 – Архітектура блоку Transformer decoder-only

Поряд із підшаром уваги кожен блок Transformer містить мережу прямого поширення (Feed-Forward Network, FFN) – два лінійні перетворення, розділені нелінійною активацією (як правило, GELU або SiLU), яка застосовується до кожного токена незалежно. Щоб уникнути згасання градієнта в глибоких моделях, вихід кожного підшару додається до його входу через залишкове з'єднання (residual connection), а результат нормалізується шаром нормалізації (Layer Normalization). У ранніх Transformer нормалізація стояла після підшару, тоді як Llama і Gemma застосовують її перед підшаром (схема Pre-LayerNorm), що робить процес навчання стабільнішим [15].

Вивід суто декодувальної моделі будується покроково: на кожному кроці до вже згенерованого тексту дописується рівно один токен, і цей токен стає частиною контексту для наступного кроку. Перед вибором токена модель отримує розподіл ймовірностей над усім словником (vocabulary), а конкретний токен відбирається згідно з обраною стратегією декодування – greedy decoding, top-p, top-k або температурного семплування [11]. Параметр temperature регулює ступінь визначеності виводу: за значень, близьких до 0, модель майже завжди обирає найімовірніший токен, що важливо для задач екстракції та класифікації, тоді як значення понад 1 підвищують різноманітність ціною стабільності. Саме тому в експериментах застосовано низьку температуру 0.1 (підрозділ 4.1).

2.2 Токенізація та її вплив на обробку українського тексту

Модель Transformer оперує не символами, а числовими ідентифікаторами, тому будь-яке звернення спершу проходить токенізацію – розбиття тексту на елементи фіксованого словника та заміну кожного з них на відповідний token ID. Те, на скільки токенів розпадається текст, визначає три практично важливі показники: обсяг обчислень (а отже, час відповіді), частку контекстного вікна, яку займає звернення, та якість подальшого розуміння тексту [26]. Для україномовних звернень цей етап є вузьким місцем, тому розглянемо його докладніше.

Сучасні LLM здебільшого спираються не на пословну, а на субсловну (subword) токенізацію – переважно за алгоритмами Byte-Pair Encoding (BPE) та SentencePiece. Словник для них не задається вручну, а виводиться зі статистики навчального корпусу: найчастотніші послідовності символів об’єднуються в окремі токени. Оскільки ці корпуси переважно англомовні, частотні англійські слова потрапляють до словника цілком і кодуються одним-двома токенами, тоді як кириличні слова, представлені в корпусі значно слабше, дробляться на дрібніші фрагменти.

Унаслідок цього українське звернення розпадається на помітно більше токенів, ніж рівний за змістом англійський текст.

Ефективність токенізації (fertility) для української мови систематично оцінено для 12 сучасних моделей LLM [26]. Fertility визначається як відношення кількості токенів до кількості слів у тексті: чим ближче fertility до 1, тим ефективніший токенізатор. Основні результати дослідження наведено у таблиці В.1 (додаток В).

Узагальнені дані для чотирьох досліджуваних моделей наведено у додатку В [26]. Найкращу ефективність токенізації українського тексту демонструє Gemma 3 (fertility 1.63), яка завдяки словнику обсягом 256K токенів включає велику кількість кириличних субслів. Qwen 3 займає проміжне положення (1.78), Llama 3.1 показує fertility 2.01, а Mistral 7B v0.3 з найменшим серед досліджуваних словником у 32K токенів – найгірший результат 2.42. Для порівняння, в усіх цих моделей fertility англійського тексту лишається в межах 1.15–1.30, тобто розрив між мовами становить від 1.31x у Gemma 3 до 1.98x у Mistral 7B (рисунок 2.2).

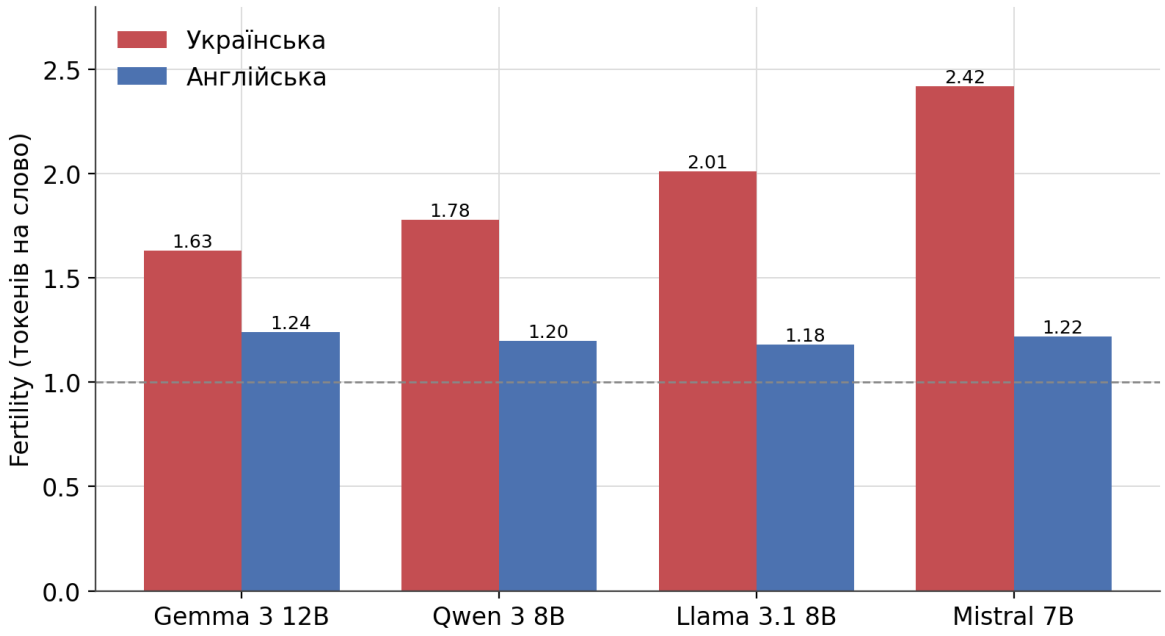


Рисунок 2.2 – Порівняння fertility токенізаторів для української та англійської мов (створено за допомогою ШІ)

Як видно з рисунка 2.2, токенізатори демонструють помітно вищу fertility для української мови порівняно з англійською, тобто витрачають більше токенів на кодування однакового за змістом тексту. Це зумовлено тим, що словники моделей формуються переважно на англомовних корпусах, тому кирилична морфологічно багата лексика частіше розбивається на дрібні субслова.

2.3 Детальна характеристика обраних моделей

2.3.1 Gemma 3 12B (Google DeepMind)

Gemma 3 – сімейство відкритих мовних моделей, розроблених Google DeepMind та випущених у березні 2025 року [12, 26, 13]. Сімейство включає моделі розмірами 1B, 4B, 12B та 27B параметрів. У даній роботі використовується модель Gemma 3 12B, яка представляє оптимальний баланс між якістю генерації та вимогами до VRAM.

Архітектурні особливості Gemma 3 12B:

– кількість параметрів: 12 мільярдів (12B);

– кількість шарів: 48 блоків Transformer;

– розмірність ембедингів: 3840;

– кількість голів уваги: 16 (з Grouped Query Attention – 8 KV-голів);

– контекстне вікно: 128K токенів;

– словник: 262 144 токени (SentencePiece) – найбільший серед досліджуваних моделей;

– мультимодальність: підтримка тексту та зображень (Vision Encoder на базі SigLIP);

– підтримка мов: понад 140 мов, включаючи українську [13];

– ліцензія: Gemma Terms of Use (дозволяє комерційне використання з обмеженнями).

Gemma 3 12B використовує оптимізований механізм уваги з Sliding Window Attention (SWA) для локальних шарів (розмір вікна 1024) та повну глобальну увагу через кожні 5 шарів, що дозволяє ефективно обробляти довгі контексти без квадратичного зростання обчислювальних витрат [12]. За результатами бенчмарку LMArena [18], Gemma 3 27B IT досягла Elo рейтингу 1339, перевершивши моделі значно більшого розміру.

2.3.2 Qwen 3 8B (Alibaba Cloud)

Qwen 3 – сімейство великих мовних моделей, розроблених командою Qwen в Alibaba Cloud та випущених у квітні 2025 року [28, 14]. Сімейство включає dense-моделі (0.6B–32B) та Mixture-of-Experts моделі (30B-A3B та 235B-A22B). У даній роботі використовується dense-модель Qwen 3 8B.

Ключовою інновацією Qwen 3 є гібридний режим мислення (Hybrid Thinking), що дозволяє моделі переключатися між режимами глибокого міркування (thinking mode) та швидкого відповідання (non-thinking mode) за допомогою спеціальних тегів /think та /no_think [14]. Це дозволяє адаптувати баланс між якістю та швидкістю відповіді в залежності від складності задачі.

Архітектурні особливості Qwen 3 8B:

– параметри: 8 мільярдів;

– шари: 36 блоків Transformer;

– розмірність ембедингів: 4096;

– голови уваги: 32 (GQA з 8 KV-голів);

– контекстне вікно: 128K токенів;

– словник: 151 936 токенів (Byte-level BPE);

– навчальні дані: 36 трильйонів токенів [14] – найбільший обсяг серед досліджуваних;

– підтримка мов: 119 мов та діалектів;

– ліцензія: Apache 2.0 – вільна ліцензія, що дозволяє необмежене комерційне використання, модифікацію та поширення коду й ваг; найбільш відкрита серед досліджуваних моделей.

2.3.3 Llama 3.1 8B (Meta)

Llama 3.1 – сімейство великих мовних моделей, розроблених Meta AI та випущених у липні 2024 року [15, 16]. Сімейство включає моделі розмірами 8B, 70B та 405B параметрів. Meta позиціонує Llama 3.1 405B як першу open-weight модель, порівнянну за якістю з GPT-4 [15].

Архітектурні особливості Llama 3.1 8B:

– параметри: 8 мільярдів;

– шари: 32 блоки Transformer;

– розмірність ембедингів: 4096;

– голови уваги: 32 (GQA з 8 KV-голів);

– контекстне вікно: 128K токенів;

– словник: 128 256 токенів (tiktoken BPE);

– навчальні дані: 15 трильйонів токенів [15];

– підтримка мов: 8 мов (англійська, німецька, французька, італійська, португальська, хінді, іспанська, тайська) [16] – українська офіційно не підтримується;

– ліцензія: Llama 3.1 Community License (дозволяє комерційне використання з обмеженнями для компаній з >700 млн активних користувачів).

Важливо зазначити, що Llama 3.1 офіційно підтримує лише 8 мов, серед яких відсутня українська [16]. Однак навчальні дані обсягом 15 трильйонів токенів неминуче включають певний обсяг українського тексту з Інтернету, що дозволяє моделі працювати з українською мовою на базовому рівні, хоча з нижчою якістю порівняно з офіційно підтримуваними

мовами. Це відображається у fertility 2.01 для української [26], що є найгіршим серед досліджуваних моделей (за винятком Mistral 7B).

2.3.4 Mistral 7B (Mistral AI)

Mistral 7B – велика мовна модель, розроблена французьким стартапом Mistral AI та випущена у вересні 2023 року [32, 17]. Це була перша модель Mistral AI, яка привернула значну увагу спільноти завдяки тому, що при розмірі лише 7,3 мільярди параметрів вона перевершувала Llama 2 13B на всіх бенчмарках [17].

Архітектурні особливості Mistral 7B:

– параметри: 7,3 мільярди;

– шари: 32 блоки Transformer;

– розмірність ембедингів: 4096;

– голови уваги: 32 (GQA з 8 KV-голів);

– контекстне вікно: 32K токенів (найменше серед досліджуваних);

– словник: 32 000 токенів (SentencePiece BPE) – найменший серед досліджуваних;

– ключова інновація: Sliding Window Attention (SWA) з розміром вікна 4096 токенів [17];

– ліцензія: Apache 2.0.

Sliding Window Attention (SWA) – механізм, при якому кожен токен «бачить» лише W попередніх токенів (де W – розмір вікна), а не всю послідовність. Це зменшує обчислювальну складність уваги з $O(n^2)$ до $O(n \cdot W)$, де n – довжина послідовності [17]. Завдяки багатошаровій структурі (32 шари) інформація може ефективно поширюватися на відстань до $W \times L = 4096 \times 32 = 131\,072$ токенів, що значно перевищує формальний розмір вікна. Mistral 7B також використовує Grouped Query Attention (GQA) з 8 KV-голів замість стандартних 32, що зменшує обсяг KV-кешу та прискорює інференс [17].

Зведене порівняння архітектурних характеристик усіх чотирьох моделей наведено у таблиці 2.2.

Таблиця 2.2 – Зведене порівняння архітектурних характеристик моделей

Характеристика	Gemma 3 12B	Qwen 3 8B	Llama 3.1 8B	Mistral 7B
Параметри	12 млрд	8 млрд	8 млрд	7.3 млрд
Шари	48	36	32	32
Розмірність	3840	4096	4096	4096
Голови уваги (KV)	16 (8)	32 (8)	32 (8)	32 (8)
Контекст	128K	128K	128K	32K
Словник	262K	152K	128K	32K
Навчальні дані	?	36T токенів	15T токенів	?
Офіц. мов	140+	119	8	?
Fertility (укр.)	1.63	1.78	2.01	2.42
Ліцензія	Gemma Terms	Apache 2.0	Llama 3.1	Apache 2.0
Дата випуску	03.2025	04.2025	07.2024	09.2023

2.4 Платформа n8n та її AI-можливості

n8n (вимовляється «n-eight-n», від «nodemation») – це платформа автоматизації робочих процесів з відкритим кодом (fair-code ліцензія), що дозволяє створювати складні конвеєри обробки даних за допомогою візуального інтерфейсу [7], [8]. Платформа заснована Яном Оберхаузером у 2019 році в Берліні та станом на 2025 рік має понад 50 000 зірок на GitHub, що свідчить про широку популярність серед розробників.

Ключовою перевагою n8n для даної роботи є можливість повністю локального розгортання (self-hosted deployment). Платформа може бути встановлена на локальному сервері через npm, Docker або Docker Compose [19], що гарантує, що жодні дані не покидають інфраструктуру

організації. Це принципово відрізняє n8n від конкурентів – Zapier [10] та Make [9], які працюють виключно у хмарі.

Вузол AI Agent [7] є центральним елементом AI-функціональності n8n. Він побудований на основі фреймворку LangChain та підтримує наступні можливості:

– підключення різних LLM-провайдерів: OpenAI, Ollama, Anthropic, Google, Azure, Groq, Mistral, Hugging Face;

– використання системного промпту для визначення ролі та поведінки агента;

– підключення інструментів (tools): HTTP Request, Code, Google Sheets, SQL, Wikipedia та ін.;

– підтримка пам'яті (memory): Window Buffer Memory, Token Buffer Memory, Zep Memory;

– реалізація парадигми ReAct [22] – автоматичний цикл Thought → Action → Observation.

Інтеграція n8n з Ollama здійснюється через вузол «Ollama Chat Model», який підключається до локального API Ollama за адресою http://127.0.0.1:11434. Вузол підтримує налаштування параметрів генерації: temperature, top_p, top_k, num_predict (максимальна кількість токенів), а також вибір конкретної моделі зі списку завантажених [7].

2.5 Ollama як середовище локального запуску LLM

Ollama [2], [3], [4] – це інструмент з відкритим кодом для запуску, управління та обслуговування великих мовних моделей на локальному комп'ютері. Проєкт заснований Джеффрі Молдейвером (Jeffrey Moldaver) та Майклом Жинь (Michael Jing) у 2023 році; станом на 2025 рік має понад 120000 зірок на GitHub [4] та є найпопулярнішим інструментом локального запуску LLM.

Архітектура Ollama побудована на базі llama.cpp – високоефективної C/C++ бібліотеки для інференсу LLM, що підтримує різні бекенди GPU-прискорення. Для GPU NVIDIA використовується CUDA, для Apple Silicon – Metal, для AMD GPU – ROCm (Linux) або експериментальний Vulkan API (Windows, Linux) [5]. У даній роботі використовується GPU AMD RX 9070 XT (архітектура RDNA 4, 16 ГБ GDDR6 VRAM) з бекендом Vulkan, що вмикається через змінну оточення OLLAMA_VULKAN=1 [6].

Ollama надає REST API, сумісний з форматом OpenAI API, що включає наступні ключові ендпоінти:

– POST /api/chat – генерація відповіді у форматі чату (з підтримкою системного промпту та історії повідомлень);

– POST /api/generate – генерація тексту за промптом;

– GET /api/tags – список завантажених моделей;

– POST /api/pull – завантаження моделі з бібліотеки Ollama.

Управління моделями здійснюється через CLI: ollama pull <model> – завантаження, ollama run <model> – запуск інтерактивного чату, ollama list – перелік завантажених моделей. Бібліотека Ollama включає понад 100 моделей різних розмірів та квантизацій [3].

2.6 Квантизація моделей: формат GGUF та вплив на якість

Квантизація (quantization) – це процес зменшення числової точності ваг моделі з метою зменшення обсягу пам'яті та прискорення інференсу. У повній точності (FP16) кожна вага займає 16 біт (2 байти); при 4-бітній квантизації – лише 4 біти (0.5 байта), що зменшує розмір моделі у 4 рази [31], [32].

Ollama використовує формат GGUF (GPT-Generated Unified Format), розроблений у рамках проєкту llama.cpp [31]. GGUF підтримує різні схеми квантизації, що відрізняються балансом між розміром моделі та якістю генерації. Основні схеми наведено у таблиці 2.3.

Таблиця 2.3 – Схеми квантизації GGUF та їх характеристики (за даними [32], [33])

Схема	Біт/вагу	Розмір 7B	Δ Perplexity	Рекомендація
FP16	16	~14 ГБ	0 (baseline)	Еталон якості
Q8_0	8	~7 ГБ	+0.004	Мінімальна втрата якості
Q6_K	6.6	~5.5 ГБ	+0.009	Висока якість
Q5_K_M	5.7	~5 ГБ	+0.023	Балансована
Q4_K_M	4.8	~4.3 ГБ	+0.054	Рекомендована [32]
Q4_K_S	4.6	~4.1 ГБ	+0.066	Прийнятна
Q3_K_M	3.9	~3.5 ГБ	+0.13	Помітна деградація
Q2_K	3.4	~3 ГБ	+0.87	Суттєва деградація

На рисунку 2.3 можна побачити, що зі зниженням розрядності квантизації perplexity зростає поступово: до рівня Q4 погіршення помірне, а на Q3 і нижче – різке.

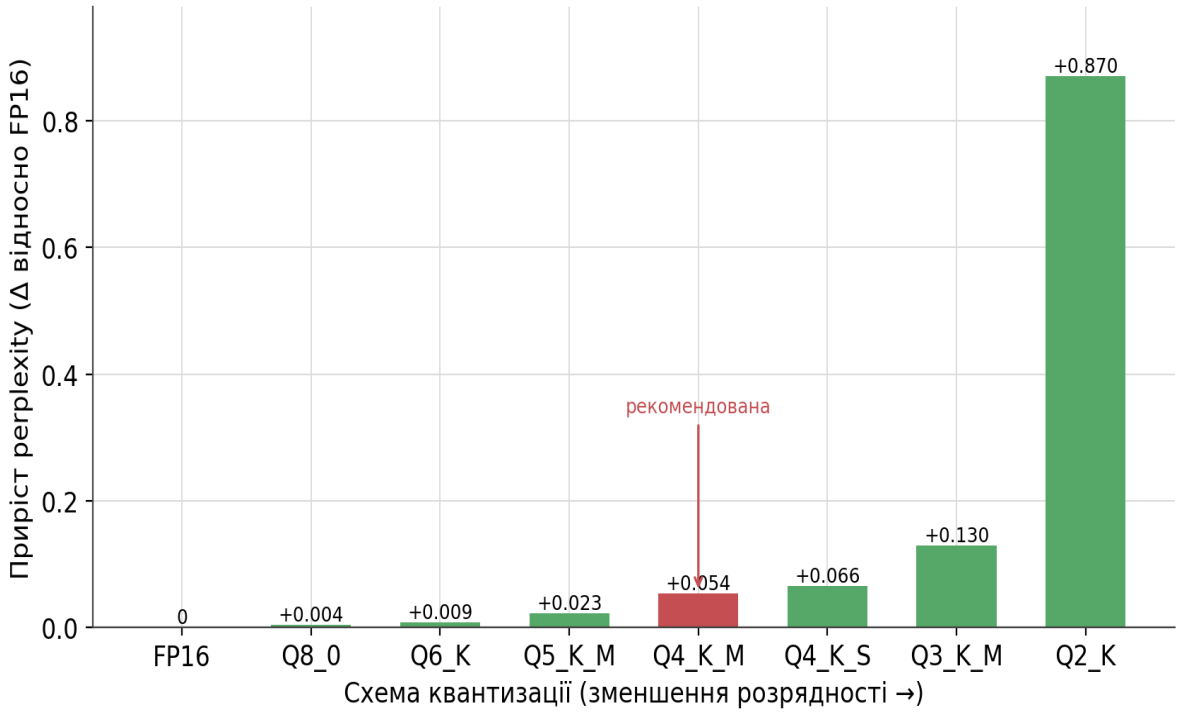


Рисунок 2.3 – Залежність perplexity від рівня квантизації для Llama 3.1 8B (створено за допомогою ШІ)

Систематичне порівняння 13 GGUF-форматів на моделі Llama-3.1-8B-Instruct проведено за бенчмарками MMLU, HellaSwag, GSM8K, IFEval та TruthfulQA [33]. Результати підтверджують, що Q4_K_M забезпечує оптимальний баланс: збільшення perplexity становить лише +0.054, при цьому розмір моделі зменшується у ~3.3 рази. Схеми Q3_K та нижче демонструють помітну деградацію на задачах, що потребують точного слідування інструкціям (IFEval).

2.7 Промпт-інжиніринг для структурованого виводу

Промпт-інжиніринг (prompt engineering) – це процес проєктування текстових інструкцій (промптів) для великих мовних моделей з метою отримання бажаного результату. У контексті даної роботи промпт-інжиніринг є критично важливим, оскільки від якості системного промпту залежить точність класифікації, коректність екстракції даних та стабільність JSON-виводу [23].

Для отримання структурованого JSON-виводу від LLM існують декілька підходів:

– structured outputs (JSON Schema) – механізм, запроваджений OpenAI у серпні 2024 року [4, 5], що гарантує 100% відповідність виводу заданій JSON-схемі на рівні декодера. Доступний лише для моделей OpenAI API, не підтримується локальними моделями через Ollama;

– JSON mode – спрощений режим, при якому модель гарантовано повертає валідний JSON, але без перевірки відповідності конкретній схемі. Підтримується Ollama через параметр format: «json»;

– промпт-інжиніринг – включення детальних інструкцій щодо формату виводу безпосередньо у системний промпт. Це основний підхід, використаний у даній роботі, оскільки він не залежить від можливостей конкретної платформи.

У даній роботі використовується комбінація промпт-інжинірингу та post-processing обробки. Системний промпт включає наступні елементи:

– чітке визначення ролі агента («You are an AI assistant for ISP customer email processing»);

– повний перелік 14 полів JSON-виводу з описом типу та допустимих значень кожного поля;

– enum-обмеження для полів category (6 значень), priority (3 значення), sentiment (3 значення);

– інструкції щодо витягнення contract_id: «число 5–8 цифр між ім'ям та номером телефону без мітки»;

– пояснення, що необхідно аналізувати body листа, а не тему (subject);

– вказівка повертати ЛИШЕ JSON без пояснювального тексту.

Для обробки випадків, коли модель все ж додає текст перед або після JSON (що трапляється у 10–20% відповідей [23]), у Code node n8n реалізовано post-processing за допомогою методів indexOf('{') та lastIndexOf('}'), що витягують JSON-об'єкт з довільного текстового оточення.

2.8 Метрики оцінки якості класифікації та екстракції даних

Для об'єктивного порівняння ефективності моделей у даній роботі використовуються стандартні метрики машинного навчання, адаптовані до специфіки задачі обробки email-звернень.

Accuracy (точність) – відсоток правильних відповідей від загальної кількості прикладів. Для полів з фіксованим набором значень (category, priority, sentiment, requires_response) accuracy обчислюється як exact match з еталонною таблицею [27]:

$$\text{Accuracy} = \frac{\text{кількість збігів з еталоном}}{\text{загальна кількість листів}} \times 100\%. \qquad (4.1)$$

Precision, Recall та F1-score – метрики, що дають детальнішу картину для кожного класу окремо [27]. Precision (точність) – це частка дійсно правильних серед передбачених позитивних, Recall (повнота) – частка знайдених позитивних серед усіх дійсно позитивних, а F1-score – їх гармонійне середнє:

$$\text{Precision} = \frac{TP}{TP+FP}, \tag{4.2}$$

$$\text{Recall} = \frac{TP}{TP+FN}, \tag{4.3}$$

$$F_1 = \frac{2\cdot(\text{Precision}\cdot\text{Recall})}{\text{Precision}+\text{Recall}}, \tag{4.4}$$

де TP (true positives) – кількість прикладів, правильно віднесених до даного класу;

FP (false positives) – кількість прикладів, помилково віднесених до класу;

FN (false negatives) – кількість прикладів класу, які модель не розпізнала [27].

Для мультикласової класифікації використовується macro-усереднення (середнє F1 по всіх класах) [27].

Метрики екстракції даних (NER) – для полів, що витягуються з тексту (sender_name, phone_number, address, contract_id), використовується exact match: результат вважається коректним, якщо витягнуте значення повністю збігається з еталоном (з нормалізацією формату: видалення пробілів, приведення до єдиного формату телефону тощо). Такий підхід дозволяє об'єктивно оцінити здатність моделі точно відтворювати критично важливі дані, де навіть незначна помилка робить результат непридатним для подальшого автоматичного опрацювання. Для поля address, де формулювання може варіюватися, використовується relaxed match: результат вважається коректним, якщо містить ключові елементи

адреси (назва вулиці та номер будинку). Це обумовлено тим, що адреса може бути записана в різних варіантах без втрати змісту, тому надмірно жорстке порівняння призвело б до заниження реальної якості екстракції.

Зведення використовуваних метрик наведено у таблиці 2.4.

Таблиця 2.4 – Метрики оцінки ефективності моделей

Метрика	Застосовується до	Метод порівняння
Accuracy	category, priority, sentiment, requires_response	Exact match з еталоном
F1-score (macro)	category (6 класів)	Precision/Recall/F1 по класах
Exact match	sender_name, phone_number, contract_id	Повний збіг (з нормалізацією)
Relaxed match	address	Збіг ключових елементів
Processing time	Кожний лист	Час у секундах (Date.now())
JSON validity	Кожна відповідь	Успішність JSON.parse()

2.9 Стандартні бенчмарки LLM та їх обмеження

Для загального порівняння можливостей великих мовних моделей у наукових публікаціях та технічних звітах використовуються стандартні бенчмарки. Найбільш поширені серед них:

– MMLU (Massive Multitask Language Understanding) [28] – бенчмарк, що оцінює знання моделі у 57 предметних галузях (від математики до юриспруденції) через завдання з множинним вибором. Із моменту появи у 2020 році MMLU став стандартним тестом загальних знань LLM;

– HellaSwag [29] – бенчмарк здорового глузду (commonsense reasoning), де модель повинна обрати найбільш ймовірне продовження речення серед 4 варіантів, включаючи adversarial-приклади;

– ARC (AI2 Reasoning Challenge) [30] – набір з 7787 наукових питань рівня початкової школи; поділяється на Easy та Challenge підмножини;

– LMArena (Chatbot Arena) [18] – crowd-sourced рейтинг LLM за методикою Elo, де реальні користувачі порівнюють відповіді двох анонімних моделей та обирають кращу.

Результати обраних моделей на основних бенчмарках наведено у таблиці 2.5.

Таблиця 2.5 – Результати моделей на стандартних бенчмарках (за даними технічних звітів [12], [14], [15], [17])

Бенчмарк	Gemma 3 12B	Qwen 3 8B	Llama 3.1 8B	Mistral 7B
MMLU (5-shot)	74.5	72.8	69.4	62.5
HellaSwag	83.2	81.5	79.2	81.0
ARC-Challenge	66.8	65.2	60.1	63.1
HumanEval (code)	60.4	62.1	54.3	30.5

Значення у таблиці – це оцінки відповідних бенчмарків (для MMLU, GSM8K, HellaSwag, IFEval та подібних – відсоток правильних відповідей), узяті з офіційних технічних звітів розробників; вищі значення відповідають кращому результату.

Однак стандартні бенчмарки мають обмеження для оцінки ефективності моделей у конкретній прикладній задачі, такій як обробка email-звернень ISP.

По-перше, всі бенчмарки проводяться англійською мовою та не відображають якість роботи моделей з українським текстом.

По-друге, MMLU, HellaSwag та ARC оцінюють загальні знання та логічні здібності, а не здатність моделі витягувати структуровані дані з неструктурованого тексту, класифікувати звернення за доменними категоріями або формувати валідний JSON.

По-третє, бенчмарки не враховують швидкість генерації, стабільність формату виводу та стійкість до галюцинацій при екстракції даних.

3 ПРОЄКТУВАННЯ СИСТЕМИ ТА ЇЇ РОЗРОБКА

3.1 Проєктування загальної архітектури системи

Система автоматизованої обробки email-звернень побудована як конвеєр (pipeline), що з'єднує три основні компоненти: джерело даних (Gmail), обробний модуль (n8n + Ollama) та сховище результатів (Google Sheets). Загальна архітектура системи представлена на рисунку 3.1.

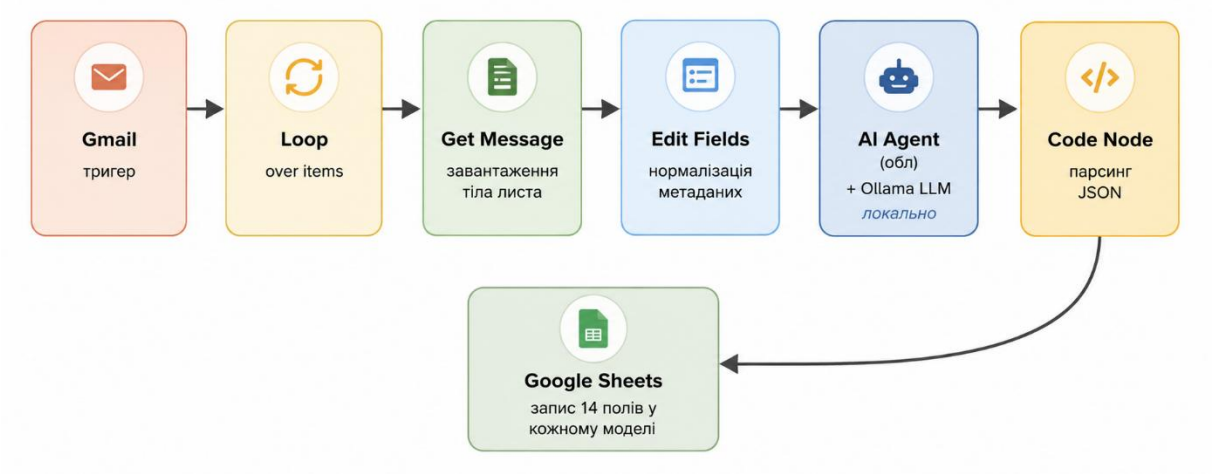


Рисунок 3.1 – Структурна схема конвеєра обробки email-звернень

Потік даних у системі організований наступним чином. Gmail Trigger вузол відстежує появу нових листів у вказаній поштовій скриньці та ініціює запуск робочого процесу (workflow). Кожний новий лист передається у вузол, який забезпечує послідовну обробку листів по одному, що необхідно для коректного вимірювання часу обробки та уникнення перевантаження GPU. Вузол Get Message отримує повний текст листа через Gmail API (параметр Simplify встановлено у false для збереження повної структури). Вузол Edit Fields формує вхідні дані для AI-агента: витягує текст листа ($json.text), message_id, sender_email та фіксує початковий час

обробки (start_time = Date.now()). AI Agent вузол відправляє підготовлений промпт до Ollama, який запускає обрану LLM та повертає текстову відповідь. Code node витягує JSON з відповіді та обчислює processing_time. Фінальний вузол Google Sheets записує результат у відповідну вкладку таблиці.

3.2 Структура робочого процесу n8n

Робочий процес (workflow) у n8n складається з 7 послідовно з'єднаних вузлів. Далі наведено детальний опис кожного вузла та його налаштувань.

Вузол 1: Gmail Trigger. Тип – Trigger. Налаштування: Poll Times – кожні 5 хвилин; Event – Message Received; Labels – INBOX. Вузол перевіряє поштову скриньку на наявність нових листів та ініціює виконання workflow для кожної нової групи листів. Використовує OAuth 2.0 для автентифікації через Google Cloud Console [34].

Вузол 2: Loop Over Items. Тип – Loop. Batch Size – 1. Забезпечує послідовну обробку листів по одному, що є критичним для двох причин: (1) GPU обробляє лише один запит одночасно, паралельна обробка призведе до черги та некоректного вимірювання часу; (2) послідовна обробка дозволяє контролювати навантаження на VRAM.

Вузол 3: Gmail – Get Message. Тип – Action. Operation – Get; Message ID – отримується з тригера ($json.id); Simplify – false (для збереження повної структури листа, включаючи headers). Вузол повертає повний об'єкт повідомлення, включаючи text (тіло листа без HTML-тегів), snippet, from, to, subject, labelIds.

Вузол 4: Edit Fields (Set). Тип – Set. Формує структуру даних для AI-агента:

– email_body – текст листа ($json.text);

– message_id – ідентифікатор повідомлення ($json.id);

– sender_email – email відправника, витягнутий з заголовка From;

– gmail_link – пряме посилання на лист https:// mail.google.com/mail/u/0/#inbox/{message_id};

– start_time – поточний час у мілісекундах (Date.now()) для подальшого обчислення processing_time.

Вузол 5: AI Agent. Тип – AI Agent (LangChain). Підключені суб-вузли: Ollama Chat Model (модель, температура 0.1, num_predict 2048). Системний промпт визначає роль агента, формат виводу, перелік полів JSON та правила класифікації. User Message містить тіло листа з Edit Fields. Детальний опис системного промпту наведено у підрозділі 3.3.

Вузол 6: Code. Тип – Code (JavaScript). Виконує три функції: (1) витягнення JSON з тексту відповіді моделі за допомогою indexOf('{') та lastIndexOf('}'); (2) обчислення processing_time = (Date.now() – start_time) / 1000 (у секундах); (3) додавання полів message_id, gmail_link, sender_email з попередніх вузлів. Детальний опис коду наведено у підрозділі 3.5.

Вузол 7: Google Sheets. Тип – Action. Operation – Append Row; Spreadsheet – ідентифікатор цільової таблиці; Sheet – вкладка, що відповідає назві моделі (gemma3_12b, qwen3_8b, llama31_8b, mistral_7b). Маппінг: 14 полів JSON → 14 колонок таблиці [35].

Описаний робочий процес, схему якого наведено на рисунку 3.2, забезпечує повний цикл автоматизованої обробки вхідної кореспонденції: від моніторингу поштової скриньки до збереження результатів аналізу в базі даних. Використання вузла Loop Over Items гарантує стабільність роботи з локальними LLM-моделями на GPU, запобігаючи перевантаженню відеопам'яті. Кожен вузол конвеєра виконує чітко окреслену функцію, що спрощує налагодження та заміну окремих компонентів без зміни загальної логіки обробки. Така архітектура дозволяє не лише класифікувати листи, а й збирати аналітику щодо продуктивності моделей у реальних умовах експлуатації.

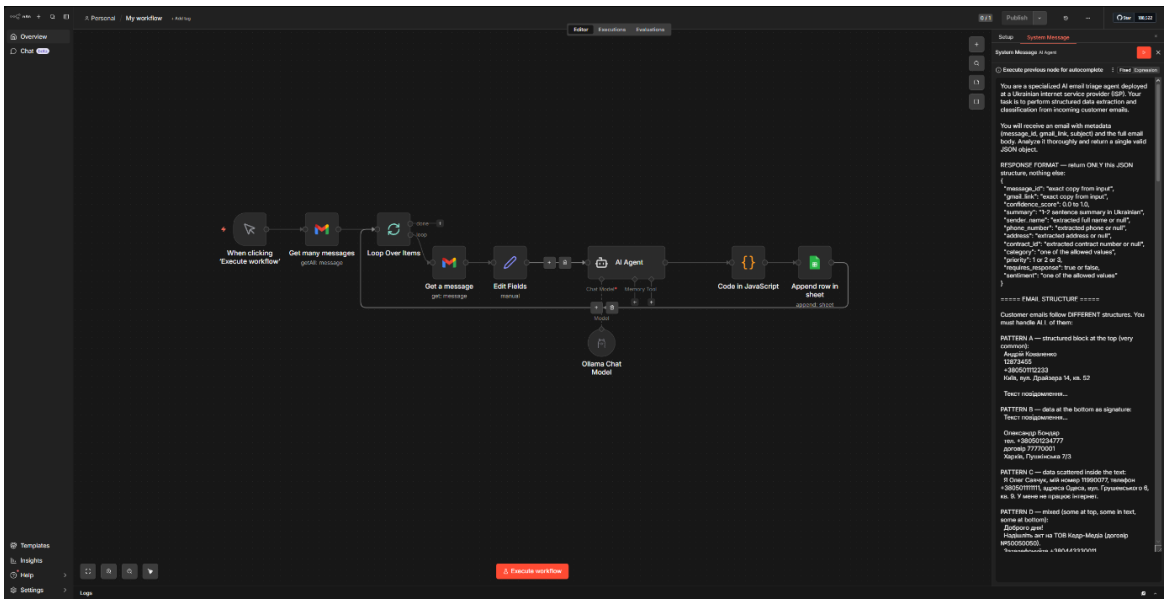


Рисунок 3.2 – Скріншот робочого процесу n8n з усіма вузлами обробки email-звернень

Як видно з рисунка 3.2, робочий процес n8n охоплює повний цикл обробки звернення – від отримання листа до запису результатів у Google Sheets, об'єднуючи вузли тригера, циклу, AI-агента та збереження даних в єдиний конвеєр.

3.3 Розробка системного промпту для AI-агента

Системний промпт є ключовим елементом, що визначає якість роботи агента. Його розробка є ітеративним процесом: початкова версія тестувалася на 10 листах, після аналізу помилок промпт уточнювався, і цикл повторювався до досягнення стабільних результатів. Повний текст розробленого системного промпту наведено у додатку Б.

Структура системного промпту включає наступні блоки.

Блок 1: визначення ролі. Модель отримує чітку роль: «You are an AI assistant that processes customer email messages for an Internet Service

Provider (ISP). Your task is to analyze the email and extract structured data in JSON format».

Визначення ролі допомагає моделі сфокусуватися на доменній задачі та уникати загальних відповідей [23].

Блок 2: специфікація JSON-виводу. Повний перелік 14 полів з описом типу, обов'язковості та допустимих значень. Для enum-полів наведено повний список значень: category (техпідтримка | бухгалтерія | підключення | відключення | спам_реклама | інше), priority (1 | 2 | 3), sentiment (positive | neutral | negative). Для boolean-поля requires_response вказано очікуваний тип (true/false).

Блок 3: правила класифікації. Детальні інструкції для визначення категорії: «If the email mentions internet speed, connection problems, router issues → техпідтримка. If the email mentions payment, invoice, debt, balance → бухгалтерія. If the email is a promotional newsletter, auto-notification, or irrelevant → спам_реклама».

Правила для priority: «1 – no response needed (spam, auto-notifications); 2 – not urgent (general questions, planned requests); 3 – urgent (outages, angry complaints, threats to cancel)».

Блок 4: інструкції з екстракції contract_id. Спеціальний блок, обумовлений особливістю українських ISP: «The contract_id is typically a 5 – 8 digit number found between the sender's name and phone number, WITHOUT any label like 'Contract:' or 'Agreement №'. If you see a standalone number in this position, treat it as contract_id».

Блок 5: формат виводу. Суворі інструкції: «Respond ONLY with a valid JSON object. Do NOT add any text before or after the JSON. Do NOT wrap the JSON in markdown code blocks».

Ця вказівка зменшує, але не усуває проблему додаткового тексту, що вимагає post-processing у Code node.

Повний текст системного промпту наведено у додатку Б.

3.4 Конфігурація Ollama та GPU-прискорення

Налаштування Ollama на робочій станції з GPU AMD RX 9070 XT потребує додаткових кроків порівняно зі стандартною інсталяцією для NVIDIA GPU [5], [6].

Апаратна конфігурація робочої станції:

– CPU: AMD Ryzen 7 7800X3D;

– GPU: AMD Radeon RX 9070 XT (архітектура RDNA 4, 16 ГБ GDDR6 VRAM);

– RAM: 32 ГБ DDR5;

– OS: Windows 11;

– Ollama: остання стабільна версія.

Для активації GPU-прискорення через Vulkan API необхідно встановити постійну системну змінну оточення OLLAMA_VULKAN=1 [5], [6]. Без цієї змінної Ollama на Windows за замовчуванням використовує CPU для інференсу моделей на AMD GPU, що призводить до збільшення часу обробки у 5–10 разів. Перевірити, що GPU використовується, можна командою ollama ps, яка відображає поточне навантаження GPU.

Додатково виявлено проблему з підключенням n8n до Ollama: при використанні адреси localhost у вузлі Ollama Chat Model виникає помилка з'єднання через невірну резолюцію IPv6 (localhost → ::1 замість 127.0.0.1). Рішення – використання явної адреси http://127.0.0.1:11434 замість http://localhost:11434.

Завантаження моделей виконується командами CLI:

– ollama pull gemma3:12b;

– ollama pull qwen3:8b;

– ollama pull llama3.1:8b;

– ollama pull mistral:7b.

3.5 Обробка відповіді моделі – Code node

Як зазначалося у підрозділі 2.7, моделі часто додають пояснювальний текст перед або після JSON-об'єкта. Для надійного витягнення JSON з довільного тексту у Code node реалізовано наступний алгоритм:

– отримання текстової відповіді моделі з вузла AI Agent ($json.output);

– пошук першої відкриваючої фігурної дужки: const start = text.indexOf('{');

– пошук останньої закриваючої фігурної дужки: const end = text.lastIndexOf('}');

– витягнення підрядка: const jsonStr = text.substring(start, end + 1);

– парсинг JSON: const result = JSON.parse(jsonStr);

– обчислення processing_time: const time = (Date.now() – $json.start_time) / 1000;

– додавання метаданих (message_id, gmail_link, sender_email, processing_time);

– повернення фінального JSON-об'єкта з 14 полями.

У випадку невдалого парсингу (JSON.parse кидає виключення) вузол повертає об'єкт з полем error: true та записує raw-відповідь моделі для подальшого аналізу помилки. Такий підхід дозволяє продовжувати обробку наступних листів навіть при помилці на одному з них.

3.6 Інтеграція з Google Sheets

Google Sheets використовується як сховище результатів обробки. Для кожної моделі створена окрема вкладка (sheet) в одній Google-таблиці. Кожна вкладка має 14 колонок, що відповідають полям JSON-виводу. Такий підхід дозволяє легко порівнювати результати різних моделей та формувати зведені звіти.

Підключення до Google Sheets API [35] реалізовано через OAuth 2.0 за допомогою Google Cloud Console: створено проєкт, увімкнено Gmail API та Google Sheets API, створено OAuth 2.0 Client ID (тип Desktop application), налаштовано redirect URI. Credentials зберігаються у n8n та використовуються вузлами Gmail Trigger, Gmail Get Message та Google Sheets.

3.7 Розробка тестового набору даних

Для об'єктивної оцінки ефективності моделей розроблено тестовий набір із 100 email-звернень, що імітують реальні листи клієнтів інтернет-провайдера. Листи написані українською мовою у вільній формі, з характерною розмовною лексикою та граматичними помилками, що відповідає реальній мовній ситуації в Україні.

Тестовий набір наведено у таблиці 3.1, він покриває наступний розподіл категорій.

Таблиця 3.1 – Розподіл тестових листів за категоріями

Категорія	Кількість листів	Частка
техпідтримка	26	26%
бухгалтерія	20	20%
підключення	16	16%
інше	15	15%
відключення	13	13%
спам_реклама	10	10%
Разом	100	100%

При формуванні тестового набору враховано наступні принципи:

– різноманітність стилів написання – від формальних до емоційних;

– наявність листів з оманливою темою (subject не відповідає body);

– листи з неповними даними (відсутній номер договору або адреса);

– листи з декількома проблемами одночасно (multi-label);

– листи різної довжини – від 2 до 15 речень;

– спам-листи різного характеру (рекламні розсилки, автоматичні сповіщення);

– листи з contract_id без мітки (просте число між ім'ям та телефоном).

Для кожного листа вручну визначено еталонні значення всіх 14 полів та занесено до еталонної таблиці. Вона використовується для автоматичного порівняння результатів моделей та обчислення метрик точності.

Відправка тестових листів реалізована через окремий workflow у n8n: Google Sheets (читання файлу emails_to_send.xlsx з полями subject та body) → Loop → Gmail (відправка кожного листа на тестову поштову скриньку). Приклади тестових листів наведено у додатку А.

3.8 Обмеження та проєктні рішення

Реалізована система є дослідницьким прототипом, спрямованим на коректне порівняння моделей, а не готовим промисловим продуктом. Низку її характеристик зумовлено свідомими проєктними рішеннями, прийнятими задля забезпечення відтворюваності та чистоти експерименту. У цьому підрозділі систематизовано ці рішення разом із обмеженнями, які вони накладають.

Послідовна обробка звернень. Конвеєр опрацьовує листи строго по одному (Batch Size = 1 у вузлі Loop), оскільки в робочій станції встановлено єдиний GPU, а Ollama обслуговує запити через спільну чергу. Паралельна подача не пришвидшила б обробку, натомість унеможливила б коректне вимірювання processing_time. Платою за це є обмежена пропускна здатність прототипу, яку у промисловому впровадженні можна підвищити горизонтальним масштабуванням.

Односкроковий агент без зовнішніх інструментів. Хоча вузол AI Agent підтримує парадигму ReAct із викликом інструментів, у системі агент працює в односкроковому режимі: на одне звернення припадає рівно один виклик моделі. Інструменти свідомо не підключалися, адже класифікація, екстракція та аналіз тональності не потребують зовнішніх запитів, а багатокрокова взаємодія підвищує ризик інкрементальних галюцинацій [25] та робить час обробки залежним від кількості ітерацій. Таким чином, односкрокова схема забезпечує чистішу основу для зіставлення.

Синтетичний характер тестової вибірки. Набір звернень сформовано вручну з імітацією реальних листів, що дозволяє гарантовано охопити всі шість категорій і навмисно закласти складні випадки (оманлива тема, contract_id без мітки, розмовна лексика, multi-label), які у випадковій вибірці траплялися б нерівномірно. Водночас синтетична вибірка не відтворює статистику реального потоку, тому абсолютні значення метрик слід трактувати як орієнтовні; відносне впорядкування моделей при цьому залишається показовим.

Обсяг вибірки та статистична похибка. Обсяг тестової вибірки обрано як компроміс між трудомісткістю ручної розмітки та репрезентативністю. За такого обсягу довірчий інтервал для окремої метрики становить орієнтовно $\pm 5 – 10$ відсоткових пунктів, тому близькі результати (наприклад, ≈84% і ≈88%) не можна вважати статистично розрізненними. З огляду на це у роботі акцентовано на виражених відмінностях, а не на дрібних розбіжностях.

Експериментальний статус бекенду Vulkan. Запуск моделей на GPU AMD здійснюється через бекенд Vulkan, який в Ollama має статус експериментального [5], [6], що зумовлює можливі розбіжності у швидкості між версіями. GPU розглядається як наявне апаратне забезпечення, а не як предмет дослідження, тому абсолютні значення часу обробки прив'язані до цієї конфігурації; натомість співвідношення швидкостей між моделями зберігатиметься.

Ручна фіксація аномалій виводу. Окремі некоректні відповіді моделей – значення поза переліком enum, орфографічні спотворення міток, вигадані дані – не фіксуються системою автоматично, оскільки модель не здатна самостійно сигналізувати про власну галюцинацію. Такі випадки виявлено під час звірення з еталонною таблицею та внесено до підсумкових таблиць розділу 4 вручну. Автоматичне виявлення аномалій потребувало б окремого валідаційного шару, що виходить за межі прототипу.

Фіксований набір полів виводу. Структуру виводу обмежено 14 полями (підрозділ 1.6); поля, що не стосуються предмета порівняння, до схеми не включалися. Це спрощує зіставлення вкладок Google Sheets між моделями, проте для промислового застосування схему доведеться розширити під потреби CRM провайдера.

Усі перелічені рішення та обмеження не знижують цінності отриманих результатів, а навпаки – чітко окреслюють межі, в яких ці результати є коректними та відтворюваними. Свідома відмова від низки промислових оптимізацій на користь простоти й прозорості конфігурації забезпечує головну мету роботи – коректне й неупереджене порівняння моделей за однакових умов. Важливо підкреслити, що більшість зазначених обмежень мають не принциповий, а суто інженерний характер і можуть бути усунені на етапі переходу від прототипу до повноцінного промислового рішення. Сформульовані обмеження водночас визначають перспективні напрями подальшого вдосконалення системи: перехід до паралельної обробки звернень, розширення тестової вибірки реальними даними, додавання валідаційного шару для автоматичного виявлення аномалій та інтеграцію з CRM-системами провайдера. Окремий інтерес у подальших дослідженнях становить також перевірка стійкості отриманих закономірностей на більших обсягах даних та за інших апаратних конфігурацій. Таким чином, розроблений прототип слід розглядати як методологічно обґрунтовану основу для порівняльного дослідження, результати якого наведено в наступному розділі.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

4.1 Методика проведення експерименту

Експеримент проведено за наступною методикою. Тестовий набір email-звернень послідовно оброблений кожною з чотирьох моделей в ідентичних умовах: одна й та сама робоча станція, однаковий системний промпт, однакові параметри генерації (temperature = 0.1, num_predict = 2048), послідовна обробка по одному листу. Перед запуском кожної моделі виконувалося «прогрівання» (warm-up) – обробка 3 тестових листів, результати яких не враховувалися, для стабілізації часових показників (завантаження моделі у VRAM, ініціалізація KV-кешу).

Результати обробки кожною моделлю (по 14 полів для кожного листа) зберігались у відповідній вкладці Google Sheets та порівнювались з еталонною таблицеюза метриками, визначеними у підрозділі 2.8.

4.2 Результати порівняльного аналізу

Зведені результати порівняльного аналізу чотирьох моделей за основними метриками наведено у таблиці 4.1.

Таблиця 4.1 – Зведені результати порівняльного аналізу моделей

Метрика	Gemma 3 12B	Qwen 3 8B	Llama 3.1 8B	Mistral 7B
Accuracy: category	≈82%	≈88%	≈68%	≈84%
Accuracy: sentiment	≈80%	≈78%	≈72%	≈80%
Accuracy: priority	≈86%	≈78%	≈84%	≈78%
Accuracy: requires_response	≈100%	≈98%	≈100%	≈100%
Extract: contract_id	88%	98%	60%	48%
Extract: phone_number	100%	100%	100%	100%
Extract: sender_name	≈96%	≈98%	≈92%	≈82%

Продовження таблиці 4.1

Метрика	Gemma 3 12B	Qwen 3 8B	Llama 3.1 8B	Mistral 7B
Extract: address	100%	≈98%	≈96%	≈98%
Сер. час (с/лист)	7,0	8,9	2,2	3,1

4.3 Аналіз результатів класифікації

За точністю класифікації категорій (category) найкращий результат продемонструвала Qwen 3 8B (≈88%); дещо нижчі показники в Mistral 7B (≈84%) та Gemma 3 12B (≈82%), а Llama 3.1 8B – найнижчий результат ≈68%. Висока точність Qwen 3 пояснюється найбільшим обсягом навчальних даних (36T токенів) [14] та кращою підтримкою мультимовності – 119 мов. Низький результат Llama 3.1 обумовлений відсутністю офіційної підтримки української мови (лише 8 мов) [16] та найгіршою fertility токенізатора для українського тексту (2.01) [26].

За точністю визначення тональності (sentiment) найкращі результати показали Gemma 3 12B та Mistral 7B (≈80%), дещо нижчий – Qwen 3 8B (≈78%), а найнижчий – Llama 3.1 8B (≈72%). Результат Gemma 3 частково пояснюється кількістю параметрів (12 млрд) та широким словником (256K), що дозволяє краще «відчувати» емоційні нюанси тексту; водночас зіставний показник значно меншої Mistral 7B свідчить, що розмір моделі тут не є вирішальним. Нижчу точність Llama 3.1 можна пов'язати з її найгіршою для української мови ефективністю токенізації, через яку емоційне забарвлення тексту вловлюється гірше.

4.4 Аналіз результатів екстракції даних

Екстракція номера телефону (phone_number) виявилася найпростішою задачею: всі чотири моделі досягли 100% точності. Це пояснюється чітким, однозначним форматом українських номерів

телефонів (+380…), який легко розпізнається моделями. Наявність сталого префікса та фіксованої довжини номера фактично зводить задачу до пошуку характерного шаблону, що не потребує глибокого семантичного аналізу.

Екстракція номера договору (contract_id) продемонструвала найбільший розкид: від 98% (Qwen 3) до 48% (Mistral 7B). Як зазначалося у підрозділі 1.2, contract_id у листах клієнтів ISP часто виглядає як просте 5 – 8-значне число без мітки, що робить його витягнення складною задачею. На відміну від телефону, тут модель змушена спиратися саме на контекст, а не на формальний шаблон. Qwen 3 8B, завдяки великому обсягу навчальних даних та кращій мультимовності, найуспішніше розпізнає контекст: число між ім'ям та номером телефону = номер договору. Mistral 7B з найменшим словником (32K) та найгіршою fertility (2.42) для української мови [26] найчастіше ігнорує або неправильно інтерпретує цей контекст. Порівняння точності екстракції даних за всіма полями для чотирьох моделей наведено на рисунку 4.1.

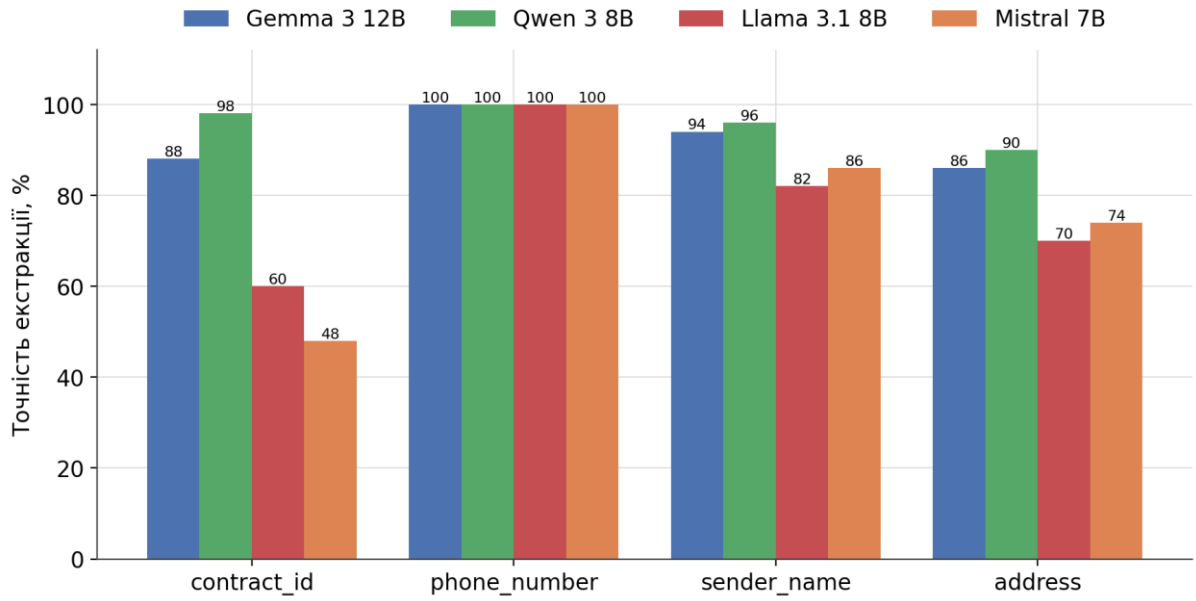


Рисунок 4.1 – Порівняння точності екстракції даних: contract_id, phone_number, sender_name, address для 4 моделей (створено за допомогою ШІ)

4.5 Аналіз швидкості обробки

Середній час обробки одного листа суттєво відрізняється між моделями: від 2.2 с (Llama 3.1) до 8.9 с (Qwen 3). Цей розрив пояснюється декількома факторами:

– кількість параметрів: Gemma 3 12B (12 млрд) обробляє лист у 7.0 с, тоді як Mistral 7B (7.3 млрд) – у 3.1 с. Більше параметрів = більше матричних множень на кожному шарі = більше часу;

– режим мислення Qwen 3: гібридний thinking-mode [14] призводить до генерації додаткових «думок» (enclosed в тегах <think>...</think>) перед формуванням відповіді, що збільшує кількість генерованих токенів та час обробки;

– fertility токенізатора: Llama 3.1, попри гіршу fertility (2.01), демонструє найшвидшу обробку (2.2 с), що пояснюється меншою кількістю шарів (32 vs 48 у Gemma) та оптимізаціями архітектури.

Детальні показники швидкості обробки для кожної моделі наведено в таблиці 4.2.

Таблиця 4.2 – Швидкість обробки: деталізація

Модель	Мін. час (с)	Макс. час (с)	Середній час (с)	Медіана (с)
Llama 3.1 8B	2	9	2,2	2
Mistral 7B	2	9	3,1	3
Gemma 3 12B	6	7	7,0	7
Qwen 3 8B	7	21	8,9	8

Як видно з таблиці 4.2, найстабільнішу обробку демонструють Gemma 3 та Mistral 7B з мінімальним розкидом часу, тоді як у Qwen 3 спостерігається найбільший максимум (21 с) через режим мислення. Такий разовий сплеск виникає, коли модель у режимі розгорнутого міркування генерує велику кількість службових токенів перед підсумковою відповіддю.

4.6 Аналіз типових помилок та обмежень

На основі аналізу результатів можна виділити наступні типові помилки моделей:

– змішування категорій техпідтримка та інше – листи, у яких клієнт емоційно скаржиться на технічну проблему, часто класифікуються як інше замість техпідтримка або навпаки. Ця помилка найчастіше трапляється у Llama 3.1 та Mistral 7B;

– пропуск contract_id – моделі (особливо Mistral 7B та Llama 3.1) часто залишають поле contract_id порожнім або повертають null, не розпізнавши число без мітки як номер договору;

– галюцинації в адресі – у декількох випадках моделі «вигадували» номер квартири або поверху, яких не було у тексті листа. Ця проблема найчастіше проявлялась у Llama 3.1 (≈2% листів);

– невалідний JSON – Llama 3.1 у ≈12% випадків генерувала JSON з помилками (незакриті дужки, відсутні коми), що вимагало post-processing або ручного виправлення;

– нечутливість до іронії – усі моделі мають труднощі з розпізнаванням іронії та сарказму в українському тексті, що призводить до невірного визначення sentiment.

Окрему увагу варто приділити аномаліям виводу, які не зводяться до звичайної неточності класифікації, а свідчать про збій на рівні формування самого значення поля. Найвиразніше такі аномалії проявилися у Llama 3.1 – моделі з найгіршою для української мови ефективністю токенізації (fertility 2.01) [26] та відсутністю офіційної підтримки української [16]. Виявлено два якісно різні типи таких збоїв: повернення значення поза переліком допустимих та орфографічне спотворення коректного значення.

Повернення неіснуючої категорії. У системному промпті поле category жорстко обмежене переліком із шести значень (техпідтримка,

бухгалтерія, підключення, відключення, спам_реклама, інше). Попри це, Llama 3.1 в окремих випадках повертала у цьому полі значення, відсутнє у переліку, – тобто мітку, що не збігається з жодною з шести дозволених категорій. Формально таке значення не є галюцинацією щодо змісту звернення: модель правильно зрозуміла, що йдеться про технічну проблему, проте згенерувала для неї власну мітку замість приписаної категорії техпідтримка. Причина лежить у площині декодування: enum-обмеження задане лише текстом промпта, а не примусовим обмеженням словника на рівні декодера (constrained decoding), якого локальний запуск через Ollama не забезпечує. Модель розкладає україномовну мітку (наприклад, техпідтримка) на кілька субслівних токенів і на кожному кроці генерації може відхилитися до статистично ближчого, але недозволеного варіанта. Для системи такий вивід рівноцінний помилці класифікації, оскільки подальша маршрутизація за невідомою категорією неможлива.

Орфографічне спотворення мітки. Другий тип аномалії – повернення значення, яке за змістом є правильним, але містить орфографічну помилку в самому записі мітки (наприклад, замість мітки техпідтримка модель повертала спотворений рядок «тезпидуца»). Такий збій безпосередньо пов’язаний із субсловною токенізацією: україномовна мітка не входить до словника як єдиний токен, а збирається з кількох фрагментів, і за низької ймовірності одного з них модель може пропустити або підмінити символ. Хоча людина-оператор легко впізнає намір, для автоматичного конвеєра такий рядок не збігається з жодним очікуваним значенням і відкидається нарівні з повністю невірною категорією. Цей випадок показовий тим, що джерелом помилки є не нерозуміння звернення, а механіка генерації багатотокенної мітки моделлю з неоптимальним для української мови токенізатором.

Показово, що такі спотворення можуть накладатися в межах одного виводу: в одному зі звернень Llama 3.1 одночасно повернула невалідну категорію «тезпидуца» та орфографічно спотворене значення

тональності «негатівній» (замість «негативний»), причому з максимальним показником confidence_score = 1,0. Це свідчить про непоказовість самооцінки впевненості: висока заявлена впевненість не корелює з коректністю виводу. Зафіксовано і протилежний випадок – confidence_score = 0,0 для коректно опрацьованого рекламного листа. Окремо відзначено нестабільність часу обробки: для Qwen 3 8B зафіксовано викид у 21 с при середньому значенні 8,9 с, що вказує на залежність латентності від довжини й складності конкретного звернення.

Спільною рисою обох аномалій є те, що модель не повідомляє про них: вона повертає спотворене значення з тією самою впевненістю, що й коректне, а поле confidence_score не знижується. Тому, як зазначено у підрозділі 3.8, такі випадки виявлено вручну під час звірення з еталонною таблицею. Практичний висновок: для моделей зі слабкою підтримкою цільової мови недостатньо текстового enum-обмеження у промпті – надійність потребує додаткового валідаційного шару, який звіряв би повернуту мітку з переліком допустимих і за розбіжності відхиляв запис або відправляв звернення на повторну обробку.

Для ілюстрації характеру помилок розглянемо три показові випадки з тестової вибірки, що відповідають типовим класам збоїв.

Приклад 1. Спотворення мітки категорії. Звернення містило опис технічної несправності («тиждень немає інтернету») разом з емоційно забарвленою претензією та погрозою розірвати договір; еталонне значення category – техпідтримка. Qwen 3 і Gemma 3 повернули коректну мітку техпідтримка, тоді як Llama 3.1 згенерувала спотворений рядок «тезпидуца». Модель правильно розпізнала технічний характер звернення, проте через субсловну токенізацію україномовної мітки припустилася помилки в самому її записі. Для автоматичного конвеєра такий рядок не збігається з жодним із допустимих значень і відкидається нарівні з повністю невірною категорією.

Приклад 2. Лист із contract_id без мітки. У тілі листа номер договору був поданий як окреме п'ятизначне число між прізвищем клієнта та номером телефону, без підпису на кшталт «Договір №». Qwen 3 правильно витягла це число, спираючись на позиційний контекст, тоді як Mistral 7B залишила поле порожнім, а в одному з варіантів сприйняла число як фрагмент адреси. Випадок підтверджує, що екстракція беземітних числових даних критично залежить від здатності моделі утримувати позиційний контекст: модель з найменшим словником (32K) і найгіршою fertility (2.42) дробить україномовне оточення на велику кількість токенів, через що зв'язок «число між іменем і телефоном = номер договору» втрачається.

Приклад 3. Галюцинація в полі address. У кількох листах, де клієнт вказував лише вулицю та номер будинку, Llama 3.1 додавала неіснуючий номер квартири або поверху, відсутній у тексті. Це класична галюцинація-доповнення: модель «добудовує» адресу до звичного повного формату, спираючись на статистичні шаблони, а не на зміст звернення [24]. Небезпека такої помилки вища за звичайний пропуск поля, оскільки вивід виглядає правдоподібно й може бути некритично перенесений у CRM. Релаксований режим порівняння для address частково згладжує оцінку, проте у промисловому застосуванні подібні доповнення потребують валідації за довідником адрес провайдера.

4.7 Інтерпретація виявлених кореляцій

Зіставлення метрик розділу 4 з архітектурними характеристиками моделей (таблиця 2.2) дозволяє виявити кілька стійких залежностей. Через невеликий обсяг вибірки (підрозділ 3.8) вони трактуються не як строгі статистичні кореляції, а як якісні закономірності.

Ефективність токенізації та точність класифікації. Простежується зворотний зв'язок між значенням fertility для української мови та точністю класифікації. Llama 3.1 з найгіршою fertility (2.01) показала найнижчу

точність по category (≈68%), тоді як Gemma 3 і Qwen 3 з кращою токенізацією (1.63 і 1.78) втримали вищі результати. Надмірне дроблення українських слів на субслівні токени розмиває семантичні межі: модель отримує не цілісні слова, а послідовність дрібних фрагментів, через що зв'язки між частинами звернення вловлюються гірше. При цьому Llama 3.1 є найшвидшою (2.2 с/лист), тобто погана токенізація знижує саме якість розуміння, а не швидкість.

Розмір словника та екстракція contract_id. Найвиразніша залежність простежується для екстракції номера договору. Результати утворюють чітку спадну послідовність за обсягом словника: Qwen 3 (152K) – 98%, Gemma 3 (256K) – 88%, Llama 3.1 (128K) – 60%, Mistral 7B (32K) – 48%. Виняток Gemma 3, словник якої найбільший, але результат нижчий за Qwen 3, пояснюється тим, що сам по собі обсяг словника не тотожний якості представлення української: вирішальну роль відіграє частка кириличних субслів і обсяг україномовних даних у навчанні, за яким Qwen 3 (36T токенів) переважає. Загальна ж тенденція підтверджує, що витягнення беземітного числа з україномовного оточення потребує цілісного утримання позиційного контексту, а надмірно фрагментована токенізація (Mistral 7B) цей контекст руйнує.

Кількість параметрів та аналіз тональності. Точність визначення sentiment виявляє зв’язок із розміром моделі: Gemma 3 12B показала ≈80% – нарівні з Mistral 7B, тобто перевага за розміром тут не є вирішальною. Це лише частково узгоджується з тим, що розпізнавання емоційного забарвлення тексту – тонша семантична задача, ніж класифікація за формальними ознаками, і потребує ширшої репрезентативної здатності моделі. Водночас залежність не є монотонною: значно менша Mistral 7B (≈80%) не поступилася більшій Qwen 3 (≈78%), а різниця між трьома найкращими моделями лежить у межах довірчого інтервалу вибірки і не є статистично значущою. Це вказує, що для тональності поряд із розміром суттєвими є особливості навчальних даних і

токенізації, а режим розгорнутого міркування Qwen 3 у цій задачі переваги не дає.

Кількість шарів та швидкість обробки. Час обробки прямо пов'язаний з обчислювальним обсягом моделі – добутком кількості шарів на розмірність представлення. Gemma 3 з 48 шарами опрацьовує лист за 7.0 с, тоді як 32-шарові Llama 3.1 і Mistral 7B – за 2.2 і 3.1 с. Окремий чинник – режим мислення Qwen 3: попри лише 36 шарів, ця модель виявилася найповільнішою (8.9 с) через генерацію додаткових службових токенів. Отже, на швидкість впливає не лише глибина мережі, а й кількість згенерованих токенів.

Узагальнюючи, найсильніші залежності спостерігаються у парах «ефективність токенізації – якість розуміння української» та «обчислювальний обсяг – швидкість». Жодна модель не є оптимальною за всіма критеріями одночасно: точність Qwen 3 досягається ціною часу обробки, а швидкість Llama 3.1 – ціною якості. Це обґрунтовує вибір моделі за пріоритетами організації, що деталізовано у наступному підрозділі.

4.8 Рекомендації щодо вибору моделі

На основі проведеного порівняльного аналізу можна сформулювати наступні рекомендації (таблиця 4.4).

Таблиця 4.4 – Рекомендації щодо вибору моделі залежно від пріоритетів

Пріоритет	Модель	Обґрунтування
Максимальна точність класифікації	Qwen 3 8B	≈88% accuracy по category, 98% contract_id
Максимальна точність sentiment	Gemma 3 12B	≈80% accuracy по sentiment
Мінімальний час обробки	Llama 3.1 8B	2.2 с/лист (у 4х швидше за Qwen 3)

Продовження таблиці 4.4

Пріоритет	Модель	Обґрунтування
Баланс якість/швидкість	Mistral 7B	≈84% category, 3.1 с/лист
Мінімальні вимоги до VRAM	Mistral 7B	7.3 млрд параметрів, ~4.3 ГБ (Q4_K_M)

4.9 Перспективи розвитку

Подальший розвиток системи може здійснюватися за наступними напрямками:

– fine-tuning моделей на доменних даних – дообучення обраної моделі наприклад, Qwen 3 8B на реальних email-зверненнях конкретного ISP може підвищити точність класифікації на 5–15%;

– RAG (Retrieval-Augmented Generation) – інтеграція бази знань провайдера (тарифи, послуги, FAQ) для автоматичного формування відповідей клієнтам;

– обробка вкладень – розширення конвеєра для аналізу вкладених файлів (скріншоти помилок, фото обладнання) з використанням мультимодальних можливостей Gemma 3 [12];

– інтеграція з CRM – автоматичне створення тикетів у CRM-системі провайдера (наприклад, Odoo, 1С) на основі структурованих даних;

– масштабування на інші канали – розширення системи для обробки повідомлень з Telegram, Viber та чат-ботів на сайті провайдера.

Реалізація наведених напрямків дозволить трансформувати дослідницький прототип у повноцінний інструмент автоматизації служби підтримки. Найбільш пріоритетним напрямком є дообучення обраної моделі на реальних даних конкретного оператора, що здатне забезпечити найвідчутніший приріст точності за відносно невеликих витрат. Отримані результати порівняльного аналізу є надійною основою для подальшого розвитку системи та її адаптації до потреб конкретного провайдера.

ВИСНОВКИ

У кваліфікаційній роботі вирішено задачу реалізації та порівняльного аналізу ефективності агентних інтелектуальних систем на базі чотирьох локальних великих мовних моделей для автоматизованої обробки клієнтських email-звернень інтернет-провайдера. Поставлену мету досягнуто: спроєктовано й реалізовано працездатний конвеєр обробки звернень, проведено експериментальне дослідження та сформульовано обґрунтовані рекомендації щодо вибору моделі.

Проведено аналіз предметної галузі. Встановлено, що ринок є високофрагментованим – понад 4100 провайдерів, з яких 63% становлять фізичні особи-підприємці, а 72% доходу розподіляється між тисячами малих та середніх операторів без ресурсів для дорогих корпоративних систем підтримки, що підтверджує потребу в доступному інструменті автоматизації. Систематизовано вісім ключових проблем автоматизації обробки звернень та виконано огляд шести категорій наявних рішень.

Досліджено теоретичні основи роботи великих мовних моделей: архітектуру Transformer та її decoder-only різновид, механізми self-attention і багатоголової уваги, принцип авторегресивної генерації. Проаналізовано вплив токенізації на обробку українського тексту: fertility для української мови варіюється від 1.63 у Gemma 3 до 2.42 у Mistral 7B. Розглянуто механізм квантизації GGUF, методи промпт-інжинірингу та виконано архітектурну характеристику кожної моделі.

Спроєктовано та реалізовано тестовий прототип у середовищі n8n із інтеграцією Ollama для локального запуску моделей. Конвеєр складається із семи послідовних вузлів, розроблено єдиний системний промпт для AI-агента, вузол post-processing для витягнення JSON та інтеграцію з Google Sheets через OAuth 2.0. Усі компоненти працюють локально, без передачі клієнтських даних на зовнішні сервери.

Розроблено тестовий набір україномовних email-звернень, що охоплює шість категорій і відтворює реальні особливості кореспонденції. Для кожного листа вручну сформовано еталонні значення всіх 14 полів, що склали еталонну таблицю для верифікації результатів.

Проведено серію експериментів та виконано порівняльний аналіз моделей за точністю класифікації, якістю екстракції даних і швидкістю обробки. Основні результати:

– Qwen 3 8B – найвища точність класифікації (≈88%) та екстракції номера договору (98%);

– Gemma 3 12B – один із найкращих результатів аналізу тональності (≈80%);

– Llama 3.1 8B – найшвидша (2.2 с на лист), проте найнижча точність класифікації (≈68%);

– Mistral 7B – оптимальний баланс якості та швидкості (≈84% за 3.1 с на лист) і найнижчі вимоги до відеопам'яті;

– усі чотири моделі досягли 100% точності екстракції номера телефону.

Виявлено якісні закономірності, що пояснюють відмінності між моделями: зворотний зв'язок між ефективністю токенізації української мови та точністю класифікації, залежність якості екстракції номера договору від характеристик словника. Окремо досліджено токенізаційні аномалії Llama 3.1, джерелом яких є генерація англомовних міток моделлю з неоптимальним для української мови токенізатором.

Підтверджено практичне значення роботи: прототип скорочує час первинної обробки звернення з 4–7 хвилин ручної роботи до 2–9 секунд автоматичної, тобто у десятки разів, за точності класифікації 68–88%. Наукова цінність полягає в тому, що порівняння виконано не на загальних англомовних бенчмарках, а на прикладній україномовній задачі. Подальший розвиток доцільно спрямувати на дообучення моделі на реальних доменних даних та розширення конвеєра на інші канали комунікації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. 4100 провайдерів сьогодні дають інтернет 8,5 млн абонентам. *HiTech.Expert*. 2024. URL: https://expert.com.ua/191162-4100-provayderiv-sogodni-dayut-internet-85-mln-abonentam.html (дата звернення: 22.06.2026).

2. Ollama : official website. *Ollama*. 2026. URL: https://ollama.com/ (date of access: 22.06.2026).

3. Ollama documentation. *Ollama*. 2026. URL: https://docs.ollama.com/ (date of access: 22.06.2026).

4. ollama/ollama. *GitHub*. 2026. URL: https://github.com/ollama/ollama (date of access: 22.06.2026).

5. Hardware support. *Ollama*. 2026. URL: https://docs.ollama.com/gpu (date of access: 22.06.2026).

6. Enabling AMD GPU acceleration for Ollama with Vulkan. *Binary WareHouse*. 2026. URL: https://www.binwh.com/en/2026/04/12/vulkan-ollama-amd-gpu/ (date of access: 22.06.2026).

7. AI agent node documentation. *n8n Docs*. 2025. URL: https://docs.n8n.io/integrations/builtin/cluster-nodes/root-nodes/n8n-nodes-langchain.agent/ (date of access: 22.06.2026).

8. Build custom AI agents with logic & control. *n8n*. 2026. URL: https://n8n.io/ai-agents/ (date of access: 22.06.2026).

9. Make.com – workflow automation platform. *Make*. 2026. URL: https://www.make.com/ (date of access: 22.06.2026).

10. Zapier: automate AI workflows. *Zapier*. 2026. URL: https://zapier.com/ (date of access: 22.06.2026).

11. Attention is all you need / A. Vaswani et al. *arXiv*. 2017. URL: https://doi.org/10.48550/arXiv.1706.03762 (date of access: 22.06.2026).

12. Gemma 3 technical report. *arXiv*. 2025. URL: https://doi.org/10.48550/arXiv.2503.19786 (date of access: 22.06.2026).

13. Gemma 3. *Google DeepMind*. 2025. URL: https://deepmind.google/models/gemma/gemma-3/ (date of access: 22.06.2026).

14. Alibaba introduces Qwen3. *Alibaba Cloud*. 2025. URL: https://www.alibabacloud.com/blog/alibaba-introduces-qwen3-setting-new-benchmark-in-open-source-ai-with-hybrid-reasoning_602192 (date of access: 22.06.2026).

15. Introducing Llama 3.1. *Meta AI*. 2024. URL: https://ai.meta.com/blog/meta-llama-3-1/ (date of access: 22.06.2026).

16. Llama 3.1 MODEL_CARD. *GitHub*. 2024. URL: https://github.com/meta-llama/llama-models/blob/main/models/llama3_1/MODEL_CARD.md (date of access: 22.06.2026).

17. Mistral 7B / A. Q. Jiang et al. *arXiv*. 2023. URL: https://doi.org/10.48550/arXiv.2310.06825 (date of access: 22.06.2026).

18. LMArena – benchmark & compare AI models. *LMArena*. 2026. URL: https://lmarena.ai/?leaderboard= (date of access: 22.06.2026).

19. A survey on large language model based autonomous agents / L. Wang et al. *arXiv*. 2023. URL: https://doi.org/10.48550/arXiv.2308.11432 (date of access: 22.06.2026).

20. The rise and potential of large language model based agents: a survey / Z. Xi et al. *arXiv*. 2023. URL: https://doi.org/10.48550/arXiv.2309.07864 (date of access: 22.06.2026).

21. Survey on evaluation of LLM-based agents / A. Yehudai et al. *arXiv*. 2025. URL: https://doi.org/10.48550/arXiv.2503.16416 (date of access: 22.06.2026).

22. ReAct: synergizing reasoning and acting in language models / S. Yao et al. *arXiv*. 2022. URL: https://doi.org/10.48550/arXiv.2210.03629 (date of access: 22.06.2026).

23. Mastering JSON prompting for LLMs. *Machine Learning Mastery*. 2025. URL: https://machinelearningmastery.com/mastering-json-prompting-for-llms/ (date of access: 22.06.2026).

24. A survey on hallucination in large language models: principles, taxonomy, challenges, and open questions / L. Huang et al. *arXiv*. 2023. URL: https://doi.org/10.48550/arXiv.2311.05232 (date of access: 22.06.2026).

25. LLM-based agents suffer from hallucinations: a survey of taxonomy, methods, and directions / X. Lin et al. *arXiv*. 2025. URL: https://doi.org/10.48550/arXiv.2509.18970 (date of access: 22.06.2026).

26. Maksymenko D., Turuta O. Tokenization efficiency of current foundational large language models for the Ukrainian language. *Frontiers in artificial intelligence*. 2025. Vol. 8. URL: https://doi.org/10.3389/frai.2025.1538165 (date of access: 22.06.2026).

27. f1_score. *scikit-learn documentation*. 2025. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html (date of access: 22.06.2026).

28. Measuring massive multitask language understanding / D. Hendrycks et al. *arXiv*. 2020. URL: https://doi.org/10.48550/arXiv.2009.03300 (date of access: 22.06.2026).

29. HellaSwag: can a machine really finish your sentence? / R. Zellers et al. *arXiv*. 2019. URL: https://doi.org/10.48550/arXiv.1905.07830 (date of access: 22.06.2026).

30. Think you have solved question answering? Try ARC, the AI2 reasoning challenge / P. Clark et al. *arXiv*. 2018. URL: https://doi.org/10.48550/arXiv.1803.05457 (date of access: 22.06.2026).

31. llama.cpp quantize README. *GitHub*. 2026. URL: https://github.com/ggml-org/llama.cpp/blob/master/tools/quantize/README.md (date of access: 22.06.2026).

32. Difference in different quantization methods – llama.cpp #2094. *GitHub*. 2024. URL: https://github.com/ggml-org/llama.cpp/discussions/2094 (date of access: 22.06.2026).

33. Kurt U. Which quantization should I use? A unified evaluation of llama.cpp quantization on Llama-3.1-8B-Instruct. *arXiv*. 2026. URL: https://doi.org/10.48550/arXiv.2601.14277 (date of access: 22.06.2026).

34. Python quickstart – Gmail API. *Google Developers*. 2025. URL: https://developers.google.com/workspace/gmail/api/quickstart/python (date of access: 22.06.2026).

35. Python quickstart – Google Sheets API. *Google Developers*. 2025. URL: https://developers.google.com/sheets/api/quickstart/python (date of access: 22.06.2026).

36. Закон України «Про електронні комунікації» від 16.12.2020 № 1089-IX. *Верховна Рада України*. 2020. URL: https://zakon.rada.gov.ua/laws/show/1089-20 (дата звернення: 22.06.2026).

37. 50+ customer support statistics & trends for 2025. *Pylon*. 2025. URL: https://www.usepylon.com/blog/50-customer-support-statistics-trends-for-2025 (date of access: 22.06.2026).

38. Ukraine ranks 93rd in world mobile Internet speed rating. *Ukrainian News*. 2024. URL: https://ukranews.com/en/news/1021112-ukraine-ranks-93rd-in-world-mobile-internet-speed-rating (date of access: 22.06.2026).

39. Call center burnout: how to prevent & overcome. *Salesforce*. 2025. URL: https://www.salesforce.com/blog/call-center-burnout/ (date of access: 22.06.2026).

40. Call center agent burnout: 2025 statistics and solutions. *JustCall*. 2026. URL: https://justcall.io/blog/call-center-agent-burnout.html (date of access: 22.06.2026).

41. Freshdesk Freddy AI documentation. *Freshworks Inc.* 2025. URL: https://www.freshworks.com/freshdesk/ai/ (date of access: 22.06.2026).

42. Zendesk AI pricing and features. *Zendesk Inc*. 2025. URL: https://www.zendesk.com/pricing/ (date of access: 22.06.2026).